Université Paris-Sud Master AIC

Traitement des images et du signal

Compte Rendu du projet

Étudiants :
Yang Mo & Li Zizhao

Superviseur :
Yohann Tendero

8 janvier 2018

Table des matières

1 Introduction

Il y a beaucoup des années qu'on pense l'ordinateur ou programmation peut "facilement" faire certaines choses qui sont mathématiquement formalisables, mais difficile à résoudre des problèmes "humaines". La vision par ordinateur est l'un de ces problèmes. Dans la vision par ordinateur, Les recherches de la détection et l'identification sont brûlantes et les applications sont grandes, en sécurité, en médecine, en logistique, en robotique... Aujourd'hui il existe de nombreux d'algorithmes pour la tâche de détection. Les algorithmes en générale peut être classés en 3 catégories :

- **Les algorithmes basés sur caractères géométriques :**
  Ce type de méthode est du genre classique et ils sont développés il y a 50 ans. Les algorithmes essaye d'analyser le profil géométrique d'un visage en terme de yeux, de nez, de bouche et de cheveux y compris les points caractéristiques dans une image. Les caractères comme la distance, onglet, sont utilisés pour la détection. Aujourd'hui ces méthodes sont moins utilisées, parce que les caractères géométriques sont juste la mesure de base pour la représentation de visage, ou plutôt dire : les informations peu profondes. aujourd'hui il est moins utilisé pour la tâche de détection.

- **Les algorithmes basés sur template**
  Ce type de méthode utilise a pré-traîné modèle pour prédire les questions de classification(visage ou non-visage) et régression(location). L'un des algorithmes de ce type le plus populaire est l'Eigenface (L'analyse en composantes principales). Le principe de cet algorithme est de statistiquement trouver les vecteurs caractéristiques qui peuvent représenter les caractères de visage. Ces vecteurs sont appelés "Eigen-

face". L'algorithme est une des méthodes la plus populaire aujourd'hui en mesure de la facilité d'implémentation et la haute précision.
Un autre algorithme très à la mode aujourd'hui est des réseaux neurones. L'algorithme de réseaux neurones a grand avantage par rapport aux autres. Sur tous algorithmes, le problème le plus important est d'avoir caractères pour représenter l'objet. C'est assez important que "Feature engineering" devient un domaine de recherche très chaud. D'autre part, les réseaux neurones sont une méthode se-organisé et se-adapté. Le processus d'apprendre est lui-même le processus de sélection de caractères. Pour la tâche de détection d'objet, il y a beaucoup de règles qui sont difficile à décrire, alors réseaux neurones peut apprendre les informations profondes dans le processus d'apprentissage notamment quand la structure de réseaux est beaucoup plus complexe.

- **Les algorithmes basés sur modèle**
  Les représentes de ce type de méthode sont Hidden Markov Model, 2D Active Appearance Models et 3D Morphable Models.

Sur toutes ces méthodes, nous avons choisi une méthode publiée par Paul Viola et Michael Jones en 2001 : « Rapid Object Detection using a Boosted Cascade of Simple Features » aussi nommé comme « Méthode Viola Jones » [1]. On la choisi non seulement en raison que c'est une méthode classique pour la détection d'objet, mais aussi pour que c'est une application réussite en utilisant théorie de machine learning en production. Cette méthode est facile à implémenter, pouvoir détecter les visages à toute échelle, et le plus important c'est la détection est en temps réel (15 images/s) sur un smart-phone.

Le but de notre projet est de comprendre cette méthode en détail et d'implémenter la méthode en Python, aussi pour réaliser le résultats qu'auteur a

mentionné dans son papier. Les détails d'algorithme et notre implémentation vont être discutés par la suite.

2 Présentation de l'algorithme

La méthode Viola Jones est une méthode sur template. Pour réaliser le système de détection, il faut suivre deux parties : la partie d'apprentissage et la partie de détection. l'apprentissage est pour apprendre les paramètres de template pour la détection. l'apprentissage est long mais n'a besoin d'être exécute qu'une seule fois. La détection exploite les données de l'apprentissage pour rapidement détecter un visage dans une image inconnue.

