Université Saâd Dahlab de Blida-1

Faculté des Sciences

Département d'informatique

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Ingénierie des Logiciels

Mémoire présenté par :

BENBLIDIA Mohamed Walid

HACHI Chakib

En vue d'obtenir le diplôme de Master

Sujet : Détection d'objets d'intérêt dans les images numériques

Encadré par : Pr. Benblidia Nadjia

Examinateur : Mr Chemcham.

Président : Mr Bala.

2015/2016

MA-004-297-1

ملخص

الكشف عن الأشياء يمكن أن يصبح مشكل شائك يصعب حله، بسبب تباين مظهرها في العالم الحقيقي. نقترح في هذه المذكرة حل مشكلة الكشف بانتهاج طريقة تنتمي إلى عائلة استخراج الخصائص، التي تتمحور حول استخراج بيانات التدرجات الموجهة (HOG) لمجموعة من صور التدريب واستخدامها لتكوين مصنف (Adaboost) منظم بتسلسل، الهدف هو الحصول على أداة عامة قادرة على الكشف على أي شيء ذا أهمية في فترة زمنية معقولة.

Résumé

La détection d'objets peut être un problème épineux à résoudre du fait de la variabilité d'apparence des objets dans le monde réel. Nous proposons dans ce mémoire de résoudre le problème de détection en utilisant une approche s'inscrivant dans la famille des extracteurs de caractéristiques. Elle consiste à extraire les histogrammes orientés gradients d'un ensemble d'images d'entrainement et de les utiliser pour entrainer un classifieur Adaboost organisé en cascade. Le but est d'obtenir un outil général capable de détecter n'importe quel objet d'intérêt en un temps raisonnable.

Abstract

Because of the variability in appearance in the real world, object detection can be a tricky problem to resolve. We propose in our study to solve the detection problem by using an extraction feature approach; which consist of extracting the HOG features from a dataset of training images and using them as an input for a classification algorithm Adaboost which will be used as a cascade trainer. The goal is to obtain a tool that can be trained to detect any desirable object.

Mots clés : Détection d'objets, Histogrammes orientés gradients, HOG, Adaboost, Boosting, classifieur en cascade.

Keywords: Object detection, Histograms oriented gradients, HOG, Adaboost, Boosting, cascade classifier.

Remerciements

Nous souhaitons exprimer nos plus sincères remerciements à nos deux promotrices professeur Benblidia Nadjia et Madame Reguieg F.Zohra pour nous avoir proposé le thème de ce mémoire. et pour nous avoir fait profiter de leur précieuse expérience tout au long de la réalisation de notre travail.

Nous exprimons également une profonde gratitude à l'égard de nos familles pour leurs encouragements, leur soutien, et leur amour inconditionnel...

Liste des tableaux

Liste des figures

Table des matières

Introduction générale

En dépit des avancées majeures dans le domaine de la robotique et de l'intelligence artificielle, doter les machines de l'aptitude de voir et de reconnaitre des objets dans une scène s'avère un des défis les plus difficiles et en même temps les plus excitants dans le monde de la vision assistée par ordinateur. En effet, ce qui rend cet accomplissement aussi ardu est la complexité de reproduire les capacités cognitives de l'homme sur machine, d'une part par l'aspect intuitif que la reconnaissance des objets semble s'opérer chez l'homme, et d'autre part par la diversité d'apparence des objets dans le monde réel. Plusieurs facteurs viennent entraver la détection de ces objets, tels que les conditions d'éclairage qui peuvent varier au cours du temps. L'ombre d'une structure voisine peut aussi modifier l'aspect et la couleur de l'objet à détecter. L'illumination de la scène peut quant à elle occasionner des réflexions spéculaires qui modifient aussi l'aspect de l'objet. Toutes ces contraintes sont liées à l'apparence de l'objet.

Dans le cadre de ce mémoire, nous tenterons de développer un outil général, s'affranchissant de connaissances a priori sur les objets, permettant de détecter n'importe quel type objet d'intérêt. Pour cela, nous proposons de conjuguer deux travaux majeurs dans le domaine de la détection d'objets, à savoir : 1) les travaux de Viola & Jones [1] pour la détection de visages. Ils introduisent la notion de classifieur adaptatif organisé sous un schéma en cascade ; et 2) les travaux de Dalal & Triggs [2] pour la détection de personnes. Ils utilisent les histogrammes orientés gradients pour extraire les caractéristiques des objets.

Pour mener à bien notre travail, nous avons adopté le plan suivant :

Le premier chapitre introduit le monde de la détection d'objet et de la vision assistée par ordinateur ; le second chapitre présente une idée non exhaustive des approches de détection qui existent dans la littérature ; le troisième chapitre est consacré à la théorie derrière la solution proposée ; et enfin, le dernier chapitre discutera de l'implémentation de notre solution ainsi que des différents résultats obtenus.

Chapitre 1 : La Vision Assistée par Ordinateur

1. Introduction

A l'heure actuelle, les systèmes d'acquisition d'images sont de plus en plus répandus : appareils numériques, webcams, téléphones portables, caméscopes..., avec une résolution de plus en plus élevée (41 megapixels pour un téléphone portable). Les applications de la vision par ordinateur sont de ce fait de plus en plus répandues : cinéma d'animation/effets spéciaux, jeux vidéos, réalité virtuelle, architecture et archéologie (reconstruction de scènes, visages, monuments) , vidéosurveillance (dans les magasins, rues ou aéroports), l'aide au guidage ou détection d'obstacles (la voiture sans conducteur de Google), logiciels de simulation d'entrainement (apprentissage de pilotes, recrues de l'armée, astronautes), la reconnaissance d'objets… et bien d'autres applications encore [3]. Malgré les avancées de la vision par ordinateur, les systèmes développés sont très loin d'égaler les performances de l'œil et du cerveau humain. En fait, toute recherche visant à reproduire une aptitude cognitive de l'homme sur machine, n'est qu'au stade embryonnaire [4].

2. Processus de la Vision Assistée par Ordinateur

La vision assistée par ordinateur VAO (ou vision artificielle) a comme principal but de permettre à une machine d'acquérir une image issue du monde réel, de la traiter, de l'analyser et de comprendre les informations y figurant, pour la prise éventuelle de décision [5]. Les principales étapes de la chaine VAO (cf. Figure 1) sont :

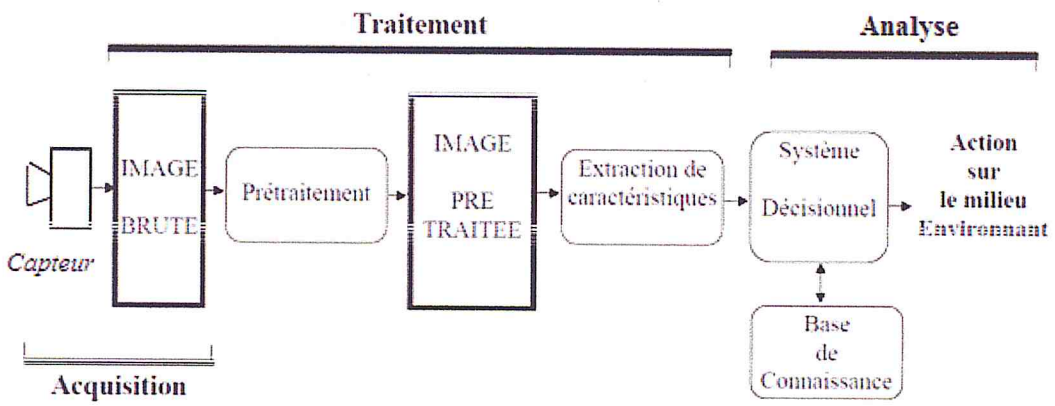


Figure 1 : Chaine de la VAO [5].

a) Acquisition de l'image

Elle correspond à l'opération qui permet d'extraire du monde réel une représentation bidimensionnelle des objets en 3D, en utilisant un système d'acquisition [7]. Ce système peut varier selon le domaine d'application: imagerie médicale (scanner, IRM), sécurité (radars, caméras de surveillance), usage quotidien (Smartphones, appareils numériques),…etc.

b) Prétraitement

La donnée brute issue de l'étape d'acquisition est souvent polluée par des informations parasites dues au dispositif optique ou électronique du capteur (comme la présence de poussière sur l'objectif) [7]. Ces bruits détériorent la qualité de l'image et peuvent handicaper sérieusement l'étape ultérieure d'extraction de caractéristiques. Le prétraitement englobe toutes les techniques qui ont pour objectif d'améliorer la qualité de l'image avant son utilisation [7]. Les méthodes les plus répandues sont le débruitage, la normalisation, le rehaussement de contraste…etc. [7].

c) Extraction de caractéristiques

C'est l'extraction de l'information pertinente, utile à une application particulière. Il s'agit de calculer un certain nombre de caractéristiques ou paramètres de différentes natures : contours, coins, périmètre, connexion (point de bifurcation), concavités, surfaces, couleur, texture, etc. Ces informations seront sauvegardées en mémoire pour être utilisées plus tard dans la phase de décision [7].

d) Système décisionnel

Elle correspond à l'étape ultime de la chaine de la VAO. Elle comporte les algorithmes de l'application tels que : détection d'objets, recherche d'image par le contenu, reconnaissance de la parole, classification automatique…etc [7].

2.1 Historique de la VAO

Les travaux sur la vision par ordinateur ne datent pas d'aujourd'hui ; dés le début des années soixante les chercheurs se sont vivement intéressés à ce domaine. Un bref parcours historique des ces avancées peut être tracé [8] :

- *1960s :* Le tout début de l'intelligence artificielle. Traitement d'images, reconnaissance de formes.
- *1970s :* Travaux fondamentaux sur la formation des images. Flux optique. Reconstruction de la scène…
- *1980s :* Vision comme mathématique appliquée : géométrie, analyse multi-échelle, modélisation probabilistique, optimisation…
- *1990s* : Conclusion des travaux sur l'analyse géométrique, la vision se mêle au graphique, les approches d'apprentissage statistiques refont surface.
- *2000s* : Avancée significative dans la reconnaissance visuelle : développement de méthodes de segmentation, classification et détection d'objets (exemple : la reconnaissance faciale).

L'une des innovations actuelles les plus notables, est le prototype de lunettes de réalité augmentée « HoloLens » présentés par Microsoft au début de l'année 2016. La réalité augmentée peut être expliquée comme une interface entre des données « virtuelles » et le monde réel. Le casque (cf. Figure 2) permettrait d'intégrer des images virtuelles à l'environnement réel. Ces images (objets, graphiques…) s'intègrent au champ de vision et sont contrôlables d'un simple geste : en agitant un doigt dans l'espace par exemple [9].

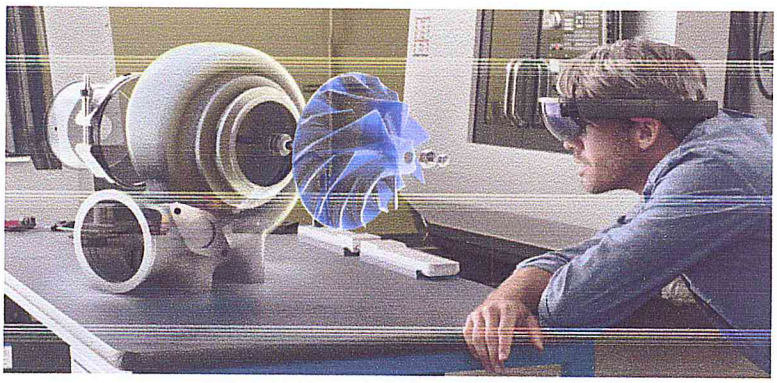


Figure 2: HoloLens de Microsoft, le premier système holographique permettant d'interagir avec des hologrammes de haute définition [9].

2.2 Vision Assistée par Ordinateur et Intelligence Artificielle

Le terme d'Intelligence Artificielle (IA) désigne toutes les techniques et méthodes qui permettent de doter des systèmes informatiques de capacités proches de celles de l'être humain [10]. La science-fiction s'est appropriée ce domaine en imaginant des robots, dotés d'une intelligence qui serait semblable à celle de l'homme. Toutefois, comme cité

précédemment, la maîtrise de l'IA par la communauté scientifique n'est qu'au premier stade de son évolution [10]; les algorithmes de vision les plus évolués n'arrivent toujours pas à égaler les capacités de perception d'un enfant de 3 ans [11].

L'intelligence artificielle est un des domaines de recherche les plus actifs, et n'est certainement pas réduit à la branche de la vision. Nous pouvons citer comme applications de l'IA [12] :

- *Les systèmes experts :* Logiciels capables de simuler le comportement d'un expert humain effectuant une tâche précise.
- *Le calcul formel :* Traitement d'expressions symboliques (fonctions mathématiques).
- *La simulation du raisonnement humain :* Mise au point de logiques de raisonnement (logiques modales, temporelles, floue, non monotones, …etc.).
- *Le traitement du langage naturel :* Compréhension de texte. Traduction dans une autre langue.
- *La résolution de problèmes :* Représentation, analyse et résolution de problèmes concrets.
- *La reconnaissance de la parole, de l'écriture, et des visages.*
- *La robotique :* La première génération de robot, est capable d'exécuter une série de mouvements préenregistrés. La deuxième génération est dotée de moyens de perception visuelle qui permettent au robot de prendre certaines décisions. Un robot de la troisième génération (objet des recherches actuelles) doit acquérir une plus grande autonomie comme se déplacer dans un environnement inconnu [12].
- *Les réseaux neuronaux :* Un modèle rudimentaire du cerveau humain, où chaque cellule neuronale est décrite comme une fonction possédant une sortie et dont les entrées sont reliées à d'autres neurones.

3. La Vision chez l'Homme

Nous ne mesurons souvent pas la difficulté de reproduire artificiellement les performances de la vision humaine. Si l'homme sait naturellement séparer et reconnaître des objets dans une image, c'est grâce à des connaissances de haut niveau. En effet, dès un âge très précoce, l'humain est déjà un expert en ce qui concerne la compréhension des objets et de la scène [11].

Ce qui rend le problème de la vision par ordinateur aussi difficile à simuler, est la transcription des pixels de l'image en des concepts mathématiques abstraits, ce qui défie les définitions philosophiques de base de ce qu'est un concept, ou de ce qui constitue un objet [13]. Selon les chercheurs en sciences cognitives, le cerveau humain consacre entre 40-70% de sa capacité au traitement des signaux reçus par l'œil [13].

Pour prétendre comprendre la nature du problème, on juge important de mentionner brièvement la notion de «vision », et plus particulièrement du mécanisme de la vision chez l'homme. La vision selon *Jitendra Malik* peut être étudiée selon trois aspects [8]:

- *La perception* : concerne l'étude des lois de la vision ; prédit ce que l'humain peut percevoir dans une image.
- *La neuroscience* : permet de comprendre les mécanismes de la rétine et du cerveau, ce qui peut être une source d'inspiration pour le développement de nouvelles approches. (ex : les réseaux neuronaux).
- *La fonction* : décrit les lois de l'optique combinées avec les statistiques du monde réel.

La correspondance entre l'humain et la machine est plus significative au niveau de l'étude de la *fonction.* Mais étant donné que les résultats seront de toute façon interprétés par l'homme, il est impératif que ce processus reste consistant avec le mécanisme de la perception humaine ; d'où la nécessité de s'inspirer d'autres sciences [8].

La perception visuelle, c'est à dire la sensation consciente de voir, s'appuie sur un organe récepteur, l'œil, dont les cellules de la rétine transmettent aux aires corticales du cerveau les informations perçues [14].

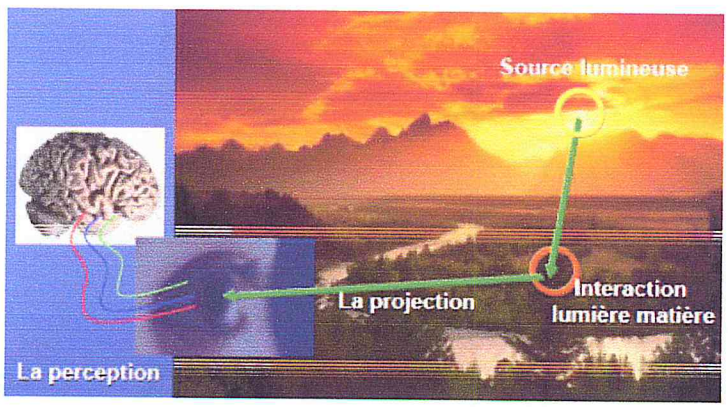


Figure 3 : Processus de la vision chez l'homme [15].

On constate alors que la vision et la perception sont deux phénomènes distincts, l'un s'opère grâce à l'œil, et l'autre au niveau du cerveau. Une correspondance entre l'homme et la machine est ainsi faite (cf. Figure 4).

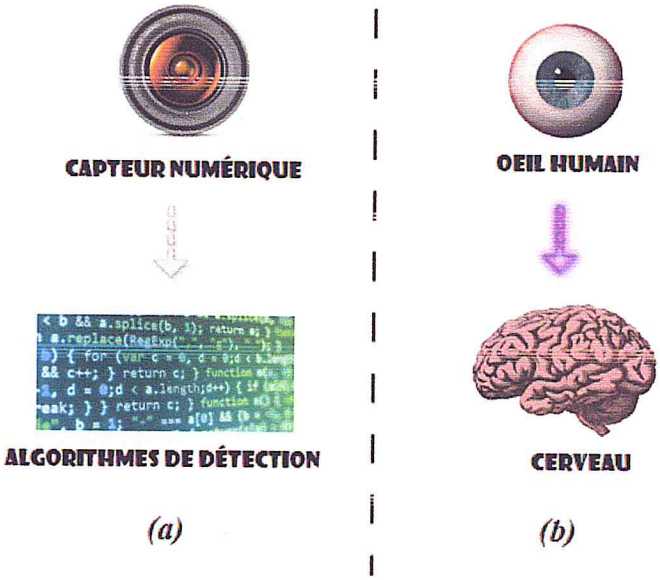


Figure 4: Correspondance du mécanisme de vision entre la machine (a) et l'homme (b).

4. Représentation des Objets

On peut compter aujourd'hui deux grands groupes de théories qui divergent sur le format de la représentation d'objets [16]. Dans le premier groupe, pour simuler les aptitudes humaines de détection d'objets, les chercheurs se sont basés sur l'hypothèse suivante: *la représentation d'un objet est conçue comme un ensemble de caractéristiques (invariants) de l'objet qui sont indépendantes des vues de ce même objet* [16]. Dans ce sens, un modèle géométrique de l'objet devait être réalisé (ensemble de formules mathématiques) et servait comme image d'apprentissage pour la détection (cf. Figure 5). L'une des approches les plus intéressantes de l'époque est celle de Biederman : «Reconnaissance Par les Composantes» (RPC) [16] qui consiste à représenter l'objet en le décomposant en des structures (primitives: *geons*) selon un schéma proposé par de Marr et Nishihara [16].

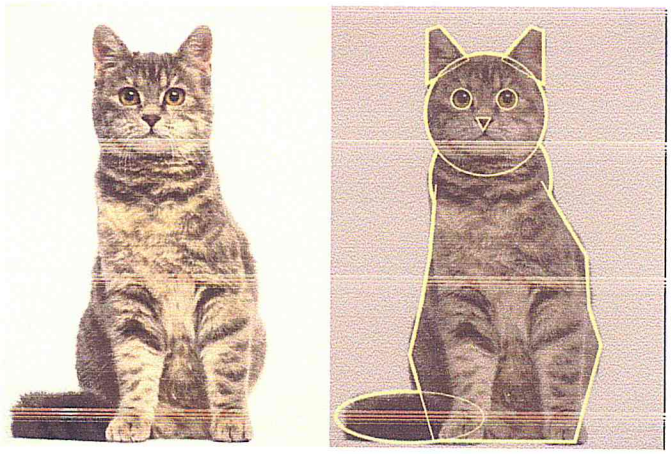


Figure 5 : Description structurelle de l'objet « chat » selon le premier groupe de théories

L'inconvénient de cette première façon de représentation des objets est qu'il est très difficile de détecter les invariants (cf. Figure 6).

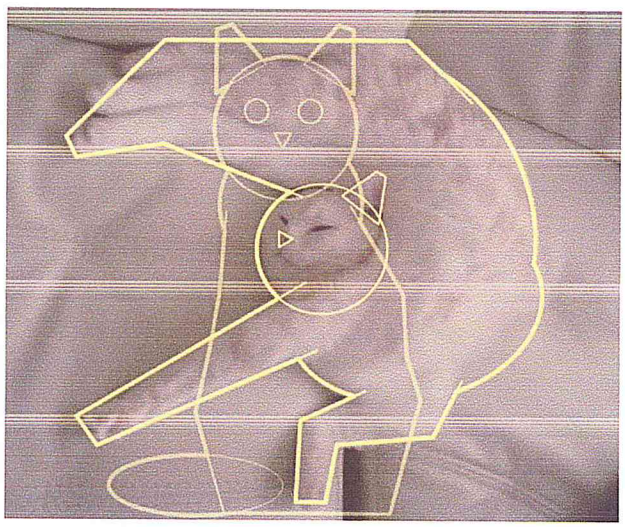


Figure 6 : L'inconvénient principal du premier groupe de théories : non robustesse face à l'invariabilité d'apparence.

Le second groupe de ces théories se base sur le paradigme suivant : *la représentation d'un objet est liée à des vues spécifiques à l'objet : n'importe quelle autre vue de l'objet pourra être déduite à l'aide de ces vues* [16]. Les modèles de ce type de représentation considère une vue comme une collections de caractéristiques : informations 2D, information 3D,... (cf. Figure 7). La reconnaissance est exprimée comme une fonction des images déjà vues. Plusieurs modèles computationnels ont été utilisés notamment par Edelman et Poggio [16].

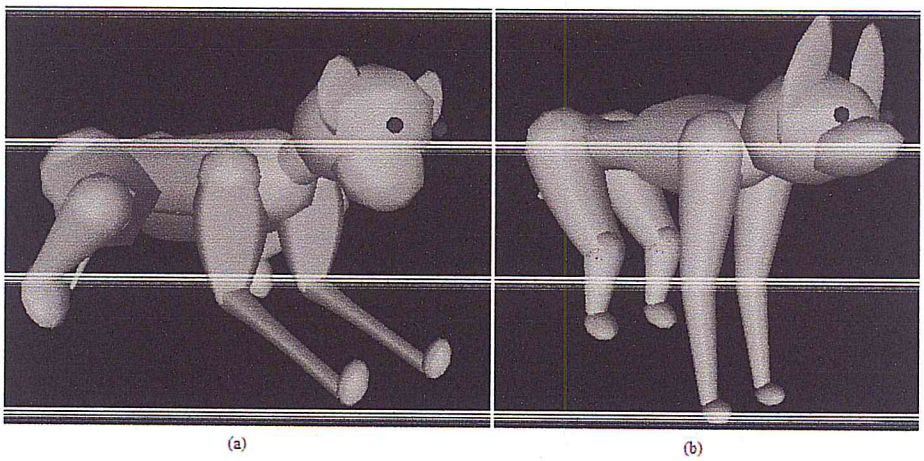


Figure 7 : Les modèles selon les travaux de Edelman : (a) Représentation d'un singe (b) représentation d'un chien

Pour remédier à ces inconvénients, de nouvelles approches ont fait surface selon la théorie du deuxième groupe [16]. Ces approches ne modélisent plus les objets par des modèles de type géométrique ou fondés sur des structures d'invariants, mais par des images réelles représentant l'objet. Pour cela, une base de données d'images assez conséquente doit être mise en place. L'ensemble des vues possibles (apparences de l'objet) est échantillonné. Ainsi, un objet est donc représenté par une connaissance extraite d'une collection d'images ; la reconnaissance/détection est basée sur l'appariement d'une nouvelle image de l'objet avec cette dernière connaissance suivant des techniques de classification [11].

Figure 8 : Collection d'images de l'objet « chat » dans la base de données d'images ImageNet organisée hiérarchiquement selon le thésaurus Wordnet.

5. Détection d'objets dans une image

En vision par ordinateur, on désigne par détection d'objet (ou classification d'objet) une méthode permettant de détecter la présence d'une instance (reconnaissance d'objet) ou d'une classe d'objets dans une image numérique. Une attention particulière est portée à la détection de visages et la détection de personnes. Ces méthodes font souvent appel à l'apprentissage supervisé et ont des applications dans de multiples domaines, tels la recherche d'image par le contenu ou la vidéo surveillance [4].

Le principal problème lié à la détection d'un objet est sa variabilité d'apparence dans le monde réel. En effet, l'objet peut prendre différentes formes sous différents angles. Il peut être sensible à la lumière, ou à l'échelle, l'objet peut paraitre partiellement occulté dans l'image…etc. [4]. Tous ces problèmes qui sont triviaux à résoudre pour le cerveau humain, sont loin d'être évidents pour la machine.

L'observation d'un objet peut être très différente selon le domaine d'activité dans lequel nous nous situons. De même, le motif à reconnaître et à classer peut-être simple ou complexe, et les outils développés sont adaptés à chaque utilisation. Par exemple, un cube sera plus facile à reconnaître qu'un visage puisqu'il a des propriétés physiques constantes dans le temps et l'espace, alors qu'un visage est sujet à beaucoup plus de changements [10].

Notion de région d'intérêt

Le résultat de la détection d'objet dans l'image est illustré à l'aide d'un rectangle autour de l'objet (cf. Figure 9), qu'on appelle aussi région d'intérêt (ROI). On parle de *vrai positif* quand une instance de l'objet est correctement détectée. Un *faux positif* se produit quand une instance négative est confondue avec l'objet d'intérêt : fausse alerte. Un *faux négatif* est généré si une instance de l'objet n'est pas détectée : échec de détection [17].

Un bon algorithme de détection devra maximiser le taux de *vrais positifs*, et minimiser le taux de *faux positifs*.

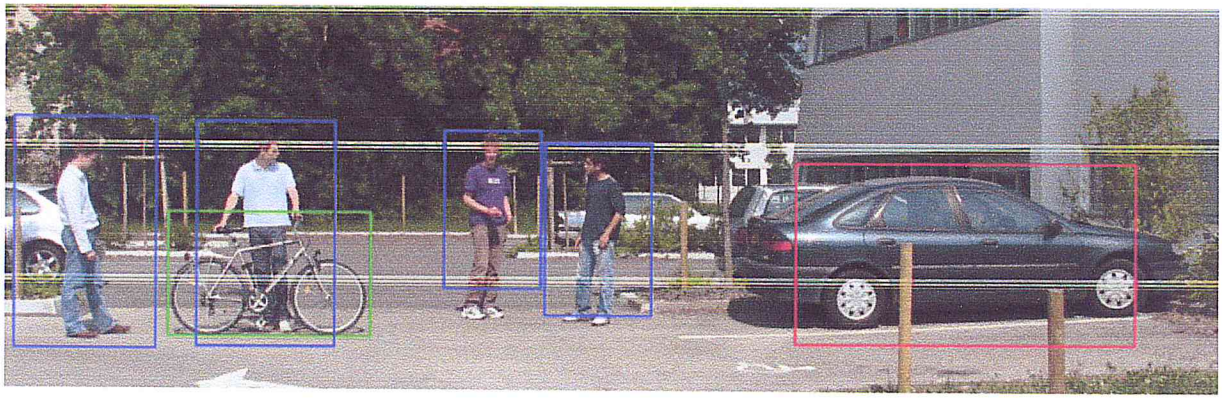


Figure 9 : Résultat final de l'application d'un algorithme de détection d'objets [4].

6. Conclusion

La reconnaissance d'objets est un problème épineux se plaçant au niveau supérieur dans la hiérarchie des tâches de vision, et constitue la partie computationnelle la plus difficile [4]. La mise au point d'algorithmes de détection de haut niveau, où chaque région est un objet sémantique ; est encore un des thèmes de recherche les plus actifs dans la discipline de la vision par ordinateur et de l'intelligence artificielle [4]. Dans le chapitre qui suit, nous présentons différentes approches permettant de réaliser un système de détection d'objets. Il ne s'agit pas de fournir une description exhaustive de l'état de l'art, ce qui ne serait être réalisable tant les méthodes sont diverses et particulièrement sensibles au domaine applicatif, mais plutôt de fournir une vue d'ensemble des systèmes qui permettent de réaliser cette tâche.

Chapitre 2: Techniques de Détection d'Objets

1. Introduction

L'expansion rapide des recherches dans le domaine de la détection/reconnaissance d'objets impose l'usage de systèmes de vision fiables. Les récentes approches de détection fusionnent les résultats obtenus par de robustes détecteurs, et ainsi, éliminent le bruit et résolvent le problème d'occlusion [18]. Pourtant, la détection d'objets reste un problème épineux à traiter, dû aussi bien à la variété des conditions d'acquisition, qu'au changement d'échelle, de localisation, d'orientation, et aux diverses poses que les objets peuvent adopter [18].

Le but d'un détecteur d'objets efficace, est la détermination précise de la localisation de l'objet dans l'image, ainsi que de son ampleur et de sa forme. Pour arriver à ce résultat, plusieurs défis devront être surmontés [18]. Ces défis sont principalement associés à la diversité de l'orientation et des poses que les objets peuvent adopter. Les conditions d'illumination, le mouvement de la caméra, les spécifications de l'appareil d'acquisition, sont autant de paramètres qui doivent être pris en ligne de considération.

2. Les approches de détection

On peut classifier les techniques de détection d'objets en quatre grandes familles [18], toutefois, les techniques décrites ci-dessous peuvent appartenir à plus d'une famille :

2.1 Approches basées sur l'extraction des caractéristiques

Les caractéristiques de l'image sont fréquemment utilisées pour représenter les contours des objets. Le but de cette famille d'approches est de développer des techniques efficaces pour la localisation de caractéristiques structurelles qui seront utilisées dans la procédure de détection. Ces approches sont particulièrement efficaces pour des applications où les conditions de pose, la prise de vue, ou de l'éclairage varient [18].

De nombreux travaux ont exploité les caractéristiques pour la détection des objets. Nous pouvons citer ci-dessous quelques uns parmi les plus importants :

2.1.1 Détection robuste d'humains dans les images thermiques

Davis & Sharma [19] ont proposé une méthode basée sur le contour pour la détection d'humains dans les images thermiques. Au début, les régions locales d'intérêt sont déterminées en faisant la soustraction du fond (Statistical background-substraction). L'information du gradient de chaque région est ensuite utilisée pour obtenir une carte de saillance. Des opérations morphologiques sont aussi employées pour compléter les contours cassés, à partir desquels les silhouettes sont formées. Pour ce type d'image, et pour l'application particulière de détection d'humains dans les images thermique, *Davis & Sharma* affirment que leur méthode donne de bien meilleurs résultats que les approches standards [19].

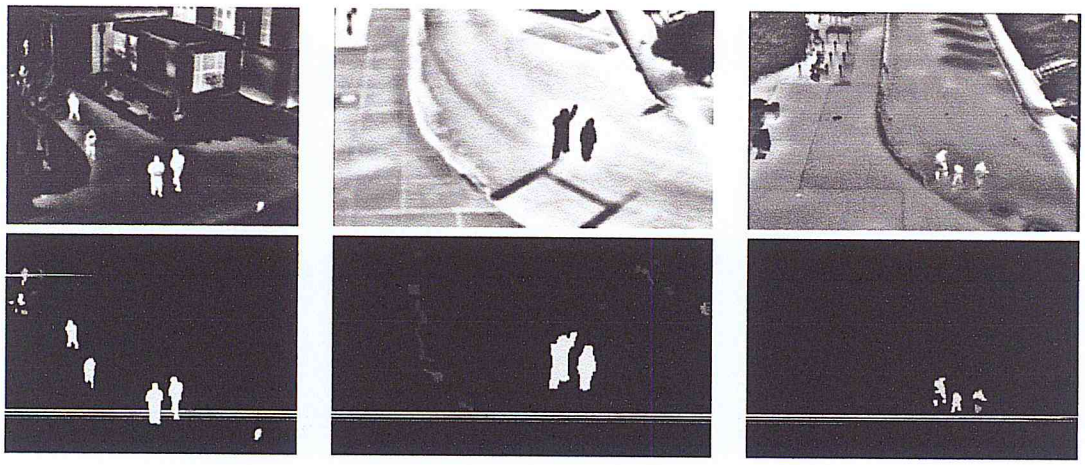


Figure 10: Différents résultats de la technique de Davis & Sharma pour la détection d'humains dans les images thermiques [19].

2.1.2 Détection robuste et automatique de véhicules dans les images aériennes

S.Salhi propose deux chaînes de traitements pour la détection de véhicules dans les images aériennes [20]. Dans le cadre de la première chaîne de traitement, le détecteur *SIFT* (*Scale Invariant Feature Transform)* développé par D.Lowe en 1999, est utilisé pour identifier les pixels de l'image susceptibles d'appartenir aux véhicules, et ainsi contourner le problème d'une segmentation onéreuse. L'usage de *SIFT* permet d'écarter les larges régions homogènes caractéristiques des images aériennes, et qui souvent sont considérées comme non pertinentes. Les points-clefs produits par le détecteur *SIFT* apparaissent en abondance sur les véhicules [20].

Après extraction des caractéristiques (les points clés), un Séparateur à Vaste Marge (*SVM*) entrainé a été appliqué afin de prédire la classe associée à chaque point-clef. Les *SVM* permettent de s'affranchir des contraintes liées à la variété de l'apparence des véhicules. Combiné au pouvoir hautement discriminant du descripteur *SIFT*, il est possible de classifier les points-clefs avec un très faible taux d'erreurs [20].

Figure 11 : Résultat de la première approche : les points clés en rouges sont abondants sur les véhicules [20].

Bien que cette approche donne de très bons résultats, elle se base essentiellement sur la capacité du descripteur. En effet, le descripteur *SIFT* n'est pas approprié pour extraire l'information locale présente dans les véhicules sombres.

La deuxième chaîne de traitement proposée par *S. Salhi* se décrit comme suit : les images sont divisées en petites régions homogènes en couleur, appelées Superpixels. L'algorithme de sur-segmentation utilisé est le *SLIC -Simple Linear Iterative Clustering-* (cf. Figure 12). Les barycentres de ces superpixels sont ensuite utilisés comme points-clefs, à partir desquels sont extraits à nouveau des descripteurs *SIFT*. L'utilisation des barycentres des superpixels comme des points-clefs a pour effet d'augmenter radicalement le nombre de points-clefs que les *SVM* doivent classifier. Pour remédier à ce problème, *S.Salhi* utilise un algorithme de détection de régions saillantes (permet d'écarter les régions non pertinentes). Cet algorithme extrait de

l'image les régions possédant une grande densité de contours. Cette approche est motivée par les travaux de *Hinz* et *Baumgartner* qui mettent en évidence un lien entre la présence de véhicules et une grande densité de contours [20].

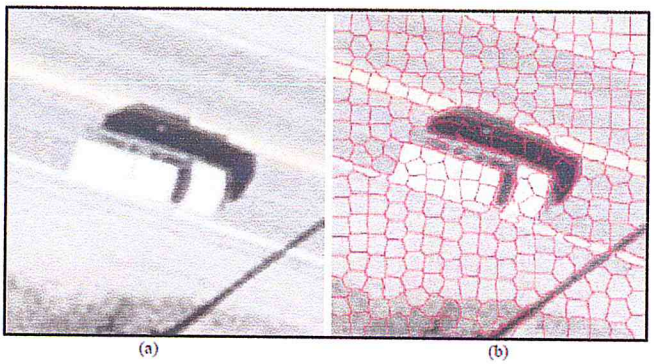


Figure 12 : Sur-segmentation obtenue à l'aide de l'algorithme SLIC (a) vue rapprochée d'un véhicule sur une route, (b) résultat de la sur-segmentation, surface moyenne couverte par les superpixels (en moyenne 45 pixels) [20].

Les résultats de la classification à l'aide de la deuxième chaine de traitement sont présentés dans la figure 13. La majorité des véhicules sont adéquatement détectés. De fausses alarmes sont présentes néanmoins dans le reste de l'image. Les emplacements de stationnement non occupés par des véhicules ne produisent pas de fausses alarmes.

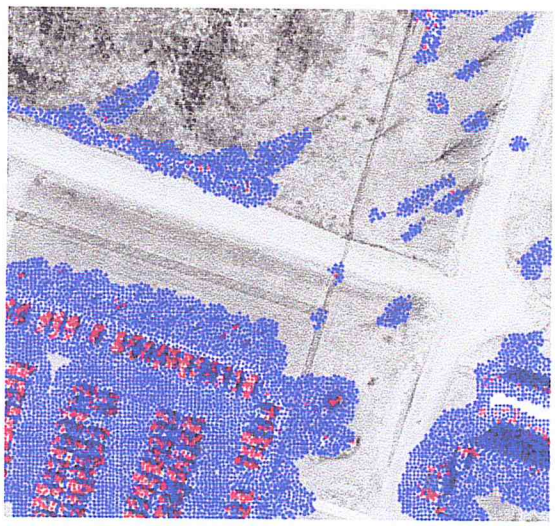


Figure 13: Résultats de la classification obtenus par les SVM à l'aide de la deuxième chaîne de traitement [20].

2.1.3 Détection d'humains dans les images et les vidéos

Dalal & Triggs présentent en 2005 une approche qui conjugue le descripteur *HOG* (*Histogrammes Orientés Gradients*), avec les *SVM* comme classificateur. Leur méthode s'avérera très efficace pour la détection d'humains [2].

L'idée derrière le descripteur *HOG* est que l'apparence et la forme locale d'un objet dans une image peuvent être décrites par la distribution de l'intensité du gradient ou la direction des contours. Ceci peut être réalisé en divisant l'image en des régions adjacentes de petite taille (les cellules), et en calculant pour chaque cellule l'histogramme des directions du gradient (orientations des contours) pour les pixels à l'intérieur de la cellule. La combinaison des histogrammes forme alors le descripteur *HOG* [4] [2].

Pour de meilleurs résultats, les histogrammes locaux sont normalisés en contraste, en calculant une mesure de l'intensité sur des zones plus larges que les cellules, appelées des blocs, et en utilisant cette valeur pour normaliser toutes les cellules du bloc. Cette normalisation permet une meilleure résistance aux changements d'illuminations et aux ombres [4] [2].

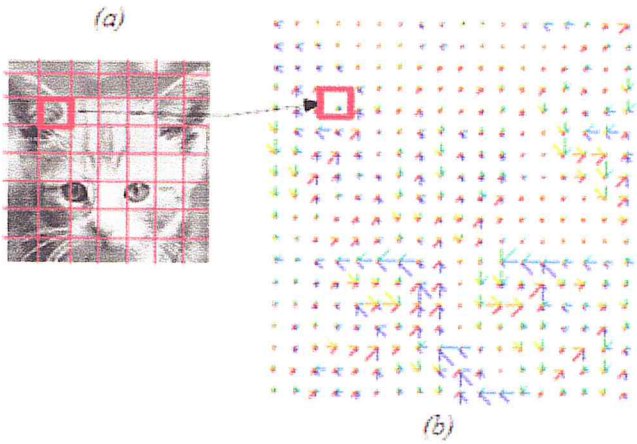


Figure 14: Extraction des gradients orientés: (a) découpage de l'image en cellules, (b) Gradients orientés de la cellule pour les trois composantes couleurs [4].

Le descripteur *HOG* du motif à détecter, sera ensuite utilisé comme fenêtre glissante pour la détection ou non de la présence de l'instance (cf. Figure 15).

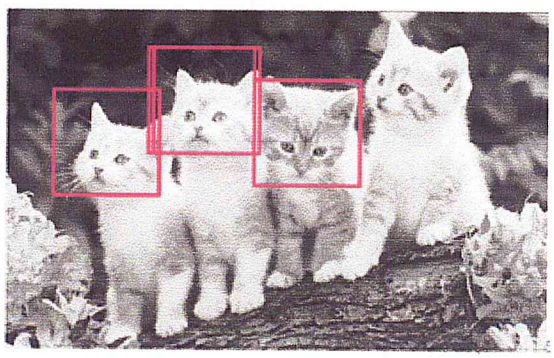


Figure 15 : Résultat après application de la technique en utilisant la tête du petit chat (figure 14 (a)) comme objet de référence [4].

Pour aspirer à une plus grande robustesse de détection, *Dalal & Triggs* proposent de renforcer leur approche en utilisant le pouvoir hautement discriminatif des *SVM*. Une phase d'apprentissage doit être implémentée : une base de données d'images assez conséquente contenant des exemples positifs : des images représentant le même objet, ainsi que des exemples négatifs : des images représentant d'autres objets. Après la phase d'apprentissage, l'algorithme pourra déterminer un modèle visuel de ce qui différencie un exemple positif d'un exemple négatif [4]. Quelques résultats de la chaine *HOG+ SVM* sont illustrés dans la figure 16.

Figure 16: Détection d'humains sur une scène urbaine en utilisant HOG+SVM.

2.2 Approches basées sur l'apparence

Dans ces méthodes, les modèles sont appris à partir d'une collection d'images représentatives de l'objet. Ces modèles sont ensuite utilisés pour la détection. Les méthodes basées sur l'apparence s'appuient sur des techniques d'analyse statistique pour trouver les caractéristiques de l'objet à détecter [18]. Les objets apparaissent différemment sous des conditions variables: changement de lumière, de couleur, de point de vue, de forme, de taille… Un seul modèle par objet ne permet pas d'assurer une bonne détection. Nous pouvons citer ci-dessous quelques méthodes appartenant à cette famille :

2.2.1 La correspondance de contour pour la reconnaissance d'objet (Edge Matching)

Y. Ramadevi et al ont proposé l'utilisation de techniques de détection de contours, telles que les détecteurs de contours *Canny, Sobel, Prewitt, Laplacien of Gaussien (LoG)* pour assurer la détection. Le changement de luminosité et de couleurs n'a souvent pas d'impact sur la détection de contours [21]. La stratégie de la méthode est comme suit :

1. Détection des contours dans le modèle et l'image.
2. Comparaison des contours de l'image pour trouver le modèle.
3. Considération du champ possible de la position du modèle.

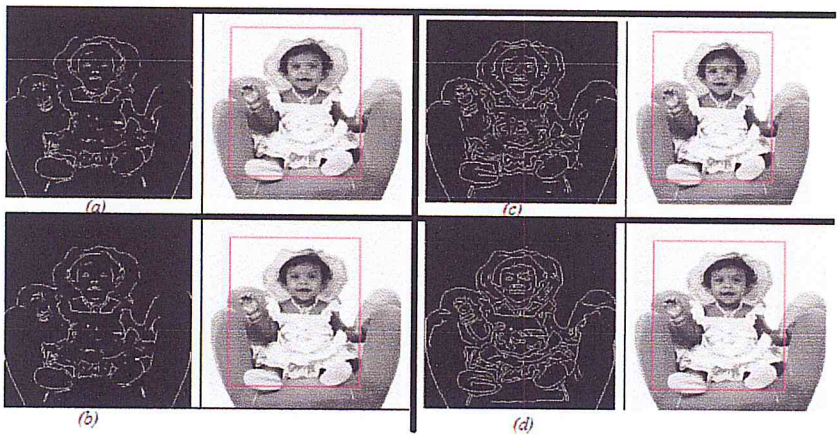


Figure 17 : résultat de détection après application de différents algorithmes de détection de contours : (a) Sobel, (b) Perwitt, (c) LoG, (d) Canny [20].

2.2.2 Les contours actifs « Snakes »

A. Fekir et al [22] proposent une méthode de détection et de suivi d'un objet dans une séquence d'images basée sur le contour actif (en anglais: snakes, tiennent leur nom de leur aptitude à se déformer comme des serpents). Les contours actifs sont définis par une courbe paramétrique pouvant être fermée ou non. Le contour actif est formé d'une série de points mobiles et répartis sur une courbe en deux dimensions placée dans la zone d'intérêt de l'image ou autour d'un objet [22].

Un snake consiste à placer aux alentours de la forme à détecter une ligne initiale de contour. Cette ligne va se déformer progressivement selon l'action de plusieurs forces qui vont l'attirer ou la repousser de la forme [22]. Ces forces sont représentées par trois énergies associées au snake :

- Une énergie interne, due uniquement à la forme du contour,
- Une énergie potentielle (externe) imposée par l'image, (qui attire la ligne du snake vers les contours réels présents sur l'image),
- Une énergie de contexte qui exprime certaines contraintes supplémentaires.

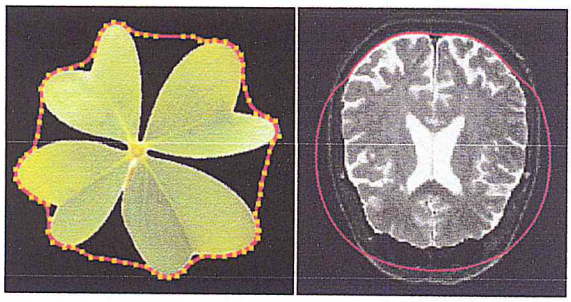


Figure 18: Illustration de la méthode des contours actifs pour la détection d'objets.

2.2.3 Segmentation avec Mean Shift

Zhu et al. [18] proposent une approche basée sur la forme: l'application de *Mean Shift* pour la segmentation, est ensuite suivie d'un raffinement manuel des frontières pour générer les contours de la carte de distance (cf. Figure 19). Cet algorithme créé une carte de confiance dans la nouvelle image basée sur l'histogramme en couleur de l'objet dans l'image originale (la densité de probabilité dans la nouvelle image, donnant pour chaque pixel de la nouvelle image une probabilité que la couleur de ce pixel appartiennent à l'image d'origine), et utilise ensuite *Mean Shift* pour trouver les sommets de confiance de la carte qui sont proches de la position de l'objet dans l'image et ainsi arriver à la détection. La faiblesse de cette approche concerne les dérives lors du suivi de l'objet, principalement lorsque la couleur de l'objet est similaire à l'arrière-plan [23].

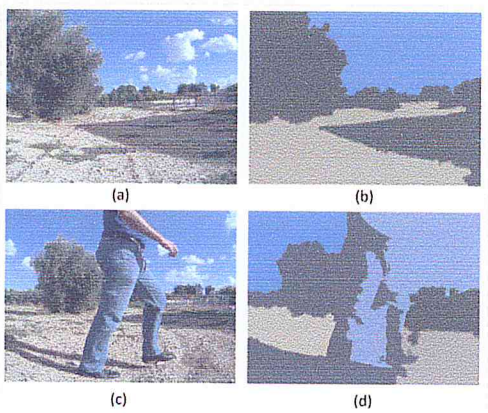


Figure 19: Segmentation avec Mean Shift :(a), (c) images originales. (b), (d) Images segmentées [24].

2.2.4 La saillance pour la détection d'objets

Jie Feng et al proposent pour la détection d'objets d'intérêt dans une image, l'utilisation de la notion de saillance (les objets qui capturent l'attention) en s'appuyant

sur le paradigme de la fenêtre glissante. La fenêtre donne la mesure de la probabilité que la région qu'elle englobe contienne un objet saillant (d'intérêt). La clé de la réussite de ce paradigme est que la mesure de la probabilité ne prend pas en considération la classe de l'objet, et est ainsi particulièrement robuste aux changements dans l'arrière-plan [25].

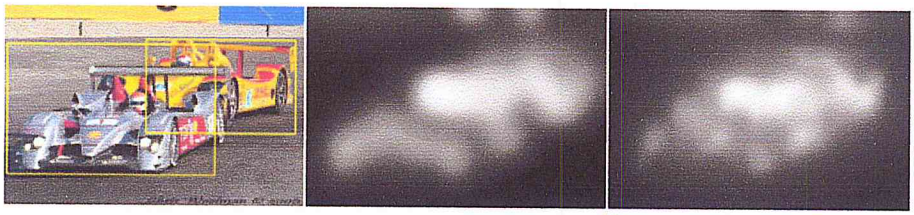


Figure 20: (a) Détection d'objets saillants, (b) 2 cartes de saillance.

Jie Feng et al, ont constaté que, malgré de grandes différences entre les objets et les arrière-plans, une propriété commune apparait: non seulement l'objet saillant ressort fortement de son voisinage immédiat, mais aussi des autres objets de l'image. En d'autres termes, il est difficile de représenter un objet saillant en utilisant le reste de l'image. Ce qui les a motivés à utiliser le paradigme de la fenêtre glissante (ne dépend pas de l'hypothèse que l'arrière plan est une zone homogène, ou que les contours de l'objet sont forts). La méthode se rapproche des facultés de l'homme à détecter un objet saillant [25].

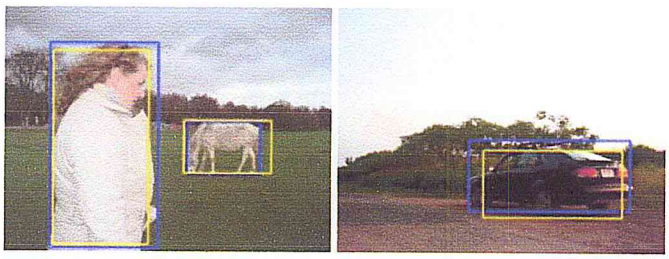


Figure 21: Résultats de détection d'objets saillants avec la méthode de Jie Feng et al [25].

2.3 Approches basées sur des connaissances a priori

Ces méthodes se basent sur l'acquisition de l'information qui constitue l'objet d'intérêt, leur principal avantage est de réduire de façon significative le taux de faux positifs [18].

Parmi les méthodes appartenant à cette famille, nous pouvons citer la méthode de *Viola et Jones* qui proposent une approche pour la détection de visages. Celle-ci consiste à considérer une connaissance a priori sur les mouvements et l'apparence d'une personne [1].

La méthode consiste à parcourir l'ensemble de l'image en calculant un certain nombre de caractéristiques (les caractéristiques pseudo-Haar). Pour réduire le temps de calcul de ces caractéristiques, Viola et Jones introduisent la notion d'images intégrales, qui permet de réduire considérablement le nombre de caractéristiques à prendre en compte. *Adaboost (Adaptative Boosting)* est ensuite utilisé pour la sélection des caractéristiques utiles, en interprétant les caractéristiques comme des classificateurs (le classificateur consiste en une combinaison linéaire des caractéristiques sélectionnées). Enfin, la méthode propose une architecture pour combiner les classificateurs boostés en un processus en cascade, ce qui apporte un net gain en temps de détection [1].

La méthode, en tant que technique d'apprentissage supervisé, est divisée en deux étapes : (1) une étape d'apprentissage du classificateur qui nécessite quelques centaines à plusieurs milliers d'exemples de l'objet que l'on souhaite détecter pour entraîner le classificateur ; et (2) une étape de détection par application de ce classificateur à des images inconnues. La détection de la présence éventuelle de l'objet dans une image est faite en parcourant celle-ci de manière exhaustive, à toutes les positions et dans toutes les tailles possibles.

Inventée à l'origine pour détecter des visages, la méthode peut également être utilisée pour détecter d'autres types d'objets comme des voitures ou des avions. *Viola* et *Jones* affirment que leur méthode est au moins 15 fois plus rapide en temps de calcul que les autres techniques existantes [1].

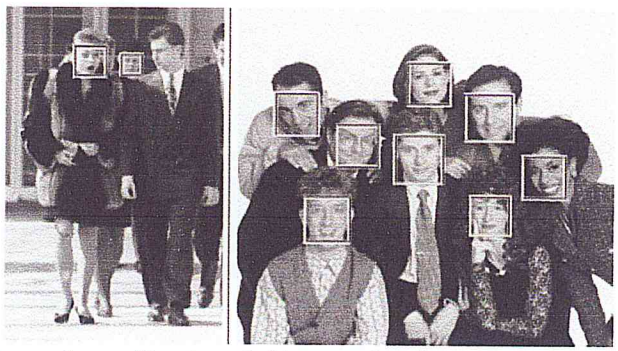


Figure 22 : Résultats après application de l'approche de Viola et Jones pour la détection de visages [1].

2.4 Méthodes de correspondance avec un modèle

La correspondance avec un modèle est une technique utilisée pour trouver de petites parties d'une image qui correspondent avec une image modèle. Ces méthodes utilisent des motifs standards pour décrire l'objet en globalité ou comme parties séparées. Ils peuvent être une alternative quand le modèle ne présente pas de fortes caractéristiques à extraire [18].

2.4.1 Détection d'objets en temps réel pour les véhicules intelligents

Gavrila et Philomin proposent un schéma de détection d'humains et de panneaux de signalisation pour les véhicules intelligents (utilisant une caméra placée au niveau du rétroviseur) [26]. Leur approche aspire à assister en temps réel le conducteur pour une conduite plus sécurisée. La méthode utilise une hiérarchie de modèles pour capturer la variété des formes de l'objet à détecter (cf. Figure 23). De bonnes hiérarchies peuvent êtres générées hors ligne en utilisant des techniques d'optimisation stochastique *(i.e. recuit simulé).* Quant à la phase de correspondance, elle est assurée en utilisant les cartes de distance *(Distance Transform)* [26].

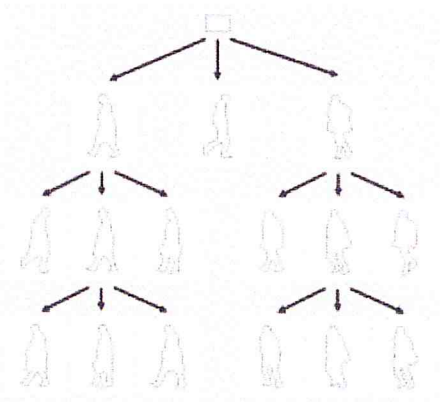
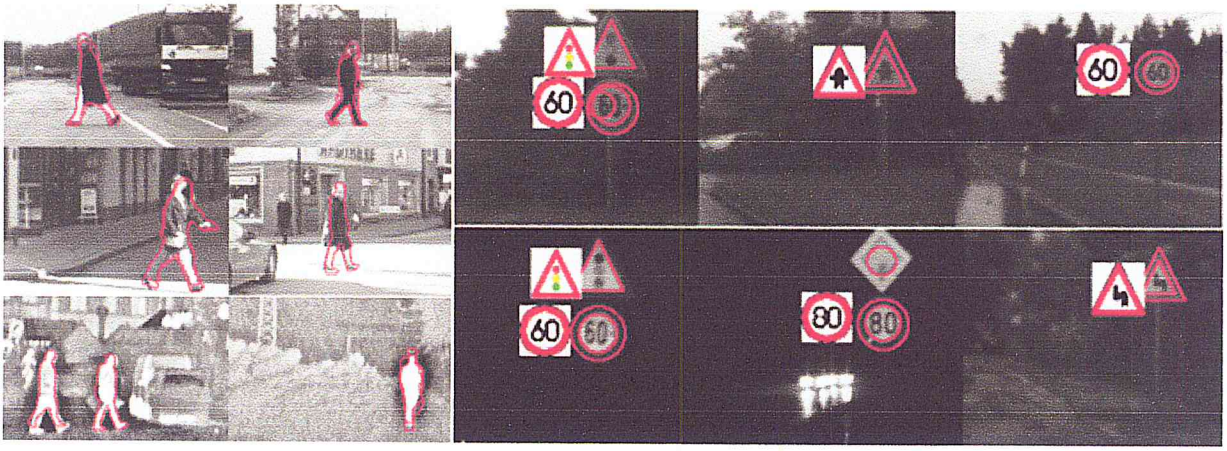


Figure 23: Hiérarchie de formes pour les piétons [26].

L'approche décrite dans [26] est basée sur la forme pour assurer la détection. Les modèles n'ont pas besoin d'être établis explicitement, ce qui est un avantage quand il s'agit d'objets non rigides tels que les piétons, ce qui la rend particulièrement efficace pour la détection d'objets avec des formes arbitraires [26].



(a) (b)

Figure 24: Résultats de la méthode de Gavrila et Philomin : (a) détection de piétons (b) panneaux de signalisation (détection et reconnaissance) [26].

2.4.2 Le détecteur *Pfinder* pour le suivi en temps réel du corps humain

Wren et al. décrivent le détecteur « *Pfinder* » pour la détection et le suivi de personnes, comme une approche basée sur la correspondance avec un modèle utilisant la soustraction de l'arrière-plan. Leur approche requière toutefois une analyse spécifique du domaine de la scène [18].

Le modèle de l'arrière-plan utilise une distribution gaussienne dans l'espace colorimétrique *YUV* pour chaque pixel, et est continuellement mis à jour. La personne est modélisée en utilisant plusieurs gouttes (*blobs* en anglais : une région de l'image dans laquelle les propriétés sont approximativement constantes, comme la couleur, l'illumination) utilisant des paramètres de couleur et d'espace et la distribution gaussienne correspondante [27]. Le changement dynamique de la goutte implique une estimation dynamique des paramètres de l'espace en utilisant le filtre de Kalman. Ensuite, pour chaque pixel, la méthode évalue sa probabilité d'appartenir à la goutte ou à l'arrière-plan. Chaque pixel est assigné soit à la goutte, soit à l'arrière-plan en utilisant le maximum a posteriori (MAP), suivi par de simples opérations morphologiques. Après cette étape, un modèle statistique pour la goutte et la texture de l'arrière-plan sont mis à jour. Enfin, le modèle de la personne (la goutte) est initialisé en utilisant une étape de détection de contour qui sert à localiser la tête, les mains et les pieds. Ce système est conçu pour la détection d'une seule personne [27].

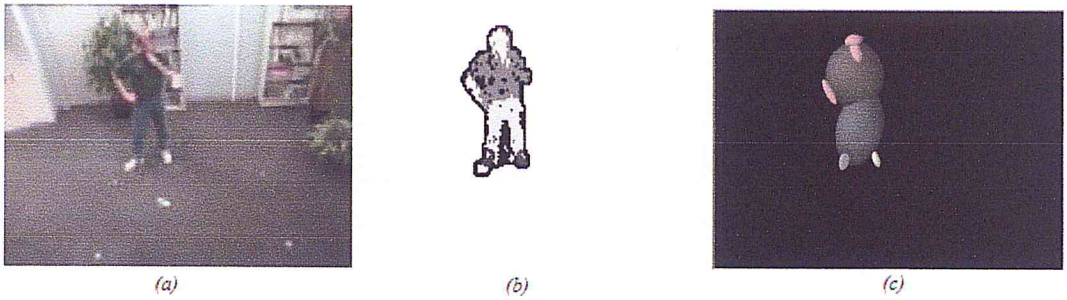


Figure 25: (a) image d'entrée, (b) segmentation, (c) une représentation en 2D de la goutte [27].

2.4.3 L'approche de Castillo et Chang pour la détection des silhouettes

Dans [28] *Castillo et Chang* présentent une approche de détection de personnes pour des applications en robotique. L'approche intègre une correspondance avec un modèle, conjuguée avec un classificateur pour la détection des composants de la silhouette (ex : torse, jambes). Cette approche s'avère robuste face à la variété de poses que la personne peut prendre. Castillo et Chang détectent en premier les contours, appliquent la carte de distance, et enfin procèdent à la correspondance avec des modèles de différentes échelles, et éliminent ainsi les régions où la silhouette n'est pas présente. La détection est ensuite confirmée à l'aide d'un classificateur *SVM* (Machine à vecteur de support) entrainé [28].

Figure 26: résultat de l'approche de Castillo et Chang [28].

2.4.4 Détection d'objet avec les réseaux neuronaux

Dumitru et al proposent dans un papier intitulé *« Scalable Object Detection using Deep Neural Networks »* l'utilisation d'un modèle de réseau de neurones basé sur la saillance pour réaliser la détection [29]. La méthode prédit un ensemble de rectangles de délimitation avec un score pour chaque rectangle (la probabilité de présence d'un objet d'intérêt). Le modèle gagnant sera le réseau qui prédira un seul rectangle contenant l'objet. Le modèle gère naturellement un nombre variable d'instances pour chaque classe. La détection en elle-même se fait également en utilisant un DNN (Deep Neural

Network) pour classifier les rectangles de délimitation suivant le score de détection [29].

L'inconvénient de cette approche est sa non robustesse face à des images contenant plusieurs instances du même objet: l'algorithme donnera plusieurs rectangles de détection pour chaque instance [29].

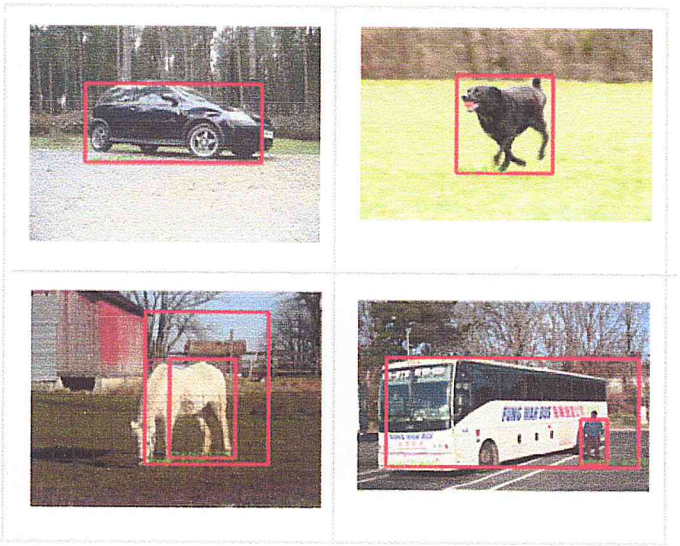


Figure 27: résultats de l'approche de Dumitru et al [29].

3. Conclusion

Nous avons vu dans ce chapitre quatre familles d'approches ainsi que les travaux majeurs s'y inscrivant. Nous nous inspirons pour la réalisation de notre application de deux grands travaux du domaine de la détection d'objets: le travail de Viola & Jones pour la détection des visages, qui introduisent la notion de classifieur en cascade ; ainsi que le travail de Dalal & Triggs, qui utilisent le gradient pour extraire les caractéristiques de l'image. La solution proposée s'inscrit donc comme appartenant à la famille des approches à base d'extraction de caractéristiques.

Chapitre 3 : Conception du Système Proposé

1. Introduction

Suite aux travaux de la littérature, nous nous sommes concentrés sur la proposition d'une approche permettant la détection de plusieurs types d'objets : visages, véhicules, et plaques de signalisation. Pour cela, nous nous sommes orientés vers l'étude des descripteurs HOG (Histogrammes Orientés Gradient) combinés avec un classifieur entrainé suivant un schéma en cascade. Le choix des extracteurs de caractéristiques HOG est motivé d'une part par leur usage fréquent dans les travaux de détection, et d'autre part par leur rapidité de calcul comparés à d'autres extracteurs de caractéristiques tels que Haar ou bien LBP (Local Binary Pattern).

2. Descriptif de l'approche choisie

L'approche que nous proposons repose sur trois étapes :

1. Approvisionnement avec les images d'entrainement : un ensemble d'images positives (contenant l'objet à détecter), et un ensemble d'images négatives quelconques (ne contenant pas l'instance de l'objet).
2. Entrainement du détecteur, consiste en deux opérations s'effectuant séquentiellement:
   - 2.1 Extraction des caractéristiques sur l'ensemble des images d'entrainement,
   - 2.2 Passage de ces caractéristiques dans un algorithme de classification *Adaboost* organisé en cascade.
3. Détection d'objets.

Le schéma qui suit illustre le fonctionnement de notre solution :

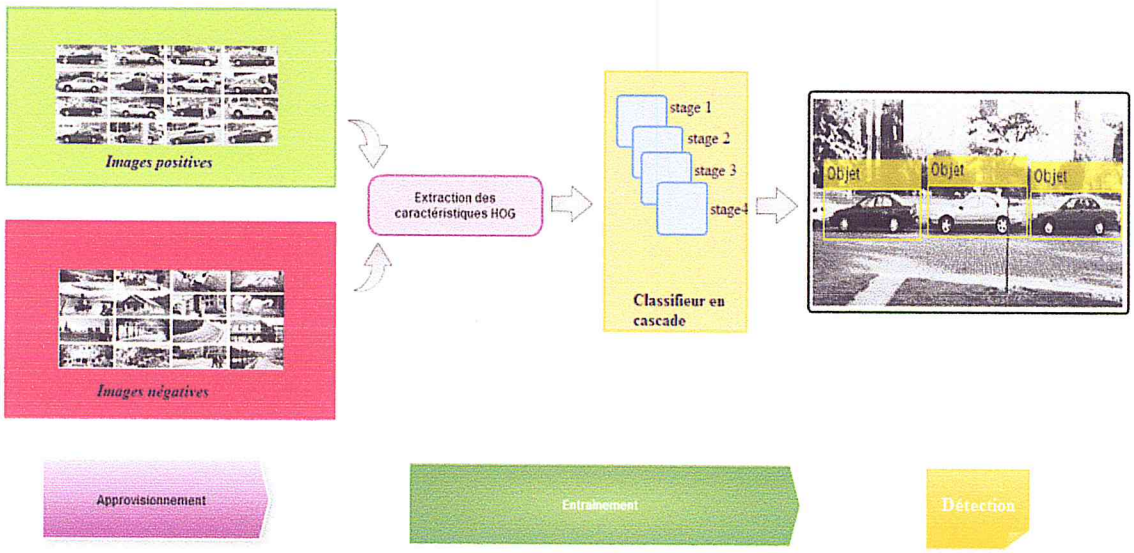


Figure 28 : Schéma de la description globale de l'application pour la détection d'objets.

3. Entrainement du détecteur

Nous allons maintenant nous attarder sur l'étape d'entrainement du détecteur, qui contient les algorithmes de l'approche. L'étape consiste en deux phases:

3.1 Extraction des caractéristiques

L'approche proposée pour la procédure de détection d'objets classe les images en se basant sur la valeur de simples caractéristiques. La principale raison pour l'utilisation d'un système basé sur les caractéristiques au dépend des valeurs des pixels, est que les caractéristiques peuvent être utilisées pour représenter un domaine de connaissance particulier. Aussi, un système basé sur les caractéristiques s'opère beaucoup plus rapidement en temps de calcul qu'un système basé sur les pixels [30].

Une caractéristique est donc un nombre réel qui code les variations du contenu pixellique à une position donnée dans la fenêtre de détection [30].

Afin d'extraire les caractéristiques, une fenêtre glissante sera appliquée sur l'ensemble de l'image. Ces caractéristiques seront les variables d'entrée du classifieur [17]. Les caractéristiques utilisées dans approche sont les histogrammes orientés gradient : HOG.

Les Histogrammes Orientés Gradients

Les HOG ont été proposés pour la première fois par Dalal & Triggs en 2005 pour la détection de personnes [2]. L'idée principale derrière le descripteur HOG est que l'apparence et la forme locale d'un objet dans une image peuvent être décrites par la distribution de l'intensité du gradient ou la direction des contours [2]. Ceci peut être fait en divisant l'image en des régions adjacentes de petite taille, appelées cellules, et en calculant pour chaque cellule l'histogramme des directions du gradient ou des orientations des contours pour les pixels à l'intérieur de cette cellule. La combinaison des histogrammes forme alors le descripteur HOG [2].

Les caractéristiques HOG sont souvent utilisées pour détecter des objets comme des véhicules ou des avions ; ceci est dû à leur particularité de capturer la forme globale d'un objet [17]. Par exemple, dans la figure suivante, on peut clairement reconnaitre la forme d'un vélo à partir des caractéristiques HOG [17]:

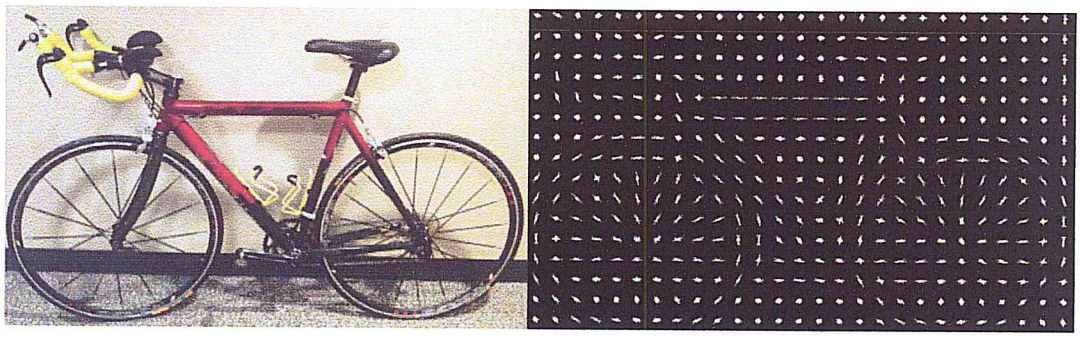


Figure 29 : Caractéristiques HOG d'une image représentant un vélo.

Les étapes de l'algorithme d'un descripteur HOG sont:

a. Calcul du gradient

La première étape de la méthode est le calcul du gradient, la méthode la plus courante pour cela consiste à appliquer un filtre dérivatif 1-D centré, dans les directions horizontales et verticales (cf. Figure 30 *(b)*). Chaque pixel aura donc deux informations : la magnitude et l'orientation du gradient. Dans le cas des images couleurs, le gradient est calculé séparément pour chaque composante, et on retient pour chaque pixel le gradient avec la plus grande magnitude [2].

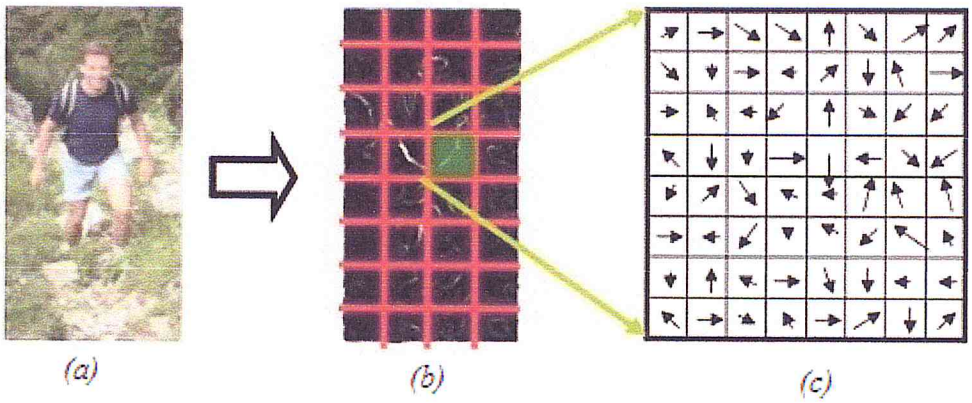


Figure 30: Calcul des gradients pour chaque cellule : (a) image originale, (b) application d'un filtre dérivatif + découpage de l'image en cellules, (c) Les gradients orientés pour chaque pixel

b. Construction de l'histogramme

La seconde étape est la création des histogrammes. Ceci est fait dans des cellules carrées de petite taille (de 4x4 à 12x12 pixels). Chaque pixel de la cellule vote alors pour une classe de l'histogramme, en fonction de l'orientation du gradient à ce point. Le vote du pixel est pondéré par l'intensité du gradient en ce point. Un histogramme donc n'est rien d'autre qu'un graphe où la fréquence de chaque gradient est illustrée (cf. Figure 31 *(b)*) [2].

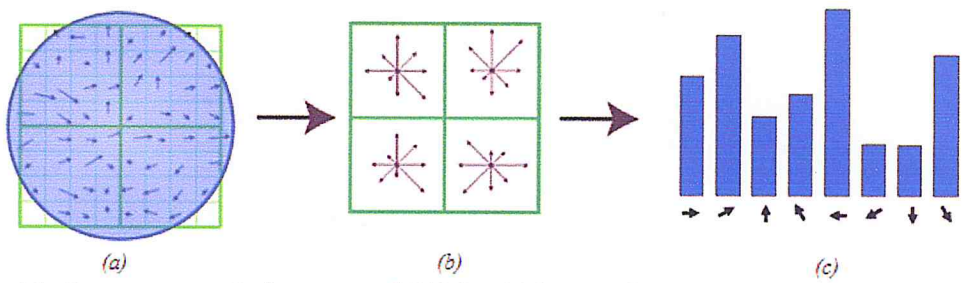


Figure 31: Le processus de formation d'HOG : (a) les gradients orientés pour chaque pixel, (b) construction des histogrammes pour chaque cellule, (c) illustration sur un graphe de l'accumulation des différents histogrammes (le descripteur final) [2].

c. Formation et normalisation des blocs

La normalisation des descripteurs est une étape importante afin d'éviter les disparités dues aux variations d'illumination. Pour cela, Dalal et Triggs regroupent plusieurs cellules dans un *bloc*, qui est l'unité sur laquelle est effectuée la normalisation. Les blocs se recouvrent : une même cellule participe plusieurs fois comme membre de blocs différents. Deux types de géométrie de blocs sont proposés: rectangulaire (R-HOG) ou circulaire (C-HOG). Les expériences faites par les auteurs ont montré qu'une

meilleure performance est obtenue pour des blocs rectangulaires contenant 3x3 cellules de 6x6 pixels chacune. Cette normalisation permet une meilleure résistance aux changements d'illuminations et aux ombres [30].

d. Accumulation des HOG

Enfin, une accumulation des différents histogrammes est effectuée pour former le descripteur final (cf. Figure 31 *(c)*). La figure qui suit résume le processus de création des HOG [2] :

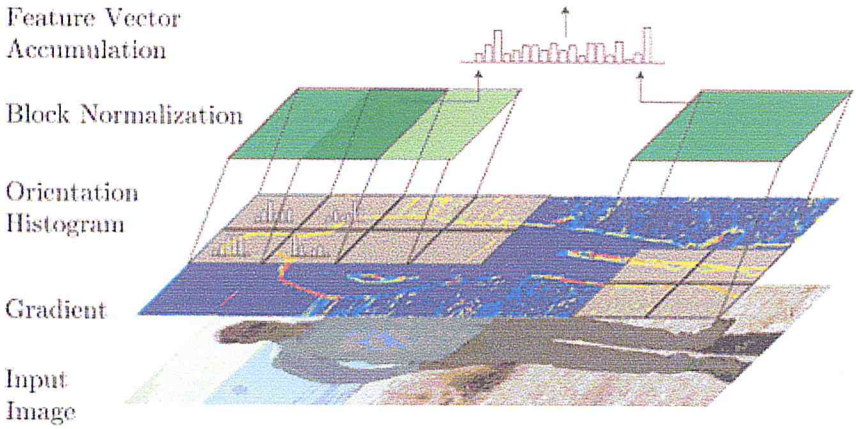


Figure 32: Illustration graphique de l'approche HOG [2].

3.2 Apprentissage : *Adaboost en cascade*

La deuxième opération après obtention des caractéristiques, est l'utilisation de ces dernières pour l'entrainement d'un classificateur et ainsi, résoudre le problème de détection. La détection d'objets d'intérêt peut être vue comme un problème de classification supervisée à 2 classes : présence/ absence de l'objet. L'entrainement d'un classifieur quelconque demande un ensemble d'exemples positifs (présence de l'objet dans l'image), et un autre ensemble d'exemples négatifs (absence de l'objet dans l'image). L'objectif est de faire apprendre à la machine ce qui différencie en termes de caractéristiques la présence ou l'absence de l'instance de l'objet à l'aide des images d'entrainement [17].

Le classifieur qu'on utilisera dans l'approche de détection se nomme *Adaboost* (*Adaptative Boosting).* Ce dernier repose sur la notion d'étages, où chaque étage (stage en anglais) est une application itérative d'un algorithme d'apprentissage naïf appelé *Decision Stump (DS).* Dans un premier temps, *DS* détermine le seuil de classification en essayant de minimiser au maximum le nombre d'échantillons mal classés (cf. Figure 35). Dans un second temps, un boosting est appliqué après chaque étape avec objectif

d'augmenter le poids des échantillons mal classés dans l'étape précédente. Les résultats des classifieurs naïfs *DS* sont ensuite combinés en cascade pour former un seul détecteur complexe. Les travaux de Viola et Jones démontrent que l'approche de classification en cascade, permet d'obtenir un taux élevé de bonnes détections, un faible taux de faux positifs, et une performance rapide, important pour des applications en temps réel ou s'exécutant sur de faibles CPU [30]. Pour comprendre le mode opératoire du classifieur en cascade *Adaboost*, nous introduisons d'abord le fonctionnement de l'algorithme d'apprentissage naïf : *Decision Stump.*

3.2.1. Arbre de décision et Decision Stump

Pour bien comprendre le rôle d'un DS, nous jugeons utile d'introduire la notion d'arbre de décision, sachant qu'un *DS* n'est qu'un cas particulier de l'arbre de décision [31].

Arbre de décision

Pour certains domaines d'application, il est essentiel de produire des classifications compréhensibles par l'utilisateur. Dans les méthodes classiques (hiérarchique, k-means, Kohonen, perceptron multi-couches), contrairement aux arbres de décision, l'information est perdue dans les classes [32].

Un arbre de décision est un algorithme de classification supervisée (les classes sont connues à l'avance) permettant de classer un ensemble d'individus décrits par des variables qualitatives ou quantitatives. L'objectif est de produire des classes de la manière la plus homogène possible [32].

Exemple d'un arbre de décision : Supposons un problème où le classificateur devra séparer un ensemble de points dans l'espace selon deux variables d'entrées (2 types de caractéristiques), décider si un patient est malade ou bien portant selon sa température et s'il a la gorge irritée.

Cet arbre de décision devra séparer entre 2 classes (malade, bien portant) selon 2 variables (température, gorge irritée) [32]:

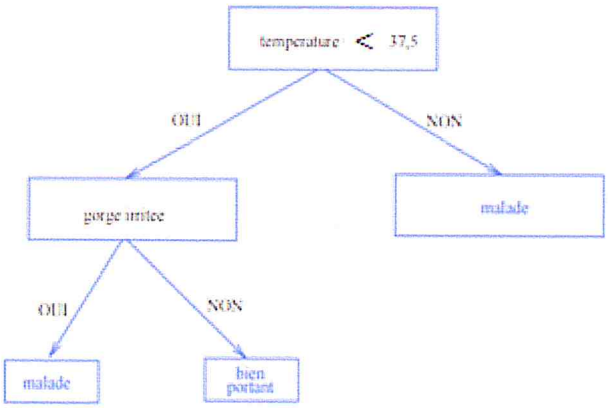


Figure 33: Arbre de décision à deux niveaux [32].

Cet arbre de décision comporte deux niveaux suivant les deux variables d'entrée : température et gorge irritée.

Decision Stump

Les *Decision Stumps* sont des arbres de décision avec un seul niveau : se composant d'un seul nœud interne (la racine), qui est immédiatement connecté avec les nœuds externes (les feuilles). Un *Decision Stump* fait une prédiction en se basant sur la valeur d'une seule caractéristique en entrée, contrairement à un arbre de décision qui en prend plusieurs [31].

Dans un problème de classification, chaque nœud d'un *Decision Stump* représente une caractéristique en attente d'être classifiée, et chaque branche représente la valeur que la caractéristique peut prendre comme on peut le voir sur la figure 34 [33].

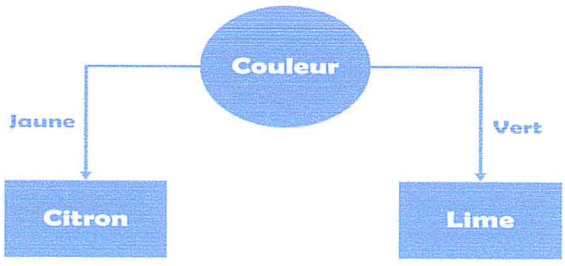


Figure 34 : Illustration sur un graphe du fonctionnement d'un Decision Stump [33]

La classification se fait en évaluant tous les seuillages possibles, et en prenant celui qui minimise au maximum le nombre d'instances mal classées [33]. Vue sa simplicité

d'opération, le taux d'instances mal classées est souvent élevé, d'où son appellation d'algorithme d'apprentissage faible ou naïf (*weak learner*).

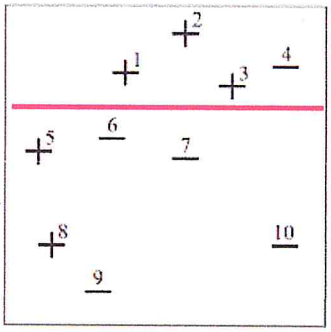


Figure 35: Illustration sur un plan du fonctionnement de Decision Stump pour classer les + et les - .

3.2.2. Cascade et Boosting

L'idée derrière un classifieur en cascade, est la combinaison de façon itérative d'algorithmes d'apprentissage faible ayant pour but d'obtenir un seul classifieur complexe. Chaque étape marque les échantillons comme appartenant soit à l'ensemble positif ou négatif. Dans le cas de la détection, les échantillons de l'image marqués positivement au premier classifieur (susceptibles de contenir l'objet d'intérêt) sont qualifiés pour une deuxième évaluation dans le deuxième classifieur, qui à son tour lance un troisième classifieur si le résultat reste toujours positif pour les mêmes échantillons, et ainsi de suite. Un résultat négatif à n'importe quelle étape provoque le rejet immédiat des échantillons du modèle en cascade [30].

Dans la phase d'entrainement, le passage de l'étage *i* vers l'étage *i+1* est précédé d'un b*oosting* : où le poids des instances mal classées du l'étage *i* sont augmentées (boostées) afin de mieux les classer à l'étage suivant (*i+1*) [17]. Le boosting permet donc de concentrer les nouveaux classifieurs sur des échantillons mal classés par les classifieurs précédents, ce qui permet d'augmenter la complexité du classifieur. Les erreurs des premiers classifieurs indiquent quels sont les échantillons difficiles à classifier correctement [30].

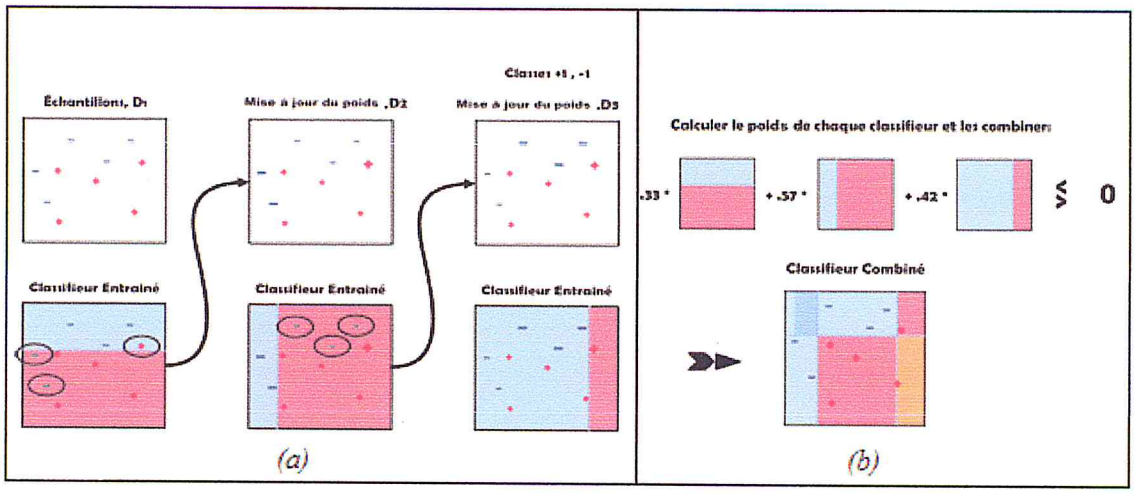


Figure 36 : Illustration du fonctionnement du classifieur Adaboost : (a) 3 DS sont appliqués successivement avec boosting des instances mal classées. (b) Somme pondérée des DS pour former le classifieur final.

Le schéma en cascade est conçu pour éliminer les régions négatives le plus rapidement possible. L'hypothèse faite par Viola et Jones dans [1] est que la majorité des fenêtres ne contiennent pas l'objet d'intérêt ; par opposition, les fenêtres positives sont beaucoup plus rares et méritent des traitements successifs. Le détecteur indique qu'un objet a été trouvé dans la fenêtre quand le résultat reste toujours positif après passage dans le dernier classificateur [17].

Une plus grande précision de détection peut être obtenue en augmentant le nombre de d'étages du classifieur [30].

Le schéma qui suit illustre le fonctionnement d'un classifieur en cascade :

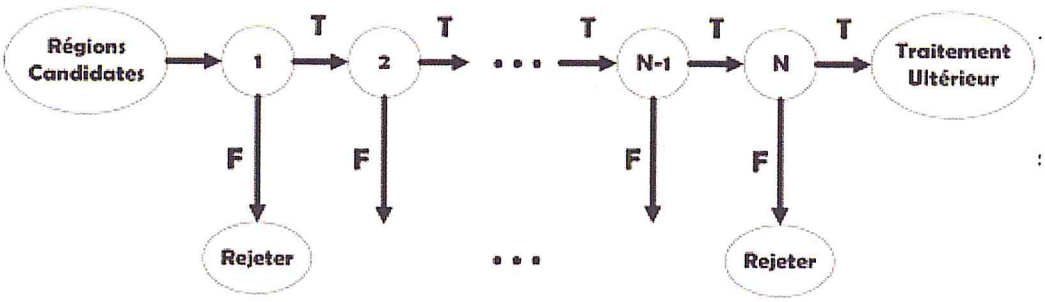


Figure 37: Description schématique du fonctionnement d'un classifieur en cascade.

Cette approche a la particularité de rejeter la plupart des régions négatives tout en détectant les instances positives [17]. La prédiction finale est donc une combinaison de plusieurs prédictions des classificateurs précédents [30].

L'algorithme de l'approche de détection peut être résumé comme suit :

Algorithme

*/*Approvisionnement avec les bases d'images*/*

X = ExCar() ;

Selon (phase)

Cas 'Entrainement'

---

*/*initialiser les poids des échantillons à la même valeur */*

Pour i allant de 1 à nombreD'échantillons

Init (Poids) ;

FinPour

*/*Adaboost */*

Pour i allant de 1 à nombreDeStages, faire

C(i) = entrainer (X, Y, poids) ;

*/*Calculer les prédictions*/*

Ypred = prédire (C(i), X) ;

*/*calculer la somme des erreurs*/*

erreur = poids * (Y = Y - Ypred)';

*/*calcul du coefficient pour le classifieur*/*

Alpha(i) = 0.5*log (1-erreur)/erreur ;

*/*mise à jour des poids*/*

Poids = poids* exp (-alpha(i) * Y * Ypred) ;

*/*normalisation des poids*/*

Poids = poids / somme (poids);

/ combiner les classifieurs*/*

C = combiner (C, C(i)* Alpha(i)) ;

FinPour

Cas 'Détection'

Tant que toute l'image n'a pas été parcourue en totalité

*/*Positionner la fenêtre de détection*/*

f = Fenêtre(I) ;

*/*Extraire les caractéristiques de la fenêtre f*/*

X' = ExCar(f) ;

*/*Passer les caractéristiques de la fenêtre sur le classifieur*/*

s=1 ;

Tant que s <= au nombre de stages

comparer(C, X') ;

Si (annotation(f)= positive)

*/*qualifier la fenêtre au stage suivant pour un autre traitement*/*

s = nextStage() ;

Sinon

*/*rejeter la fenêtre*/*

Rejeter(f) ;

Fsi

FinTanque

*/*Tracer les régions d'intérêt autour des fenêtres qualifiées au dernier stage */*

Si (annotation(f)=positive)

RIO (f) ;

FinSi

GlisserLaFenêtre(I) ;

FinTanque

FinSelon

Fin

Annotations et remarques

- C(i) : classifieur au ieme stage.
- X : L'ensemble des échantillons.
- Y : Un vecteur représentant la classe d'appartenance de l'échantillon : +1, positive, -1 négative, définis à l'étape de détermination manuelle des régions d'intérêt.
- Poids : Un vecteur de poids indiquant l'importance de mieux classer l'échantillon. Les échantillons ont tous le même poids au début de l'algorithme, les échantillons mal classés aux premiers stages voient leurs poids boosté.
- C(i) = entrainer (X, Y, poids) : la fonction d'entrainement fait en sorte de prendre la meilleure classification, qui minimise le nombre d'échantillons mal classés
- Ypred : Un vecteur représentant la prédiction au stage i de la classe de l'échantillon X.
- erreur : la somme des erreurs de classification : si Y est différent de Ypred. L'erreur est toujours > 0.5, en effet le classifieur *Decision Stump* doit donner un meilleur résultat qu'une classification hasardeuse.
- Y - Ypred est un vecteur de 0 et de 1 montrant l'erreur du classifieur pour chaque échantillon.
- Alpha(i) : Représente le coefficient du classifieur i selon la valeur de 'erreur'.
- Mise à jour des poids : une bonne classification se produit si et Y et Ypred ont le même signe : Y*Ypred est positif, sachant que alpha(i) est toujours positif, le résultat de exp (-alpha(i) * Y * Ypred) sera négligeable : le poids n'est pas boosté. Si par contre le signe de Y*Ypred est négatif (mauvaise classification), le poids se verra boosté.
- comparer(C, X') : Compare le résultat entre les caractéristiques de la fenêtre avec ceux du classifieur.

4. Conclusion

L'approche que nous proposons s'inspire des deux travaux suivants : Viola et Jones pour la détection de visages avec l'utilisation des caractéristiques pseudo-Haar, ainsi que le travail de Dalal et Triggs utilisant les HOG pour la détection de personnes. La démarche adoptée consiste à proposer un système à base des extracteurs de caractéristiques vus précédemment. Ce système sera combiné avec un classifieur en cascade simple. Nous allons prouver dans le chapitre qui suit que notre système permet de détecter différents types d'objets (visages, voitures, plaques) de façon indépendante.

Chapitre 4 : Application, Tests et Interprétations

1. Introduction

On présentera ci-dessous l'implémentation de la solution proposée au chapitre 3. L'environnement matériel et logiciel utilisés pour sa réalisation ; les datasets (bases d'images) qui ont servi pour l'entrainement du classifieur en cascade ; les résultats obtenus avec notre application, ainsi que leurs interprétations.

2. Environnement de travail

2.1 Environnement matériel

La mise en œuvre et les tests de l'application ont été réalisés sur deux ordinateurs de type PC :

- Lenovo B590, windows 7 64bits, Processeur Intel (R) Core™ i5-3230M CPU @ 2.60 GHZ, 4 Go de Ram.
- Hp 630, windows 7 64bits, Processeur Intel (R) Core™ i3-2310M CPU @ 2.10 GHZ, 4 Go de Ram.

2.2 Environnement logiciel

L'application a été développée à l'aide de l'outil de programmation *Matlab version 8.6 (R2015b)* de l'éditeur *Mathworks* qui intègre un langage de haut niveau dédié principalement au calcul matriciel ainsi qu'au traitement des signaux et des images.
La compilation sous *Matlab* se fait en appelant la fonction « Deploytool » qui utilise le fichier *.m* (code source) et le fichier *.fig* (spécification de l'interface utilisateur GUI) pour ainsi générer le fichier exécutable.
Pour pouvoir exécuter l'application sous Windows sans avoir à installer l'intégralité de l'environnement de développement Matlab, l'éditeur *Mathworks* fournit le composant *Matlab Compiler 9.0*.

3. Bases d'images utilisées

Les bases d'images utilisées pour l'entrainement des classifieurs proviennent de sources variées. Chaque type d'objet demande une base d'images spécifique se composant d'un ensemble d'exemples positifs : des images contenant l'objet en question, et un ensemble

d'exemples négatifs : des images ne contenant pas l'objet. Les images négatives doivent de préférence contenir l'arrière plan qui soit en relation avec l'objet. Pour l'exemple de la détection de plaques de signalisation, les images négatives doivent contenir le contexte dans lequel les plaques se trouvent généralement: routes, devantures de magasin, véhicules, piétons…etc.

Trois types d'objets on été utilisés pour tester la robustesse de notre solution : visages (vue frontale), et plaques de signalisation.

4. Fonctionnement de l'application « O.D.Trainer »

Le processus de détection se passe en deux temps : phase d'entrainement du classifieur ; et phase de détection des objets. Nous allons maintenant illustrer avec plus de détails le fonctionnement de ces deux phases.

4.1 Entrainement du classifieur

L'entrainement d'un classifieur en cascade est réalisé sous *Matlab par* une succession d'étages, où chaque étage est une combinaison de classifieurs naïfs (Decision Stumps). La sortie du classifieur fournit un détecteur sous format xml.

Après réglage des paramètres et extraction des caractéristiques, l'entrainement procède à chaque étage (via les Decision Stumps) au classement des caractéristiques comme appartenant soit aux positifs (caractéristiques présentes sur l'objet d'intérêt) soit aux négatifs (caractéristiques non présentes sur l'objet). Un boosting est appliqué après chaque étage: les caractéristiques mal classées dans l'étage i voient leur poids boostés pour une meilleure classification à l'étage $i+1$. Les décisions prises à chaque étage sont combinées sous forme de moyenne pondérée. Ce qui permet d'obtenir au final un classifieur complexe avec une plus grande précision.

Le schéma qui suit illustre le fonctionnement de l'entrainement :

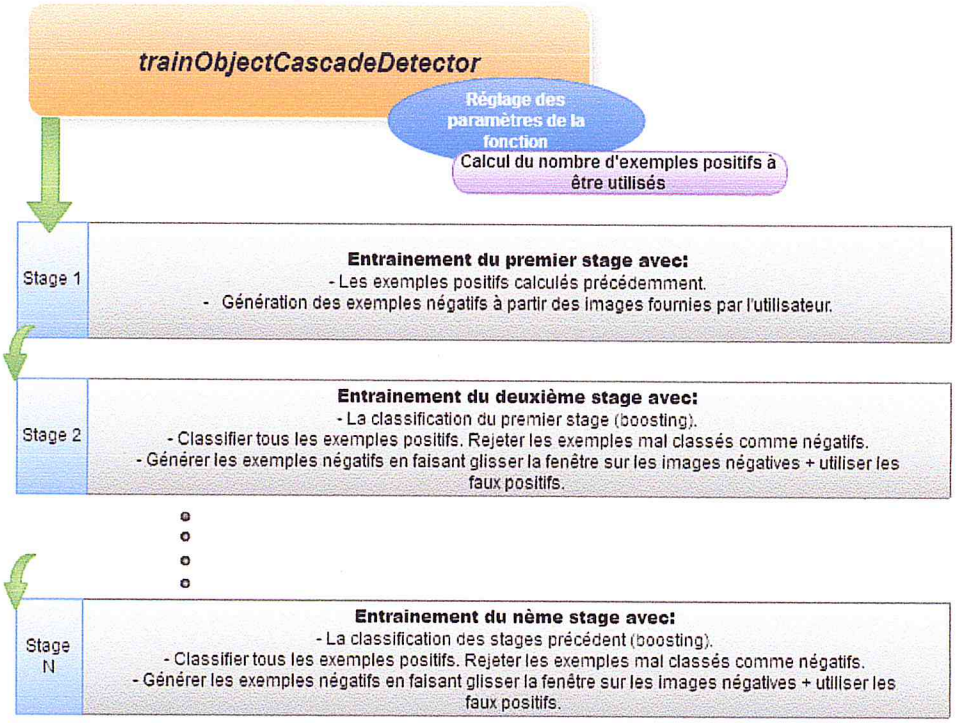


Figure 38: Schéma illustrant le fonctionnement de l'entrainement

L'entrainement du classifieur en cascade requiert un ensemble d'échantillons positifs et un ensemble d'échantillons négatifs. On spécifie les exemples positifs en utilisant *Matlab* pour marquer les régions contenant l'objet (cf. Figure 39). L'application peut générer des échantillons négatifs automatiquement à partir des régions non marquées. Le fichier exporté en sortie (extension *.mat)* est un tableau contenant des éléments de type structure représentant les instances positives. Les champs composant le type structuré sont:

- *imageFilename :* une chaine de caractère spécifiant le nom de l'image. L'image peut être en couleur, en échelle de gris ou bien indexée par n'importe quel format supporté par la fonction IMREAD de Matlab (permet de lire les images).
- *objectBoundingBoxes*: une matrice M*4 [x y largeur hauteur] de rectangles spécifiant la localisation des objets dans les M images.

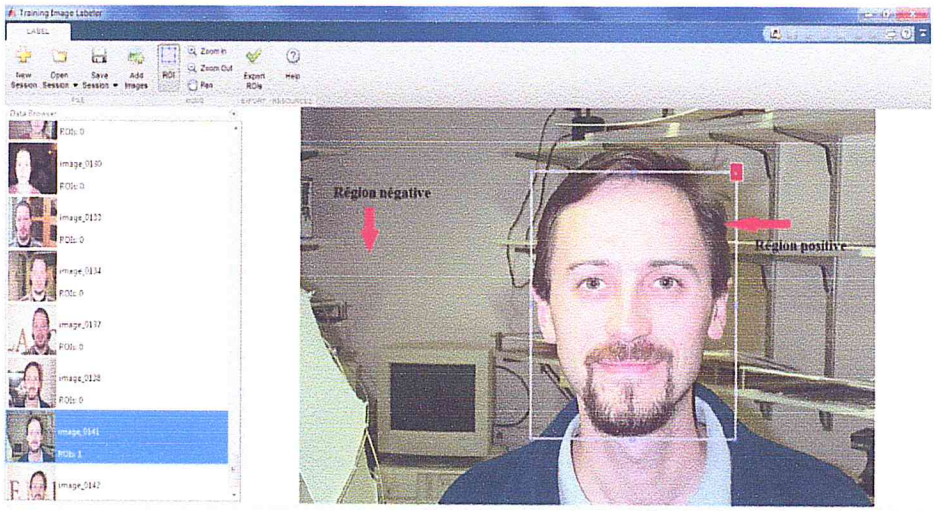


Figure 39: Interface de la fonction d'entrainement pour la spécification des régions d'intérêt dans les images positives.

Le nombre d'échantillons positifs que le classifieur utilisera à chaque étage est calculé automatiquement. Cette valeur dépend du nombre d'étages ainsi que du taux de vrai positif défini. La formule est la suivante :

$$Nombre\ d'échantillons\ positifs = [\ Nombre\ total\ d'échantillons\ positifs\ /\ (1 + (Nombre\ d'étages - 1) * (1 - Taux\ de\ vrai\ positif))\ ]$$

Les régions classées positivement à partir des images négatives fournies par l'utilisateur (fausses détections), sont utilisées comme exemple négatif pour entrainer l'étage suivant. De cette façon, chaque étage corrige les détections des étages précédents. La figure suivante illustre ce mécanisme :

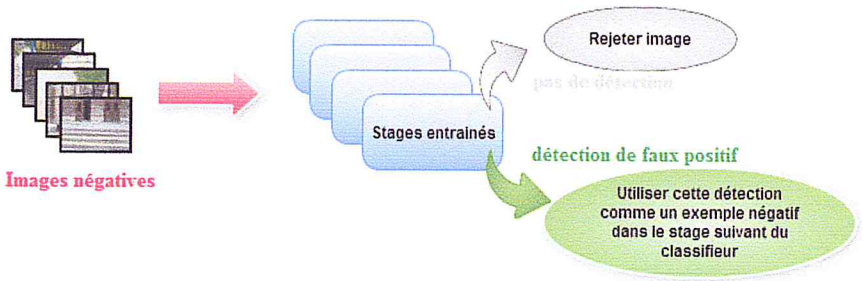


Figure 40: Utilisation des images négatives fournies par l'utilisateur au cours de l'entrainement.

A la fin de l'entrainement, le fichier XML généré contiendra les caractéristiques indiquant la présence de l'objet d'intérêt dans l'image.

4.2 Phase de détection

La deuxième phase consiste simplement à appliquer le classifieur généré à partir de la première phase sur une nouvelle image. La détection des objets se fait en glissant une fenêtre sur la totalité de l'image. La variation de la taille de la fenêtre permet de capturer les objets à différentes échelles. Le classifieur en cascade détermine à chaque étage si la fenêtre considérée est positive (contient l'objet d'intérêt) ou négative (ne contient pas l'objet d'intérêt).

Si la région parcourue est annotée avec un label négatif, la classification de cette région se termine, et le détecteur glisse la fenêtre vers l'emplacement suivant. Si par contre le label de la région est positif, le classifieur qualifie la région considérée à l'étage suivant pour un autre traitement. A la fin, le détecteur détermine la présence de l'objet que si la région est toujours annotée positivement au dernier étage. La figure suivant illustre ce procédé :

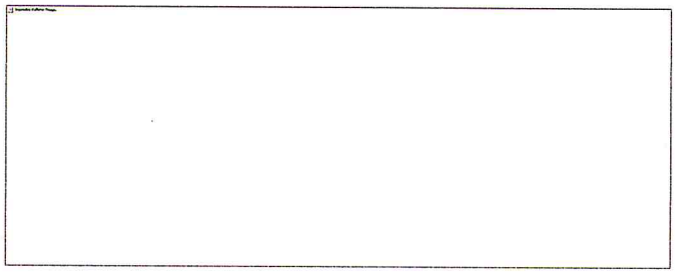


Figure 41: Illustration de la phase de détection à l'aide d'un classifieur en cascade. T : (région annotée positivement), F (région annotée négativement). Ci : succession des classifieurs naïfs.

L'intérêt d'un classifieur sous un schéma en cascade est que les étages sont conçus pour rejeter les régions négatives le plus rapidement possible dans le processus de détection. Le rejet permet d'éviter un traitement supplémentaire des régions classées négativement dans les étages suivants de la cascade. L'hypothèse qui a été faite est que la vaste majorité des fenêtres ne contiennent pas l'objet d'intérêt. Inversement, les vrais positifs sont plus rares, et méritent qu'on s'attarde dessus avec des traitements successifs.

4.3 Les paramètres d'entrainement

Pour obtenir le résultat souhaité, certains paramètres doivent être ajustés avant d'entrainer le classifieur:

- Le nombre d'étages : augmenter ce paramètre peut résulter à un détecteur plus robuste mais qui requiert plus de temps d'entrainement. Aussi, un nombre d'étage élevé demande plus d'images d'entrainement, cela est dû au fait qu'à chaque étage, un certain nombre d'échantillon positifs et négatifs sont éliminés. Ce nombre dépend du taux de vrai positif et du taux de fausse alarme spécifiés.

- Le taux de vrai positif : ce paramètre permet de spécifier pour chaque étage combien d'échantillons positifs sur l'image peuvent êtres mal classés. Plus cette valeur est grande, et plus la complexité de chaque étage augmente (le nombre de classifieurs à chaque étage augmente, ainsi que le temps d'entrainement). La valeur doit être entre 0.8 et 1. Le taux de vrai positif global du détecteur est calculé comme suit :

$$Taux\ de\ vrai\ positif^{Nombre\ d'étages}$$

- Le taux de fausses alarmes: ce paramètre définie le taux de fausses détection accepté à chaque étage. La valeur doit être entre 0 et 0.75. On peut se permettre de spécifier un taux élevé de fausses alarmes si le nombre d'étages est grand. Le taux de fausses alarmes global du classifieur en cascade est calculé comme suit :

$$Taux\ de\ faux\ positifs^{Nombre\ d'étages}$$

Plus le nombre d'étages est grand, et plus le taux de faux positif global du classifieur diminue.

- Le facteur d'échantillon négatif : ce paramètre détermine le nombre d'échantillons négatifs à utiliser à chaque étage. La formule est la suivante:

*Le nombre d'échantillons négatif = facteur d'échantillon négatif * le nombre d'échantillons positifs utilisés à chaque étage.*

- La taille de l'objet d'entrainement: ce paramètre permet de définir la taille de l'objet à détecter dans l'image (sa hauteur et sa largeur en pixels) pour une meilleure précision de détection. Les images d'entrainement seront redimensionnées selon la taille spécifiée.

- Le choix du nombre d'étages : deux manières de procéder existent :

- Peu d'étages avec un taux bas de faux positif : les étages dans ce cas de figure sont plus complexes (contiennent plus de weak learners), et donc le temps d'entrainement sera plus conséquent.
- Plus d'étages avec un taux élevé de faux positif : les étages dans ce cas sont moins complexes (contiennent peu de weak learners).

En général, il est préférable d'utiliser la deuxième approche car le taux global de faux positif diminuera de façon exponentielle avec le nombre d'étages (*Taux de faux positifs* $^{\wedge Nombre\ d'étages)}$, ce qui aura comme effet de minimiser le risque de rejet d'une région positive. Par exemple, si le taux de faux positif est de 50% pour chaque étage, alors le taux de faux positif global d'un classifieur à deux étages sera de 25%, (3 étages, 12.5%)...etc. Si dans un étage, une région contenant l'objet est annotée négativement, sa classification est définitive, et on ne peut pas revenir en arrière. Pour éviter cette situation, on augmente le taux de faux positif: même si le détecteur fait une fasse détection, on peut toujours l'éliminer dans les étages suivants. Il est à préciser que plus le nombre d'étages est grand, et plus le nombre d'images d'entrainement que le classifieur devra utiliser sera élevé.

- Le nombre d'échantillons positifs à fournir : Le nombre d'échantillons disponibles pour entrainer chaque étage dépend du taux de vrai positif. Si par exemple, ce taux est de 90% alors 10% des échantillons positifs sont rejetés comme négatifs, et les 90% restant du total des échantillons positifs sont disponibles pour le prochain étage. Plus le nombre d'images positives fournies est grand, et plus le nombre d'étage et le taux de vrai positif qu'on peut spécifier sera grand, et donc une plus grande performance du détecteur sera obtenue.

- Le nombre d'échantillons négatifs à fournir : L'entrainement peut parfois se terminer avant d'avoir atteint le nombre d'étages spécifié. Ceci se produit lorsque la fonction n'arrive pas à générer assez d'exemples négatifs. Il faut dans ce cas fournir d'avantage d'images négatives.

- Taille de l'objet d'entrainement : pour une précision maximale, il est préférable de préciser la taille de l'objet. Toutefois, pour un entrainement rapide, il est préférable de préciser une taille plus petite que l'objet en question.

4.4 Organigramme

L'organigramme suivant schématise l'utilisation de notre application « O.D.Trainer » :

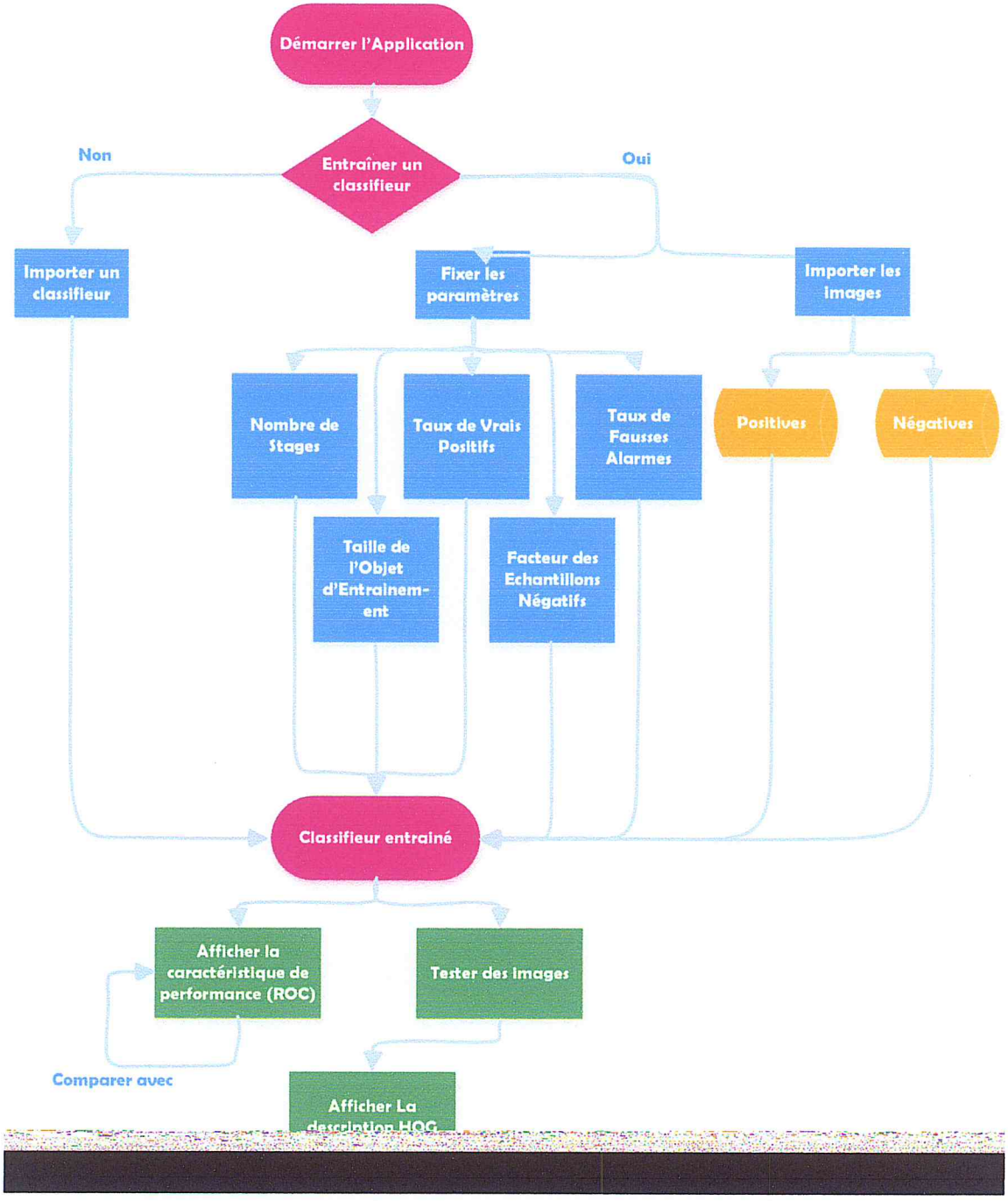


Figure 42 : Organigramme illustrant l'utilisation de « O.D Trainer » .

4.4 Interface utilisateur

L'interface (GUI) facilite la prise en main des différentes fonctionnalités de notre application. Elle fournit les modules permettant le réglage des paramètres d'entrainement, ainsi que la manipulation des données vers les fonctions d'entrainement et de test.

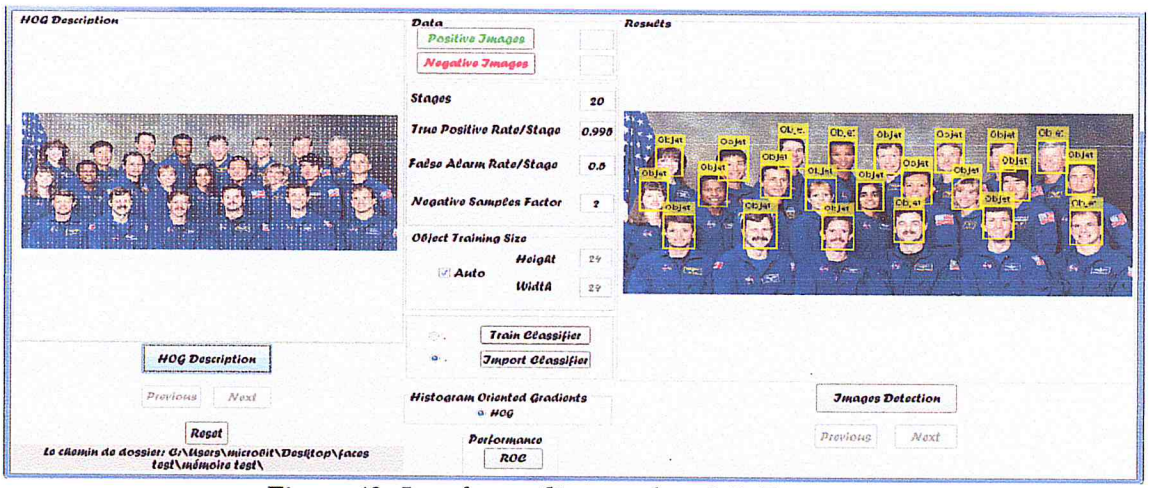


Figure 43: Interface utilisateur de « O.D Trainer ».

Ci-dessous est précisé le fonctionnement de chaque module de l'interface :

- *'Positive Images'* : permet l'importation des images positives préalablement étiquetées sous format (.mat).
- *'Negative Images'* : permet l'importation des images négatives à partir d'un dossier.
- *'Stages'* : permet de spécifier le nombre d'étages qu'aura le détecteur.
- *'False Alarm Rate/ Stage'* : permet de spécifier le taux maximum de fausses alarmes autorisé par étage.
- *'True Positive Rate'* : permet de spécifier le taux minimum de vrais positifs requis à chaque étage.
- *'Negative Samples Factor'* : permet de spécifier le nombre d'échantillons négatifs à utiliser à chaque étage.
- *'Object Training Size'* : permet de spécifier une taille proche de celle de l'objet à détecter.
- *'Train Classifier'* : permet de démarrer le processus d'entrainement selon les différents paramètres définis.
- *'Import Classifier'* : permet l'importation d'un classifieur préalablement entrainé.

- *'Test Images'* : permet d'importer les images de test. Le résultat sera affiché dans le panneau 'Results'.
- *'HOG Description'* : permet d'illustrer les caractéristiques (HOG) sur les images de test dans le panneau 'HOG Description'.
- *'Previous'* : permet la visualisation des précédentes images.
- *'Next'* : permet la visualisation des images suivantes.
- *'Reset'* : permet la réinitialisation de toute l'interface.

5. Expérimentation et résultats

On présentera ci-dessous les spécificités des bases d'images utilisées, les différents paramètres d'entrainement, ainsi que les résultats obtenus pour les différents tests pour les trois types d'objets : visages, véhicules, plaques de signalisation.

5.1 Détection de visages (vue frontale)

5.1.1 Base d'images utilisée

- Images positives : 450 images de 896 x 592 pixels au format PNG acquises à « *California Institute of Technology* » [34] ; représentant des visages sous une vue frontale. La base se compose de photos de 27 personnes de différentes races ethniques et sous différentes conditions : expressions des visages, luminosité, image de fond.
- Image négatives : 1610 images négatives de 320 x 240 pixels au format PNG représentant différents paysages : laques, habitations, forêts, montagnes, paysages urbains…etc.

5.1.2 Paramètres d'entrainement

Les paramètres d'entrainement fixés pour les différents tests sont indiqués ci-dessous:

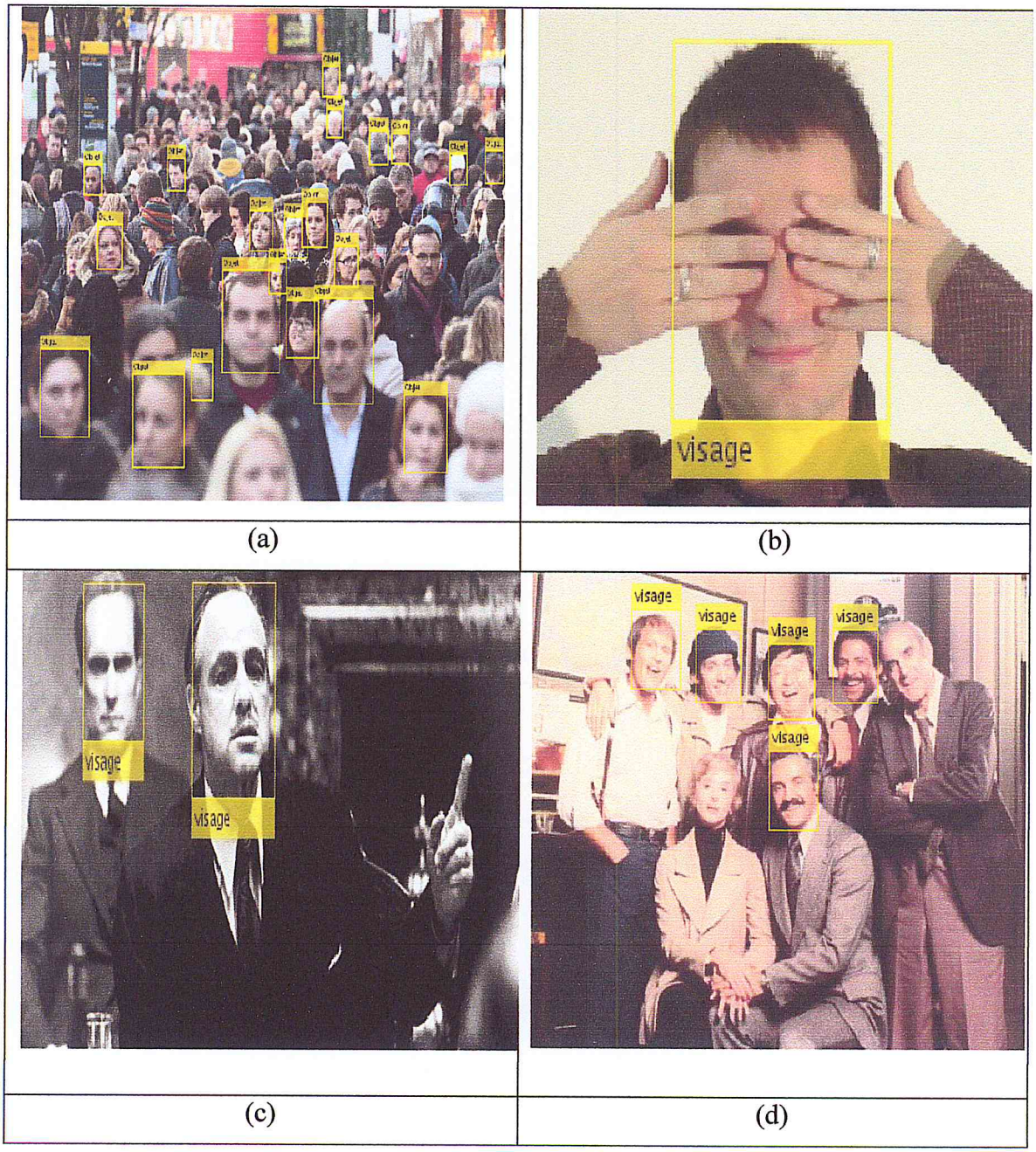
Tests / paramètres	Nombre d'étages	Taux vrai positif	Taux faux positif	Facteur échantillon négatif	Taille objet d'entrainement
Test 1	15	0.995	0.5	2	42x32
Test 2	12	0.995	0.4	2	42x32
Test 3	11	0.990	0.4	2	42x32

Tableau 1 : Les paramètres utilisés pour l'entrainement de trois détecteurs de visages

5.1.3 Résultats des Tests

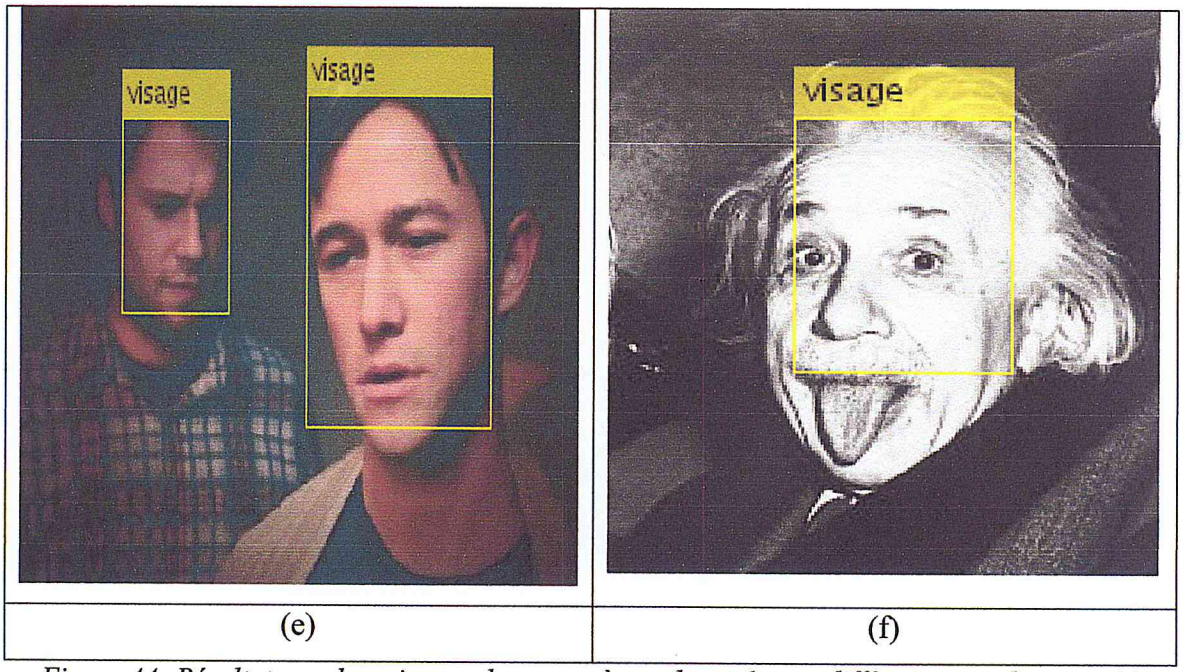
Les résultats obtenus pour les différents tests sont :

- *Résultats pour Test 1*
  - ✓ Le taux de vrai positif global atteint est de 92%.
  - ✓ Le taux de faux positif global atteint est de 0.003%.
  - ✓ Le nombre de Decision Stumps (répartis dans les 15 étages) est de 112.



(a) (b)

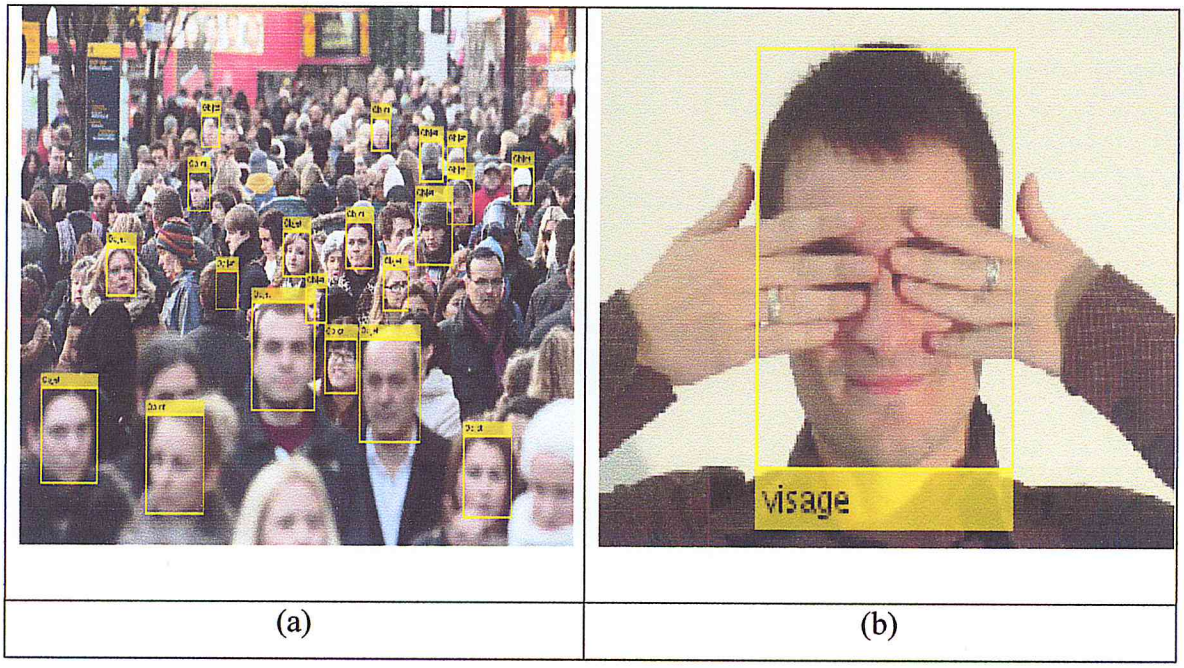
(c) (d)



(e) (f)

Figure 44: Résultats sur les visages des paramètres de test1 sous différentes conditions : (a) changement d'échelle, (b) occlusion, (c) flou, (d) orientation, (e) changement de luminosité, (f) expression faciale.

- *Résultats pour Test 2*
  - ✓ Le taux global de vrai positif atteint est de 94%.
  - ✓ Le taux global de faux positif atteint est de 0.0017%.
  - ✓ Le nombre de Decision Stumps (répartis dans les 12 étages) est de 104.



(a) (b)

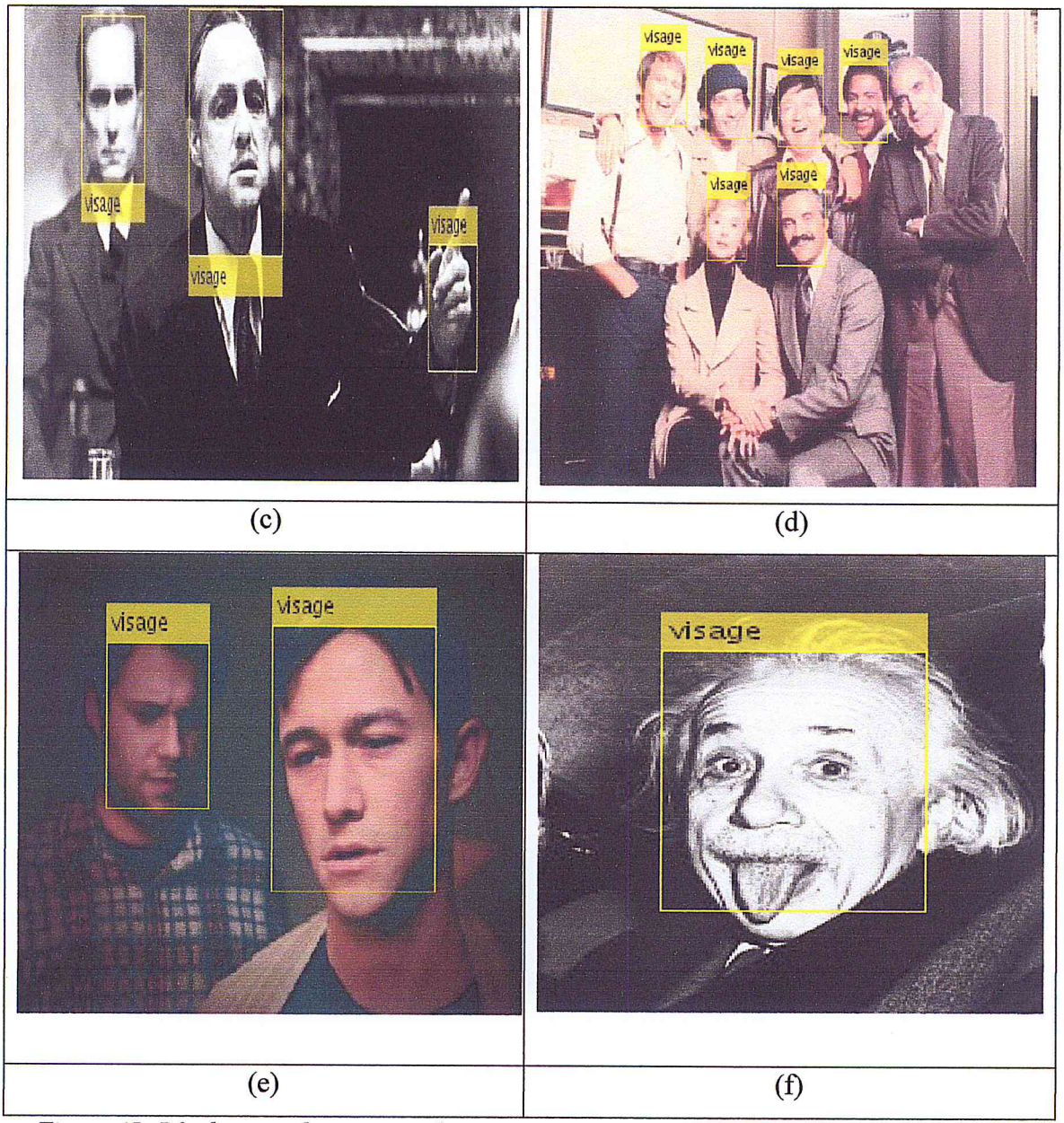


Figure 45: Résultats sur les visages des paramètres de test2 sous différentes conditions : (a) changement d'échelle, (b) occlusion, (c) flou, (d) orientation, (e) changement de luminosité, (f) expression faciale.

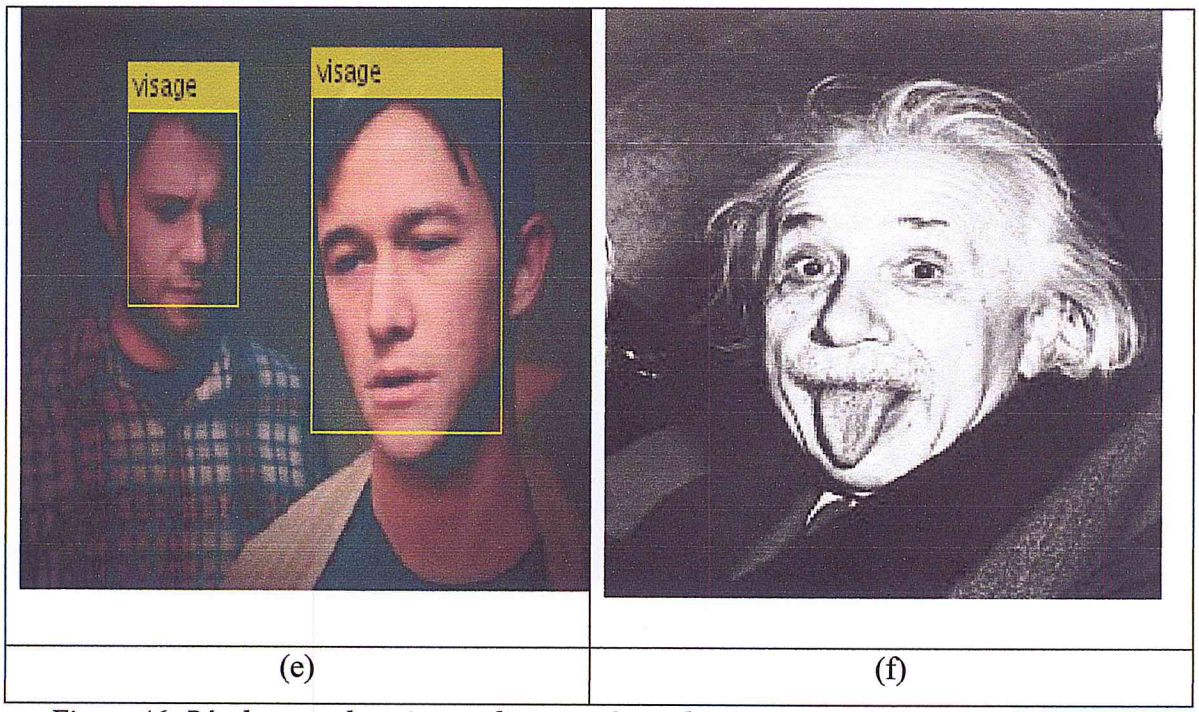
- *Résultats pour Test 3*
  - ✓ Le taux global de vrai positif atteint est de 89%.
  - ✓ Le taux global de faux positif atteint est de 0.0042%
  - ✓ Le nombre de Decision Stumps (répartis dans les 11 étages) est de 60.

(a)

(b)

(c)

(d)



(e) (f)

Figure 46: Résultats sur les visages des paramètres de test3 sous différentes conditions : (a) changement d'échelle, (b) occlusion, (c) flou, (d) orientation, (e) changement de luminosité, (f) expression faciale.

5.1.4 Commentaires

Les résultats obtenus montrent une robustesse des détecteurs face :

- ✓ au changement d'échelle : test1(a), test2(a), test3(a).
- ✓ aux occlusions: test1 (b), test2 (b), test3 (b).
- ✓ au flou : test1(c), test2(c).
- ✓ au changement de luminosité : test1(d), test2(d), test3(d).
- ✓ à différentes expressions faciales : test2(f).

On note également :

- × quelques fausses alarmes dans : test1(a), test2(a), test3(a), test2(c).
- × des échecs de détections lorsque les visages sont penchés.
- × un échec de détection du visage flou dans test3(c).
- × un mauvais encadrement du visage dans test1(f) : la région contenant la langue a été classée comme négative.
- × un échec de détection dans test3(f).

5.1.5 Discussion

Les détecteurs de visages obtenus avec les paramètres précisés précédemment présentent de bons résultats face au changement de luminosité, de l'échelle et aux

occlusions. On note cependant une faiblesse face à l'orientation des visages, ainsi que quelques fausses alarmes lorsqu'une région de l'image possède le même ratio qu'un visage. Les meilleurs résultats ont été obtenus dans test2 : taux global de vrai positif le plus élevé et le taux global de faux négatif le plus bas. Nous présentons ci-dessous les courbes ROC illustrant la performance des détecteurs pour les différents tests :

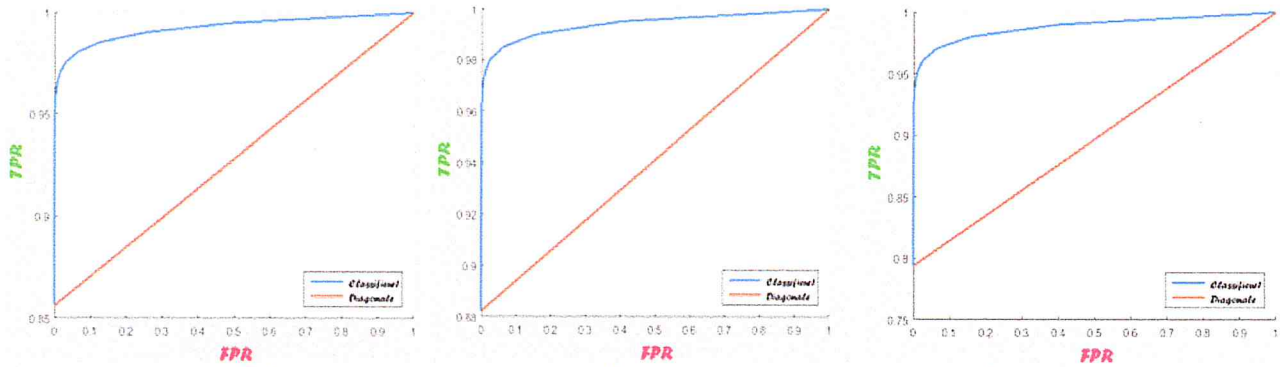


Figure 47: Résultats comparatifs des courbes ROC pour les trois tests : (a) test1, (b) test2, (c) test3

5.2 Détection de véhicules (vue latérale)

Un deuxième exemple d'application de notre système s'est fait en considérant les véhicules.

5.2.1 Bases d'images utilisées

- Images positives : une combinaison de deux bases de données : la première se composant de 550 images en échelle de gris de 100 x 40 pixels au format PNG acquises à l' «University of Illinois at Urbana-Champaign » [35]. La deuxième base se compose de 79 images en couleur de 240 x 260 pixels au format PNG acquises à « Darmstadt University of Technology » [35]. Les deux bases illustrent des véhicules sous un angle de vue latéral et sous différentes conditions d'acquisition : occlusion, changement d'échelle, et double sens.

- Image négatives : une combinaison de deux bases de données : la première de 500 images négatives en niveaux de gris de 100 x 40 au format PNG. La deuxième base contient 1371 images en couleur de 279 x 186 pixels au format PNG illustrant différents paysages : autoroutes, animaux, humains, paysage urbain, habitations, mers…etc.

5.2.2 Paramètres d'entrainement

Le tableau 2 représente les paramètres d'entrainement fixés pour trois tests :

Tests / paramètres	Nombre d'étages	Taux vrai positif	Taux faux positif	Facteur échantillon négatif	Taille objet d'entrainement
Test 1	3	0.999	0.05	4	30 x 80
Test 2	7	0.995	0.5	2	30 x 80
Test 3	9	0.995	0.6	2	30 x 80

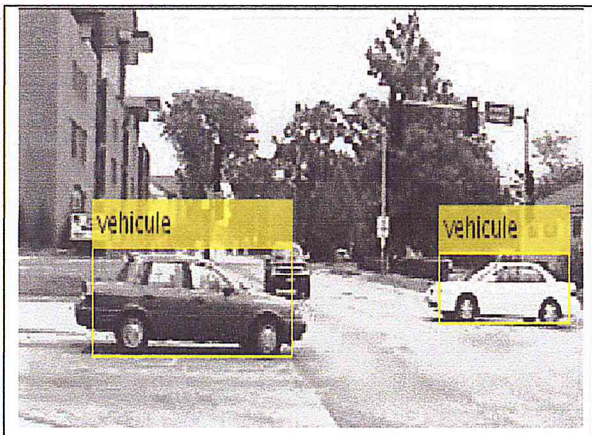
Tableau 2 : Les paramètres utilisés pour l'entrainement de 3 détecteurs de véhicules (vue latérale)

Compte tenu du nombre réduit d'images positives disponibles pour le traitement (550), le nombre d'étages choisi ainsi que le taux de faux positif doivent être relativement bas pour assurer un bon résultat.

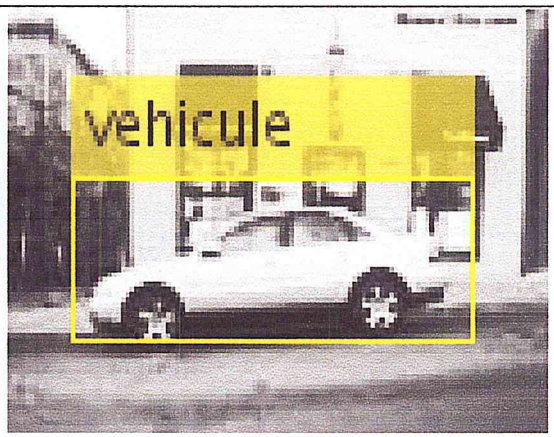
5.2.3 Résultats des Tests

A l'issue des différentes expérimentations, nous avons obtenu les résultats suivants :

- *Résultats pour Test 1*
  - ✓ Le taux de vrai positif global atteint est de 99%.
  - ✓ Le taux de faux positif global atteint est de 0.013%.
  - ✓ Le nombre de Decision Stumps (répartis dans les 3 étages) est de 56.



(a)



(b)

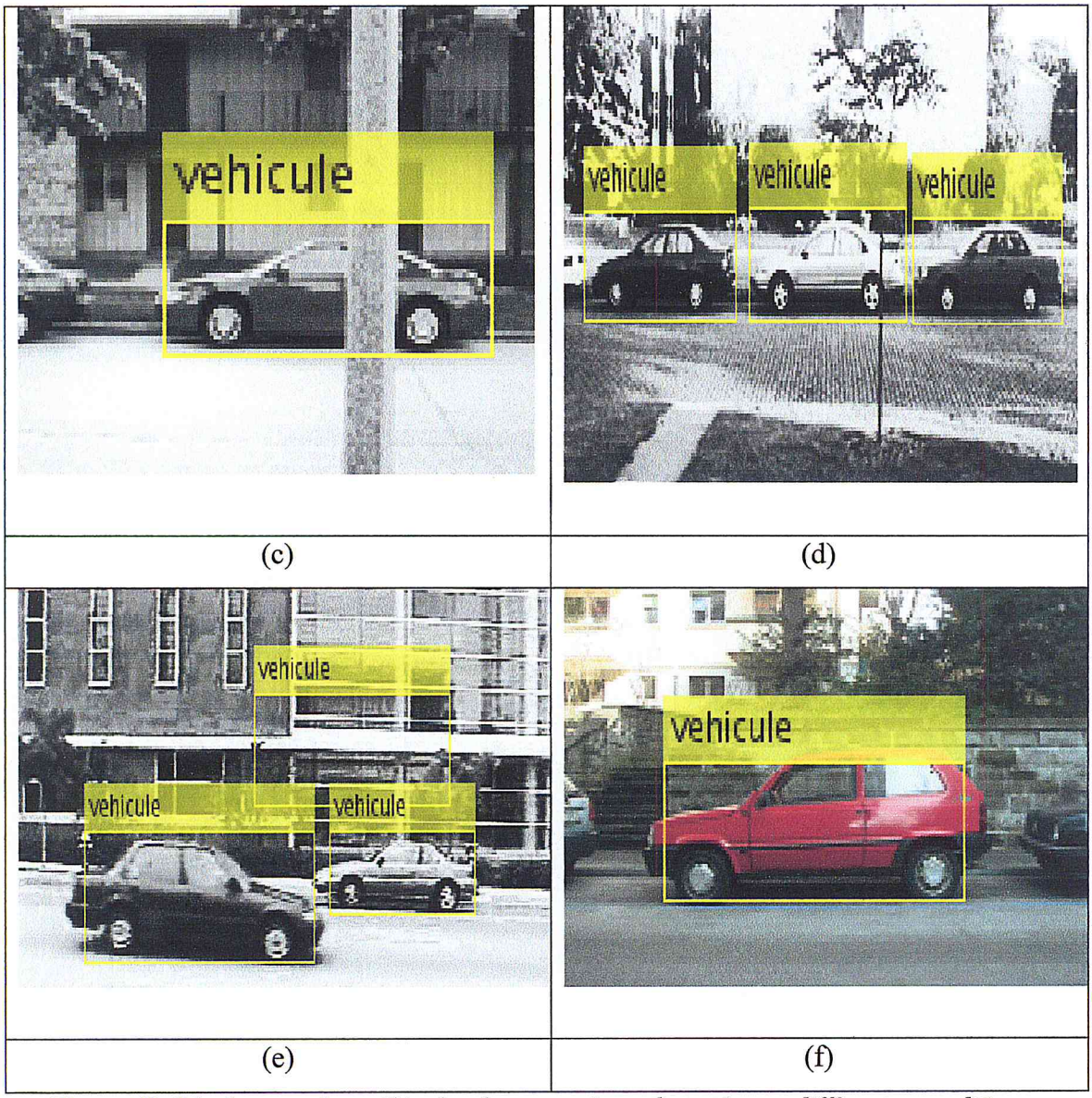


Figure 48: Résultats sur les véhicules des paramètres de test1 sous différentes conditions : (a,e) changement d'échelle, (b) inclinaison (c) occlusion, (d,f) changement de luminosité.

- *Résultats pour Test 2*
  - ✓ Le taux de vrai positif global atteint est de 96%.
  - ✓ Le taux de faux positif global atteint est de 0.78 %.
  - ✓ Le nombre de Decision Stumps (répartis dans les 7 étages) est de 35.

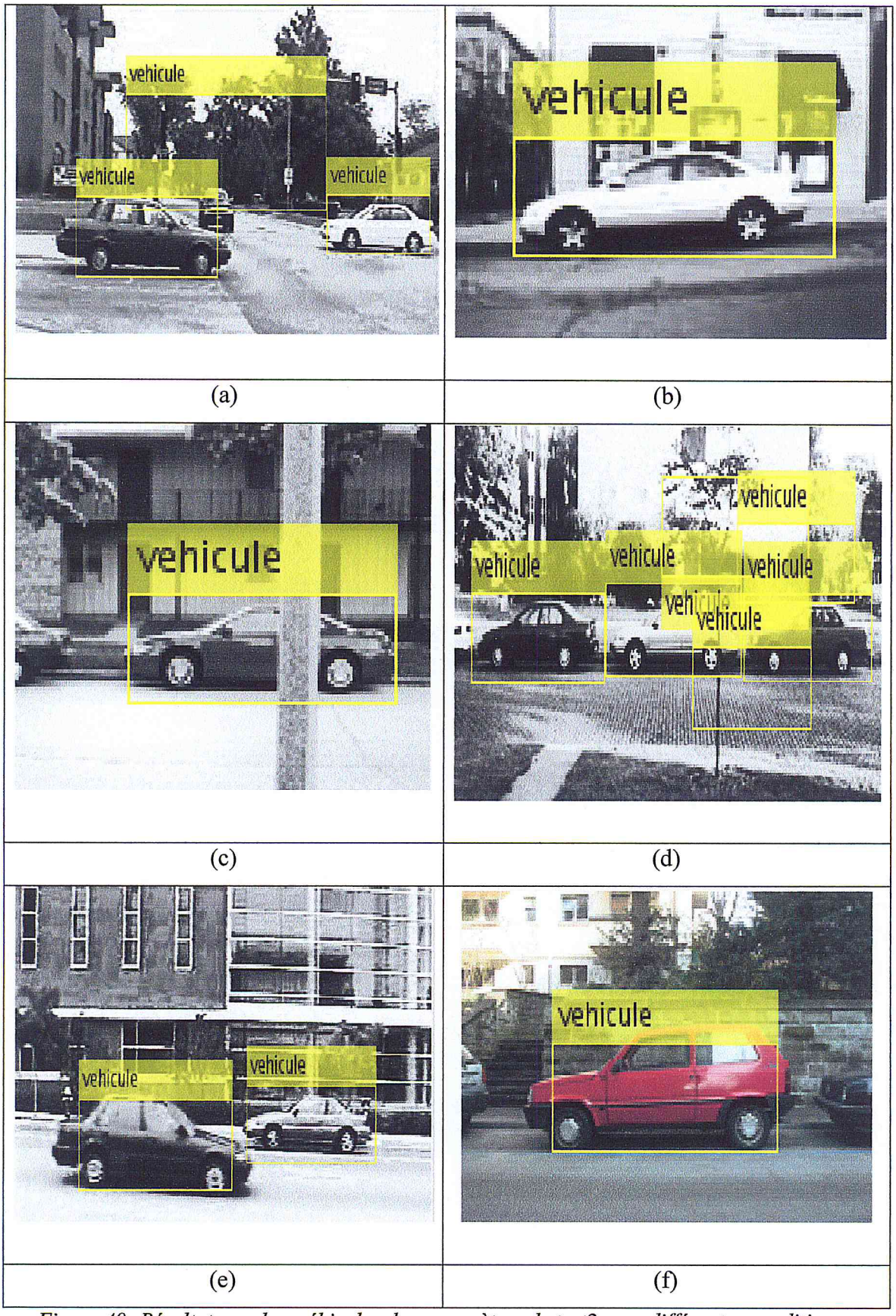
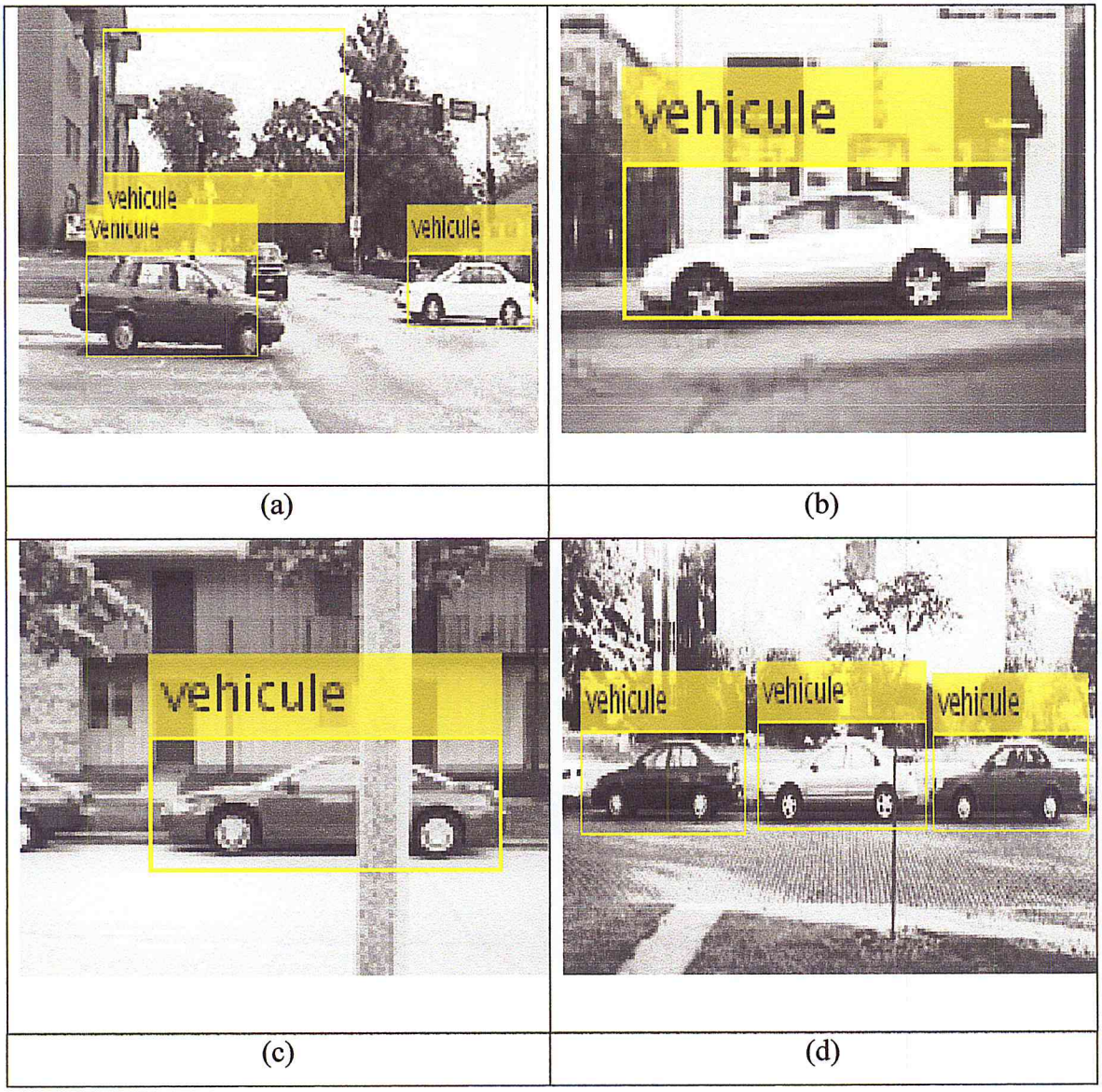


Figure 49: Résultats sur les véhicules des paramètres de test2 sous différentes conditions : (a,e) changement d'échelle, (b) inclinaison (c) occlusion, (d,f) changement de luminosité.

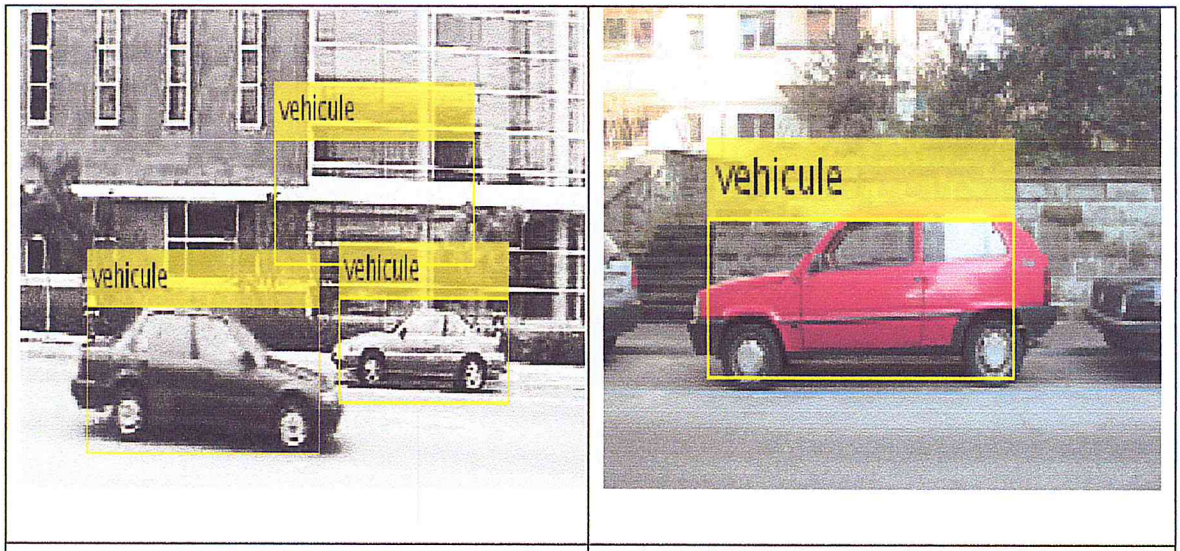
➢ *Résultats pour Test 3*

- ✓ Le taux de vrai positif global atteint est de 95%.
- ✓ Le taux de faux positif global atteint est de 1%.
- ✓ Le nombre de Decision Stumps (répartis dans les 9 étages) est de 40.



(a) (b)

(c) (d)



(e) (f)

Figure 50: Résultats sur les véhicules des paramètres de test3 sous différentes conditions : (a,e) changement d'échelle, (b) inclinaison (c) occlusion, (d,f) changement de luminosité.

5.2.4 Commentaires

On remarque une robustesse des détecteurs face :

- ✓ au changement d'échelle: test1(a), test2(a), test3(a), test1(e), test2(e), test3(e).
- ✓ aux occlusions: test1 (c), test2 (c), test3 (c).
- ✓ à une légère inclinaison: test1(b), test2(b), test3(b).
- ✓ à la luminosité des véhicules: détecte aussi bien les véhicules sombres que clairs.

On note également :

- ✗ quelques fausses alarmes dans test2(a), test2(d), test3(a) : le taux de faux positif global atteint ne permet pas d'éliminer ces régions négatives.
- ✗ quelques fausses alarmes dans test1(e) et test3(e) : la structure du bâtiment en arrière plan a été confondue avec la forme d'un véhicule.
- ✗ un mauvais encadrement du véhicule dans test1(f), test2(f) et test3(f) : le véhicule rouge présente un ratio légèrement différent (véhicule moins long) de la taille de l'objet spécifiée avant l'entrainement.

5.2.5 Discussion

Les détecteurs de véhicules obtenus présentent une robustesse face au changement d'échelle, aux occlusions et aux inclinaisons légères, ainsi qu'à la couleur. Cependant, on note la présence de fausses alarmes, notamment dans test2 (le taux global de faux

positif atteint à la fin de l'entrainement est de 0.78%). Ceci peut être expliqué par le fait que le taux de faux positif par étage défini avant l'entrainement est élevé (0.5) par rapport au nombre d'images d'entrainement disponibles. Une plus grande base aurait permis un plus grand nombre d'étages et de ce fait, plus de chance d'éliminer les régions négatives aux étages suivants.

Les meilleurs résultats ont été obtenus dans test 1 : le taux de faux positif est le plus bas, le taux de vrai positif le plus élevé, et le nombre de DS utilisés pour l'entrainement est le plus haut (ce qui indique une plus grande complexité de traitement dans chaque étage). La figure ci-dessous présente les courbes ROC illustrant la performance pour les trois tests :

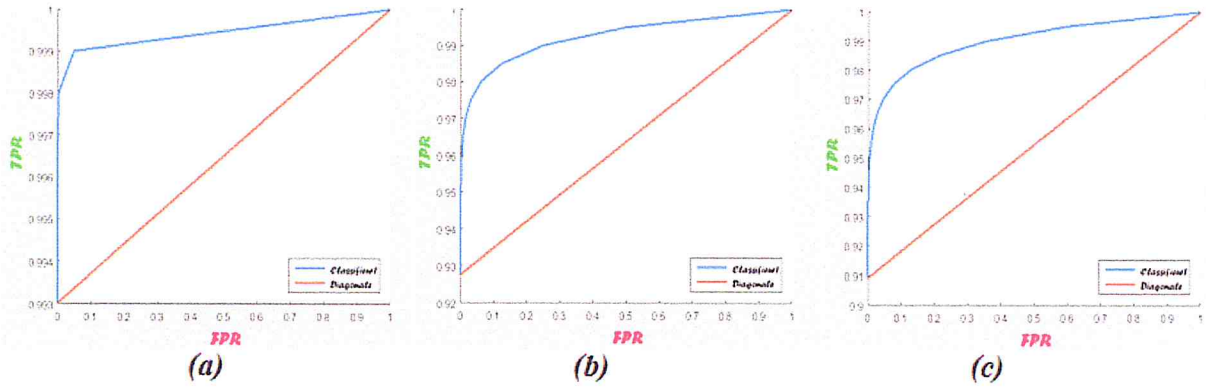


Figure 51 : Résultats comparatifs des courbes ROC pour les trois différents tests : (a) test1, (b) test2, (c) test3.

5.3 Détection de plaques de signalisation

5.3.1 Base d'images utilisée

- Images positives : 616 images de 1500 x 1024 pixels au format JPG contenant des plaques de signalisation de différentes formes (circulaires, triangulaires, carrées, octogonale) et de différentes couleurs (jaunes, bleues, rouges, blanches) et sous différentes conditions (luminosité, rotation, occlusion, différentes échelles, déformées). La base d'images a été acquise au niveau de ''Computer Vision Research Group (GRAM UAH) Université Alcalá de Henares, en Espagne'' [36].

- Image négatives : 2887 images de 320 x 240 au format PNG ne contenant pas de plaques de signalisation: routes, véhicules, arbres, habitations, parcs…etc.

5.3.2 Paramètres d'entrainement

Les paramètres d'entrainement fixés pour les différents tests sont mentionnés ci-dessous:

Tests / paramètres	Nombre d'étages	Taux vrai positif	Taux faux positif	Facteur échantillon négatif	Taille objet d'entrainement
Test 1	10	0.990	0.3	2	32 x 32
Test 2	15	0.999	0.1	2	32 x 32
Test 3	20	0.995	0.5	2	32 x 32

Tableau 3 : Paramètres utilisées pour l'entrainement de 3 détecteurs de plaques de signalisation.

5.3.3 Résultats des Tests

- *Résultats pour Test 1*
  - ✓ Le taux de vrai positif global atteint est de 90%.
  - ✓ Le taux de faux positif global atteint est de 0.0006%.
  - ✓ Le nombre de Decision Stumps (répartis dans les 10 étages) est de 541.

(a) (b)

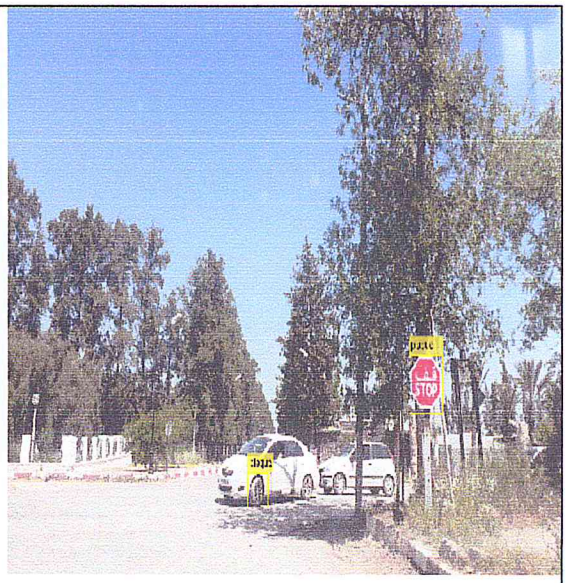
(c) (d)

(e) (f)

Figure 52: Résultats sur les plaques des paramètres de test1 sous différentes conditions : (a,b,f) changement de formes, (c) déformation, (d) occlusion, (e) changement d'échelle.

- *Résultats pour Test 2*
  - ✓ Le taux de vrai positif global atteint est de 98%.
  - ✓ Le taux de faux positif global atteint est de 0.0000000000001%.
  - ✓ Le nombre de Decision Stumps (répartis dans les 9 étages) est de 1407.

(a)



(b)

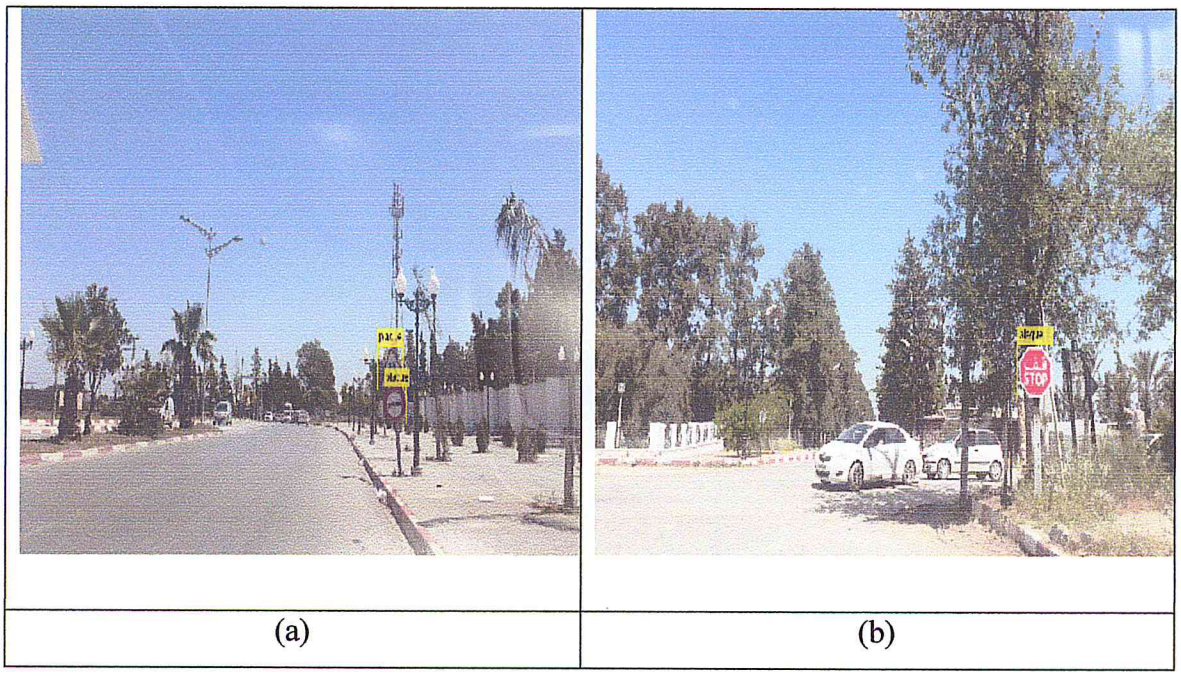
(c)

(d)

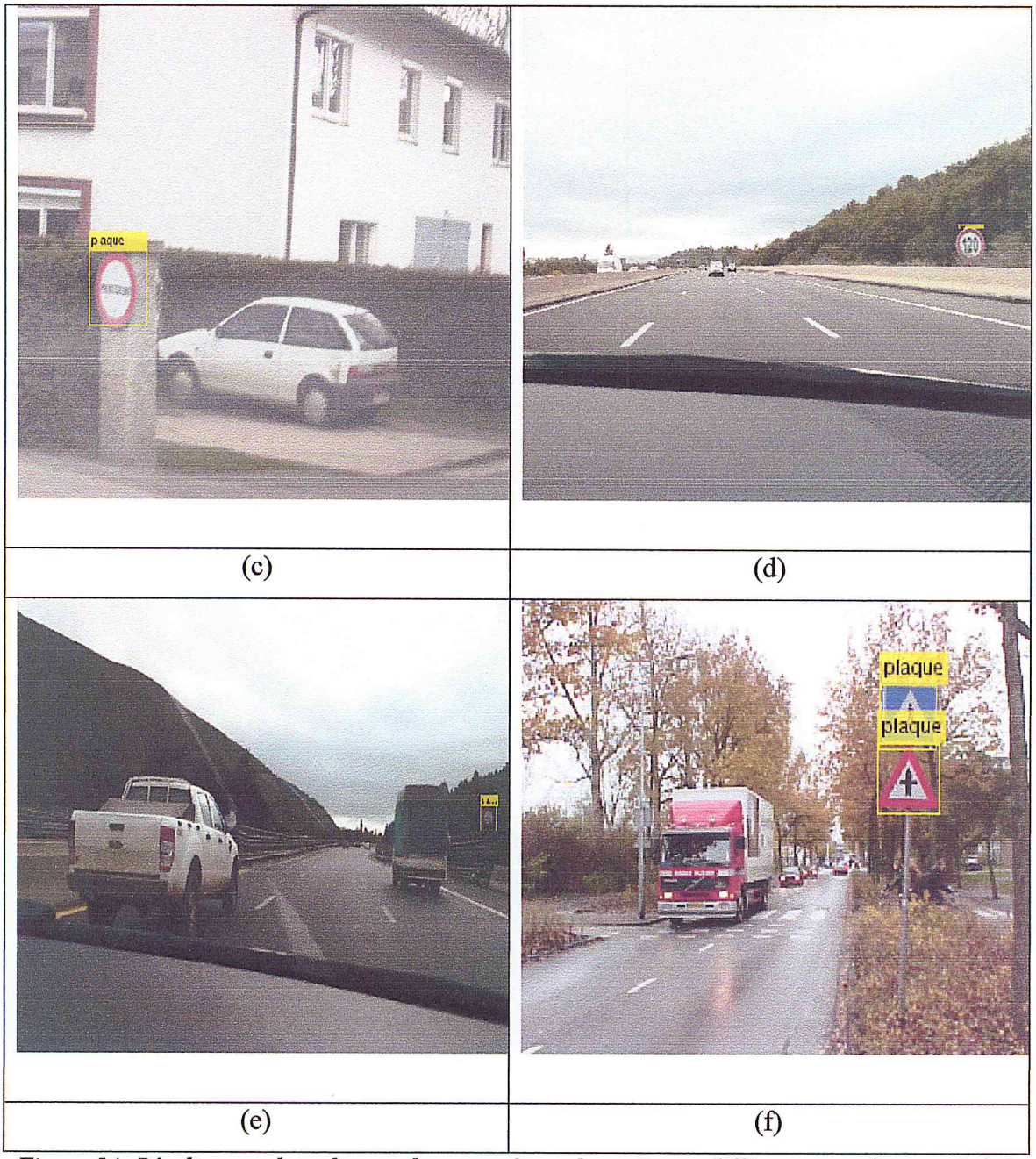
(e) (f)

Figure 53: Résultats sur les plaques des paramètres de test2 sous différentes conditions : (a,b,f) changement de formes, (c) déformation, (d) occlusion, (e) changement d'échelle.

- *Résultats pour Test 3*
  - ✓ Le taux de vrai positif global atteint est de 90%.
  - ✓ Le taux de faux positif global atteint est de 0.00009%.
  - ✓ Le nombre de Decision Stumps (répartis dans les 20 étages) est de 948.



(a) (b)



(c) (d)

(e) (f)

Figure 54: Résultats sur les plaques des paramètres de test3 sous différentes conditions : (a,b,f) changement de formes, (c) déformation, (d) occlusion, (e) changement d'échelle.

5.3.4 Commentaires

D'après les résultats obtenus, nous remarquons une robustesse des détecteurs face :

- ✓ à la forme: triangulaire: test1(a,f), test2(a,f), test3(a,f). Carrée: test1(f), test2(f), test3(f). Circulaire: test1(a,c,d,e), test2(a,c,d,e). Octagonale: test1(b), test2(b), test3(b). Les formes circulaires légèrement déformées sont aussi détectées : test1(c), test2(c), test3(c).

- ✓ à l'occlusion : test1(d), test2(d), test3(d): la plaque est légèrement cachée par la végétation.
- ✓ à la basse de luminosité : test1(e), test2(e), test3(e).
- ✓ à différentes échelles : lointaines : test1(e), test2(e), test3(e) ; ou proches : test1(f), test2(f), test3(f).

On note également :

- * de fausses alarmes présentes dans test1(c), test2(b,c). La forme circulaire des roues a été confondue avec la présence d'une plaque circulaire. Ce problème peut être réglé en fournissant des images contenant des roues de véhicules (ou autre objet similaire) dans l'ensemble des images négatives.

5.3.5 Discussion

Le détecteur de plaque de signalisation obtenu donne de très bons résultats face aux changements de couleur, de luminosité, de forme, d'échelle, aux occlusions et aux déformations légères. Les faux positifs peuvent être évités en enrichissant la base d'images négatives avec les objets provocant les fausses alarmes. Ainsi, lors de la phase d'entrainement, les régions annotées positivement à partir des images négatives seront considérées comme échantillons négatifs et seront passées à l'étage suivant.

Diminuer le taux de faux positif (0.1 dans test2) a pour effet d'augmenter de façon considérable le nombre de DS utilisés pour l'entrainement (1407). Ce qui nous permet de spécifier un nombre moins élevé d'étages mais avec une plus grande complexité. L'objectif est d'éviter que le taux global de vrai positif diminue tout en gardant un taux global de faux positif très bas.

Les courbes ROC illustrant la performance du détecteur pour les différents tests sont présentés dans la figure 55 :

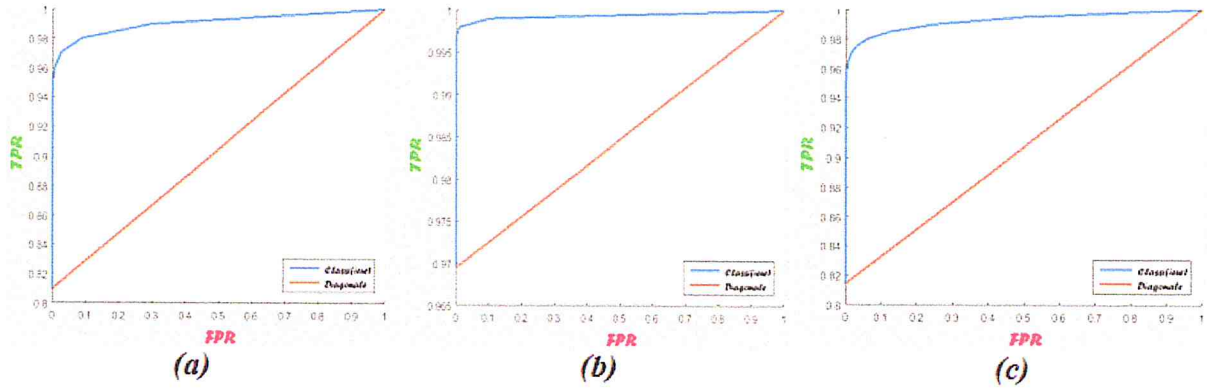


Figure 55 : Résultats comparatifs des courbes ROC pour les différents tests : (a) test1, (b) test2, (c) test3.

6. Discussion générale des résultats

En résumé, notre approche permet d'obtenir de bons résultats pour n'importe quel type d'objet à reconnaitre. Toutefois, elle nécessite un réglage précis des paramètres lors de la phase d'apprentissage. Pour cela, plusieurs choix sont possibles ; le choix à prendre est conditionné principalement par le volume de la base de données d'images disponible. Le tableau suivant décrit la conduite à prendre avant d'entamer l'entrainement du détecteur [17]:

Conditions	Considérations
Une grande base d'images d'entrainement (+1000)	• Augmenter le nombre d'étages (>15) • Augmenter le taux de faux positif (>0.45).
Un ensemble d'images d'entrainement réduit (entre 100 et 1000)	• Réduire le nombre d'étages (<10) • Réduire le taux de faux positif (<0.4).
Pour réduire la probabilité de rater un objet	• Augmenter le taux de vrai positif. (un taux élevé peut augmenter le nombre de fausses alarmes).
Pour réduire le nombre de fausses alarmes	• Augmenter le nombre d'étages (si la base d'images est suffisamment grande). • Réduire le taux de faux positif (ceci

	aura pour effet de rehausser la complexité de chaque étage en augmentant le nombre de DS).

Tableau 4: recommandations lors du réglage des paramètres d'entrainement.

7. Conclusion

Nous avons présenté dans ce dernier chapitre l'environnement de travail dans lequel on a travaillé. Nous avons aussi décrit le fonctionnement de l'approche, l'interface de l'application réalisée, les tests pour trois types d'objets différents, ainsi que des recommandations générales pour réussir un bon entrainement. Les résultats obtenus sont assez satisfaisants compte tenu du fait que la taille des bases d'images utilisées est moyenne par rapport à d'autres approches de détection d'objets (9832 images positives utilisées par Viola et Jones).

Conclusion générale

Tout au long de ce mémoire nous avons proposé une approche de détection d'objets basées sur l'extraction des caractéristiques, qui s'affranchit d'une connaissance a priori sur l'objet. Elle nous permet donc de répondre à la problématique posée en introduction ; celle d'aboutir à un outil permettant la détection de n'importe quel objet d'intérêt.

L'approche consiste dans un premier temps à extraire les caractéristiques HOG des images d'entrainement (positives et négatives) et d'utiliser ces informations comme données d'entrée du classifieur. Le classifieur utilisé est *Adaboost*, lequel est organisé sous un schéma en cascade, où les résultats des classifieurs naïfs (les *Decision Stumps*) sont boostés puis combinés pour aboutir au final à un classifieur complexe : le détecteur.

Les résultats obtenus témoignent d'une performance assez robuste de l'approche choisie. Les différents paramètres permettent de personnaliser l'entrainement selon la taille de la base d'images disponible au départ, et selon le résultat souhaité.

Bien évidemment, un résultat parfait n'existe pas dans le monde de la recherche, et des améliorations futures peuvent être rajoutées. L'inconvénient principal de l'approche réside dans l'utilisation de simples caractéristiques pour décrire la forme des objets, ce qui rend le détecteur généré très sensible aux changements de ratio des objets. Par exemple, l'entrainement d'un seul détecteur pour gérer différentes vues de véhicules ne donnera pas de bons résultats, ceci est dû à la différence de ratio que les véhicules peuvent avoir : forme allongée d'une vue latérale, et forme presque carrée pour une vue frontale. Une solution pourrait être d'appliquer des filtres supplémentaires sur les caractéristiques afin d'extraire d'autres informations liées à la nature des objets, tel que la texture.

Un autre inconvénient à noter, est celui du temps d'entrainement. Celui-ci dépend principalement de la taille de la base ainsi que de la résolution des images. Nous avons dû en effet abandonner l'entrainement d'un détecteur de piétons dont le temps avait dépassé 15 jours (près de 55 heures pour l'entrainement du premier étage uniquement). La base contenait 2862 images positives contenant en moyenne 5 instances de piétons, et 3751 images négatives.

Bibliographie

[1] Paul Viola, