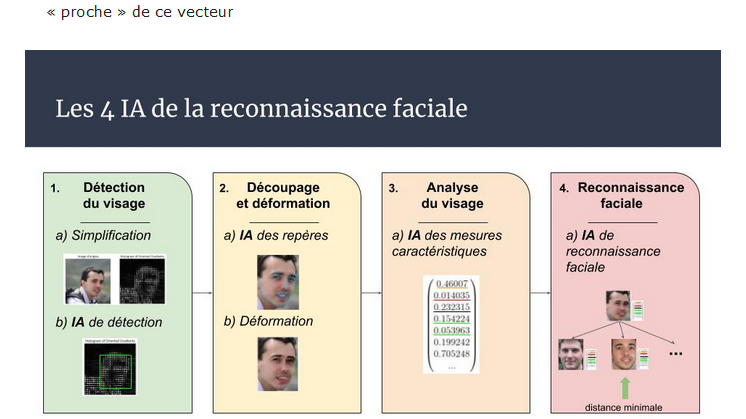
# Reconnaissance Faciale

La reconnaissance faciale (ou du visage) est une méthode d'identification biométrique d'un individu qui consiste à comparer les données d'une image numérique ou capturée en direct avec le dossier de la personne concernée.

Si les applications les plus courantes des systèmes de reconnaissance faciale concernent la sécurité, elles touchent cependant de plus en plus d'autres domaines. Citons par exemple Kinect, le système de jeu à reconnaissance de mouvement, qui utilise cette technologie pour différencier les joueurs, ou encore certains systèmes de paiement mobile qui s'en servent pour authentifier les utilisateurs de manière sécurisée.

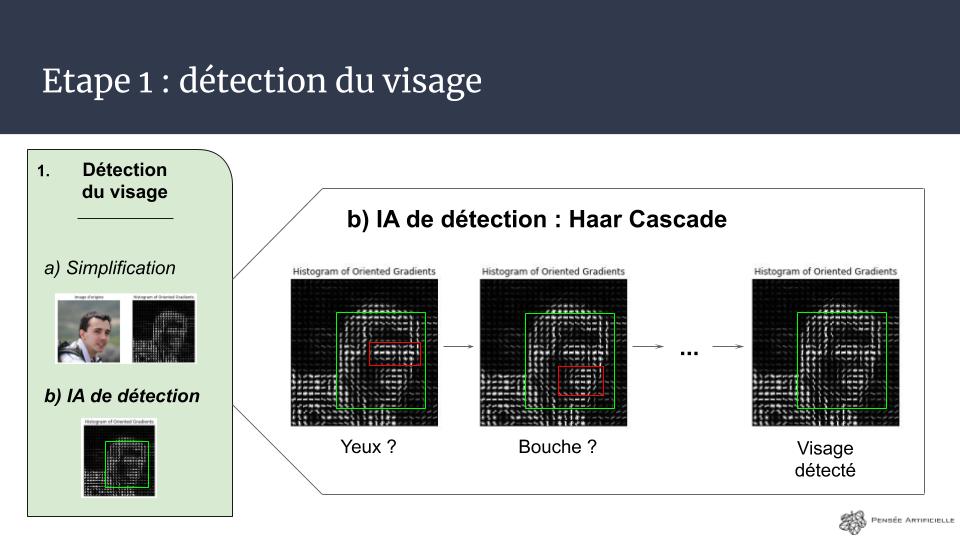
La reconnaissance faciale n’est pas une intelligence artificielle (IA) qui reconnait les gens, mais, quatre programmes IA différents, qui s’enchaînent pour arriver à identifier quelqu’un ou à vérifier une identité :

* Un premier programme IA trouve le visage dans l’image (avec l’algorithme Haar Cascade)
* Puis le visage est découpé et déformé pour être recentré et réaligné par rapport à l’image,
* Ensuite un autre programme analyse l’image (apprentissage) ;
* Enfin, la dernière IA cherche dans notre base de donnée quelle est la personne la plus « proche » de ce visage



Dans notre cas, nous allons mettre en place toute la chaîne ci-dessus à l’exception de la partie 2 (découpage et déformation) car on se placera toujours face à la caméra pour les reconnaissances.

## Détection du visage

Cette partie consiste de détecter un visage dans une vidéo webcam comme décrit dans la figure ci-dessous :

D’après notre schéma, dans cette étape nous devons :

* Convertir l’image en niveaux de gris
* Envoyer cette image à l’algorithme Haar Cascade qui s’occupera de trouver les visages
* Tracer des rectangles autour de tous les visages trouvés et découper l’image autour du premier.

Il s’agit de faire un algorithme, qui va scanner l’image de gauche à droite et de haut en bas afin que de chercher les objets décrit dans le fichier XML : détection de visage (haarcascade\_frontalface.xml), détection de yeux (haarcascade\_profilface.xml), et détection des éléments du corps (haarcascade\_fullbody.xml).

## Analyse du visage

On commence donc par charger notre fichier haarcascade et par le configurer en indiquant comment il doit fusionner les visages qu’il identifie (en effet, il va trouver plusieurs fois le même visage en décalant simplement sa zone de quelques pixels).

