Détection de visage

Fonctionnement d'OpenCV

Rafik LACHAAL

Généralité

Bien faire la différence entre détection de visages et reconnaissance faciale :

- La détection de visage nous dit si oui ou non il y a un visage sur une image.
- La reconnaissance faciale nous dit à qui appartient ce visage.

Quatre étapes pour la détections de visages :

1. Machine learning et jeux d'apprentissages étiqueté :
   - Images avec un visage = Positifs
   - Images sans visage = Négatifs
2. Il faut extraire des caractéristiques depuis ces données :
   - Qu'est ce qui caractérise un visage ? Le nez, les yeux, la bouche, …
   - Mais qu'est ce qui caractérise l'absence de visage ?
   - Problème : trop de caractéristiques → étape 3

Généralité

3. Adaboost pour réduire le nombre de caractéristiques tout en préservant la qualité du modèle :
   - On passe de quelques centaines de millier à plusieurs milliers de caractéristiques.
   - Pas suffisant pour réduire le temps d'exécution → étape 4.
4. Classificateur en cascade :
   - Nous permet d'accélérer la détection de visage.

Idée d'application : Une appli qui reconnaitrais les visages sur une image et les échangerais de position.

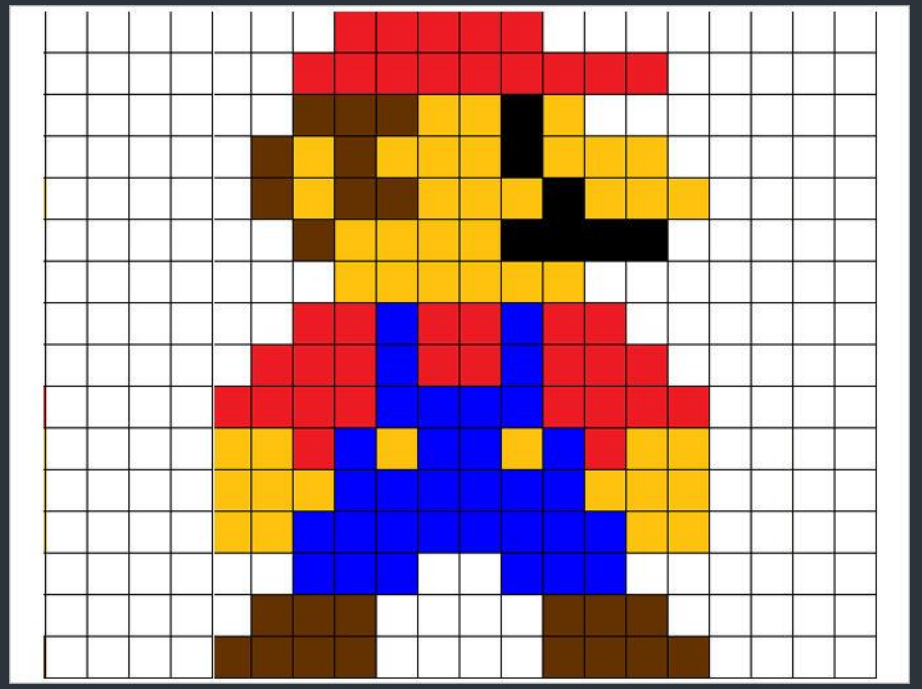
Les caractéristiques (features)

Définition d'une caractéristique :

C'est une propriété **quantifiable** partagé par des exemple partagée en machine learning.

Quantifiable :

- C'est-à-dire que l'on peut réécrire sous forme **numérique**.
- Exemple une image est composé de pixel chacun étant définit par un vecteur contenant 3 valeur donnant les niveaux de couleur (rouge, vert, bleu).
- A partir de ces pixels on peut extraire d'autre caractéristiques plus élaborés comme des formes ou des tailles.



Les caractéristiques (features)

On peut lister les caractéristiques utilisées en reconnaissance d'image comme suit :

- Les bordures :
  - Dans notre exemple de Mario quand on passe d'un pixel blanc à un pixel d'une autre couleur on sait qu'on à atteint une bordure.
  - De manière plus général on sait que l'on à atteint un contour quand on a un changement **brusque** de couleur (plusieurs nuance d'une même couleur sont des couleurs différentes mais ne marque pas un contour).
- Les lignes :
  - Plusieurs pixels ayant la même couleur (ou à peut près la même couleur) qui se suivent.
- Le changement de texture :
  - Se traduit par l'alternance de pixels de couleurs différentes de manière croisé (comparable à un dégradé).

