

UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE DE
LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR :
SOUKAINA ALAOUI ISMAILI

BIG DATA ET RECONNAISSANCE DES VISAGES

AVRIL 2022

Université du Québec à Trois-Rivières

Service de la bibliothèque

Avertissement

L'auteur de ce mémoire ou de cette thèse a autorisé l'Université du Québec à Trois-Rivières à diffuser, à des fins non lucratives, une copie de son mémoire ou de sa thèse.

Cette diffusion n'entraîne pas une renonciation de la part de l'auteur à ses droits de propriété intellectuelle, incluant le droit d'auteur, sur ce mémoire ou cette thèse. Notamment, la reproduction ou la publication de la totalité ou d'une partie importante de ce mémoire ou de cette thèse requiert son autorisation.

Résumé

De nos jours, la reconnaissance faciale et le Big Data sont deux facettes qui n'avaient pratiquement rien en commun, mais les progrès récents dans plusieurs domaines et principalement la vision par ordinateur les a conduits à converger pour effectuer des tâches à l'aide de l'apprentissage automatique profond telles que la détection, la reconnaissance et la classification.

Dans ce rapport, nous avons développé un système de reconnaissance faciale basé sur la technique Local Binary Pattern Histogram (LBPH) et la technique de Haar Cascade Classifier pour traiter la reconnaissance en temps réel du visage humain à partir de la caméra de l'ordinateur. Ce système prépare dans un premier temps, les données à partir desquelles on veut extraire de l'information et faire la reconnaissance, il s'agit de photos et de vidéos contenant de l'information sur le visage humain, ensuite il utilise l'algorithme Haar Cascade Classifier pour effectuer la détection de l'objet visage humain. Par la suite, le système procède à l'extraction des caractéristique des points de visage pour encoder la forme de celui-ci en binaire à l'aide de la technique LBP. Finalement la classification des histogrammes générés qui nous permet de voir s'il s'agit de la bonne personne qu'on veut identifier ou non.

Mots-clés : boosting, extraction des caractéristiques, détection de visage, reconnaissance de visage.

Abstract

Nowadays, facial recognition and Big Data are two facets that had practically nothing in common, but recent advances in several fields and mainly computer vision has led them to converge to perform tasks using deep machine learning such as detection, recognition and classification.

In this report, we developed a facial recognition system based on Local Binary Pattern Histogram (LBPH) technique and Haar Cascade Classifier technique to process real-time recognition of human face from computer camera. This system first prepares the data from which we want to extract information and perform recognition, these are photos and videos containing information on the human face, then it uses the algorithm Haar Cascade Classifier to perform human face object detection. Thereafter, the system proceeds to the extraction of the characteristics of the face points to encode the shape of the latter in binary using the LBP technique. Finally the classification of the generated histograms which allows us to see if it is the right person that we want to identify or not.

Keywords : boosting, feature extraction, face detection, face recognition.

Remerciements

Je tiens à remercier, au terme de ce travail, toutes les personnes dont l'intervention, de près ou de loin, a contribué à l'aboutissement de ce mémoire. Tout d'abord, j'adresse mes sincères remerciements à Mr Fathallah Nouboud qui m'a fait l'honneur de m'encadrer et qui n'a épargné aucun effort pour le bon déroulement de ce travail, avec ses remarques et ses consignes qui m'ont été d'un grand apport. Je tiens à lui témoigner ma reconnaissance et ma gratitude pour la qualité de son encadrement, son soutien et ses précieux conseils, pour le temps précieux qu'il m'a consacré, pour son suivi, sa disponibilité et ses conseils qui m'ont été d'une grande aide pour mener à bien ce projet. Je tiens à témoigner toute ma reconnaissance envers lui pour l'expérience enrichissante et pleine d'intérêt qu'il m'a fait vivre durant cette maîtrise. Mes vifs remerciements s'adressent également à ma petite famille, mon mari et mes enfants, qui m'ont comblé de leur soutien et m'ont voué un amour inconditionnel. Vous êtes pour moi un exemple de courage et de sacrifice continu, je vous remercie pour votre compréhension, votre soutien et votre tendresse... Je m'acquitte, enfin, volontiers d'un devoir de gratitude et de remerciements à tous mes enseignants pour la qualité de l'enseignement : François Meunier, Ismaili Biskri, William Flageol, Louis Houde et Usef Faghihi, qu'ils ont bien voulu me prodiguer durant mon cursus à l'université afin de me fournir une formation efficiente et ciblée.

Table des matières

0.1	Introduction générale	11
1	Problématique et contexte général	12
1.1	La vision par ordinateur	13
1.2	le Big Data	15
1.3	La reconnaissance faciale	16
1.4	Le Big Data et la reconnaissance de visage	17
1.5	Architecture d'un système de reconnaissance faciale	18
1.5.1	Préparation des données	19
1.5.2	Détection de visage	20
1.5.3	Extraction des caractéristiques	21
1.5.4	Classification des visages	22
1.5.5	Identification et vérification	23
2	Etat de l'art	24
2.1	Introduction	25
2.2	Approche Eigen Face	25
2.2.1	Introduction	25
2.2.2	Technique ACP	26
2.2.3	Technique SVD	27

2.2.4	Technique SVM	28
2.3	Approche classification avec descripteur par bloc	29
2.3.1	Introduction	29
2.3.2	L'histogramme de LBP	30
2.4	L'approche de Deep Learning avec les réseaux de neurone	30
2.5	Approche par la logique floue	31
2.5.1	Introduction	31
2.5.2	Fuzzification	31
2.5.3	Inférence floue	31
2.5.4	Défuzzification	32
2.5.5	Détection de visage utilisant la logique floue	32
3	Approche pour la reconnaissance des visages en utilisant le Big Data	34
3.1	Préparation des données	35
3.2	Détection de visage	37
3.2.1	Le classificateur Haar Cascade	37
3.2.2	Entraînement du classificateur pour détecter le visage	39
3.2.2.1	Le boosting	39
3.2.2.2	L'algorithme Adaboost	40
3.3	Extraction des caractéristiques	48
3.3.1	L'histogramme LBP	48
3.3.1.1	Définition	48
3.3.1.2	Calcul de l'opérateur LBPH :	49
3.3.1.3	Réalisation de l'histogramme	52
3.4	Classification	54
3.5	Identification	56

4	Implémentation et résultats	57
4.1	Environnement de travail	58
4.2	Implémentation	58
4.2.1	Détection de visage	59
4.2.1.1	Etape 1 : Les images positives et les images négatives .	59
4.2.1.2	Etape 2 : Préparation des images	60
4.2.1.3	Etape 3 : création du fichier Vector des images positives	62
4.2.1.4	Etape 4 : Le Haar-Training	62
4.2.1.5	Etape 5 : création du fameux fichier haarCascadeClassifieur personnalisé	64
4.2.2	Reconnaissance de visage	68
4.3	Expérimentation	70
4.4	Résultats et perspectives	74
4.5	Conclusion générale	77
4.6	Annexe 1 : Détection de visage	78
4.7	Annexe 2 : Reconnaissance et identification	80
4.8	Annexe 3 : Interface graphique	82

Table des figures

1.1	Vision par ordinateur	13
1.2	Big Data et les 3V	15
1.3	Architecture d'un système de reconnaissance faciale	18
1.4	Préparation des données	19
1.5	Détection de visage	20
1.6	Extraction des caractéristiques	21
1.7	Classification des visages	22
1.8	Identification et vérification	23
2.1	La décomposition en valeurs singulières	27
2.2	système de décomposition matricielle avec SVD	28
2.3	Classification SVM	29
3.1	Base de données	36
3.2	Caractéristique de bords	38
3.3	Caractéristique de lignes	38
3.4	Caractéristiques centrées	38
3.5	Arbre de l'algorithme Adaboost	41
3.6	L'algorithme Adaboost	42

3.7	Random forest	43
3.8	Entraînement de classificateurs avec Adaboost	45
3.9	Histogramme LBP :	53
3.10	Processus de classification	54
3.11	Identification du visage	56
4.1	Images positives	60
4.2	Images négatives	60
4.3	Code pour détecter le visage	66
4.4	Détection de visage	67
4.5	Détection de plusieurs visages	67
4.6	Les caractéristiques Haar	68
4.7	Entraînement avec l'histogramme LBP	69
4.8	Extraction des caractéristiques avec LBP	70
4.9	Histogramme LBP et la mesure de similarité	71
4.10	Reconnaissance faciale	72
4.11	Interface graphique	73
4.12	Reconnaissance faciale avec éclairage faible	74
4.13	Reconnaissance faciale avec différentes déviations angulaires	75
4.14	Reconnaissance faciale avec éclairage insuffisant	75
4.15	Précision de la méthode LBPH	76

Liste des tableaux

1.1	Les données et le Big Data	16
3.1	Représentation d'une image en pixels	48
3.2	Calcul de l'opérateur LBP :	50

0.1 Introduction générale

Pour la plupart des gens, une image numérique n'est qu'une image. Mais pour un ordinateur, une image est une tapisserie de pixels et le scientifique a la possibilité de décomposer et d'analyser cette image. L'extraction d'informations à partir d'images fixes et même de vidéos, à l'aide de techniques de traitement d'images, y compris des algorithmes, sera monétisée dans un proche avenir.

De nombreuses recherches antérieures concernant l'identification faciale automatisée ont contourné le problème de savoir quels aspects du stimulus facial sont nécessaires pour la reconnaissance, considérant que des mesures prédéterminées étaient appropriées et efficaces [34]. Cela a suggéré qu'une méthode de théorie de l'information de codage et de décodage des images faciales pourrait fournir un aperçu du contenu informationnel de ces images, en se concentrant sur les caractéristiques locales et globales importantes. Ces caractéristiques pourraient ou non avoir une relation directe avec l'idée intuitive des propriétés du visage comme les cheveux, les yeux, le nez et les lèvres. Dans le concept de la théorie de l'information, le but est d'obtenir des informations utiles dans une image faciale, ensuite de les coder selon la méthode utilisée, puis de comparer le codage du visage à une base de données de modèles qui ont été codés de la même manière [24].

Chapitre 1

Problématique et contexte général

Sommaire

1.1	La vision par ordinateur	13
1.2	le Big Data	15
1.3	La reconnaissance faciale	16
1.4	Le Big Data et la reconnaissance de visage	17
1.5	Architecture d'un système de reconnaissance faciale	18
1.5.1	Préparation des données	19
1.5.2	Détection de visage	20
1.5.3	Extraction des caractéristiques	21
1.5.4	Classification des visages	22
1.5.5	Identification et vérification	23

1.1 La vision par ordinateur

Il existe deux formes de la vision : la vision de la machine et la vision par ordinateur (CV). Alors que la première est largement à usage industriel, les caméras sur un tapis roulant dans une installation industrielle en étant un exemple, la seconde consiste à apprendre aux ordinateurs à extraire et à comprendre des données cachées dans des images numériques et des vidéos.

Ainsi, on peut dire que le but de la vision par ordinateur est de développer des algorithmes qui permettent à l'ordinateur de «voir».



FIGURE 1.1 – Vision par ordinateur

La vision par ordinateur est donc un domaine complet qui traite un haut niveau de programmation en alimentant les images/vidéos d'entrée pour effectuer automatiquement, et à l'aide de l'apprentissage automatique en profondeur, des tâches telles que la détection, la reconnaissance et la classification.

Le rôle principal de la vision par ordinateur est de déterminer si une image contient un certain objet. Les algorithmes analysent automatiquement les images et extraient des informations.

La vision par ordinateur automatisée n'est pas encore suffisamment fiable pour détecter des anomalies ou suivre des objets. Par conséquent, dans de tels cas, les humains sont toujours dans la boucle d'analyse pour évaluer la situation. La vision par ordinateur essaie d'apprendre une fonction vraiment complexe, et on a souvent l'impression que nous n'avons pas assez de données pour la vision par ordinateur et cela même si les ensembles de données sont de plus en plus volumineux. Souvent, nous n'avons simplement pas autant de données que ce dont nous avons besoin [14].

La vision par ordinateur peut améliorer les résultats des moteurs de recherche internet. La technologie peut contextualiser les images dans les recherches. Ce qui signifie qu'il peut étiqueter et identifier les objets, et même les paramètres de vos photos téléchargées.

Voici quelques exemples de ce que la vision par ordinateur permet de faire :

- Les techniques de reconnaissance faciale utilisées par les réseaux sociaux et autres entreprises pour marquer des personnes sur des photographies sont basées sur le CV. Facebook a développé cette technologie pour identifier quelqu'un même si le visage de cette personne était masqué ;
- Identification visuelle biométrique de personnes ;
- Surveillance et alerte de sécurité ; détection d'anomalie ;
- Les applications du CV incluent la réalité augmentée, la biométrie, la reconnaissance faciale, l'analyse gestuelle et la robotique ;
- Suivi d'objets en mouvement ; évitement de collision ; profondeur stéréoscopique ;
- Les développeurs sont sur le point de lancer des moteurs de recherche pour des services de partage d'images tels qu'Instagram qui passeront au crible et donneront un sens aux données du flux de photos téléchargées par des millions d'utilisateurs ;
- Les drones télécommandés utilisent la vision par ordinateur pour reconnaître les objets ;
- Certaines sociétés informatiques comme Microsoft, par exemple, ont une division Cognitive Services. Cette division propose des API pour aider les utilisateurs à extraire des informations riches à partir d'images afin de catégoriser et traiter les données visuelles ;
- Les détaillants au Royaume-Uni et aux États-Unis ont commencé à utiliser Computer Vision pour permettre aux clients « d'essayer » des meubles et du papier peint dans leur propre chambre avant d'acheter.

1.2 le Big Data

Au cours de ces dernières années, plus de données sont produites à des taux plus rapides que jamais et en différents formats, textes, email, photos, vidéos, musique, documents et autres, cela a conduit à l'apparition d'un certain nombre de technologies qui corrigent des aspects spécifiques de ces problèmes d'où le Big Data.

Le Big Data désigne littéralement un ensemble de données massives autrement dit mégadonnées ou encore grosses données. Ces données peuvent être structurées, semi-structurées ou non structurées. Ces données sont collectées et analysées afin d'être utilisées pour extraire des informations qui peuvent servir à différentes applications dans des projets de machine learning.

L'analyse du Big Data se fonde sur les trois dimensions aussi appelées les trois V :

- la vérocité concerne la grande rapidité à laquelle les données sont créées, stockées et analysées ;
- Le volume se rapporte au coté illimité du Big Data avec un nombre de données qui ne cesse de croître ;
- la variété représente les différentes sources de provenance des données.

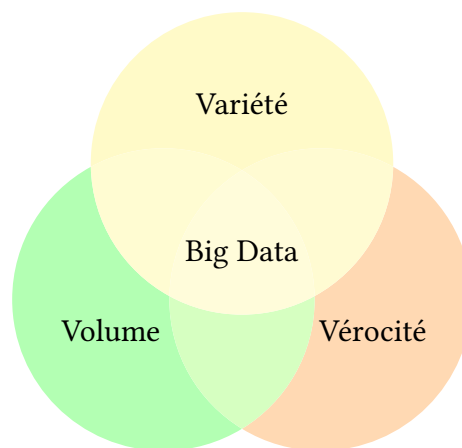


FIGURE 1.2 – Big Data et les 3V

Quand on parle du Big Data on évoque directement les données, nous distinguons alors de deux types de données, les données structurées et les données non structurées.

Le terme Big Data fait référence au volume conséquent de données à leurs types et aux outils permettant leurs traitement.

TABLE 1.1 – Les données et le Big Data

Les données structurées	sont les données qui ont été organisées pour faciliter le traitement et simplifier l'analyse, il peut s'agir d'éléments comme le nom, l'âge qui respectent un format défini, c'est ce types de données qu'on utilise pour remplir un formulaire.
Les données non structurées	sont les données qui ne respectent pas un format défini ou de structure, on parle des photos, de vidéos ou du texte, elles sont plus difficile à analyser, en s'appuyant sur des outils analytiques on peut les faire parler

1.3 La reconnaissance faciale

Le visage de chaque individu est unique, tout comme ses empreintes digitales. La reconnaissance faciale consiste à reconnaître automatiquement une personne grâce à son visage et cela, via une caméra, un ordinateur ou un téléphone intelligent, et grâce à un logiciel avec ses algorithmes d'identification.

Ses applications se sont rapidement multipliées : vidéo de surveillance, biométrie, robotique, espionnage, paiement automatique dans des magasins sans caisse, indexation d'images et de vidéos, recherche par le contenu, criminologie, etc. L'une des premières tentatives de reconnaissance de visage est faite par le japonais Takeo Kanade en 1973 dans sa thèse de doctorat à l'université de Kyoto [?].

Toutes les grandes sociétés travaillent et/ou appliquent déjà la reconnaissance faciale : Facebook, Google, Apple, Microsoft, etc. En Chine, on peut citer Alibaba, Tencent, Baidu, etc. Ce qui ne va toutefois pas sans susciter de vives inquiétudes quant aux atteintes à la vie privée.

En 2018, une start-up chinoise à la pointe de la reconnaissance faciale fondée en 2014 par un ancien du Massachusetts Institute of Technology MIT, s'impose comme une championne de l'intelligence artificielle avec SenseTime. Elle procure sa technologie aux principaux fournisseurs de téléphones intelligents chinois : Huawei, il est probable que

le visage de la personne connectée soit déjà enregistré dans l'immense base de données de SenseTime.

Les techniques de la reconnaissance faciale pourraient s'étendre dans la surveillance sociale malgré les craintes que cela suscitent. Ainsi, on apprenait en mai 2019 que la ville de San Francisco a approuvé une ordonnance interdisant à la police et aux services municipaux d'utiliser la reconnaissance faciale sur ses résidents[1].

1.4 Le Big Data et la reconnaissance de visage

La reconnaissance faciale et le Big Data sont deux facettes qui n'avaient pratiquement rien en commun, mais les progrès récents dans plusieurs domaines connexes tels que les réseaux sociaux ainsi que les progrès dans la nationalisation des bases de données faciales des forces de l'ordre les ont conduits à converger. Les applications à venir telles que le marquage de visage dans les hubs de réseautage social tels que Facebook, en plus d'être un élément crucial dans des applications critiques telles que l'appréhension criminelle par le biais de la correspondance de mug shot, la recherche de personnes disparues, etc.

Le principal avantage de l'analyse Big Data (BDA) est l'augmentation significative de la précision et l'amélioration des performances qu'elle peut offrir en raison de sa capacité à proposer un nombre massif d'images pour la tâche à accomplir. N = toute la taille de l'échantillon, c'est-à-dire toutes les caractéristiques faciales à comparer (contre environ cinq ou six points dans les mécanismes conventionnels).

Les technologies BDA populaires incluent : Apache Hive, Apache Giraph et les œuvres déjà importantes de Horton, Hadoop, etc.

Yahoo a développé Apache Giraph en 2010, sur la base d'un important article de Pregel publié par Google. La technologie Apache Giraph est basée sur les stratégies du calcul distribué et du système de traitement itératif de graphes et repose de manière décisive sur les trois composants critiques du calcul et du traitement parallèles : le calcul simultané, la communication et la synchronisation des barrières. Ces propriétés font d'Apache Giraph un candidat idéal pour aider à proposer des systèmes de reconnaissance faciale efficaces. Facebook continue également d'élargir sa présence dans le domaine de la BDA en optant pour la méthodologie Apache Giraph afin de proposer une nouvelle recherche de graphe social qui peut effectivement évoluer jusqu'à un billion d'arêtes[34].

1.5 Architecture d'un système de reconnaissance faciale

Le processus de détection et d'identification des visages est une technique d'apprentissage automatique, en apprenant et en extrayant les caractéristiques physiques de l'humain. Faire correspondre ces caractéristiques avec les images testées peut identifier la personne ou empêcher ces personnes de se reconnaître.

Cette technologie révolutionnaire d'identification progresse à grand pas, elle s'installe un peu partout, elle est plus fiable que l'oeil humain. Un système de reconnaissance faciale est capable alors de détecter un visage et l'identifier dans une simple image ou une vidéo grâce aux algorithmes de l'apprentissage automatique.

Certains algorithmes se concentrent uniquement sur des images à haute résolution ; certains d'entre eux se concentrent sur les basses résolutions. Récemment, les chercheurs se sont concentrés sur les différentes vues frontales des images, sous différents angles, différents éclairages.