Pour réaliser cela, il va avoir trois étapes, la première est de constituer une base de données de visage (dataSet) à noter, et à côté de chaque visage l’identité de la personne ensuite nous mettrons en forme toutes ces données dans des tableaux pour notre fonction d’apprentissage. On réalisera à la suite un programme dans un troisième temps pour mettre une identité au-dessus de chaque personne.

### Création d’une base donnée d’images (dataSet) :

Ce premier programme consiste à partir du programme détection de visage, d’enregistrer dans un dossier dataSet à l’aide de la fonction DetectM

ultiScale qui permet de récupérer chacune de visage de la webcam. Ensuite on donne un format avec la fonction **imwrite** avec lequel on stocke l’mage dans le dataSet. Et laisse tourner le programme à un moment et les images sont enregistrées au fur et à mesure dans le répertoire dataSet. (Voir en annexe le diagramme d’activité)

### Apprentissage

Nous allons maintenant parcourir l’arborescence du répertoire dataSet avec la fonction walk de la librairie OS pour construire deux tableaux, un avec l’ensemble des images et un avec l’identité à côté. L’identité étant le nom du répertoire. Nous pouvons maintenant faire l’apprentissage à l’aide de ces deux tableaux et écrire les fichiers qu’on pourrait utiliser à la suite. Ainsi il va avoir deux fichiers, la fonction d’entrainement ne va mettre dans le fichier résultat que les caractéristiques de visage et juste un identifiant. La correspondance entre identifiant et le vrai nom (labelle) sont enregistrées dans un fichier avec le format pickles qui est un format de fichier particulier et qui contient le vrai labelle.

Pour cela :

* On parcourt le dossier dataset
* Pour chaque image trouvée, on extrait le visage
* Puis on réalise une prédiction dessus (id et labelle)
* Et on stocke ce visage dans un dictionnaire qui va représenter notre base de données entrainée au format yml ; ( voir en annexe le diagramme d’activité)

### Identification

Maintenant qu’on a les deux fichiers, un avec les caractéristiques de visage et avec l’identité labelle, on va pouvoir ouvrir ces deux fichiers et les lire avec des fonctions **Read** et placer des noms au-dessus d’un visage.

Nous procédons de la même manière que le programme détection d’objet, nous ajoutons juste la fonction Recognizer.predict qui prend en paramètre l’image récupérée du fichier de la webcam et qui renvoie deux variables, la première est l’identifiant et la deuxième un indice de confiance, plus l’indice de confiance est bas, plus on est sûr que l’identité est bonne et plus il est élevé, moins la fonction est sûr d’elle dans sa prédiction. La valeur qu’on a assez souvent est en général entre 80 et 100. Nous l’avons donné 95, si l’indice de confiance est inférieur à 95 on cherche dans le fichier labelle, l’identité de la fonction de prédiction, si l’indice est supérieur à 95 on affiche l’identité inconnue. Et on les affiche avec la couleur verte si l’identité est connue ou avec la couleur rouge si l’identité est inconnue grâce aux fonction cv2.putText et cv2.rectangle. ( voir en annexe le diagramme d’activité)

## Amélioration de la reconnaissance

Dans cette partie, nous allons modifier les trois programmes avec lesquels nos résultats n’étaient pas trop mauvais mais contiennent quelques faux positives(erreurs).

### Amélioration du dataSet

Cette première modification concerne le dataSet, il est important quand on constitue un dataSet de lui donner des informations les plus divers possibles. Dans notre cas, quand on donne le visage, il va falloir donner de visage avec des expressions différentes sous une luminosité différente, des angles légèrement différents et le nombre maximum d’image enregistré au lieu d’enregistrer toutes les images on enregistre une sur quatre et ce qui va permettre d’avoir un dataSet meilleur avec des images plus variées.

La deuxième chose qu’on a fait, c’est d’écarter ou supprimer des images floues pour éviter les erreurs sur l’identification pour ça nous avons utilisé la fonction Resize de l’open cv et en fin une troisième chose qu’on a fait, c’est de normaliser la taille des images en les redimensionnant à une taille fixe avant de les mettre dans le dataSet ce qui va nous permettre d’obtenir résultat bien meilleur.

### Amélioration de l’apprentissage

Dans le programme d’apprentissage, comme évoqué dans la partie dataSet, nous allons réinitialiser la taille des images avec la fonction Resize de la librairie openCV ensuite on l’applique une opération mathématique Laplacienne pour savoir si l’image est floue ou pas. Et cette opération Laplacienne renvoie un indice qui constitue un indice d’honnêteté. Si l’indice d’honnête est inférieur à 250 on exclut l’image, sinon on l’ajoute dans le dataSet. Ce qui permettra d’écarter les images légèrement floues et un certains nombres de faux positive ça évitera de mettre sur certains visages des identités qui ne sont des bonnes identités.

### Amélioration de l’identification

Dans le dernier, on fait la même chose, juste avant la fonction prédiction on réinitialise la taille des images. Et on ne fait rien d’autre comme modification.

# Annexe