Les caractéristiques (features)

Pour notre projet de reconnaissance facial nous allons utiliser **OpenCV**.

Pour une plus grande rapidité de calcule ce dernier utilise les **caractéristiques (pseudo-) Haar** pour extraire les caractéristiques propres aux visages.

Déroulement de l'extraction :

- Tout d'abord les images sont ramenées en nuance de couleur noire/blanc. **Les pixels ne sont plus codés qu'avec une seule valeur.**
- Ensuite l'image et découpé en plusieurs zones rectangulaires adjacentes, **des fenêtres de détection de 20 x 20 pixels généralement.**
- Enfin, on va sommer les intensités (la valeurs) des pixels de chacun des rectangle et calculer la différence avec la somme le rectangle adjacent.

  → **Pour chaque fenêtre on aura donc une valeur qui lui est propre.**

Les caractéristiques (features)

Chacune de ces valeurs nous permettra d'identifier si notre fenêtre contient :

- des bordures horizontales (1)
- des bordures verticales (2)
- des lignes (3)
- un changement de texture (3)

Pour chacun de ces cas de figure on aura une valeur différente.

Les caractéristiques (features)

- **Une caractéristique de pseudo-Haar est donc un nombre réel** qui code les variations du contenu pixellique à une position donnée dans la fenêtre de détection.

$$Caractéristique\ de\ Haar = Somme\ rectangle1\ - somme\ rectangle2$$

- Une image peut donc être traduite comme une suite de caractéristique de Haar que l'on peut directement injecter dans un classificateur.
  - Pour une petite image classique ~ 150 000 caractéristiques différente.
  - C'est encore trop d'information à traiter.

Sélectionner les meilleurs caractéristiques avec Adaboost

Le Boosting : Hypothésis boosting (en français : Renforcement d'hypothèse)

- Toute méthode qui va combiner plusieurs mauvais classificateurs pour en faire un seul bon.
- L'idée général est d'entraîner plusieurs classificateur l'un après l'autre chacun s'efforçant de corriger sont prédécesseur.
- Les deux principales méthodes de boosting sont :
  - **L'Adaboost** (algorithme itératif)
  - **Le gradient Boosting**
- On dit aussi que se sont des méta-algorithme

Sélectionner les meilleurs caractéristiques avec Adaboost

Fonctionnement général d'Adaboost :

- A chaque observation Adaboost va assigner un **poids $D(i)$** pour leur donner une certaine importance (au départ toutes les observations ont toutes le même poids).
- A chaque itération il va :
  1. Entraîner un classificateur de base (un arbre de décision par exemple)
  2. Puis il va faire des prédictions sur le jeu d'entraînement.
  3. Il va ensuite calculer l'erreur de prédiction ε comme étant la somme des poids des observations mal classées.

$$\varepsilon_t = \sum D_t(i)[y_i \neq h(x_i)]$$

Pour chaque prédicteur on à aussi un poids qui est calculé :

$$\alpha_t = \eta . \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$$

Sélectionner les meilleurs caractéristiques avec Adaboost

4. Les poids des observations sont ensuite mis à jour :
   - **Le poids des obs bien classées sont reste inchangés.**
   - **Le poids des obs mal classées sont accrus** et calculé en fonction de α, donc de ε. On dit qu'elles sont boostées.
5. L'algorithme réitère les opérations précédente avec un nouveau classificateur et les poids nouvellement calculés

- L'algorithme prends fin lorsque tout les prédicteur ont été parcourus ou que l'on a trouvé un prédicteur parfait.

Pour faire des prédiction Adaboost va calculer toute les prédictions retournées par chacun des prédicteurs, chacun d'eux retournera soit 0 soit 1.

- Adaboost sommera le poids des prédicteurs retournant 0.
- Et il sommera le poids des prédicteurs retournant 1.

La prédiction retourné sera celle pour la quel la somme sera la plus élevé.

L'avantage d'Adaboost, c'est qu'en pondérant de plus en plus les observations mal classé il va se concentrer de plus en plus sur celle-ci pour chercher à bien les classer.

Sélectionner les meilleurs caractéristiques avec Adaboost

OpenCV va utiliser Adaboost pour sélectionner les meilleurs caractéristiques (les plus représentatives) parmi une liste de caractéristiques.

- Pour ce faire il va sélectionner une caractéristique de manière aléatoire et entraîner un mini classificateur pour classifier nos images.
- Pour chacune de nos caractéristiques il verra à quel point le classificateur est mauvais.
- C'est comme ça qu'il va pouvoir classer les caractéristiques en fonction de leur utilité et créer une liste restreinte.
- Ainsi on peut passer de 150 000 caractéristique à environ 5000.

Classificateur en cascade

Plutôt que de prendre toutes les caractéristiques en même temps pour savoir sil y a un visage ou non on peut les séparer en plusieurs étapes grâce à un **classificateur en cascade**.

On groupe nos millier de caractéristiques sur différent niveaux.

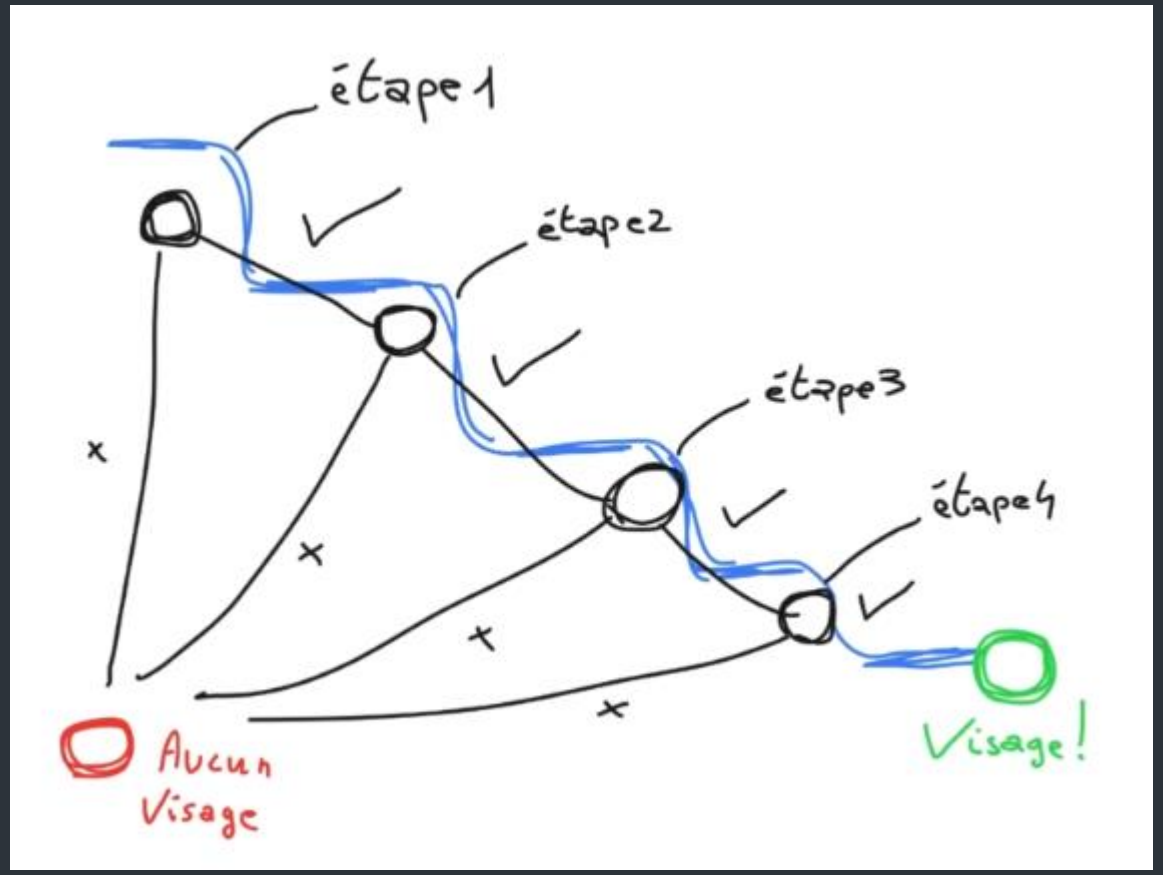
Ainsi on n'a pas besoin de vérifier toutes les caractéristiques en une seul étapes

L'or de l'étape on va vérifier si certaine caractéristiques nécessaire pour former un visage sont bien présente :

- Si oui alors on passe à l'étape 2 et on recherche d'autre caractéristique.
- Si non on s'arrête et on conclu qu'il n'y a pas de visage.