La méthode Viola Jones utilise l'algorithme AdaBoosting pour s'entraîner à découvrir des caractéristiques spéciales des visages. Le Boosting est une famille d'algorithme d'apprentissage entraînant un large ensemble de classifieurs faibles, tous ces classifieurs votent pour construire un classifieur fort. Les détails pour la marche de Adaboost va être présenter plus tard.

2.1 Haar Like features

Caractère est le plus intéressante de la méthode. Cette méthode utilise 'Haar-like' Caractères. Un "Haar-like" caractère est un produit scalaire entre image et quelque Haar-like template. Pour plus préciser, on met I et P représente image (aux niveaux gris) et un template, on assume que les tailles sont tous NxN. Un caractère peut être défini comme ci-dessous :

$$\sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i,j)1_{P(i,j)\ is\ white} - \sum_{1\leq i\leq N}\sum 1\leq j\leq N I(i,j)1_{P(i,j)\ is\ black}$$

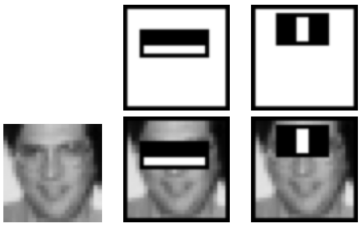

FIGURE 1 – Haar-like caractères sélectionné

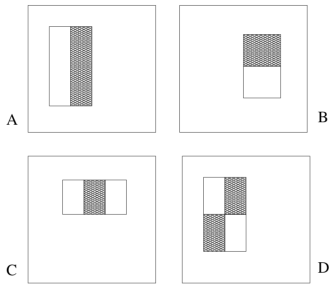

FIGURE 2 – Haar-like caractères templates

On peut imaginer que un caractère comme cela peut probablement aider de juger un image est "visage" ou "non-visage". Et il existe très de nombreuse Haar-like caractères. Pour une image de taille 24 x 24, on peut tirer plus de 180000 caractères, cela pose deux problèmes, 1 : il faut sélectionner Haar-like caractères utiles. 2 : il faut un moyen de calculer rapidement la valeur de Haar-like caractères.

2.2 Calcul de l'image intégrale

Pour calculer rapidement la valeur de Haar-like caractères, l'image intégrale est introduit. Au lieu de calculer pour chaque pixel, on peut utiliser programmation dynamique pour faciliter le calcul. Pour plus précis, on crée une matrice de la même taille de image : N x N. Chaque positionne de

l'image enregistre le somme de valeur pixel à son gauche et au dessus, c'est pour cette raison que cette méthode est appelée "l'image intégrale". On définit l'intégral de image I ci dessous :
$II(i,j) = \sum_{1 \leq s \leq i} \sum_{1 \leq t \leq j} I(s,t)$

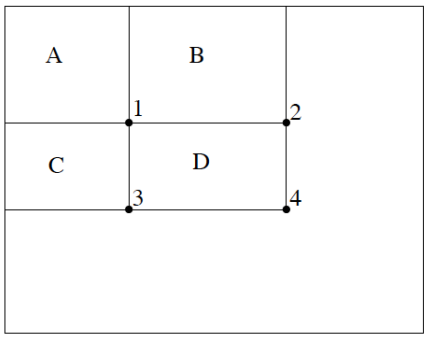

FIGURE 3 – Image Intégrale

Avec telle matrice, on peut calculer le somme d'une région rectangulaire facilement. On monte un exemple d'une figure dessus, on veut calculer le somme de pixels d'une région D qui peut être calculé comme $4 + 1 - 2 - 3$ (pour la valeur de matrice dans chacune des positions). En utilisant cette méthode, on peut rapidement calculer la valeur d'un n'importe quel Haar-like caractère.

2.3 Le processus d'apprentissage

Dans cette partie, on va décrire le processus d'apprentissage d'AdaBoost et de cascade attentionnée. On utilise les images de petites tailles 19 x 19 comme trainset. Les images soit positives (visage), soit négative(non-visage). La résolution 19 * 19 est aussi le base de fenêtre pour la détection. Donc le problème devient un problème de classification. Le but d'apprentissage est

d'entraîner notre modèle de détection sorti une prédiction de dire l'entrée est visage ou non-visage.

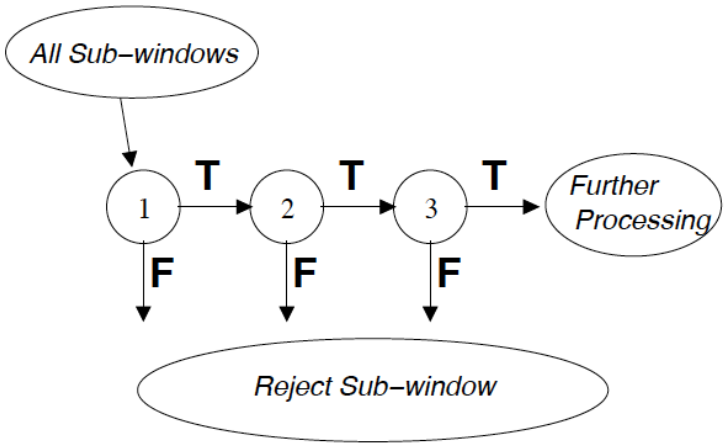

FIGURE 4 – Le processus de cascade attentionnée

Mais avec un seul classifieur fort composé de certain caractères, on ne peut pas avoir un haut taux de détection. Donc on essaye de concaténer les classifieurs forts afin d'avoir un haut taux de détection. Aussi, après chaque passage de classifieur fort, on rejette les exemples négatifs, autrement dit, il n'y a que des exemples positifs qui passent dans le classifieur fort suivant. Ce genre de mesure nous aider de accélérer le processus d'apprentissage.

2.4 Le processus de détection

Après l'entraînement, on va utiliser le modèle pour la détection. Par rapport à la partie "Apprentissage", la détection est rapide. On assume que l'image entrée est de la taille M x N, d'abord, on utilise la fenêtre de détection de base résolution (dans notre cas, 19 * 19) glisser pixel par pixel

dans l'ensemble de image, et dans chaque position on fait la prédiction si la région dans la fenêtre est visage ou non-visage. Après, la fenêtre grandi son taille (x1.5 chaque fois par exemple) pas en pas et répéter le processus de détection jusqu'à finir la détection en tout échelle. Pour un image de 500 x 500, il y a plus que 100000 fenêtres à vérifier. La méthode de "sliding window" demande concrètement énorme de calcul et l'avantage de la structure cascade est évident. la fenêtre négative va être refusée dans certaines premières décision stumps. Seulement la fenêtre qui semble positive va passer plus profond de la cascade donc la détection est beaucoup plus vite. En plus, l'image d'intégral aussi accélère le calcul. Après on va avoir un ensemble de fenêtres positives overlaping. On calcule le moyenne de ces fenêtres et sortie le résultat final.

3 Implémentation de l'algorithme

On implémente l'algorithme d'après les étapes proposées par Viola et Jones. Dans notre archives, on a quatre principaux programmes qui correspondent aux approches : le calcul de l'image intégrale, la construction des caractères Haar-like, l'implémentation d'AdaBoost, l'implémentation de cascade attentionnée. Ensuite, on les intègre dans une fonction main afin d'apprendre et d'évaluer notre système de détection de visages.

Quant à la base de données, on utilise celle-là de CBCL[2] qui est une base de données de visages verticales frontales de taille $19 * 19$ au niveau de gris, y compris 2429 visages, 8548 non-visages pour le processus d'apprentissage, 472 visages, 19572 non-visages pour le processus de test. Dans la base de données, il y a presque tous les visages de différentes races, de différents âges, et bien sûr, de femmes et de hommes avec différents attributs de visage pour l'ensemble de visages. Il existe aussi des images comme une partie de

visage qui peut être déranger le processus de classification dans l'ensemble de non-visages.

Comme parlé dans la partie précédente, la proposition du concept de l'image intégrale est pour but de faciliter le calcul de caractères. Dans notre implémentation, dans un premier temps, on calcule l'image intégrale en rajoutant une colonne supplémentaire et une ligne supplémentaire. Ensuite, on calcule le somme des pixels dans une région rectangulaire qui est la base de caractères Haar-like.

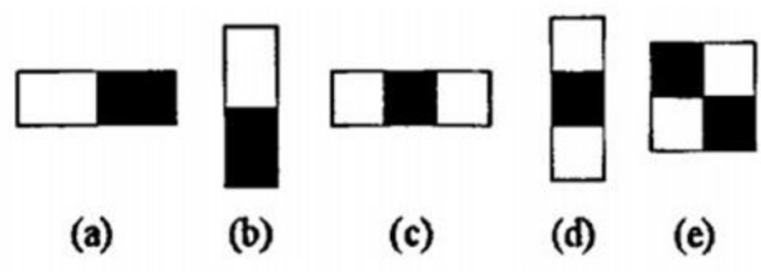

FIGURE 5 – Caractères Haar-like

Dans l'article original, l'auteur définit 5 types de caractères montrant dans Figure 5. Le phase suivant consiste à implémenter ces 5 caractères dont on définit la taille de caractère et la position que le caractère applique. Ensuite, on implémente le classifieur faible, c'est à dire on vote sur l'image pour chaque caractère.

Une fois que les caractères soient prêts, on passe dans l'étape d'AdaBoost pour construire un classifieur fort. Le classifieur fort comprend plusieurs classifieurs faibles, qui sont pris par le somme d'erreurs minimum pondérées de la classification faible. Il faut faire attention qu'à chaque itération, on enlève

le classifieur choisi puisqu'on ne le prend qu'une seule fois et met à jour les poids des classifieurs faibles. Le processus complet de'Adaboost est montré dans Figure 6.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

FIGURE 6 – L'algorithme d'AdaBoost

La cascade attentionnée consiste à concaténer plusieurs classifieurs forts.

Ici, on n'implémente qu'une version simple : on ne considère pas le taux de faux négatif de chaque étape ni le taux de détection de chaque étape. Ce qu'on a fait, c'est d'enlever les exemples négatifs à chaque étape.

4 Présentation des résultats

Dans le papier, l'auteur utilise une base de données de 9832 visages (4916 visages et leurs images miroir verticales) et 10000 non-visages en sélectionnant au hasard de sous-fenêtres à partir d'un ensemble de 9544 images qui ne contiennent pas de visages. Les images sont de taille $24 * 24$. Ils utilisent une architecture de cascade de 38 couches avec respectivement 1, 10, 25, 25, 50 classifieurs forts dans les cinq premières couches pour détecter les visages verticales frontales. Le nombre total de caractères utilisés dans toutes les couches est 6061. Chaque classifieur fort dans la cascade a été formé avec 9832 visages d'entraînement et 10000 non-visages. Pour la première couche de la cascade, les exemples non-visages viennent de la sélection au hasard de sous-fenêtres à partir d'un ensemble d'images qui ne contiennent pas de visages. Les exemples non-visage utilisés pour former les couches suivantes ont été obtenus en balayant la cascade partielle à travers les images non-visages et en recueillant des exemples faux positifs. Un maximum de 10000 de ces sous-fenêtres non-visages ont été collectées pour chaque couche.

Vu que la taille de notre base de données et la capacité de calcul de notre machine, on n'arrive pas à tout copier le processus proposé par Viola et Jones. Ici, on présente notre architecture propre de la cascade : on calcule les caractères au carré de taille 8 à 10 pour les images de taille $19 * 19$. Au total, on obtient 2496 caractères, ensuite, on les passent dans une cascade de 3 trois couches avec respectivement 1, 2, 5 caractères. Les résultats de classification

sur le test set sont listés dans Figure 7. Si on calcule les caractères au carré de taille 6 à 10 pour les images de taille $19 * 19$. Au total, on obtient 11552 caractères, ensuite, on les passent dans une cascade de 5 trois couches avec respectivement 1, 2, 2, 5, 5 caractères. Les résultats de classification sur le test set sont listés dans Figure 8.

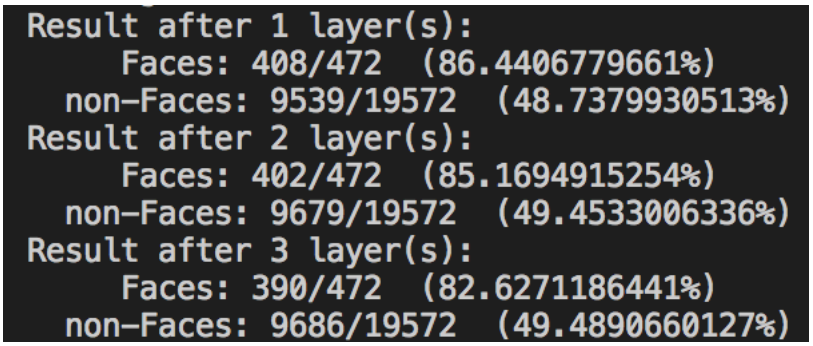

FIGURE 7 – Résultat de classification avec une cascade de trois couches

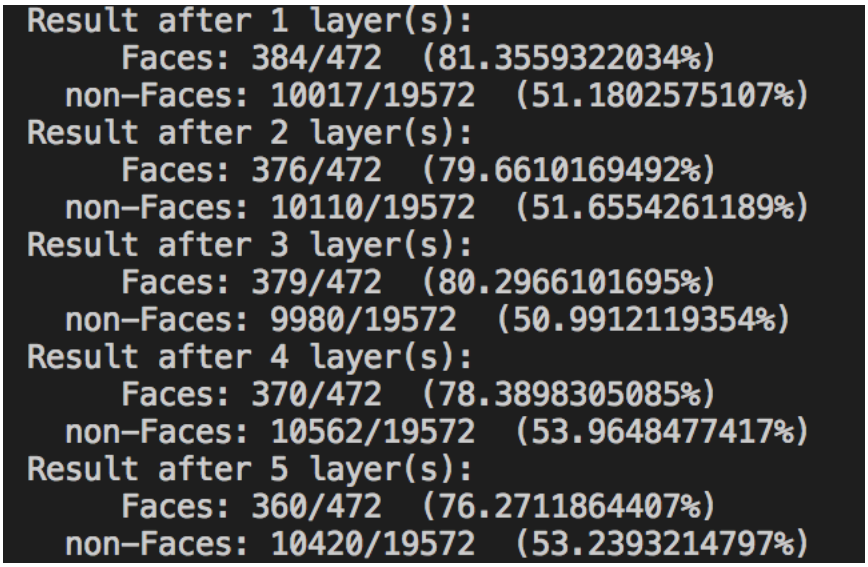

FIGURE 8 – Résultat de classification avec une cascade de cinq couches

A cause de la capacité de notre ordinateur, on n'a pas pu choisir un grand nombre de caractères ni un classifieur fort de grosse taille qui nous induit un très long processus d'apprentissage. Aussi, la méthode Voila Jones a besoins de nombreux exemples de non-visages pour l'apprentissage. Donc on s'en fout sur le problème de taux faux négatif mentionné dans la partie de cascade de l'article. Le seuil utilisé par le classifieur faible est toujours 0.

Ici, on n'évalue que la qualité de classifieur, on voit que sur le test set de visages, on a un taux de détection d'envirion 80%, mais sur le test set de non-visages, on obtient un résultat comme la recherche aléatoire. On pense que c'est parce qu'on rejette beaucoup d'exemples négatifs pendant l'apprentissage, si on n'applique qu'un classifieur fort avec 5 caractères, sur le test set de non-visages, on obtient un taux de détection d'environ 70% même si dans ce cas-là, le taux de détection sur le test set de visages est 23%.

On n'a pas implémenté la détection sur une "vrai" image car on pense ce n'est pas la partie la plus importante dans ce projet. Le but de ce projet est l'implémentataion d'algorithme.

5 Conclusion et Amélioration

Dans ce projet, on a réalisé l'implémentation de l'algorithme Viola et Jones en implémentant l'image intégrale, les caractères Haar-like, l'AdaBoost et la cascade attentionnée. On a entraîné un modèle de cascade sur un ensemble d'entraînement, et puis on l'a testé sur un ensemble de test, avec une architecture simple, on arrive à un taux de détection d'environ de 80%.

Il existe des pistes qu'on peut améliorer notre implémentation. Par exemple,

avec l'implémentation actuelle, tester le système avec plus de caractères et avec des classifieurs plus forts qui consiste à contenir plus de classifieurs faibles dans un classifieur fort.

Le phase suivant est d'implémenter un processus de cascade adaptative qui est contrôlée par le taux de faux négatif de chaque couche et le taux de détection de chaque couche, aussi que le taux de faux négatif défini par l'utilisateur. La cascade adaptative est aussi l'endroit où on introduit le mécanisme de la régularisation de seuil.

Référence

[1] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[2] Cbcl face database #1. http ://cbcl.mit.edu/software-datasets /FaceData2.html.