Malgré la diversité de ces algorithmes, ils respectent presque la même architecture et le même workflow des étapes.

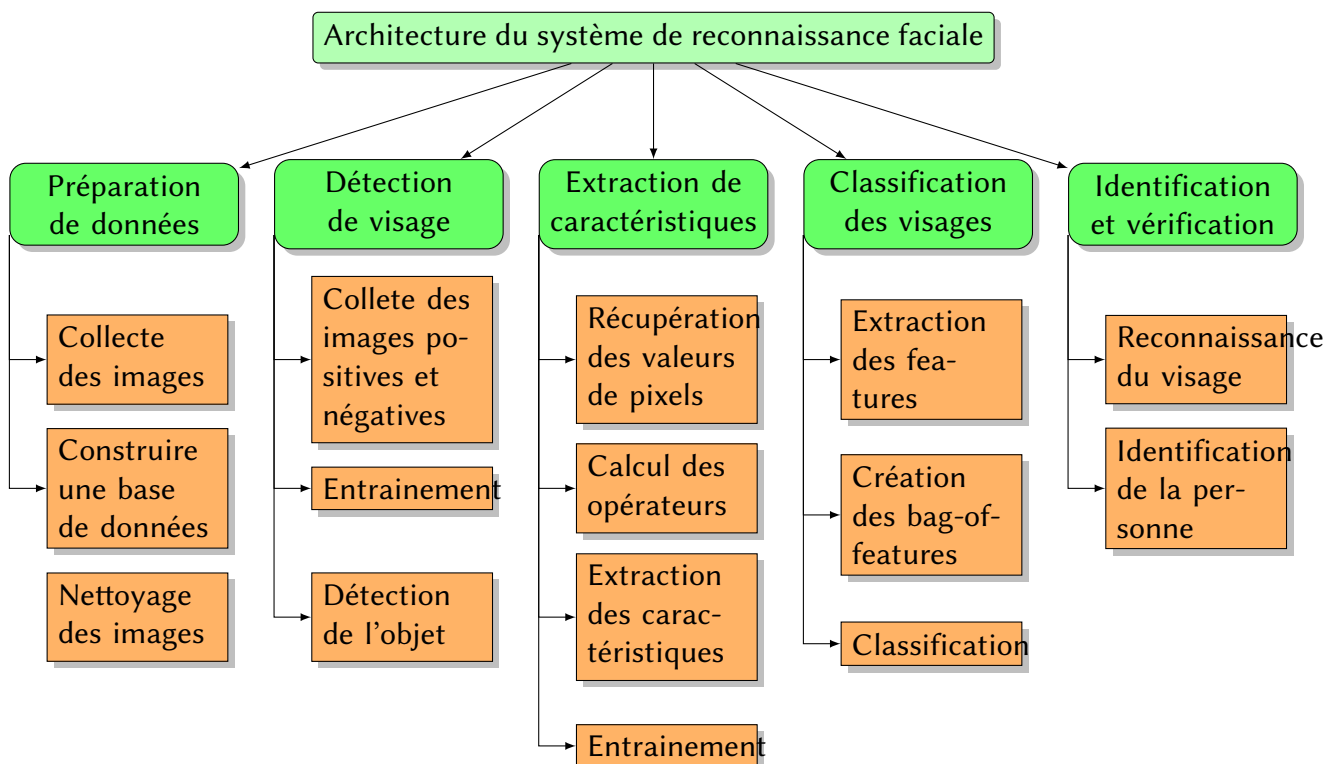


FIGURE 1.3 – Architecture d'un système de reconnaissance faciale

Un système de reconnaissance faciale doit dans un premier temps préparer les données à partir desquelles on veut extraire de l'information et faire la reconnaissance. Ces données sont sous forme de photos et de vidéos contenant de l'information qui est le visage humain. Ensuite il utilise des algorithmes appropriés de l'apprentissage profond pour faire la détection de l'objet visage humain, par la suite le système procède à l'extraction des caractéristiques des points de visage pour encoder la forme du visage en binaire ou numérique, ça peut être le nez, les yeux, la bouche. Finalement la classification du visage qui permet d'identifier et vérifier le visage dans une scène.

1.5.1 Préparation des données

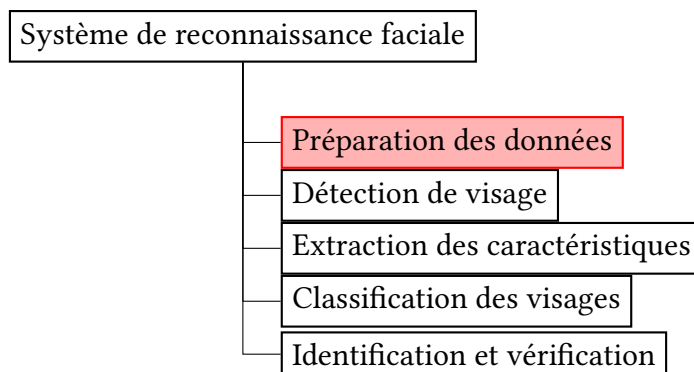


FIGURE 1.4 – Préparation des données

Les données peuvent être des simples images ou des séquences d'images dites vidéos chargées à partir d'une base de données, d'une caméra de surveillance ou autres. Les données doivent être nettoyées et converties en niveau de gris afin d'être utilisées efficacement et sans bruit.

On peut utiliser des bases de données publiques comme :

- Labelled Faces in the Wild (LFW)
- Youtube faces
- Kaggle

Ou encore des bases de données privées, ainsi qu'on peut préparer nos propres bases de données.

1.5.2 Détection de visage

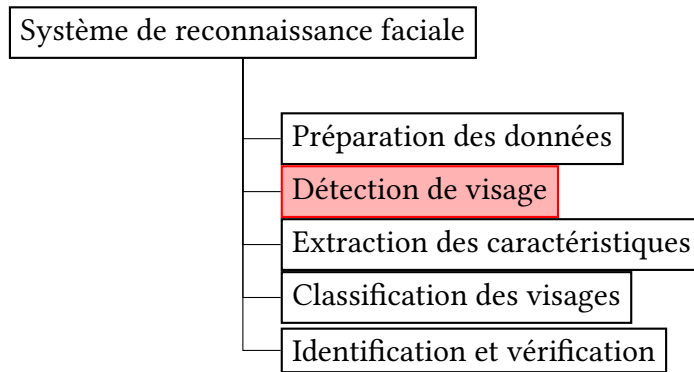


FIGURE 1.5 – Détection de visage

La détection de visage est la pierre angulaire de tous les algorithmes d'analyse faciale, y compris l'alignement du visage, la modélisation du visage, le ré-éclairage du visage, la reconnaissance du visage, la vérification ou l'authentification du visage, le suivi de la pose de la tête, le suivi ou la reconnaissance de l'expression faciale, la reconnaissance du sexe ou de l'âge, et bien d'autres. Ainsi, les ordinateurs peuvent identifier clairement le visage, après ils commencent à vraiment comprendre les pensées et les intentions des gens à l'aide de l'apprentissage automatique [20].

La détection des visages, est une partie importante de la reconnaissance faciale en tant que première étape de la reconnaissance automatique des visages.

La détection de visage peut se faire à l'aide des algorithmes de deep learning. Cette étape se base sur l'entraînement du modèle afin de reconnaître et de détecter l'objet recherché dans les données. Avant de coller un nom à un visage, le logiciel doit comprendre d'abord qu'il est en présence d'un être humain. La détection de visage est la première étape de la reconnaissance faciale, pour cela, il faut éduquer la machine en lui disant qu'il s'agit d'un visage dans des millions d'images contenant le visage et qu'il s'agit d'autre chose que le visage dans des millions d'images ne contenant pas le visage. Le système comprend alors les caractéristiques du visage que ça soit de face ou de profil.

1.5.3 Extraction des caractéristiques

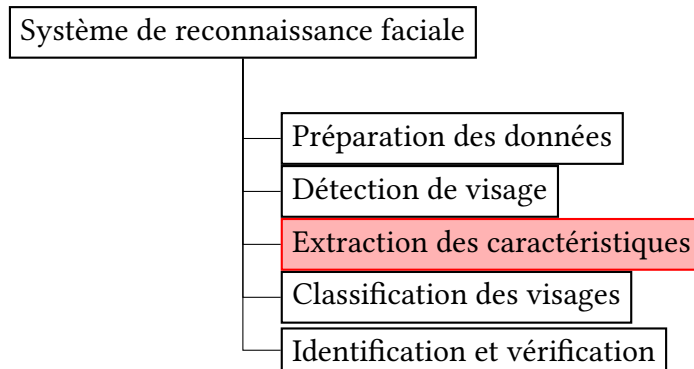


FIGURE 1.6 – Extraction des caractéristiques

Cette étape consiste à isoler les éléments du visage qu'on va mesurer et à détecter les points caractéristiques basiques comme les yeux, le contour de la bouche et le contour du nez pour reconstruire la forme 3D d'un visage afin de différencier les personnes. Pour cela, on calcule la distance entre les éléments du visage et ainsi que tous les éléments qui peuvent illustrer que deux visages ne sont pas identiques.

L'importance des traits du visage pour la reconnaissance faciale ne peut être sur-estimée. De nombreux systèmes de reconnaissance faciale ont besoin de traits faciaux en plus du visage holistique, comme le suggèrent des études en psychologie. Il est bien connu que même les méthodes d'appariement holistiques, par exemple, les visages propres proposés par Turk et Pentland [33] et les Fisherfaces, proposés par Belhumeur et al [27], nécessitent des emplacements précis des principales caractéristiques faciales telles que les yeux, le nez et la bouche pour normaliser le visage détecté. Les traits du visage peuvent être de différents types : région [25], [5], point clé (repère) ou Landmarks [21], [16] et contour [32], [17].

En règle générale, les entités ponctuelles clés fournissent une représentation plus précise et cohérente à des fins d'alignement que les entités basées sur la région, avec une complexité et une charge de calcul inférieures à celles de l'extraction d'entités de contour. Trois types de méthodes d'extraction de caractéristiques peuvent être distingués [10] :

- (1) méthodes génériques basées sur les arêtes, les lignes et les courbes ;
- (2) méthodes basées sur des modèles qui sont utilisées pour détecter les traits du visage tels que les yeux ;

- (3) méthodes d'appariement structurel qui tiennent compte des contraintes géométriques sur les caractéristiques.

1.5.4 Classification des visages

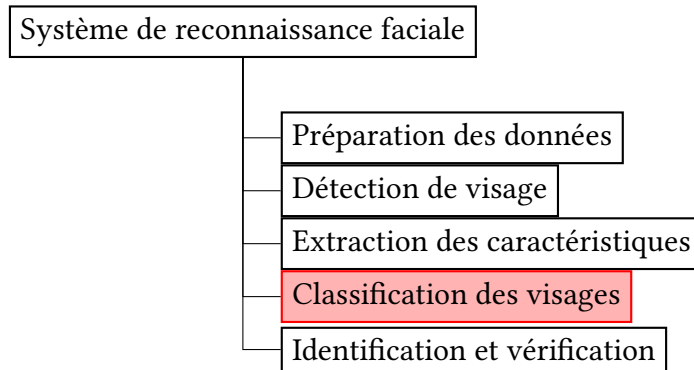


FIGURE 1.7 – Classification des visages

La classification d'images est un problème classique des domaines du traitement d'images, de la vision par ordinateur et de l'apprentissage automatique.

La classification est un arrangement systématique en groupes et catégories en fonction de ses caractéristiques. La classification des images a vu le jour pour réduire l'écart entre la vision par ordinateur et la vision humaine en entraînant l'ordinateur avec les données. La classification d'images est obtenue en différenciant l'image dans la catégorie prescrite en fonction du contenu de la vision.

La classification des images fait référence à un processus de vision par ordinateur qui permet de classer une image en fonction de son contenu visuel. Par exemple, un algorithme de classification d'images peut être conçu pour indiquer si une image contient ou non une figure humaine. Bien que la détection d'un objet soit triviale pour l'homme, la classification robuste des images reste un défi dans les applications de vision par ordinateur.

1.5.5 Identification et vérification

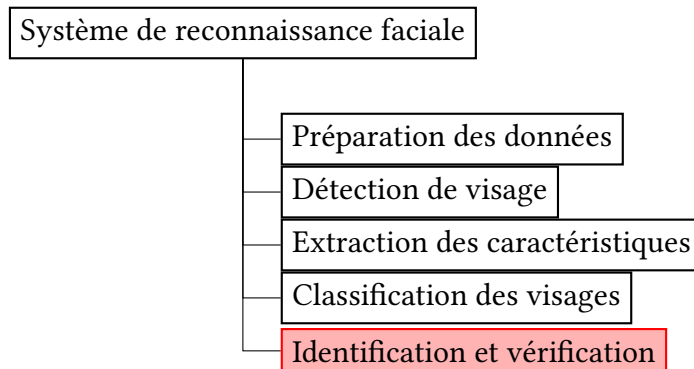


FIGURE 1.8 – Identification et vérification

L'identification signifie que le système est capable de reconnaître l'identité de la personne détectée dans l'image ou la vidéo afin de dire à qui appartient le visage détecté.

Grâce à des techniques d'apprentissage automatique, le système peut choisir le visage qui correspond le mieux aux visages détectés à l'aide du modèle réalisé à partir de la base de données. Cette correspondance peut se réaliser à l'aide des calculs mathématiques comme la distance euclidienne ou la vraisemblance maximale.

Après la vérification de la correspondance du visage avec le modèle entraîné, le système récupère le nom de la personne devant la caméra et l'affiche afin de l'identifier.

Chapitre 2

Etat de l'art

Sommaire

2.1	Introduction	25
2.2	Approche Eigen Face	25
2.2.1	Introduction	25
2.2.2	Technique ACP	26
2.2.3	Technique SVD	27
2.2.4	Technique SVM	28
2.3	Approche classification avec descripteur par bloc	29
2.3.1	Introduction	29
2.3.2	L'histogramme de LBP	30
2.4	L'approche de Deep Learning avec les réseaux de neurone	30
2.5	Approche pa la logique floue	31
2.5.1	Introduction	31
2.5.2	Fuzzification	31
2.5.3	Inférence floue	31
2.5.4	Défuzzification	32
2.5.5	Détection de visage utilisant la logique floue	32

2.1 Introduction

L'objectif de la détection de visages est de détecter et de localiser les visages dans l'image, d'extraire le visage humain à utiliser dans d'autres zones afin de faire la reconnaissance et l'identification de la personne.

De nos jours, il existe de nombreux algorithmes différents pour accomplir la détection ou la reconnaissance des visages, tels que les visages de Fisher, la visages propres, la transformation de fonction invariante à l'échelle et les caractéristiques Speed UpRobust [8].

Plusieurs perspectives existent sur le système de détection et de reconnaissance des visages; certains des projets se concentrent uniquement sur des images à haute résolution; certains d'entre eux se concentrent sur les basses résolutions. Récemment, des chercheurs se sont concentrés sur les différentes vues frontales des images, sous différents angles, différents éclairages, etc.

2.2 Approche Eigen Face

2.2.1 Introduction

Eigenfaces fait référence à une approche de la reconnaissance faciale basée sur l'apparence qui cherche à capturer la variation dans une collection d'images de visage et à utiliser ces informations pour encoder et comparer des images de visages individuels dans une approche holistique. Plus précisément, les Eigenfaces sont les principales composantes d'une distribution de faces, ou de manière équivalente, les vecteurs propres de la matrice de covariance de l'ensemble des images de visage, où une image à N pixels est considérée comme un point (ou vecteur) dans un espace à N dimensions. L'idée d'utiliser des composants principaux pour représenter des visages humains a été développée par Sirovich et Kirby (Sirovich et Kirby 1987) et utilisée par Turk et Pentland (Turk et Pentland 1991) pour la détection et la reconnaissance des visages. L'approche Eigenface est considérée par beaucoup comme la première technologie de reconnaissance faciale fonctionnelle. Elle a servi de base à l'un des meilleurs produits technologiques de reconnaissance faciale du commerce [38].

Cette méthode de reconnaissance utilise l'analyse en composantes principales (ACP) ou la méthode de décomposition en valeurs singulières (SVD) afin de diminuer la dimension de l'espace de travail pour simplifier les données et leur interprétation.

SVD / ACP ont été appliquées à une grande variété de problèmes en science et en ingénierie.

2.2.2 Technique ACP

L'analyse des composantes principales est une technique de réduction de dimensions qui consiste à convertir un ensemble de variables en un ensemble de combinaisons linéaires de plus en plus faibles que la variabilité des données initiales.

Cette méthode permet de transformer un espace de données en un autre espace de dimension inférieure tout en minimisant la perte d'informations.

La réduction de la dimensionnalité des données est un domaine actif en informatique. C'est un problème fondamental dans de nombreux domaines différents, en particulier dans la prévision, la classification de documents, la bioinformatique et la reconnaissance d'objets ou dans la modélisation de processus technologiques complexes. Dans de telles applications, les ensembles de données avec des milliers de fonctionnalités ne sont pas rares. Toutes les fonctionnalités peuvent être importantes pour certains problèmes, mais pour certains concepts cibles, seul un petit sous-ensemble de fonctionnalités est généralement pertinent [15].

L'ACP est une méthode de transformation de l'ensemble de données initiales représentées par des échantillons vectoriels en un nouvel ensemble d'échantillons vectoriels avec des dimensions dérivées. L'idée de base peut être décrite comme suit : un ensemble d'échantillons vectoriels à n dimensions $X = x_1, x_2, x_3, x_4, \dots, x_n$ doit être transformé en un autre ensemble $Y = y_1, y_2, y_3, y_4, \dots, y_n$ de même dimensionnalité, mais les y ont les propriétés que la plupart de leur contenu informationnel est stocké dans les premières dimensions. Ainsi, nous pouvons réduire l'ensemble de données à un plus petit nombre de dimensions avec une faible perte d'informations [15]

$$Y = A.X$$

Basée sur une analyse en composantes principales (ACP), la méthode des Eigen Faces repose sur une utilisation des premiers vecteurs propres comme visages propres, d'où le terme Eigen Faces [4].

Les eigenfaces sont un ensemble de vecteurs propres utilisés dans le domaine de la vision artificielle afin de résoudre le problème de la reconnaissance du visage humain. Le recours à des eigenfaces pour la reconnaissance a été développé par Sirovich

et Kirby (1987) et utilisé par Matthew Turk et Alex Pentland pour la classification de visages. Cette méthode est considérée comme le premier exemple réussi de technologie de reconnaissance faciale. Ces vecteurs propres sont dérivés de la matrice de covariance de la distribution de probabilité de l'espace vectoriel de grande dimension des possibles visages d'êtres humains [30].

2.2.3 Technique SVD

La décomposition en valeurs singulières (SVD) est l'une des factorisations matricielles les plus importantes de l'ère du calcul. Elle forme la base mathématique d'un puissant système de reconnaissance faciale et facilite le calcul des soi-disant faces propres (eigenfaces), qui fournissent une représentation efficace des images dans la reconnaissance faciale.

La décomposition en valeurs singulières d'une matrice A est la factorisation de A en produit de trois matrices $A = UDV^T$, où les colonnes de U et V sont orthonormées et la matrice D est diagonale à entrées réelles et positives.

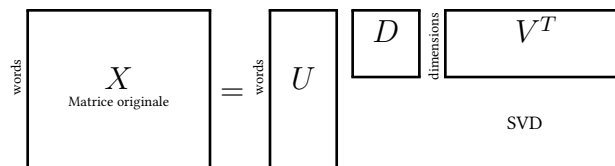


FIGURE 2.1 – La décomposition en valeurs singulières

La SVD est utile dans de nombreuses tâches. Ici, nous mentionnons quelques exemples. Tout d'abord, dans de nombreuses applications, la matrice de données A est proche d'une matrice de faible rang et il est utile de trouver une matrice de faible rang qui soit une bonne approximation de la matrice de données. Nous allons montrer qu'à partir de la décomposition en valeurs singulières de A , on peut obtenir la matrice B de rang k qui se rapproche le mieux de A ; en fait, nous pouvons le faire pour chaque k .