Si on valide toutes les étapes alors on a un visage.

Du fait que l'on ai pas à vérifier toutes les caractéristiques à chaque étape on gagne en temps de vérification.



Paramètre d'échelle

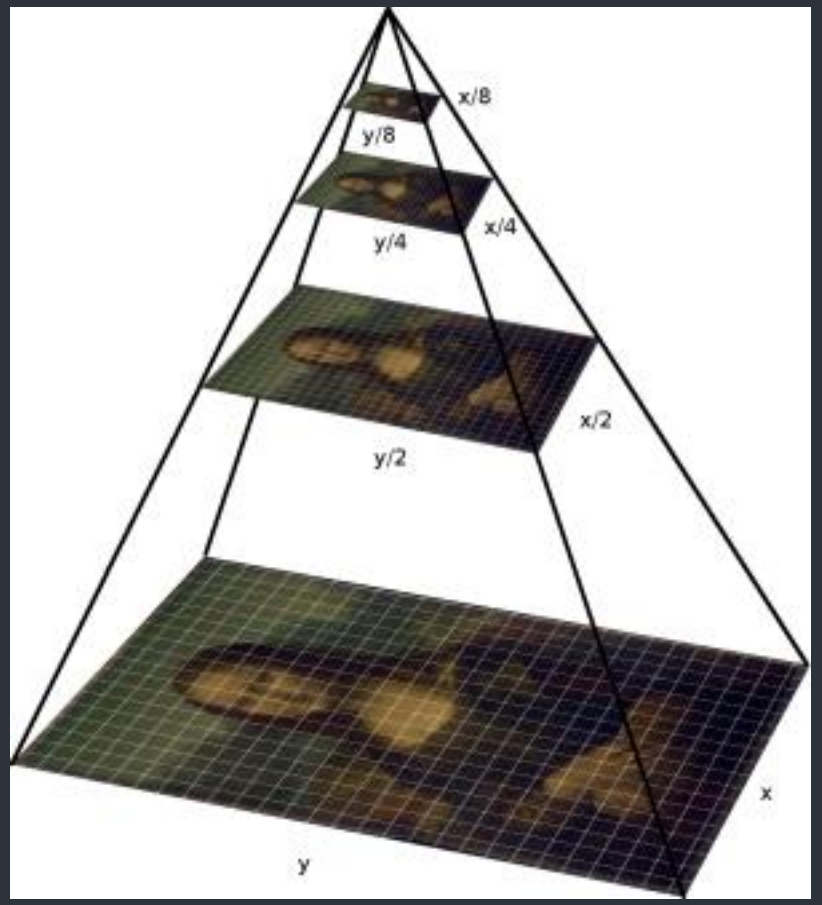
Ce paramètre d'échelle va nous permettre d'ajouter plus de robustesse à notre projet de détection de vissage en ajoutant plus de robustesse à notre image. L'idée est de :

- Pouvoir s'adapter à toutes taille de visage
- Pouvoir s'adapter à toute forme de visage
- Pourvoir s'adapter si le visage et au premier plan ou pas.

La solution : On va utiliser un pyramide d'image

Cela signifie proposer l'image testé à plusieurs échelles différentes en la redimensionnant.

Notre algorithme sera invariant à l'échelle.



Paramètre d'échelle

Le paramètre d'échelle est de combien on rétrécit une image à chaque étape de la pyramide.

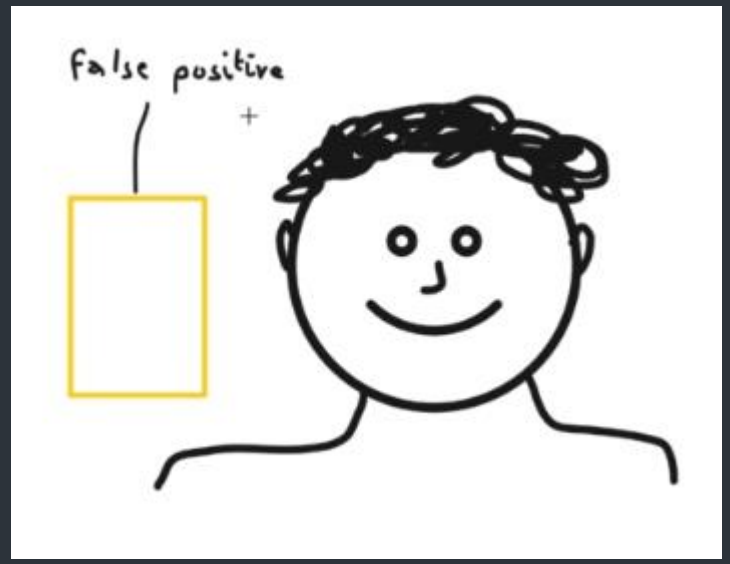
- Il est souvent noté en pourcentage.
- Par exemple un paramètre d'échelle de 10% on va réduire l'image de la pyramide de 10% à chaque étage de la pyramide.
- Les bonne valeur pour ce paramètre d'échelle sont généralement comprise entre 1,05 et 1,10 (≡ 5 et 10 %).
  - Plus se facteur est petit et plus la détection prendra du temps car on teste plus d'image.
  - En revanche plus il est grand et plus on à de chance de manquer un visage.

C'est un paramètre d'OpenCV qu'il faudra faire varier.

Paramètre du nombre minimum de voisins

Un problème récurrent en détection de visage est la détection de **faux positif**.

C'est-à-dire détecter un visage là ou il n'y en a pas.



Cela est notamment dû au paramètre d'échelle qui redimensionne notre image à chaque niveau de la pyramide.

Paramètre du nombre minimum de voisins

Pour lutter contre cela on à un autre paramètre : **Nombre minimum de voisins**

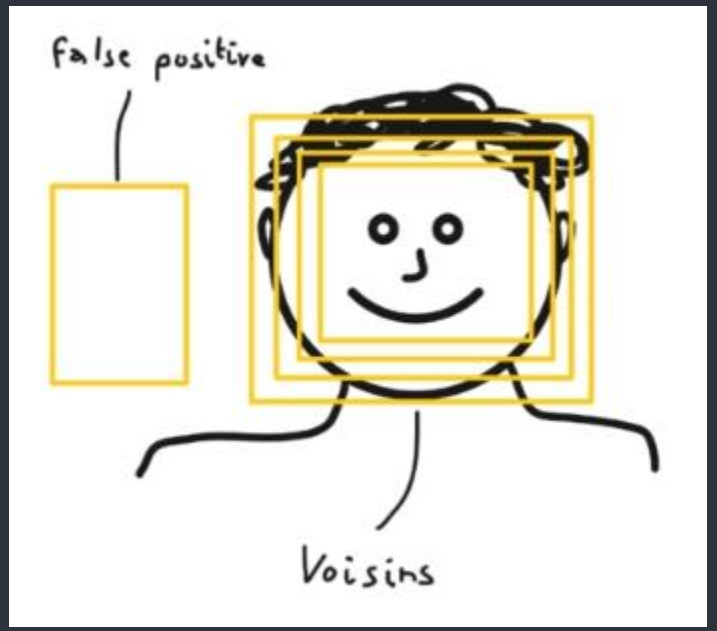
Sont rôle est de détecter si une détection de visage est un false positive ou un vrais positif.

Comment ça marche ? :

Grâce au paramètre d'échelle un même visage peut être détecté plusieurs fois.

On obtient donc plusieurs détection d'image enchevêtrés les une dans les autres.

- → C'est ce que l'on appel des **voisins**
- → Le paramètre du nombre minimum de voisin détermine combien de voisin on à besoin pour valider définitivement un visage.
- → Ce paramètre est donc un seuil.



Paramètre du nombre minimum de voisins

Pour résumer :

Sont rôle est de détecter si une détection de visage est un false positive ou un vrais positif.

- Contrôle le nombre de faux positives
- Seuil de notre détection de visage :
  - Nombre minimum de voisins qu'un potentiel visage à besoin pour être validé définitivement.
  - Les bonnes valeurs sont comprises en 3 et 6.
- Plus cette valeur sera élevé et plus grande sera la confiance dans les visage détectés.
- Attention si la valeur est trop forte on peut avoir moins de détection et manquer un vrais visage.

On passe à la pratique

Lien recommandé pour installer **OpenCV** :

https://anaconda.org/conda-forge/opencv