De plus, la décomposition en valeurs singulières est définie pour toutes les matrices (rectangulaires ou carrées) contrairement à la décomposition spectrale plus couramment utilisée en algèbre linéaire. Le lecteur familier avec les vecteurs propres et les

valeurs propres (nous ne supposons pas la familiarité ici) se rendra également compte que nous avons besoin de conditions sur la matrice pour assurer l'orthogonalité des vecteurs propres. En revanche, les colonnes de V dans la décomposition en valeurs singulières, appelées vecteurs singuliers droits de A , forment toujours un ensemble orthogonal sans hypothèses sur A . Les colonnes de U sont appelées les vecteurs singuliers à gauche et forment également un ensemble orthogonal. Une conséquence simple de l'orthogonalité est que pour une matrice A carrée et inversible, l'inverse de A est $VD^{-1}U^T$.

$$A = U\Sigma V^T$$

FIGURE 2.2 – système de décomposition matricielle avec SVD

Dans le meilleur ajustement des moindres carrés, on minimise la distance à un sous-espace. Un autre problème consiste à trouver la fonction qui convient le mieux à certaines données. Ici une variable y est une fonction des variables x_1, x_2, \dots, x_d et on souhaite minimiser la distance verticale, c'est-à-dire la distance dans la direction y , au sous-espace des x_i plutôt que de minimiser la distance perpendiculaire au sous-espace ajusté aux données.

2.2.4 Technique SVM

Machine à vecteurs de support (SVM) est un ensemble d'algorithmes de machine learning permettant de faire une classification pour résoudre des problème comme classifier les images afin de faire la reconnaissance faciale. Cette méthode est connue par sa

grande flexibilité ainsi que sa simple utilisation même pour les débutants en apprentissage automatique.

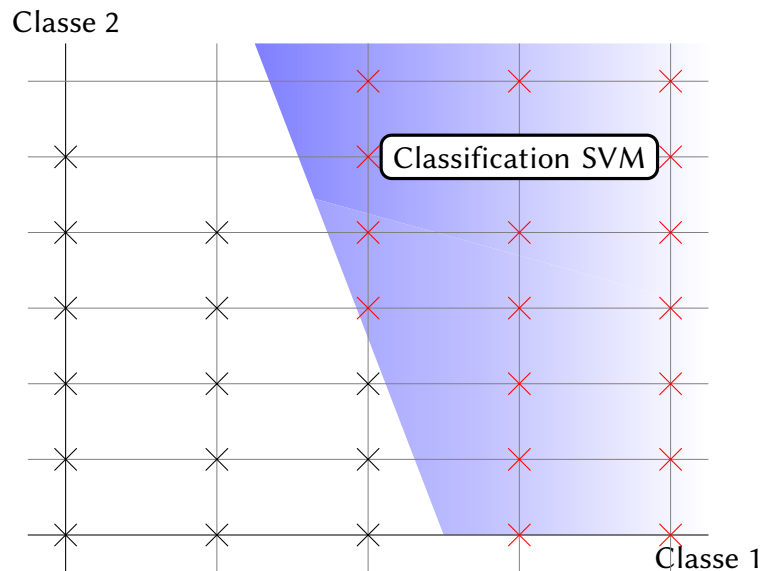


FIGURE 2.3 – Classification SVM

Le but principal du SVM est de former deux groupes de données distincts séparés entre eux à l'aide d'une frontière de telle sorte que la distance entre les groupes de données soit maximale.

2.3 Approche classification avec descripteur par bloc

2.3.1 Introduction

Cette approche divise l'image en des blocs formant une grille. Un descripteur est appliqué pour chaque bloc, et ainsi l'image est constituée en concaténant les différents descripteurs.

Le descripteur le plus répandu est l'histogramme de LBP. Suite à l'application de ce dernier sur toutes les images de la base de données, on classe nos images pour identifier les visages avec un algorithme de classification comme SVM.

2.3.2 L'histogramme de LBP

L'opérateur des motifs binaires locaux (LBP) a été proposé à la fin des années 90 par Ojala [26].

L'idée de cet opérateur de texture est d'assigner à chaque pixel un code dépendant des niveaux de gris de son voisinage. Le niveau de gris du pixel central (i_c) de coordonnées (x_c, y_c) est comparé à celui de ses voisins (i_n) selon l'équation :

$$LBP(x_c, y_c) = \sum_{p-1}^{m=0} s(i_n - i_c)2^n,$$

$$s(i_n - i_c) = 1 \text{ si } i_n i_c \geq 0,$$

$$0 \text{ si } i_n - i_c < 0,$$

avec p est le nombre de pixels voisins. Dans notre travail, nous considérons un voisinage de 3×3 d'où $p=8$ voisins. Nous obtenons donc, comme pour une image en niveaux de gris, une matrice contenant des valeurs des LBP comprises entre 0 et 255 pour chaque pixel. Un histogramme est calculé en se basant sur ces valeurs pour former le descripteur LBP-B [36].

2.4 L'approche de Deep Learning avec les réseaux de neurone

La formation d'un réseau de neurones pour la tâche de détection de visage est un défi car la difficulté à caractériser l'image prototypique «sans visage» [29].

Contrairement à la reconnaissance faciale, dans laquelle les classes à discriminer sont des visages différents, les deux classes à discriminer dans la détection de visage sont « image contenant des visages » et « image ne contenant pas de visages ». Il est facile d'obtenir un échantillon représentatif des images contenant des visages, mais beaucoup plus difficile d'obtenir un échantillon représentatif de celles qui n'en contiennent pas. Les informations sur les fonctionnalités doivent être stockées dans la base de données à des fins de récupération. La récupération d'informations peut être effectuée en utilisant une approche de réseau neuronal qui a le potentiel d'incarner à la fois des données de visage numériques et structurelles [11]. Cependant, les approches de réseau de neurones n'ont été démontrées que dans une base de données limitée. Ainsi, le besoin d'énormes échantillons d'images de visage/non-visage rend le processus d'apprentissage très difficile pour le réseau de neurones.

2.5 Approche pa la logique floue

2.5.1 Introduction

La logique floue est une extension de la logique booléenne créée par Lotfi Zadeh en 1965 en se basant sur sa théorie mathématique des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En introduisant la notion de degré dans la vérification d'une condition, permettant ainsi à une condition d'être dans un autre état que vrai ou faux, la logique floue confère une flexibilité très appréciable aux raisonnements qui l'utilisent, ce qui rend possible la prise en compte des imprécisions et des incertitudes.

En logique floue, un ensemble flou contient plusieurs valeurs. L'ensemble flou est connu par un degré d'appartenance (ou degré de vérité). On utilise un continuum de valeurs logiques entre 0 (complètement faux) et 1 (complètement vrai). Une fonction d'appartenance est utilisée pour mapper un item X dans le domaine des nombres réels à un intervalle de 0 à 1, ce qui permet un degré de vérité [6].

2.5.2 Fuzzification

«Fuzzification» : rendre une entrée classique en valeurs linguistiques.

Des valeurs d'entrée sont traduites en concept linguistique comme des ensembles flous. Les fonctions d'appartenance sont appliquées aux mesures et des degrés de vérité sont établis pour chaque proposition.

La fuzzification des variables est une phase délicate du processus mis en œuvre par la logique floue. Elle est souvent réalisée de manière itérative et requiert de l'expérience.

2.5.3 Inférence floue

Dans un système à la logique floue, toutes les règles sont déclenchées en parallèle. Des règles peuvent être déclenchées partiellement.

Sélection monotonique : Le niveau d'appartenance de vérité d'une conséquence peut être déterminé à partir du niveau d'appartenance de vérité de l'antécédent. Des règles peuvent avoir des antécédents multiples.

Chaque partie est calculée simultanément et résolue à un seul chiffre en utilisant des opérations d'ensembles. Des règles peuvent avoir des conséquences multiples.

Toutes les parties sont affectées également par les antécédents. On doit obtenir un seul chiffre net qui représente la sortie du système expert. On combine les ensembles flous en un seul ensemble flou et la « défuzzification » de l'ensemble résulte en un seul chiffre.

2.5.4 Défuzzification

Le processus de transformer un fait flou en un fait net est la défuzzification. Quelques méthodes existent, comme l'appartenance maximale, la méthode du centroïde, et la méthode des moyennes pondérées [6].

2.5.5 Détection de visage utilisant la logique floue

La détection de visage peut se faire par détection de la couleur de la peau, la forme de la tête ou par des méthodes détectant les différentes caractéristiques du visage. Cette étape est autant plus délicate quand l'image acquise contient plusieurs objets de visage ou un fond non uniforme qui crée une texture perturbant la bonne segmentation du visage. Cette étape est dépendante de la qualité des images acquises. Dans la littérature scientifique, le problème de localisation de visages est aussi désigné par la terminologie «détection de visages». Les performances globales de tout système automatique de reconnaissance dépendent amplement des performances de la détection de visages. Dans l'étape de détection, on identifie et on localise le visage dans l'image acquise au départ, indépendamment de la position, de l'échelle, de l'orientation et de l'éclairage.

C'est un problème de classification où on assigne l'image à la classe visage ou à la classe non visage. On traite une procédure simple de détection du visage, premièrement pour segmenter la région de la peau à partir d'une image, et deuxièmement, pour décider si ces régions contiennent un visage humain ou non.

Prenons un exemple basée sur la segmentation hybride de la couleur de la peau en utilisant trois espaces de couleur RVB, YCbCr et HIS et les caractéristiques du visage humain en utilisant l'entropie. Dans le but d'extraire la caractéristique, plutôt que de regarder l'image entière du visage, et de mettre l'entropie basée sur la sélection de la région de la peau, qui sélectionne des segments très informatifs de l'image du visage, par

rapport à l'entropie de l'image ORL en utilisant la distance euclidienne. Le nombre d'or et la taille de la région de la peau déterminent où se trouve cette région, visage ou pas à travers le système flou. La logique floue a été très bien acceptée par le public. L'approche fournit une méthode appropriée pour l'extraction d'informations. La méthode proposée a été mise à l'essai sur différents types d'images réelles et ses performances sont tout à fait satisfaisantes. Précision de détection 94,74%.

Chapitre 3

Approche pour la reconnaissance des visages en utilisant le Big Data

Sommaire

3.1	Préparation des données	35
3.2	Détection de visage	37
3.2.1	Le classificateur Haar Cascade	37
3.2.2	Entraînement du classificateur pour détecter le visage	39
3.2.2.1	Le boosting	39
3.2.2.2	L'algorithme Adaboost	40
3.3	Extraction des caractéristiques	48
3.3.1	L'histogramme LBP	48
3.3.1.1	Définition	48
3.3.1.2	Calcul de l'opérateur LBPH :	49
3.3.1.3	Réalisation de l'histogramme	52
3.4	Classification	54
3.5	Identification	56

Le processus de détection et d'identification des visages est une technique d'apprentissage automatique, en apprenant et en extrayant les caractéristiques physiques de l'humain. Faire correspondre ces caractéristiques avec les images testées peut identifier la personne ou empêcher ces personnes de se reconnaître. Il existe plusieurs défis et paramètres variables dans la détection et l'identification des visages comme l'éclairage, les différentes poses, les expressions de changement, les images d'entrée de mauvaise qualité, etc.

Il existe plusieurs perspectives différentes sur le système de détection et de reconnaissance des visages; certains des projets se concentrent uniquement sur des images à haute résolution; certains d'entre eux se concentrent sur les résolutions basses. Récemment, des chercheurs se sont concentrés sur les différentes vues frontales des images, sur les différents angles, sur les différents éclairages, etc [12].

L'approche de reconnaissance faciale proposée comporte quatre étapes principales : un module d'acquisition d'images, un module d'extraction de caractéristiques, un module d'apprentissage de la base de données de classificateurs et un module de classification. Initialement, les ensembles de données de visage sont collectés par le module d'acquisition d'images. Ensuite, une série de caractéristiques sont extraites en appliquant le module d'extraction de caractéristiques. Ces caractéristiques faciales sont utilisées pour analyser les points de repère du visage qui représentent des informations sur l'identité humaine. Dans le processus suivant, le classificateur est formé pour reconnaître le visage. Dans le dernier module, le système reconnaît l'image du visage et récupère des informations sur la personne dans la base de données.

3.1 Préparation des données

Comme tout système utilisant l'apprentissage automatique, nous avons besoin d'une base de données qui va nous servir à faire l'entraînement de nos fichiers de classification.

Il existe des bases de données publiques dont la fonctionnalité principale est l'utilisation pour les tests de recherche en reconnaissance faciale. On note :

- la base de données CMU Pose qui a été construite entre octobre et décembre 2000, des chercheurs ont collecté une base de données de 41 368 images de 68 personnes. En étendant la salle 3D CMU, ils ont pu photographier chaque personne sous 13 poses différentes, 43 conditions d'éclairage différentes et avec

4 expressions différentes. On appelle cette base de données la base de données CMU Pose, Illumination, and Expression (PIE)[31].

- La base de données LFW qui contient 13 233 images de visages cibles. Certaines images contiennent plus d'un visage, mais c'est le visage qui contient le pixel central de l'image qui est considéré comme le visage définissant l'image. Les visages autres que le visage cible doivent être ignorés comme "arrière-plan" [22].

On peut aussi construire notre propre base de données, pour cela j'ai collecté plus de 2200 images d'environ 40 personnalités publiques, de ma petite famille et de moi même avec un système qui capture environ 50 échantillons d'images de la personne en face de la caméra de l'ordinateur, ainsi que des images dites négatives qui ne contiennent aucun visage et contiennent toutes sortes d'objets différents.

Une grande base de données peut augmenter la précision et l'efficacité de l'algorithme ainsi qu'elle rend l'entraînement plus robuste et précis.

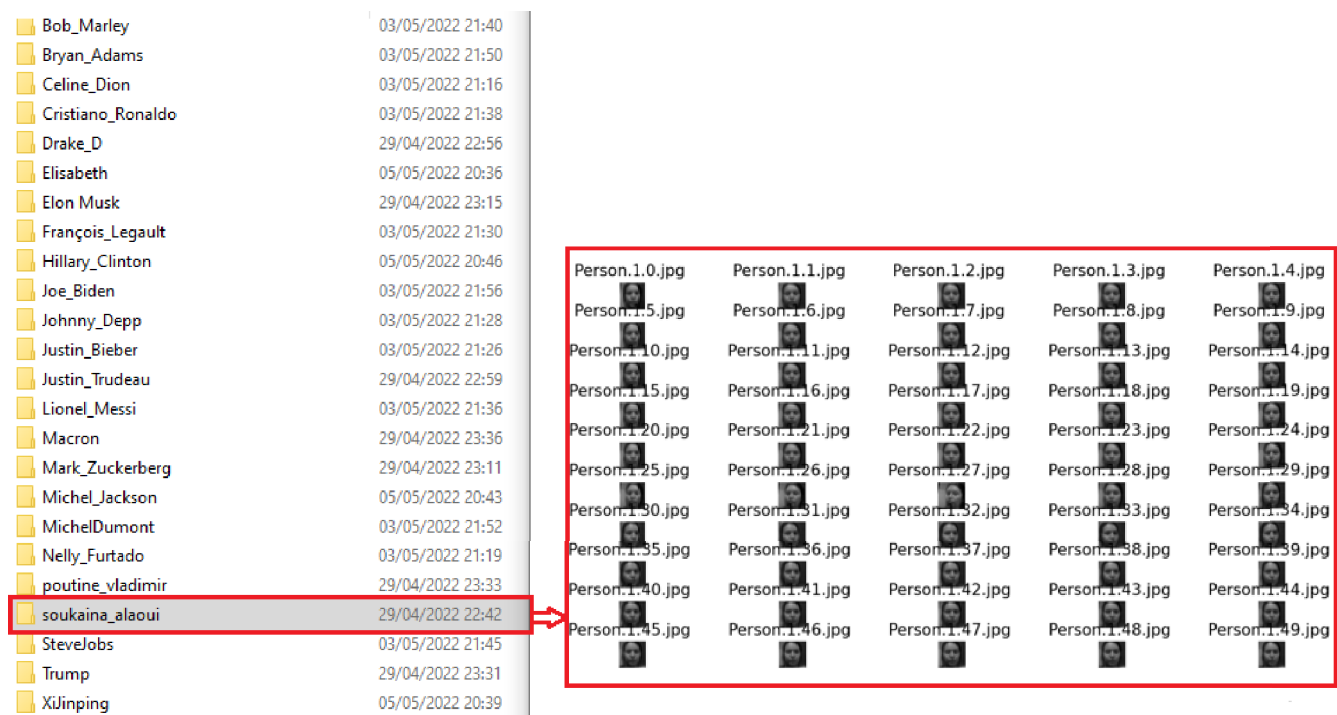


FIGURE 3.1 – Base de données

Ces images doivent être sauvegardées dans des dossiers annotés afin de pouvoir nous aider à identifier chaque visage dans l'image et indiquer à qui il appartient.

Nous avons plusieurs images pour chaque personne que nous voulons reconnaître.

Ces images serviront de données d'entraînement afin que notre reconnaissance faciale LBP puisse apprendre les caractéristiques de chaque individu.

Une image est une projection optique bidimensionnelle, mais le monde dans lequel nous souhaitons donner un sens visuellement est tridimensionnel. A cet égard, la vision est de l'« optique inverse » : il faut inverser la projection $3D \rightarrow 2D$ pour retrouver les propriétés de l'objet dans l'espace de l'image; notons que l'inversion $2D \rightarrow 3D$ d'une telle projection est, strictement impossible mathématiquement.

A ce stade, le jeu de données est pré-traité pour le processus d'extraction des caractéristiques. Les images du jeu de données ont été converties en images en niveaux de gris, puis normalisées ces images pour de bons résultats de reconnaissance. Pour la détection de caractéristiques, des modules Haar ont été utilisés pour détecter ces caractéristiques locales dans une image d'entrée donnée. Ici, l'image d'entrée fait référence à l'image numérique capturée par la caméra.

Après avoir détecté les caractéristiques, le classificateur classera l'image d'entrée en tant qu'image de visage.

3.2 Détection de visage

La détection de visage signifie que le système est capable de comprendre qu'il y a un visage humain dans une image ou vidéo.

Lorsque nous voulons développer un système de reconnaissance de visage, nous devons utiliser d'abord la détection de visage, cela signifie également que nous trouvons ou détectons le visage dans l'image et le système nous dit à qui appartient le visage détecté.

3.2.1 Le classificateur Haar Cascade

Bien qu'il existe de nombreux algorithmes différents pour effectuer la détection des visages, chacun a ses propres faiblesses et forces. Certains utilisent des contours et d'autres sont encore plus complexes impliquant des modèles, des réseaux de neurones ou des filtres. Ces algorithmes souffrent du même problème : ils sont coûteux en terme de temps de calcul.

Viola et Jones [35] ont conçu un algorithme, appelé Haar Classifiers, pour détecter rapidement tout objet, y compris les visages humains, à l'aide de cascades de classificateurs AdaBoost basées sur des caractéristiques de type Haar et non sur des pixels. La base de la détection d'objets du classificateur Haar réside dans les Haar-like features.

Ces caractéristiques, plutôt que d'utiliser les valeurs d'intensité d'un pixel, utilisent le changement des valeurs de contraste entre les groupes rectangulaires adjacents de pixels. Les écarts de contraste entre les groupes de pixels sont utilisés pour déterminer les zones relativement claires et sombres.

Deux ou trois groupes adjacents avec une variance de contraste relative forment une caractéristique de type Haar.

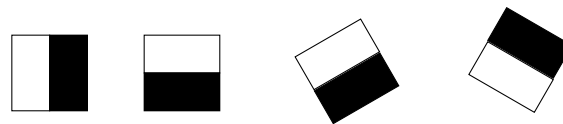


FIGURE 3.2 – Caractéristique de bords

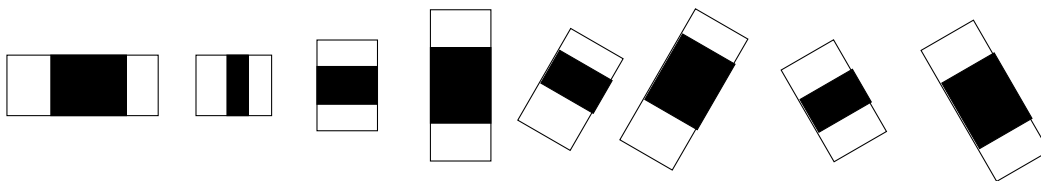


FIGURE 3.3 – Caractéristique de lignes

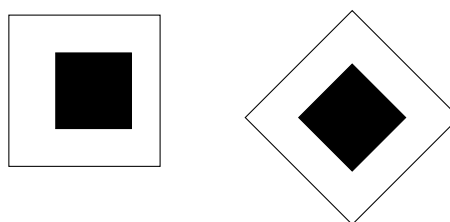


FIGURE 3.4 – Caractéristiques centrées

Les caractéristiques de Haar peuvent facilement être mises à l'échelle en augmentant ou en diminuant la taille du groupe de pixels examiné. Cela permet d'utiliser des fonctionnalités pour détecter des objets de différentes tailles.

Nous implémenterons notre cas d'utilisation en utilisant le classificateur Haar Cascade. Le classificateur Haar Cascade est une approche efficace de détection d'objets qui a été proposée par Paul Viola et Michael Jones dans leur article [35].

Il s'agit essentiellement d'une approche basée sur l'apprentissage automatique dans laquelle une fonction en cascade est entraînée à partir d'un grand nombre d'images positives et négatives. Sur la base de la formation, il est ensuite utilisé pour détecter les objets dans les autres images.

3.2.2 Entraînement du classificateur pour détecter le visage

La détection des caractéristiques faciales humaines, telles que la bouche, les yeux et le nez, nécessite que les cascades du classificateur Haar soient d'abord entraînées. Afin de former les classificateurs, l'algorithme AdaBoost et les algorithmes de fonctionnalité Haar doivent être implémentés.

3.2.2.1 Le boosting

Le boosting est une classe d'algorithmes d'apprentissage machine d'ensemble qui impliquent de combiner les prédictions de nombreux apprenants faibles.

Un apprenant faible est un modèle très simple, bien qu'il ait une certaine compétence sur l'ensemble de données. Le boosting était un concept théorique bien avant qu'un algorithme pratique ne puisse être développé, et l'algorithme AdaBoost (amplification adaptative) a été la première approche réussie de l'idée.

En général, le boosting fournit une interface propre entre le boosting et l'algorithme d'apprentissage faible sous-jacent : à chaque tour, l'algorithme de boosting fournit un ensemble de données pondérées à l'apprenant faible, et l'apprenant faible fournit un prédicteur en retour.

Le seul moyen pour l'algorithme de boosting d'apprendre à partir des données consiste à appeler l'algorithme d'apprentissage de base. Cependant, si l'apprenant de base est simplement appelé à plusieurs reprises, toujours avec le même ensemble de données d'apprentissage, nous ne pouvons pas nous attendre à ce que quelque chose d'intéressant se produise ; au lieu de cela, nous nous attendons à ce que le même classificateur de base, ou presque, soit produit encore et encore, de sorte que ce petit classificateur est gagné sur l'exécution de l'apprenant de base une seule fois. Cela montre que l'algorithme de boosting, s'il veut s'améliorer par rapport à l'apprenant de base, doit en quelque sorte manipuler les données qu'il lui fournit.

Comme le boosting reste indépendant de l'apprenant faible, il est connu sous le nom d'algorithme *meta-learning*.

Le schéma de pondération (comment α est défini dans Algorithm peut être varié pour modifier les algorithmes de boosting globaux. Dans cette section, nous allons implémenter AdaBoost qui choisit α comme :

$$\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t}$$

,

où γ est le soi-disant *edge* de $h_t(x)$, et défini comme :

$$\gamma_t = \sum_{i=1}^N D_t(i) y_i h_t(x_i)$$

,

3.2.2.2 L'algorithme Adaboost

L'algorithme AdaBoost de Freund et Schapire [10] a été le premier algorithme de boosting pratique, et reste l'un des plus largement utilisés et étudiés, avec des applications dans de nombreux domaines.

L'algorithme AdaBoost implique l'utilisation d'arbres de décision très courts (à un niveau) en tant qu'apprenants faibles qui sont ajoutés séquentiellement à l'ensemble.

Expérimentalement, sur des données issues de nombreuses applications du monde réel, AdaBoost s'avère également très efficace. Pour avoir une idée des performances globales d'AdaBoost, nous pouvons la comparer à d'autres méthodes comme Random Forest, qui suit une méthodologie importante dans l'apprentissage automatique, car différents algorithmes peuvent présenter des forces relatives qui varient considérablement d'un ensemble de données à l'autre.

Adaboost se base sur les arbres à une seule profondeur avec l'attribution des poids différents pour chaque noeud, contrairement à Random Forest qu'on représente avec un arbre ayant une profondeur longue.

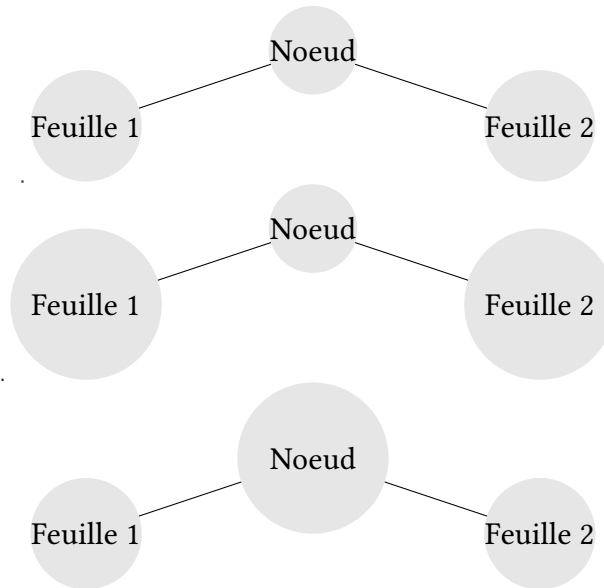


FIGURE 3.5 – Arbre de l’algorithme Adaboost

Chaque modèle suivant tente de corriger les prédictions faites par le modèle avant lui dans la séquence. Ceci est réalisé en pesant l’ensemble de données d’entraînement pour mettre davantage l’accent sur les exemples d’entraînement sur lesquels les modèles antérieurs ont fait des erreurs de prédiction.

Nous sommes maintenant prêts à décrire en détail l’algorithme de boosting AdaBoost, qui intègre ces idées, et dont le pseudocode est représenté par l’algorithme 3.6. AdaBoost procède par ronds ou appels itératifs à l’apprenant de base.

Pour choisir les ensembles de formation fournis à l’apprenant de base à chaque tour, AdaBoost maintient une distribution sur les exemples de formation. La distribution utilisée au t -ième tour est notée D_t , et le poids qu’elle attribue à l’exemple d’apprentissage i est dénoté $D_t(i)$.

Intuitivement, ce poids est une mesure de l’importance de classer correctement l’exemple i sur le tour en cours. Au départ, tous les poids sont fixés de manière égale, mais à chaque tour, les poids des exemples mal classés sont augmentés de sorte que, effectivement, les exemples difficiles obtiennent un poids successivement plus élevé, forçant l’apprenant de base à concentrer son attention sur eux [13].

Ces étapes de l’algorithme peuvent être présentées comme suit :

<p>Entrée : Base de données $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$; Algorithme d'apprentissage de base E; Nombre de cycles d'apprentissage T;</p> <p>Traiter :</p> <ol style="list-style-type: none"> 1. $D_1(x) = 1/m$ # initialiser la répartition du poids 2. pour $t = 1, \dots, T$; 3. $h_t = E(D, D_t)$; # former un classificateur h_t à partir de D en cours de diffusion D_t 4. $\epsilon_t = P_{\mathbf{x} \sim D_t} h_t(\mathbf{x}) \neq f(\mathbf{x})$; évaluer l'erreur de h_t 5. if $\epsilon_t > 0.5$ then break 6. $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$; # déterminer le poids de h_t 7. $D_{t+1}(\mathbf{x}) = \frac{D_t(\mathbf{x})}{Z_t} \mathbf{x} \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\mathbf{x}) = f(\mathbf{x}) \\ \exp(\alpha_t) & \text{if } h_t(\mathbf{x}) \neq f(\mathbf{x}) \end{cases}$ $= \frac{D_t(\mathbf{x}) \exp(-\alpha_t f(\mathbf{x}) h_t(\mathbf{x}))}{Z_t}$ # mettre à jour la distribution, où Z_t est un facteur de normalisation qui permet D_{t+1} d'être une distribution. 8. end <p>Sortie : $H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$</p>
--

FIGURE 3.6 – L'algorithme Adaboost

L'algorithme d'apprentissage consiste à commencer par un arbre de décision, à rechercher les exemples dans l'ensemble de données d'apprentissage qui ont été mal classés et à ajouter plus de poids à ces exemples. Un autre arbre est entraîné sur les mêmes données, bien que désormais pondéré par les erreurs de classification. Ce processus est répété jusqu'à ce qu'un nombre souhaité d'arbres soit ajoutés.

Le boosting a également été initialement proposé comme méthode de comité, bien que contrairement au bagging, le comité des apprenants faibles évolue au fil du temps et les membres votent de manière pondérée. Le boosting semble dominer l'ensachage sur la plupart des problèmes et est devenu le choix préféré.

Contrairement à l'algorithme Adaboost, l'algorithme Random Forest (Breiman, 2001) qui est une modification substantielle du bagging construit une grande collection d'arbres décorrélés, puis les moyenne. Sur de nombreux problèmes, les performances des forêts aléatoires sont très similaires à celles du boosting, et elles sont plus simples à entraîner et à régler. Par conséquent, les forêts aléatoires sont populaires et sont implémentées dans une variété de packages.

Les forêts aléatoires sont une combinaison de prédicteurs d'arbres tels que chaque arbre dépend des valeurs d'un vecteur aléatoire échantillonné indépendamment et avec la même distribution pour tous les arbres de la forêt contrairement à Adaboost qui affecte des poids différents aux noeuds. L'erreur de généralisation pour les forêts converge petit à petit à une limite à mesure que le nombre d'arbres dans la forêt devient grand. L'erreur de généralisation d'une forêt de classificateurs d'arbres dépend de la force des arbres individuels dans la forêt et de la corrélation entre ceux-ci.

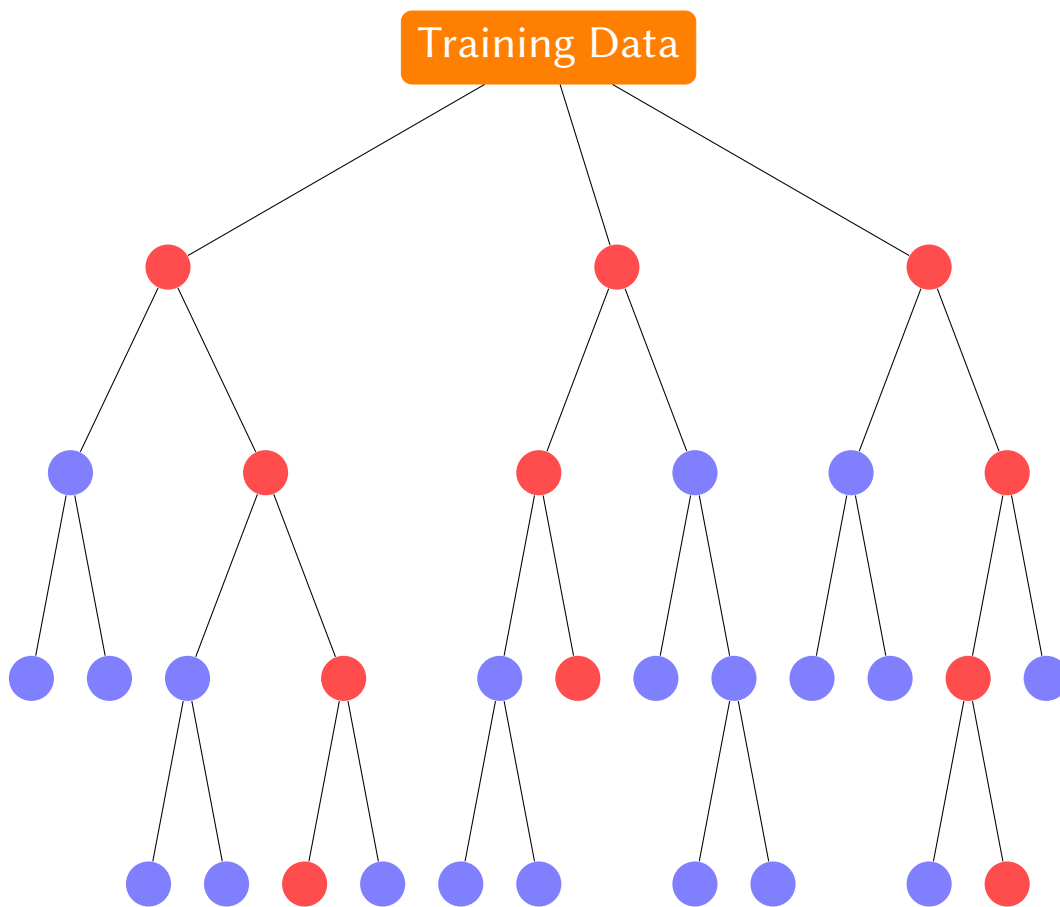


FIGURE 3.7 – Random forest

Donc on pourrait dire que AdaBoost est un algorithme indéniablement performant et Random Forest est au moins aussi bon, sinon meilleur. Mais AdaBoost est aussi déroutant que réussi ; il a enfreint les règles de base des statistiques en ajustant de manière itérative même des ensembles de données bruyants jusqu'à ce que chaque point de données de l'ensemble d'apprentissage soit ajusté sans erreur. Encore plus déroutant, du

moins pour les statisticiens, il continuera à itérer un algorithme déjà parfaitement ajusté qui réduit l'erreur de généralisation.

La vision statistique du boosting comprend AdaBoost comme une optimisation par étapes d'une perte exponentielle, qui suggère (exige !) la régularisation de la taille de l'arbre et le contrôle du nombre d'itérations ainsi que le poids. En revanche, un Random Forest n'est pas une optimisation ; il semble fonctionner mieux avec de grands arbres et autant d'itérations que possible.

Il est largement admis qu'AdaBoost est efficace parce qu'il s'agit d'une optimisation, tandis que les forêts aléatoires ou Random Forest fonctionnent bien parce qu'elles fonctionnent. Les auteurs de cet article [3] mettent en lumière cette conjecture en fournissant une nouvelle intuition étayée par des exemples pour montrer comment AdaBoost et la forêt aléatoire réussissent pour la même raison.

L'entraînement des classificateurs Adaboost se fait comme suit : chaque classificateur collecte un nouvel ensemble d'échantillons négatifs pour chaque étape, et le nombre d'échantillons négatifs fixe la limite pour la taille de l'ensemble. Il utilise les informations des étapes précédentes pour déterminer lequel des «échantillons candidats» sont mal classés.

Intuitivement, pour qu'un classificateur apprenant soit efficace et précis dans ses prédictions, il doit remplir trois conditions :

- (1) il doit avoir été formé sur « suffisamment » d'exemples de formation ;
- (2) il devrait fournir un bon ajustement à ces exemples de formation (ce qui signifie généralement qu'il devrait avoir une faible erreur de formation) ;
- (3) il devrait être « simple ».

La figure 3.8 présente en illustrateur de l'entraînement des classificateurs Ada-boost, le premier classificateur entraîné regarde dans l'image s'il s'agit d'un visage ou non, s'il peut le détecter on passe au classificateur suivant, sinon on le rejette et on retourne à la fenêtre de décision pour former un nouveau classificateur et tester ces capacités de détection et ainsi de suite jusqu'à la formation de tous les classificateurs qui peuvent détecter le visage.

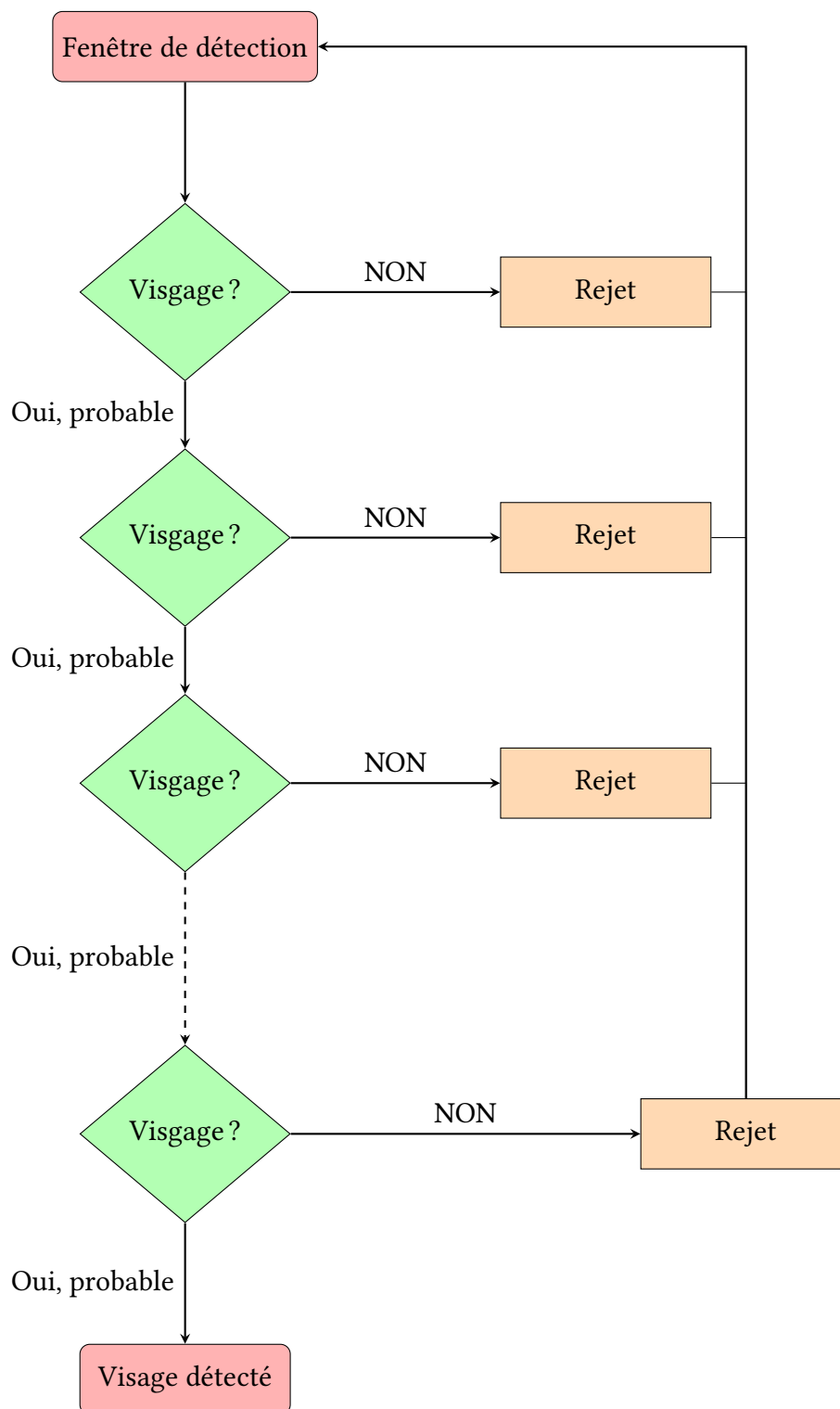


FIGURE 3.8 – Entrainement de classificateurs avec Adaboost

On peut expliquer le processus de cet entrainement des classificateurs avec l’algorithme Adaboost comme suit :

$$D_1(i) = \frac{1}{n} \quad (3.1)$$

$$\text{for } t = 1 \text{ to } T \quad (3.2)$$

$$\text{train } h_t \text{ using } D_t \quad (3.3)$$

$$e_t = P_{D_t}[h_t(x) \neq y] \quad (3.4)$$

$$\alpha_t = 1/2 \log \frac{1 - e_t}{e_t} \quad (3.5)$$

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \quad (3.6)$$

$$(3.7)$$

Le classifieur final est :

$$f(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right) \quad (3.8)$$

$$D_T(i) = 1/n \frac{\exp(-\sum_t \alpha_t y_i h_t(x_i))}{\prod_{t=1}^{T-1} Z_t} \quad (3.9)$$

$$\prod_t Z_t = 1/n \sum_i \exp(-y_i f(x_i)) \quad (3.10)$$

$$1/n \sum_i [f(x_i) \neq y_i] \leq 1/n \sum_i \exp(-y_i f(x_i)) = \prod_t Z_t \quad (3.11)$$

$$\leq \prod_t \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \quad (3.12)$$

$$\leq 1/n \sum_i \exp(-y_i \sum_t \alpha_t h_t(x_i)) \quad (3.13)$$

$$\prod_t Z_t = \prod_t 2\sqrt{e_t(1 - e_t)} = \prod_t \sqrt{1 - 4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2} \quad (3.14)$$

$$\gamma_t = 1/2 - e_t, \quad (3.15)$$

si $\gamma_t \geq \gamma > 0$, donc $e^{-2T\gamma^2}$. L'erreur d'entraînement chute de façon exponentielle, ainsi on obtient le meilleur classificateur possible.

Souches de décision : Chaque prédicteur correspondra à une souche de décision, celle-ci est simplement juste une paire caractéristique-seuil. Notons que pour chaque caractéristique f_i , nous pouvons avoir plusieurs seuils possibles que nous noterons τ . Étant donné une instance d'entrée à classer, une souche de décision correspondant à la caractéristique f_i et au seuil τ prédira +1 si l'instance d'entrée a une valeur de caractéristique

f_i supérieure au seuil τ ; sinon, il prédit -1 .

Détermination du seuil approprié :

Pour créer les différents seuils pour chaque caractéristique f_i , nous devons :

1. trier les exemples de formation par leurs valeurs f_i ;
2. supprimer les valeurs en double ;
3. construire des seuils à mi-chemin entre des valeurs de caractéristiques successives.

Nous devons également ajouter deux seuils pour chaque fonctionnalité : un en dessous de toutes les valeurs observées pour cette fonctionnalité et un au-dessus de toutes les valeurs pour cette fonctionnalité. En supprimant les valeurs en double, nous réduisons le nombre de seuils à vérifier.

(*Remarque* : La procédure de recherche est la même que celle utilisée pour sélectionner un seuil pour définir une division binaire sur une caractéristique continue dans les arbres de décision, sauf que nous maximisons l'erreur d'apprentissage au lieu du gain d'information ou de l'indice de Gini.)

Souche de décision optimale : Le seuil optimal τ^* pour la caractéristique f_i correspond au seuil qui maximise :

$$\tau^* = \operatorname{argmax}_{\tau} \left| \frac{1}{2} - \operatorname{error}(\tau) \right|$$

,

où $\operatorname{error}(\cdot)$ est calculé comme la somme des poids ($D_t(i)$) des instances mal classées. Dans ce cas, puisque nous avons affaire à des valeurs absolues, si un seuil a une erreur de 0,2 et qu'un autre seuil produit une erreur de 0,9, nous préférons ce dernier car il a une différence plus élevée par rapport à 0,5 (pour de telles souches de décision, nous devons inverser la prédiction de sorte que pour les instances dont la valeur de caractéristique est supérieure à τ , nous prédisons -1 et $+1$ sinon. Nous devons nous en souvenir lorsque nous combinons de telles souches de décision avec d'autres pour évaluer $H(x)$).

3.3 Extraction des caractéristiques

3.3.1 L’histogramme LBP

3.3.1.1 Définition

L’histogramme à motifs binaires locaux est une technique très utilisée dans le domaine de la reconnaissance faciale. Cette technique utilise un bloc de 3*3 pixels extrait à partir d’une image, et elle est particulièrement intéressée par le pixel central pour pouvoir calculer l’opérateur LBP.

Les images sont représentées sous la forme de tableaux rectangulaires de nombres représentant des intensités d’image à des emplacements particuliers. Chaque élément d’un tel tableau est appelé un pixel. Une image couleur peut être représentée sous forme de trois matrices distinctes appelées «plans de couleur», contenant des composants rouge, vert et bleu sous forme d’images monochromatiques. Une image avec un bord oblique pourrait ressembler à :

0	0	0	1	1	0	0	1	1	0
0	0	1	12	10	17	15	18	1	0
0	2	5	15	14	18	12	20	25	0
0	1	17	20	15	20	14	16	56	60
0	8	22	18	20	35	28	40	43	46
0	10	25	27	32	38	40	41	39	46
0	12	20	30	40	38	30	38	46	40
0	5	15	36	38	40	30	89	77	76
5	12	11	10	42	55	45	56	60	72
20	15	13	20	18	60	63	90	98	74

TABLE 3.1 – Représentation d’une image en pixels

L’opérateur LBP se repose sur les valeurs du 9 pixels qui constituent ce bloc, à savoir, le pixel central et les 8 pixels voisins ou encore les 26 voisins si on a une grille XY.

Cette technique d’extraction de caractéristiques nous aide par la suite à faire la classification qui va nous servir à réaliser un modèle d’apprentissage en profondeur en anglais le deep learning.

Dans cette section, l’algorithme de détection de visage basé sur LBPH est présenté. L’algorithme LBPH est la combinaison des descripteurs Local Binary Patterns (LBP) et

Histogramme des Gradient Orientés (HOG). LBP est un moyen simple mais puissant d'extraire et d'étiqueter les pixels d'une image. En utilisant le LBPH, nous pouvons facilement représenter des images de visage avec un simple vecteur.

3.3.1.2 Calcul de l'opérateur LBPH :

Dans cette section, nous allons voir comment on peut se servir des valeurs de pixels pour calculer l'opérateur LBP, ou encore, comment l'algorithme calcule les valeurs de chaque pixel de l'image pour créer notre modèle à l'aide de l'histogramme à motifs binaires locaux.

L'opérateur LBP se définit comme suit :

$$LBP(x_c, y_c) = \sum_{p=1}^{m=0} s(i_n - i_c)2^n$$

$$s(i_n - i_c) = 1 \text{ si } i_n - i_c \geq 0$$

$$0 \text{ si } i_n - i_c < 0.$$

Pour un bloc de 9 pixels 3*3, on peut le définir comme suit :

$$LBP = \sum_n^7 s(i_n - i_c)2^n$$

$$s(i_n - i_c) = 1 \text{ si } i_n - i_c \geq 0$$

$$0 \text{ si } i_n - i_c < 0$$

avec i_c la valeur du pixel central, et i_n la valeur des pixels voisins qui sont les pixels autour du pixel central sachant que n allant de 0 jusqu'à 7. Si la différence des valeurs i_c et i_n est négative on pose $s(i_n - i_c) = 1$ sinon on aura $s(i_n - i_c) = 0$.

Ces valeurs de pixels peuvent être les valeurs de l'intensité ou de luminosité, c'est ce type de caractéristiques de pixels que nous examinons lorsque nous procédons à faire le calcul de l'opérateur LBP afin de pouvoir maximiser la variation ou les fonctionnalités qui sont pertinentes pour l'identité, nous ne regardons pas la couleur dans ce cas. La plupart des gens examinent l'image en niveau de gris, mais cela n'empêche pas de faire cette procédure en couleur, mais dans ce cas nous allons avoir trois dimensions, donc nous allons nous intéresser à 27 pixels et non seulement à 9.

Dans la matrice de visage, les caractéristiques extraites de l'image servent à comparer les valeurs des pixels voisins avec les valeurs des pixels centraux pour enfin générer un code binaire.

Pour détecter les visages dans une photo RBG (couleur), nous devons d'abord convertir l'image en une image en niveaux de gris. Pour chaque pixel $P_{i,j}$, c'est un vecteur qui contient trois valeurs qui doivent représenter le degré de rouge, de bleu et de vert. Nous convertissons l'image RBG en image à l'échelle y en :

$$G_{i,j} = (0.2989, 0.5870, 0.1140)^T \cdot P_{i,j},$$

où $G_{i,j}$ représente le pixel correspondant dans l'image. L'opérateur LBP va comparer la valeur centrale avec ses P valeurs voisines sur le cercle de rayon R et attribue à la valeur voisine 1, si la valeur centrale est plus grande que la voisine, il attribue la valeur 0 sinon.

Dans ce qui suit nous allons faire la démonstration de l'opérateur LBP pour un pixel, notons que les valeurs des pixels sont ordonnées comme suit :

i_0	i_1	i_2
i_7	i_c	i_3
i_6	i_5	i_4

Considérons un exemple simple d'un bloc de 3*3 pixels extrait à partir d'une image et calculons l'opérateur LBP qui va transformer ces valeurs en des valeurs binaires formant une seule valeur du pixel central.

5	9	1
4	4	6
7	2	3

TABLE 3.2 – Calcul de l'opérateur LBP :

Pour n=0 :	$LBP = s(i_0 - i_c)2^0 = s(5 - 4)2^0 = 1$
Pour n=1 :	$LBP = s(i_1 - i_c)2^1 = s(9 - 4)2^1 = 1$
Pour n=2 :	$LBP = s(i_2 - i_c)2^2 = s(1 - 4)2^2 = 0$
Pour n=4 :	$LBP = s(i_4 - i_c)2^4 = s(3 - 4)2^4 = 0$
Pour n=5 :	$LBP = s(i_5 - i_c)2^5 = s(2 - 4)2^5 = 0$
Pour n=6 :	$LBP = s(i_6 - i_c)2^6 = s(7 - 4)2^6 = 1$
Pour n=7 :	$LBP = s(i_7 - i_c)2^7 = s(4 - 4)2^7 = 1$

On peut présenter les valeurs binaires calculées comme suit :

1	1	0
1		1
1	0	0

On peut aussi obtenir ces valeurs binaires en comparant la valeur de chaque pixel voisin avec celle du pixel central, si la valeur du pixel voisin est supérieure à la valeur du pixel central alors on met 1 sinon on met 0, cette méthode est aussi appelée le seuillage du pixel central par rapport aux autres pixels voisins.

Ces valeurs binaires signifient essentiellement que si nous avons une transition de 0 à 1 ou de 1 à 0, alors il s'agit des bords ou du contour qui vont nous permettre par la suite de définir et détecter l'objet recherché, ce sont les transitions des zones claires du visages à des zones sombres du visage. Nous aurons donc un descripteur invariant d'illumination des bords.

L'opérateur LBP réduit l'influence de l'éclairage et renvoie la texture en considérant chaque pixel d'une image qui exclut les pixels limites.

Ensuite, nous récupérons les valeurs binaires ou dites bits calculées successivement de i_0 jusqu'à i_7 , c'est-à-dire, qu'ont fait une rotation dans le sens de l'aiguille d'une montre, pour les transformer en un octet afin de former le nombre LBP binaire comme suit :

1	1	0	1	0	0	1	1
---	---	---	---	---	---	---	---

Cette séquence de 8 bits que nous avons obtenue forme un nombre binaire généré à partir des valeurs de pixels va être convertie en nombre décimal pour obtenir la valeur du pixel central afin de former notre système :

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

On obtient alors :

$$1*2^0 + 1*2^1 + 0*2^2 + 0*2^3 + 1*2^4 + 0*2^5 + 1*2^6 + 1*2^7 = 1 + 2 + 16 + 64 + 128 = 211$$

Donc 211 est le code LBP généré.

En utilisant le résultat LBP, nous pouvons générer un histogramme pour cette image et former un vecteur de données pour décrire les motifs de l'image d'origine.

3.3.1.3 Réalisation de l'histogramme

Ce calcul doit être fait pour chaque pixel de l'image en implémentant les mêmes instructions afin d'obtenir les codes LBP correspondant et en le remplaçant dans le pixel de l'image concerné. Nous transformons ces valeurs obtenues en un histogramme en fonction du nombre d'apparitions ou de fréquence de chaque nombre, donc fondamentalement en regardant les statistiques, nous obtenons les caractéristiques de la distribution de ces nombres décimaux, notons qu'il y a que 256 valeurs différentes.

L'histogramme représente la fréquence des occurrences du résultat LBP pour chaque pixel. A partir de la dernière partie, après avoir fait l'opérateur LBP, la valeur de chaque pixel est comprise entre 0 et 255. Ainsi, l'histogramme ne contient que 256 positions.

Cependant, nous ne considérons pas directement l'image entière. Nous divisons l'image en plusieurs morceaux d'image. nous calculons les valeurs de 256 positions pour chaque élément d'image étant donné N_x et N_y , représentant le nombre de cellules dans la direction horizontale et verticale respectivement.

Ensuite, nous concaténons tous les histogrammes de tous les morceaux d'image pour former un histogramme plus significatif.

Par la suite, nous utilisons ce qu'on appelle des modèles binaires locaux uniformes et ils n'ont que 59 valeurs différentes possibles au lieu de 256, nous obtenons donc des statistiques vraiment robustes.

Et ensuite nous réalisons des histogrammes que nous obtenons pour chaque image afin de trouver la similarité entre elles, et sur la base de cela nous pouvons prédire qu'il s'agit de la même personne, si la similarité est élevée avec une image d'entraînement connue.

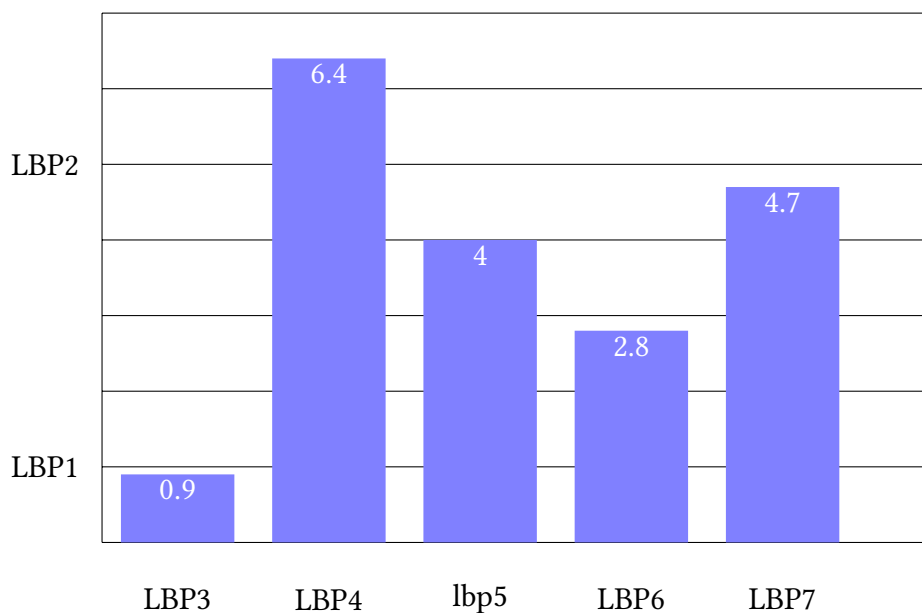


FIGURE 3.9 – Histogramme LBP :

L'avantage de cette technique c'est qu'elle est invariante par rapport à l'éclairage, autrement dit, si nous changeons l'éclairage de l'image ou de la scène, toutes les valeurs de pixels augmenteront et la différence relative entre elles restera la même, donc notre modèle binaire restera le même indépendamment de la valeur d'éclairage en général.

3.4 Classification

Cette étape consiste à faire la classification qui est effectuée en calculant les similarités entre des histogrammes directs. Cependant, une perte de données spatiales conduit à considérer la même méthodologie pour l'illustration de l'image faciale [9].

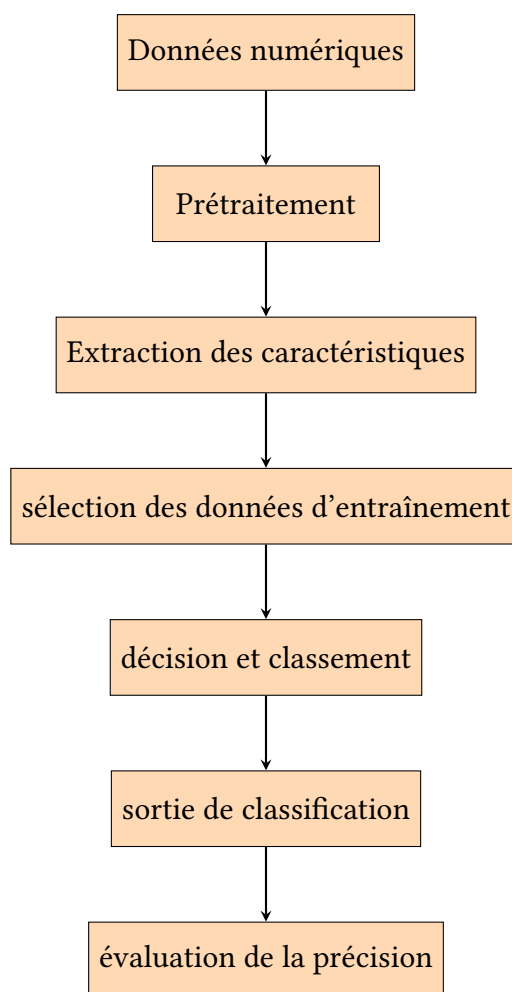


FIGURE 3.10 – Processus de classification

Le système exécute un processus de reconnaissance faciale. D'abord, il détecte tous les visages dans l'image, puis il extrait les caractéristiques faciales de l'image de visage de test d'entrée. Une fois ce vecteur de caractéristiques d'entrée est réalisé, il est comparé au modèle de jeu de données d'image formé à l'aide du classificateur Haar Cascade.

Si le vecteur de caractéristiques de test d'entrée correspond au modèle formé, il reconnaît le visage et récupère les informations sur cette image à partir de la base de données. Le système peut détecter plusieurs visages dans l'image si une nouvelle image

de visage entre dans le système, il reconnaît le visage à partir des informations de la base de données [2].

Le classificateur formé vise à faire correspondre l'image avec les images du modèle formé qui est calculé par l'une des méthodes suivantes :

L'intersection d'histogrammes :

$$D(S, M) = \sum_i (\min(S_i, M_i))$$

;

Statistique de log-vraisemblance :

$$L(S, M) = - \sum_i S_i \log M_i$$

;

Statistique du Chi-deux (X_2) :

$$X^2(S, M) = \sum_i \frac{(S_i - M_i)^2}{S_i + M_i}$$

;

avec S et M deux histogrammes LBP.

La recherche des deux histogramme qui se ressemblent le mieux peut se faire aussi à l'aide du calcul de la distance euclidienne comme suit :

$$d(p, q) = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

.

Compte tenu de la représentation caractéristique d'un échantillon de visage, l'étape de classification vise à calculer un score pour l'échantillon et, selon un seuil de décision, accepter ou rejeter l'échantillon. Les méthodes de mesure de similarité, telles que la corrélation normalisée (NC) [37], sont les classificateurs les plus simples et les plus populaires. Des modèles statistiques plus complexes tels que les réseaux de neurones (NN) [23] ou les machines à vecteurs de support (SVM) [18] sont également utilisés. NC, NN ou SVM sont des approches discriminantes.

3.5 Identification

Après avoir fait la correspondance entre les histogrammes afin de classifier les visages et les reconnaître, on procède à l'identification qui est l'étape finale du processus de reconnaissance faciale.

Cette partie consiste à faire la reconnaissance de visage détecté auparavant.

On parcourt tout le répertoire des images qui est sous forme d'une arborescence dans la base de données des visages. Ensuite on construit deux tableaux, le premier contient l'ensemble de photos et le deuxième contient l'identité de chaque personne qui est le nom du répertoire. S'il y a des images dans le répertoire, on va récupérer le nom du dossier contenant l'image et on le stocke dans le premier tableau qui identifie les visages et on enregistre toutes les images correspondantes dans le deuxième tableau des images.

On répète ce processus d'identification pour tous les dossiers des images de notre base de données et, à l'aide des classificateur entraînés, on pourra facilement faire l'identification du visage en faisant la correspondance entre les données stockés et le visage détecté.

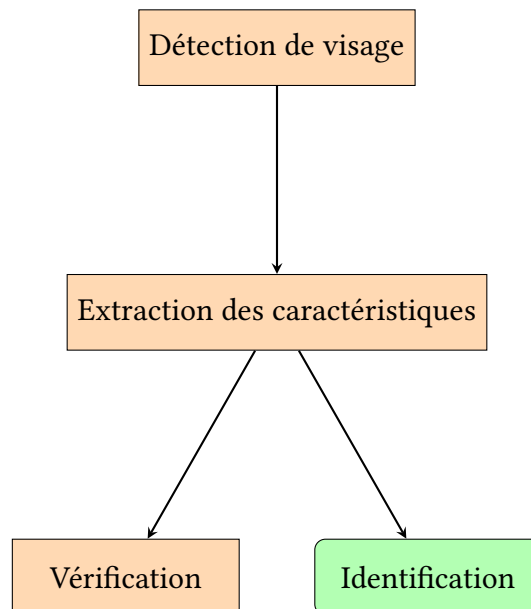


FIGURE 3.11 – Identification du visage

Chapitre 4

Implémentation et résultats

Sommaire

4.1	Environnement de travail	58
4.2	Implémentation	58
4.2.1	Détection de visage	59
4.2.1.1	Etape 1 : Les images positives et les images négatives	59
4.2.1.2	Etape 2 : Préparation des images	60
4.2.1.3	Etape 3 : création du fichier Vector des images positives	62
4.2.1.4	Etape 4 : Le Haar-Training	62
4.2.1.5	Etape 5 : création du fameux fichier haarCascade-Classifier personnalisé	64
4.2.2	Reconnaissance de visage	68
4.3	Expérimentation	70
4.4	Résultats et perspectives	74
4.5	Conclusion générale	77
4.6	Annexe 1 : Détection de visage	78
4.7	Annexe 2 : Reconnaissance et identification	80
4.8	Annexe 3 : Interface graphique	82

4.1 Environnement de travail

L'application de la reconnaissance faciale avec l'approche proposée a été implémentée en Python, à l'aide de la bibliothèque de traitement d'image Opencv, le classificateur Haar ainsi que l'algorithme LBPH en utilisant la caméra de l'ordinateur.

Heureusement, Intel a développé la bibliothèque open source consacrée à faciliter la mise en œuvre de programmes liés à la vision par ordinateur appelée Open Computer Vision Library (OpenCV).

Pour entraîner les classificateurs, deux ensembles d'images sont nécessaires. Un ensemble d'images d'une scène qui ne contient pas l'objet, dans ce cas un trait de visage, qui va être détecté, cet ensemble d'images est appelé images négatives. L'autre ensemble d'images, les images positives, contiennent une ou plusieurs instances de l'objet. L'emplacement des objets dans les images positives est spécifié par : le nom de l'image, le pixel supérieur gauche, la hauteur et la largeur de l'objet [28].

Afin de produire la détection des caractéristiques faciales la plus robuste possible, l'ensemble d'images positives d'origine doit être représentatif de la variance entre différentes personnes, y compris la race, le sexe et l'âge.

Il existe de nombreux formats d'images différents utilisés pour stocker et transmettre des images sous forme compressée, car les images brutes sont de grandes structures de données qui contiennent beaucoup de redondance (par exemple, des corrélations entre des pixels proches) et sont donc hautement compressibles. Différents formats sont spécialisés pour la compressibilité, la maniabilité ou les propriétés des imprimantes et des navigateurs, on note les formats jpeg, png, jpg, bmp,...[7].

Pour réaliser et mettre en œuvre les travaux ci-dessus, du matériel et des logiciels ultérieurs sont nécessaires pour développer l'approche proposée. J'ai utilisé les outils et environnements suivants : système d'exploitation Windows10, Python 2.7/3.10, bibliothèque OpenCV, bibliothèque Numpy, et bibliothèque Pillow principalement et un ordinateur intel CORE i3.

4.2 Implémentation

Le processus de détection et d'identification des visages est une technique d'apprentissage automatique, en apprenant et en extrayant les caractéristiques physiques de

l'humain. Dans cette partie, on va faire correspondre ces caractéristiques avec les images testées afin d'identifier une personne et la reconnaître. Il existe plusieurs paramètres difficiles et variables dans la détection et l'identification des visages, tels que l'éclairage, les différentes poses, les expressions de changement, les images d'entrée de mauvaise qualité, etc.

4.2.1 Détection de visage

Pour réussir à faire la détection du visage, on doit posséder d'un fichier de classificateur HaarCascade, dans ce qui suit nous allons découvrir ensemble les principales étapes afin de créer ce fichier.

Les étapes nécessaires pour créer un fichier Haar cascade classifier sont :

- Collection d'images d'entraînement positives et négatives
- Marquage d'images positives à l'aide des outils objectmarker.exe
- Création d'un fichier .vec (vecteur) basé sur des images marquées positives à l'aide de createsamples.exe
- Formation du classificateur à l'aide de haartraining.exe
- Exécution du classificateur à l'aide de cvHaarDetectObjects()

Donc, comment cela fonctionne ? Ce sont d'énormes fichiers xml individuels avec beaucoup d'ensembles de fonctionnalités et chaque xml correspond à un type très spécifique de cas d'utilisation.

4.2.1.1 Etape 1 : Les images positives et les images négatives

Les images positives sont celles qui contiennent l'objet que nous souhaitons détecter et les images négatives sont les images qui ne contiennent pas l'objet et contiennent toute sorte d'objets différents que celui que nous voulons détecter.



FIGURE 4.1 – Images positives

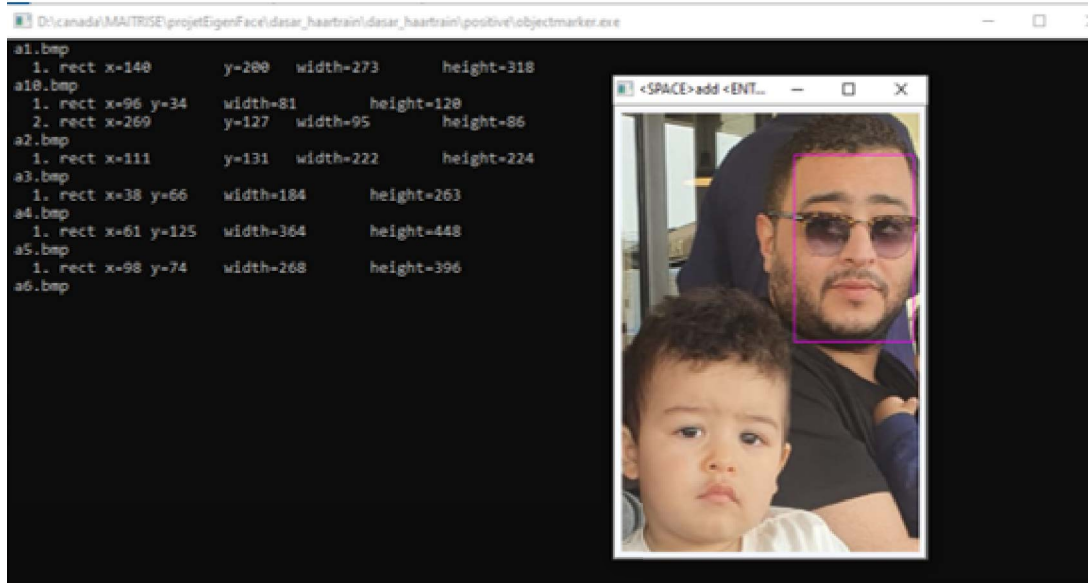


FIGURE 4.2 – Images négatives

4.2.1.2 Etape 2 : Préparation des images

Pour entraîner le classificateur, nous avons besoin d'un fichier texte contenant les noms des images négatives, pour le créer nous devons exécuter le fichier batch bg.bat

Ensuite, nous préparons les images positives afin de créer le fichier vector qui contient les noms des images positives ainsi que l'emplacement des objets dans chaque image. Pour cela il faut avoir des images de type .bmp dans le dossier rowdata, par la suite il faut exécuter l'application ObjectMarker afin de marquer notre objet dans chaque image. ObjectMarker ouvre deux fenêtres, une pour afficher l'image positive, dont je souhaite marquer l'objet, et la deuxième pour afficher les informations nécessaires sur l'image, à savoir le nom, les coordonnées de la position de l'objet après l'avoir marqué.



Un fichier Infor.txt a été créé contenant toutes les informations des images comme suit :

rawdata/a1.bmp 1 140 200 273 318

rawdata/a10.bmp 2 96 34 81 120 269 127 95

rawdata/a2.bmp 1 111 131 222 224

rawdata/a3.bmp 1 38 66 184 263

rawdata/a4.bmp 1 61 125 364 448

rawdata/a5.bmp 1 98 74 268 396

rawdata/a6.bmp 2 152 36 104 164 24 214 128 148

rawdata/a7.bmp 1 83 103 248 356

rawdata/a8.bmp 1 37 35 90 124

rawdata/a9.bmp 4 128 28 60 68 285 78 47 62 388 129 44 58 472 69 50 68

Le premier nombre de chaque ligne définit le nombre d'objets existants dans l'image donnée, et les quatre nombres qui suivent définissent les données de la position de l'objet dans l'image. Pour les images qui contiennent n fois l'objet, nous aurons n fois les quatre chiffres successivement.

4.2.1.3 Etape 3 : création du fichier Vector des images positives

On exécute le fichier batch createsamples qui contient la commande suivante :

```
createsamples.exe -info positive/info.txt -vec vector/facevector.vec -num 100000  
-w 24 -h 24
```

-info positive/info.txt Chemin du fichier d'informations positives

-vec vector/facevector.vec Chemin du fichier vectoriel de sortie

-num 100000 Nombre de fichiers positifs à compresser dans un fichier vectoriel

-w 24 Largeur des objets

-h 24 Hauteur des objets

4.2.1.4 Etape 4 : Le Haar-Training

Il faut adapter le fichier haarTraining selon le nombre d'images positives et négatives qu'on a pour entraîner le programme.

```
haartraining.exe -data cascades -vec vector/vector.vec -bg negative/bg.txt -npos 10 -nneg 200 -nstages 15 -mem 1024 -mode AI  
-rem -nonsym
```

-data cascades chemin pour stocker la cascade de classificateurs

-vec data/vector.vec Chemin qui indique l'emplacement du fichier vectoriel

-bg negative/bg.txt Chemin qui pointe vers le fichier d'arrière-plan

-npos 10 Nombre d'échantillons positifs inférieur à no. fichiers bmp positifs

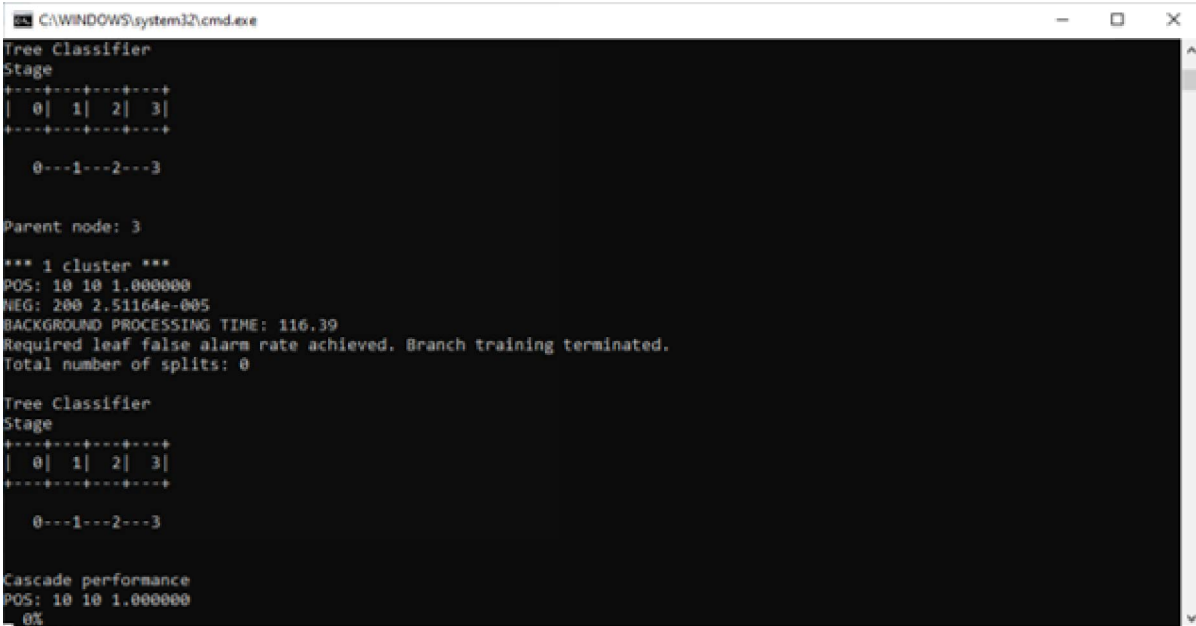
-nneg 200 Nombre d'échantillons négatifs (patches) supérieur à npos

-nstages 15 Nombre d'étapes prévues pour la formation

-mem 1024 Quantité de mémoire affectée en Mo

-w 24 -h 24 Taille de l'échantillon

-nonsym on utilise ceci si l'objet n'est pas symétrique horizontalement



```
C:\WINDOWS\system32\cmd.exe
Tree Classifier
Stage
+---+---+---+---+
| 0| 1| 2| 3|
+---+---+---+---+
0---1---2---3

Parent node: 3

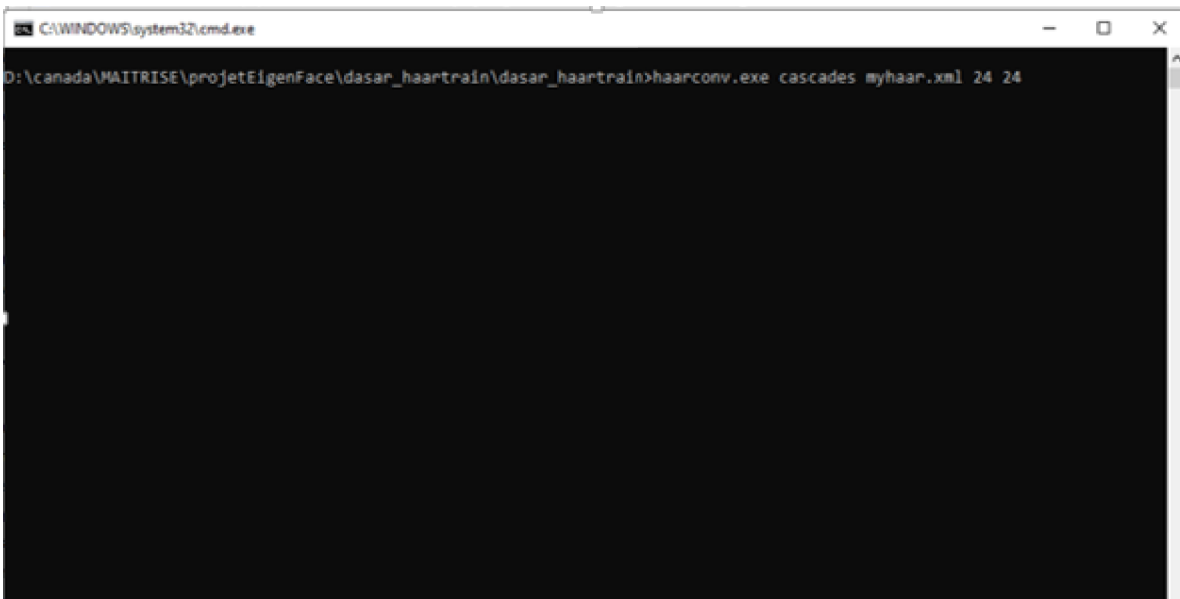
*** 1 cluster ***
POS: 10 10 1.000000
NEG: 200 2.51164e-005
BACKGROUND PROCESSING TIME: 116.39
Required leaf false alarm rate achieved. Branch training terminated.
Total number of splits: 0

Tree Classifier
Stage
+---+---+---+---+
| 0| 1| 2| 3|
+---+---+---+---+
0---1---2---3

Cascade performance
POS: 10 10 1.000000
0%
```

4.2.1.5 Etape 5 : création du fameux fichier haarCascadeClassifier personnalisé

Après avoir fini l'étape de l'entraînement de notre fichier, on obtient les catalogues numérotés définissant les stages d'entraînement réalisés. Maintenant, nous devons combiner et regrouper les différents entraînements effectués pour obtenir notre fichier entraîné en exécutant le fichier batch convert.bat



```
C:\WINDOWS\system32\cmd.exe
D:\canada\VAITRISE\projetEigenFace\dasar_haartrain\dasar_haartrain>haarconv.exe cascades myhaar.xml 24 24
```

Le fichier myhaar.xml a été créé avec succès, nous pouvons l'utiliser dans notre programme pour détecter l'objet que nous cherchons dans l'image qui est le visage.

Dans ce projet, on va utiliser notre fichier haarcascade pour détecter les visages. Importons cv2 et utilisons également la fonction CascadeClassifier d'OpenCV pour pointer vers l'emplacement où nous avons stocké le fichier XML.

```
import cv2

cap=cv2.VideoCapture(0)
f_c=cv2.CascadeClassifier("D:\\canada\\WAITRISE\\projetEigenFace\\dasar_haartrain\\dasar_haartrain\\myhaar.xml")
```

Maintenant, la deuxième étape consiste à charger l'image et à la convertir en échelle de gris.

$$G_{i,j} = (0.2989, 0.5870, 0.1140)^T \cdot P_{i,j}$$

Généralement, les images que nous voyons se présentent sous la forme d'un canal RVB (Rouge, Vert, Bleu). Ainsi, lorsque OpenCV lit l'image RVB, il stocke généralement l'image dans le canal BGR (bleu, vert, rouge). Aux fins de la reconnaissance d'image, nous devons convertir ce canal BGR en canal gris. La raison pour laquelle le canal gris est facile à traiter et nécessite moins de calcul est qu'il ne contient qu'un seul canal de noir-blanc.

```
image_gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
```

Maintenant, après avoir converti l'image de RVB en gris, nous allons maintenant localiser les caractéristiques exactes de notre visage. Voyons comment nous pourrions implémenter cela dans le code.

```
faces=f_c.detectMultiScale(image_gray,1.3,2)
```

Dans ce morceau de code, nous utilisons une fonction intégrée qui s'appelle detectMultiScale et on lui donne en paramètre l'objet dans lequel on a chargé notre image après avoir été convertie en niveau de gris.

Cette fonction nous aidera à trouver les caractéristiques / emplacements de notre image. On utilisera toutes les fonctionnalités de l'objet `f_c` pour détecter les fonctionnalités de la nouvelle image.

Les paramètres que nous passerons à cette fonction sont :

- La variable d'échelle de gris;
- `scaleFactor`, paramètre spécifiant dans quelle mesure la taille de l'image est réduite;
- `minNeighbors`, paramètre spécifiant le nombre de voisins que chaque rectangle candidat doit avoir pour être conservé.

À partir de l'étape ci-dessus, la fonction `detectMultiScale` renvoie 4 valeurs : coordonnée `x`, coordonnée `y`, largeur (`w`) et hauteur (`h`) de la caractéristique détectée du visage. Sur la base de ces 4 valeurs, nous allons dessiner un rectangle autour du visage.

```

while b==True:
    b, image=cap.read()
    if b==True:
        image_gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        faces=f_c.detectMultiScale(image_gray,1.3,2)
        for(x,y,h,w) in faces:
            cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
            visageDetected=image_gray[y:y+h,x:x+w]
            id_, conf =recognizer.predict(visageDetected)
            if conf>=45 and conf<=85:
                font= cv2.FONT_HERSHEY_SIMPLEX
                name= labels[id_]
                color= (255,255,255)
                stroke= 2
                cv2.putText(image, name, (x,y), font, 1, color, stroke, cv2.LINE_AA)
            img_element="mon_visage.png"
            cv2.imwrite(img_element,visageDetected)
            cv2.imshow("soukaina face recognition", image)
            out.write(cv2.cvtColor(image_gray, cv2.COLOR_GRAY2BGR))
            path='D:/canada/MAITRISE/projetEigenFace/capturePhoto.jpg'
            cv2.imwrite(path,image_gray)
            if cv2.waitKey(10)==ord('q'):
                break
cap.release()
out.release()

```

FIGURE 4.3 – Code pour détecter le visage

Ce bout de code lance la caméra de l'ordinateur et permet de détecter le visage et enregistrer la séquence vidéo avec le visage détecté ainsi qu'une image du visage.

Pour stopper l'exécution du code, nous allons appuyer sur 'q', et enfin nous libérons la source avec la *cap.release()* et *ou.release()*.

L'algorithme capte bien le flux de la caméra et parvient parfaitement à détecter le visage.

Nous avons la sortie du programme ci-dessus sous forme de vidéo afin de détecter le visage devant la caméra :

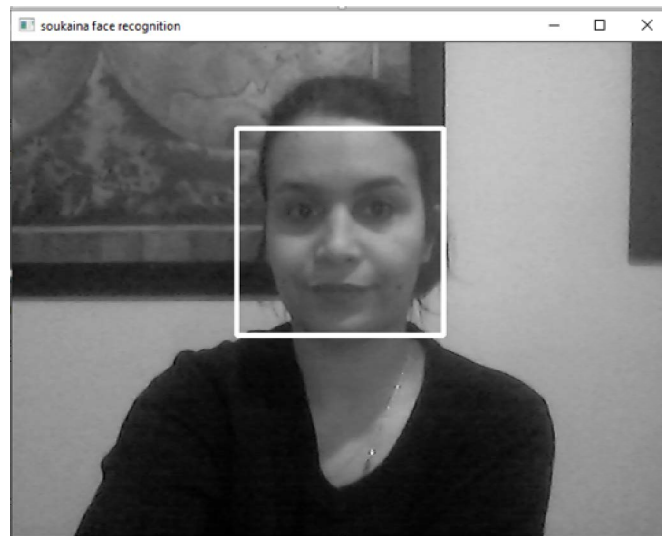


FIGURE 4.4 – Détection de visage

La détection de visage est généralement utilisée comme une étape de pré-traitement pour trouver des régions qui sont potentiellement des visages et des membres humains dans les images.

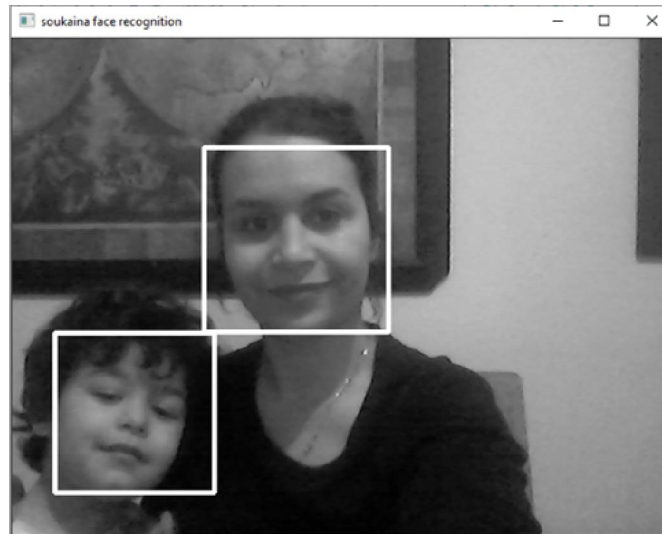


FIGURE 4.5 – Détection de plusieurs visages

Bien que le classificateur haarcascade soit assez utile, cette approche présente peu d'inconvénients.

La partie la plus difficile étant de spécifier avec précision la valeur des paramètres de `scaleFactor` et `minNeighbors` de la fonction `detectMultiScale`. Il est assez courant de se heurter à des scénarios dans lesquels nous devons régler les deux paramètres image par image, ce qui est un gros problème lorsqu'il s'agit d'un cas d'utilisation de détection d'image.

4.2.2 Reconnaissance de visage

Cette partie consiste à faire la reconnaissance du visage détecté auparavant à l'aide des caractéristiques Haar et de l'algorithme Adaboost.

D'abords on utilise le fichier Haar Cascade Classifier créé pour détecter tous les visages dans les images de la base de données à l'aide de la fonction `os.walk()` qui parcourt tout le répertoire des images qui est sous forme d'une arborescence.

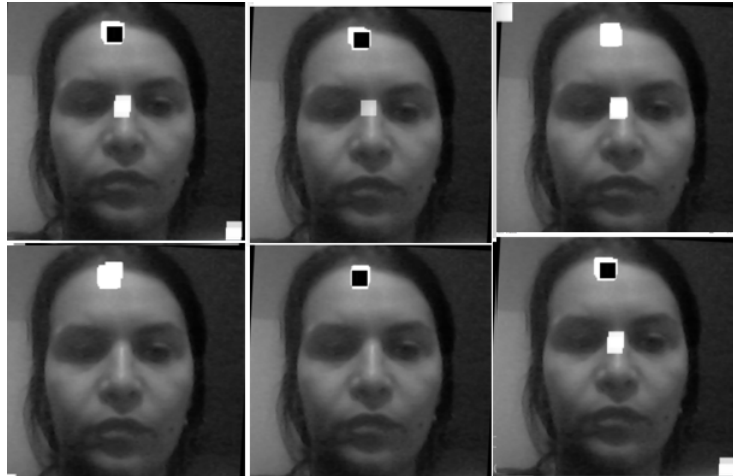


FIGURE 4.6 – Les caractéristiques Haar

Ensuite on construit deux tableaux, le premier contient l'ensemble de photos et le deuxième contient l'identité de chaque personne qui est le nom du répertoire. S'il y a des images dans le répertoire, on va récupérer le nom du dossier contenant l'image, donc par exemple on pourra être dans le répertoire images/soukaina_alaoui ce qu'on veut comme nom ou identité, c'est juste soukaina_alaoui, donc à l'aide de la fonction *split()* on pourra fractionner la phrase selon le caractère slash et récupérer que le dernier argument.

Par la suite, on va parcourir tout le tableau files contenant les fichiers afin de récupérer ceux qui se terminent par .png ou .jpg et les stocker dans le tableau des X, si le label qu'on est en train de traiter est déjà présent dans le tableau, on passe, si ce n'est pas le cas on l'ajoute tout simplement et enfin on récupère l'ID en fonction du label du nom de la personne dans mon tableau d'ID et on l'ajoute cette fois-ci au tableau Y qui contient donc les identités des différents visages, et la boucle nous aide à former deux grands tableaux, le premier pour stocker l'ID de chacune des photos et le deuxième stocke l'ensemble des identités des images.

Enfin on pourra faire l'apprentissage automatique, l'entraînement se base sur les deux tableaux présentés ci-dessus et il est important que ces deux tableaux aient le même nombre d'éléments, et on lance notre *recognizer.train()* afin de générer un fichier entraîné qui va nous permettre de reconnaître les visages et les identifier par la suite.

Le code suivant présente la partie de la génération du fichier d'entraînement à l'aide de l'histogramme LBP :


```
for root, dirs, files in os.walk(image_dir):
    for file in files:
        if file.endswith("png") or file.endswith("jpg") or file.endswith("jpeg") or file.endswith("PNG"):
            path=os.path.join(root, file)
            label= os.path.basename(root).replace(" ", "-").lower()
            if not label in label_ids:
                label_ids[label]=current_id
                current_id+=1
            id_ = label_ids[label]
            pil_image= Image.open(path).convert("L")
            size= (550,550)
            final_image= pil_image.resize(size, Image.ANTIALIAS)
            image_array = np.array(final_image, "uint8")
            faces= faces_cascade.detectMultiScale(image_array, scaleFactor=1.5, minNeighbors=5)

            for(x,y,w,h) in faces:
                roi= image_array[y:y+h, x:x+w]
                x_train.append(roi)
                y_labels.append(id_)

with open("labels.pickle", 'wb') as f:
    pickle.dump(label_ids, f)
recognizer.train(x_train, np.array(y_labels))
recognizer.save("trainer.yml")
```

FIGURE 4.7 – Entraînement avec l’histogramme LBP

Cette technique LBP nous permet de faire l’extraction des caractéristiques des différentes parties de l’image sélectionnées afin de calculer son histogramme et le comparer avec les histogrammes des autres images pour pouvoir mesurer la similarité entre elles et sortir avec un fichier bien entraîné qui peut comparer l’image reçue avec celles dans la base de données afin de faire l’identification.



FIGURE 4.8 – Extraction des caractéristiques avec LBP

L'idée principale de cette technique est de faire la représentation locale de chaque pixel de l'image afin d'étudier leurs propriétés et leurs caractéristiques. Comme on peut voir dans la figure ci-dessus, la technique LBP détecte les bords, les lignes, les taches, les contours et les structures planes qui peuvent être mesurés et estimés par un histogramme afin de le comparer par d'autres histogrammes dans le but de calculer la similarité et d'en déduire s'il s'agit de la personne à identifier ou pas.

4.3 Expérimentation

D'abord, nous allons prendre un exemple simple qui est de faire la classification entre deux visages, une méthode très simple est de calculer leurs motifs binaires locaux ensuite les histogrammes normalisés de ces derniers.

Pour cela, nous prenons deux images de la même personne et la troisième image sera d'une personne différente. Pour classifier ces images, nous devons mesurer la similarité entre les deux images qui représente la même classe et l'image différente, et ce à l'aide de la différence euclidienne entre leurs histogrammes normalisés des motifs binaires locaux. La figure suivante montre l'expérimentation de cette technique.

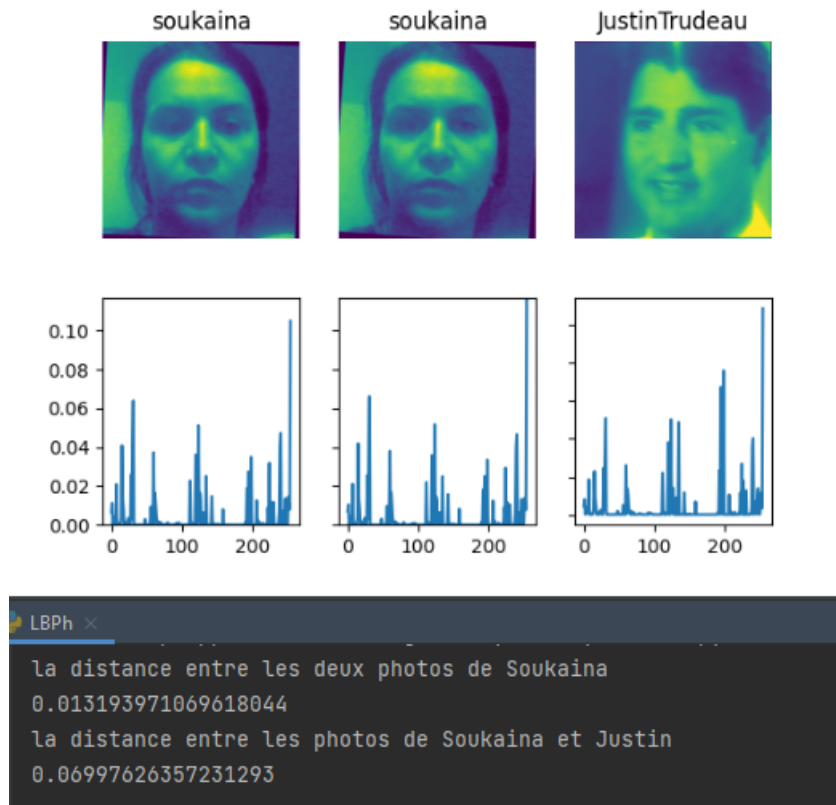


FIGURE 4.9 – Histogramme LBP et la mesure de similarité

Comme nous pouvons voir les histogrammes des images de Soukaina se ressemblent avec une distance euclidienne proche de 0, par contre l'histogramme de Justin est clairement différent et la mesure de dissimilitude est élevée, donc nous pouvons dire que le calcul de la distance euclidienne entre les histogrammes des images nous aide à déterminer si les visages se ressemblent et s'il s'agit de la même personne, cette technique est efficace et nous donne des résultats précis avec un coût optimal.

Cette partie indique que les expériences exécutées correspondent aux performances de l'algorithme de motifs binaires locaux de reconnaissance faciale appliqué dans ce système. Il est nécessaire de décrire les conditions expérimentales qui doivent être acceptables pour le système proposé. Ainsi, les expériences proposées doivent être exécutées en changeant certains facteurs qui sont significatifs dans la méthode d'apprentissage et de reconnaissance du système, comme la position du visage, l'éclairage...

Pour tester cette approche de reconnaissance faciale en utilisant les motifs binaires locaux et voir sa précision, l'auteur de ce mémoire s'est placée devant la caméra avec des

photo de plusieurs personnes publiques et nous avons été reconnus par le système, et en temps réel, comme le montre la figure 4.10 suivante :

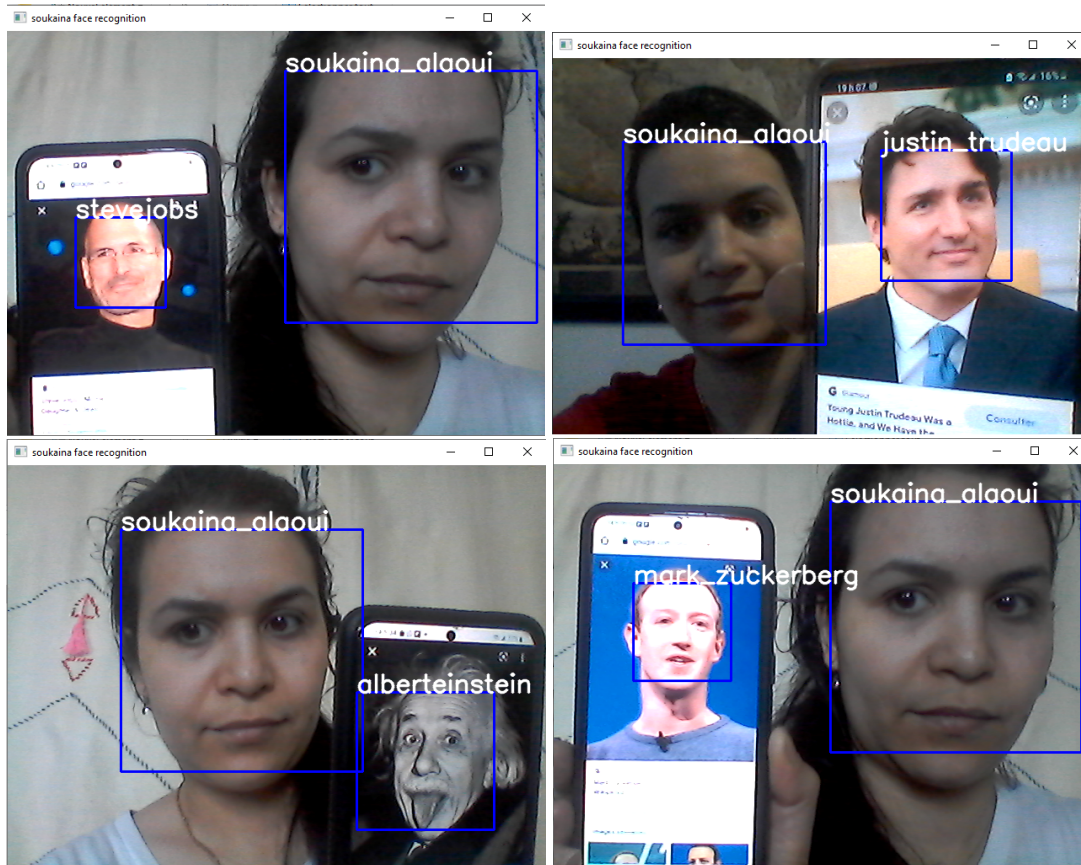


FIGURE 4.10 – Reconnaissance faciale

Afin de mettre en place l'implémentation de cette approche de reconnaissance faciale, nous avons réalisé une application avec Python et openCV pour faciliter la manipulation et l'utilisation du code.

Pour réaliser une interface graphique avec python afin de présenter l'application à l'écran pour l'utilisateur, il faut utiliser la bibliothèque tkinter [3]. La bibliothèque tkinter permet de construire une interface graphique contenant des boutons, des zones de texte et plusieurs outils graphiques pour permettre aux utilisateurs la possibilité d'avoir une application facile à manipuler et d'utiliser les fonctionnalités offertes sans avoir à saisir ou comprendre les lignes de code.

L'interface graphique de notre application de reconnaissance de visage est très simple, elle contient des boutons d'interaction, pour effectuer plusieurs tâches telles que la reconnaissance faciale l'enregistrement de la vidéo d'identification du visage et la

sauvegarde d'une photo du visage identifié, et du texte explicatif, elle peut être présentée comme suit :



FIGURE 4.11 – Interface graphique

Le bouton Reconnaissance de visage lance la caméra et exécute le programme de détection et de reconnaissance de visage,

Le bouton Ouvrir permet d'ouvrir la photo ou la vidéo capturée du visage identifié à partir du répertoire dans lequel l'image ou la vidéo de la personne reconnue a été enregistrée.

Le bouton Quitter permet de quitter l'application.

4.4 Résultats et perspectives

D'après ce qui précède, nous observons que cette méthode est assez simple en terme d'extraction des motifs binaires locaux et donne des résultats satisfaisants en terme de temps et de précision. Les résultats produits par le prototype développé montrent qu'une personne peut être détectée et reconnue en quelques fractions de secondes et en temps réel.

On peut donc croire, que nous avons obtenu de bons résultats à partir de diverses analyses expérimentales de cette technique, ils fournissent également des résultats acceptables pour l'occlusion, la variation de pose et l'illumination.

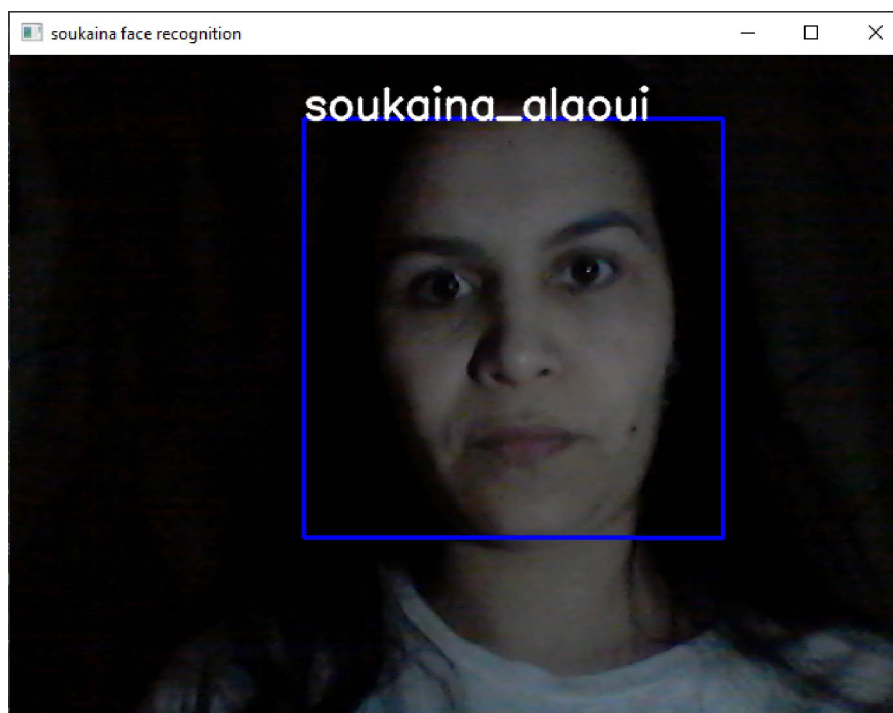


FIGURE 4.12 – Reconnaissance faciale avec éclairage faible

La vraie variation de pose se produit lorsque la position du sujet est à un angle différent par rapport à la caméra, et le système peut reconnaître le visage même si dans l'ensemble de données, seules les images frontales sont stockées comme indiqué dans la figure 4.10. Plusieurs chercheurs [19] [2] ont obtenu de bons résultats même avec différentes déviations angulaires et à différentes résolutions en utilisant cette approche.

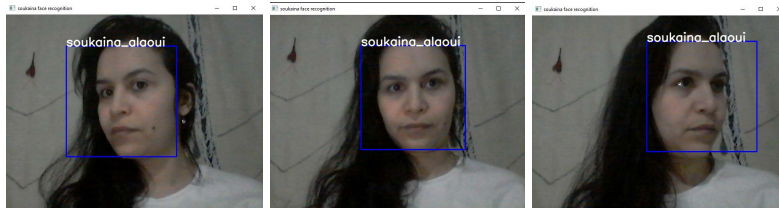


FIGURE 4.13 – Reconnaissance faciale avec différentes déviations angulaires

Comme illustré ci-dessus, le test a été fait avec trois variations angulaires différentes, une à environ 30° à gauche, la deuxième c'est une position frontale face et la troisième à environ 30° à droite et on obtient toujours le même résultat, qui est l'identification de la personne en temps réel.

Cependant, comme tout système de reconnaissance faciale, on peut remarquer des imperfections, à chaque fois que l'éclairage de la pièce était insuffisant, le taux de reconnaissance diminuait, puisqu'il y avait un nombre important de faux positifs. Un autre défi était celui du matériel, la reconnaissance faciale nécessitant un matériel informatique performant et plus particulièrement une caméra haute définition avec une haute résolution.

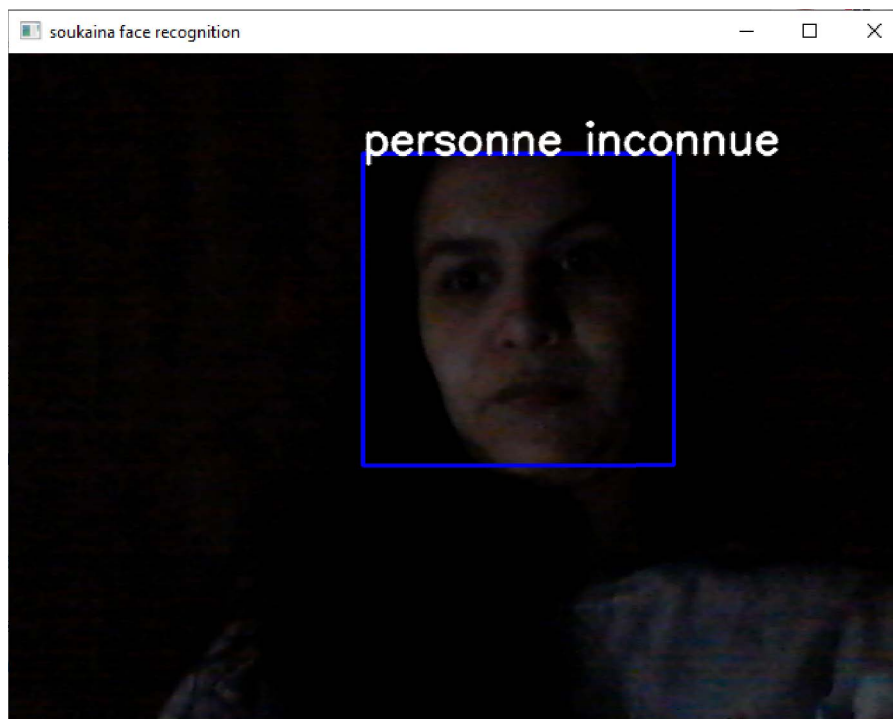


FIGURE 4.14 – Reconnaissance faciale avec éclairage insuffisant

La précision de cette approche est d'une valeur plus que 80% avec une résolution faible de l'image. Les auteurs de [19] ont eu une précision de 78.4%, on peut obtenir de meilleurs résultats avec une haute résolution d'image et une base de données volumineuse.

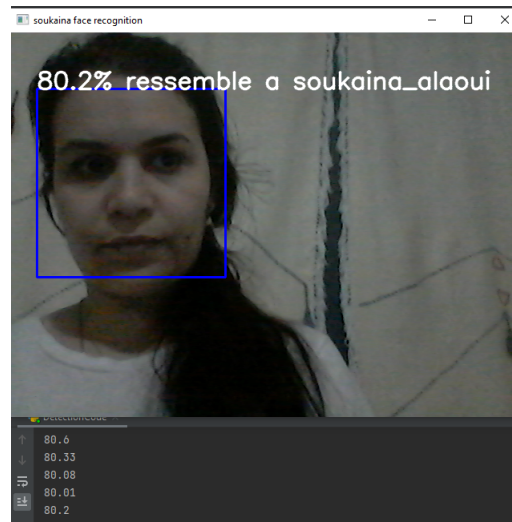


FIGURE 4.15 – Précision de la méthode LBPH

De plus, le nombre d'images par personne dans la base de données de l'ensemble d'apprentissage ayant des variations de pose et d'éclairage plus diverses joue un rôle essentiel dans la détermination de la précision de la reconnaissance et peut augmenter la précision et l'efficacité de l'approche.

Comme l'apprentissage automatique est très important de nos jours, il existe de nombreux domaines où ce travail peut être étendu.

Le système développé pourrait identifier les visages humains en temps réel et être intégré à Google Maps comme exemple d'application, pour suivre n'importe quel sujet d'intérêt. En outre, les systèmes de reconnaissance faciale peuvent également être utilisés dans le développement de systèmes de présence automatisés et pour enquêter sur des activités criminelles.

4.5 Conclusion générale

La technologie de reconnaissance faciale a parcouru un long chemin au cours des vingt dernières années. Aujourd'hui, les machines sont capables de vérifier automatiquement les informations d'identité pour les transactions sécurisées, pour les tâches de surveillance et de sécurité, pour le contrôle d'accès aux bâtiments etc. Ces applications fonctionnent généralement dans des environnements contrôlés et les algorithmes de reconnaissance peuvent tirer parti des contraintes environnementales pour obtenir une grande précision de reconnaissance. Cependant, les systèmes de reconnaissance faciale de nouvelle génération vont avoir une application généralisée dans les environnements intelligents - où les ordinateurs et les machines ressemblent davantage à des assistants utiles.

Pour atteindre cet objectif, les ordinateurs doivent être capables d'identifier de manière fiable les personnes à proximité d'une manière qui s'inscrit naturellement dans le schéma des interactions humaines normales. Ils ne doivent pas nécessiter d'interactions particulières et doivent se conformer aux intuitions humaines sur le moment où la reconnaissance est probable. Cela implique que les futurs environnements intelligents devraient utiliser les mêmes modalités que les humains et présenter à peu près les mêmes limites. Ces objectifs semblent maintenant atteints - cependant, des recherches substantielles restent à faire pour que la technologie de reconnaissance des personnes fonctionne de manière fiable, dans des conditions très variables en utilisant des informations provenant de modalités uniques ou multiples.

4.6 Annexe 1 : Détection de visage

```

1  import numpy as np
2  import cv2
3  import pickle
4
5
6  cap=cv2.VideoCapture(0)
7  f_c=cv2.CascadeClassifier("D:/canada/MAITRISE/projetEigenFace/
   dasar_haartrain/dasar_haartrain/myhaar.xml")
8  recognizer = cv2.face.LBPHFaceRecognizer_create()
9  recognizer.read("trainer.yml")
10 labels = {}
11 with open("labels.pickle", 'rb') as f:
12     og_labels= pickle.load(f)
13     labels = {v:k for k,v in og_labels.items()}
14
15 def manup():
16     fourcc=cv2.VideoWriter_fourcc(*'MJPG')
17     width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH) + 0.5)
18     height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT) + 0.5)
19     size = (width, height)
20     out=cv2.VideoWriter('D:/canada/MAITRISE/projetEigenFace/
   captureDetection.avi',fourcc,20.0,(int(cap.get(3)), int(cap.get
   (4))))
21     b=True
22     while b==True:
23         b, image=cap.read()
24         if b==True:
25             image_gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
26             faces=f_c.detectMultiScale(image_gray,1.3,2)
27             for(x,y,h,w) in faces:
28                 cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
29                 visageDetected=image_gray[y:y+h,x:x+w]
30                 id_, conf =recognizer.predict(visageDetected)
31                 if conf>=45 and conf<=85:
32                     font= cv2.FONT_HERSHEY_SIMPLEX
33                     name= labels[id_]
34                     color= (255,255,255)
35                     precision= float("{0:.2f}".format((100*(1-(
   conf)/300))))
36                     acc.append(precision)
37                     print(precision)

```

```
38         stroke= 2
39         cv2.putText(image, name, (x,y), font, 1,
color, stroke, cv2.LINE_AA)
40         img_element="mon_visage.png"
41         cv2.imwrite(img_element, visageDetected)
42         cv2.imshow("soukaina face recognition", image)
43         out.write(cv2.cvtColor(image_gray, cv2.
COLOR_GRAY2BGR))
44         path='D:/canada/MAITRISE/projetEigenFace/
capturePhoto.jpg'
45         cv2.imwrite(path, image_gray)
46         if cv2.waitKey(10)==ord('q'):
47             break
48         cap.release()
49         out.release()
50         cv2.destroyAllWindows()
51         return 0
52
53 manup()
```

4.7 Annexe 2 : Reconnaissance et identification

```
1  import os
2
3  import cv2
4  import numpy as np
5  from PIL import Image
6  import pickle
7
8  BASE_DIR= os.path.dirname(os.path.abspath(__file__))
9  image_dir = os.path.join(BASE_DIR, "images")
10 faces_cascade = cv2.CascadeClassifier("D:/canada/MAITRISE/
    projetEigenFace/dasar_haartrain/dasar_haartrain/myhaar.xml")
11 recognizer = cv2.face.LBPHFaceRecognizer_create()
12
13 current_id=0
14 label_ids = {}
15 y_labels = []
16 x_train = []
17
18 for root, dirs, files in os.walk(image_dir):
19     for file in files:
20         if file.endswith(".png") or file.endswith(".jpg") or file.
    endswith(".jpeg") or file.endswith(".PNG"):
21             path=os.path.join(root, file)
22             label= os.path.basename(root).replace(" ", "-").lower()
23             if not label in label_ids:
24                 label_ids[label]=current_id
25                 current_id+=1
26             id_ = label_ids[label]
27             pil_image= Image.open(path).convert("L")
28             size= (550,550)
29             final_image= pil_image.resize(size, Image.ANTIALIAS)
30             image_array = np.array(final_image, "uint8")
31             faces= faces_cascade.detectMultiScale(image_array,
    scaleFactor=1.5, minNeighbors=5)
32
33             for(x,y,w,h) in faces:
34                 roi= image_array[y:y+h, x:x+w]
35                 x_train.append(roi)
36                 y_labels.append(id_)
37
38 with open("labels.pickle", 'wb') as f:
39     pickle.dump(label_ids, f)
```

```
40  
41 recognizer.train(x_train, np.array(y_labels))  
42 recognizer.save("trainer.yml")
```

4.8 Annexe 3 : Interface graphique

```
1  import tkinter
2  from tkinter import *
3  import cv2
4  from PIL import Image, ImageTk
5  from tkinter import filedialog
6  import numpy as np
7  from tkinter import messagebox
8  import tkinter as tk
9  import shutil, os
10 import subprocess
11 import customtkinter
12 import tkinter
13 import sys
14 import os
15 from tkinter import ttk
16
17 customtkinter.set_appearance_mode("System") # Other: "Light", "Dark
18
19
20
21 PATH = os.path.dirname(os.path.realpath(__file__))
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

41     # ===== create two CTkFrames =====
42     imagee = Image.open("D:/canada/MAITRISE/projetEigenFace/fond
43     .jpg").resize((self.WIDTH, self.HEIGHT))
44     self.bg_image = ImageTk.PhotoImage(imagee)
45     self.image_label = tkinter.Label(master=self)
46     self.image_label.place(relx=0.5, rely=0.5, anchor=tkinter.
47     CENTER)
48     self.image_label = tkinter.Label(master=self, image=self.
49     bg_image)
50     self.image_label.place(relx=0.5, rely=0.5, anchor=tkinter.
51     CENTER)
52     self.frame_left = customtkinter.CTkLabel(master=self,
53     borderwidth="2", relief="raised", highlightcolor="black",
54     highlightthickness=1,
55     width=200, bg_color
56     ='black',
57     height=App.HEIGHT
58     -40
59     )
60     self.frame_left.place(relx=0.32, rely=0.5, anchor=tkinter.E)
61
62     self.frame_right = customtkinter.CTkLabel(master=self,
63     borderwidth="2", relief="raised", highlightcolor="black",
64     highlightthickness=1,
65     width=420,
66     height=App.HEIGHT
67     -40, bg_color='black'
68     )
69     self.frame_right.place(relx=0.365, rely=0.5, anchor=tkinter.
70     W)
71
72     # ===== frame_left =====
73
74     self.button_1 = customtkinter.CTkButton(master=self.
75     frame_left,
76     text="Detection",
77     command=self.
78     detection,
79     border_width=0,
80     fg_color='#B6B7B6', bg_color='black',
81     corner_radius=8)
82     self.button_1.place(relx=0.5, y=180, anchor=tkinter.CENTER)
83
84     self.button_2 = customtkinter.CTkButton(master=self.

```

```

70         frame_left ,
71                                     text=" Ouvrir photo " ,
72                                     command= self . ouvrir ,
73                                     border_width=0,
74         fg_color= '#B6B7B6' , bg_color= 'black' ,
75                                     corner_radius=8)
76         self . bouton_2 . place (relx =0.5 , y=230 , anchor=tkinter . CENTER)
77
78         self . bouton_3 = customtkinter . CTkButton (master= self .
79         frame_left , fg_color= '#B6B7B6' , bg_color= 'black' ,
80                                     text=" Ouvrir Video " ,
81                                     command= self . video ,
82                                     border_width=0,
83                                     corner_radius=8)
84         self . bouton_3 . place (relx =0.5 , y=280 , anchor=tkinter . CENTER)
85
86         # ===== frame_right =====
87
88         # ===== frame_right =====
89
90         self . label_info_1 = customtkinter . CTkLabel (master= self .
91         frame_right ,
92                                     text= " Detection
93         de visage \n" +
94                                     " utilisant
95         le fichier \n" +
96                                     " Haar
97         Cascade personalise " ,
98                                     width=250 ,
99                                     height=100 ,
100                                    corner_radius=20 ,
101                                    bg_color= 'black' ,
102                                    fg_color= (" #
103         B6B7B6 " ) , # <- custom tuple-color
104                                    justify=tkinter .
105         CENTER)
106
107         self . label_info_1 . place (relx =0.5 , rely =0.05 , anchor=tkinter .
108         N)
109         self . label_info_2 = customtkinter . CTkLabel (master= self .
110         frame_right ,
111                                     text=" Charger l'

```

```

102         image" ,
103                                     width=250,
104                                     height=250,
105                                     corner_radius
106         =110, bg_color='black',
107                                     fg_color=( "#
108         B6B7B6" ),
109                                     # <- custom tuple
110                                     justify=tkinter .
111         CENTER)
112         self.label_info_2.place(relx=0.5, rely=0.35, anchor=tkinter .
113         N)
114
115         # ===== frame_right <- =====
116
117         self.button_5 = customtkinter.CTkButton(master=self .
118         frame_right ,
119                                     height=25,
120                                     text=" Quitter" ,
121                                     command=self .
122         terminer_click ,
123                                     border_width=0,
124                                     bg_color='black',
125                                     corner_radius=8,
126                                     fg_color='#B6B7B6')
127         self.button_5.place(relx=0.50, rely=0.95, anchor=tkinter .
128         CENTER)
129
130
131         def detection(self):
132             subprocess.Popen("C:/Users/hp/PycharmProjects/pythonProject1
133             /DetectionCode.py", shell=True)
134
135         def ouvrir(self, event=0):
136             global my_image
137             self.filename = filedialog.askopenfilename(initialdir="D:/

```



```

canada/MAITRISE/projetEigenFace/" ,
134                                     filetypes=[('
Picture Files', '*.jpg')]
135     my_image = ImageTk.PhotoImage(Image.open(self.filename).
resize(((self.label_info_2).width, self.label_info_2.height)))
136
137     my_image_label = customtkinter.CTkLabel(master=self.
label_info_2, corner_radius=100, width=(self.label_info_2).width
, height=self.label_info_2.height, image=my_image)
138     my_image_label.place(relx=0.5, rely=0.5, anchor=tkinter.
CENTER)
139     def terminer_click(self, event=0):
140         """ Affiche un message popup puis termine le programme. """
141
142         reponse = messagebox.askquestion('Question', 'Desirez-vous
vraiment quitter ?')
143
144         # askquestion retourne 'yes' ou 'no'
145
146         if reponse == 'yes':
147             self.quit()
148
149         def video(self, event=0):
150             filename = filedialog.askopenfilename(initialdir="D:/canada/
MAITRISE/projetEigenFace/" ,
151                                     filetypes=[('Video
Files', '*.avi')])
152
153             print(filename)
154             cap = cv2.VideoCapture(filename)
155             ret = True
156             while ret == True:
157                 ret, frame = cap.read()
158                 if ret == True:
159                     cv2.imshow('frame', frame)
160                     if (cv2.waitKey(250) & 0xFF == ord('q')):
161                         break
162
163             cap.release()
164             cv2.destroyAllWindows()
165
166         def on_closing(self, event=0):
167             self.quit()
168         def start(self):

```

```
169         self.mainloop()
170     if __name__ == "__main__":
171         app = App()
172         app.start()
```

Bibliographie

- [1] La belle histoire des révolutions numériques, chapitre : l'inquiétante reconnaissance faciale. (2) :340.
- [2] Pratik Nimbal Gopal Krishna Shyam Abhishek Pratap Singh, SunilKumar S Manvi. Face recognition system based on lbph algorithm.
- [3] Justin Bleich Abraham J. Wyner, Matthew Olson. Explaining the success of adaboost and random forests as interpolating classifiers.
- [4] Loubna BEDOUI. Authentification de visages par la méthode d'analyse discriminante linéaire de fischer. (6) :22.
- [5] David CristinacceTim Cootes. Facial feature detection using adaboost with shape constraints.
- [6] Gabriel Cormier. Chapitre 11. logique floue. page 2.
- [7] J G Daugman. Computer science tripos : 16 lectures.
- [8] C.N Ravi² Dr C.Sunil Kumar¹ and J.Dinesh³. Human facerecognition and detection system with genetic and ant colonyoptimization algorithm iosr journal of computer engineering.
- [9] Dr.Bharati. Image classification process.
- [10] Rahmita Wirza O.K. Rahmat Elham Bagherian. Facial feature extraction for face recognition : a review.
- [11] Hazem M. El-Bakry Mohy A. Abo Elsoud. Human face recognition using neural networks.
- [12] Fayaz Ali Dharejo Abddul Ghaffar Hira Memon Farah Deebea¹, Aftab Ahmed. Lbph-based enhanced real-time face recognition.
- [13] Robert E. Schapire Yoav Freund. Boosting foundations and algorithms.
- [14] Daniel Gutierrez. Where are we with computer vision ?
- [15] S. Rankov J. Novakovic. Classification performance using principal component analysis and different value of the ratio r.

- [16] V. Kruger Jim Gemmell, Kentaro ToyamaK. Hierarchical wavelet networks for facial feature localization.
- [17] Takeo Kanade Jing XiaoS. BakerIain Matthews. Real-time combined 2d+3d active appearance models.
- [18] J. Kittler K. Jonsson, J. Matas and Y.P. Li. Learning support vectors for face verification and recognition (2000).
- [19] Semih Aslan2 Kamal Chandra Paul1. An improved real-time face recognition system at low resolution based on local binary pattern histogram algorithm and clahe.
- [20] Ashu Kumar · Amandeep Kaur · Munish Kumar. Face detection techniques : a review.
- [21] Norbert Krüger Christoph von der Malsburg Laurenz Wiskott, Jean-Marc Fellous. Face recognition by elastic bunch graph matching.
- [22] Gary B. Huang Erik Learned-Miller. Labeled faces in the wild : Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014.
- [23] S. Marcel. A symmetric transformation for lda-based face verification.
- [24] Dhia Alzubaydi Mohamed Husein*. Mobile face recognition application using eigen face approaches. (4).
- [25] Yeon-Sik Ryu Se-Young Oh. Automatic extraction of eye and mouth fields from a face image using eigenfeatures and multilayer perceptrons.
- [26] MÄENPÄÄ T. OJALA T., PIETIKÄINEN M. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. 24(7) :971–987.
- [27] David Kriegman Peter N. Belhumeur Joao P. Hespanha. Face recognition : Eigenfaces vs. fish-erfaces : Recognition using class specific projection.
- [28] Dr. John Fernandez Phillip Ian Wilson. Facial feature detection using haar classifiers.
- [29] H. A. Rowley and T. Kanade. Neural network - based face detection.
- [30] Marc SILANUS. Reconnaissance faciale par eigenfaces. (7).
- [31] Simon Baker Terence Sim and Maan Bsat. The cmu pose, illumination, and expression (pie) database of human faces.
- [32] Christopher J Taylor Timothy F. CootesGareth J. Edwards. Active appearance models.
- [33] M. Turk and A. Pentland. Eigenfaces for recognition,.

- [34] Rituparna Jb Tushar Aggrawalb K N Balasubramanya Murthya S Natarajanb Vinay Aa, Vinay S Shekhara. Cloud based big data analytics framework for face recognition in social networks using machine learning. (3).
- [35] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features.
- [36] N. Vandenbroucke A. Ahmad et D. Hamad W. Ben Soltana, A. Porebski. Contribution des descripteurs de texture lbp à la classification d'images de dentelles. page 3.
- [37] J. Kittler Y. Li and J. Matas. On matching scores of lda-based face verification.
- [38] Sheng Zhang and Matthew Turk. Eigenfaces. (5).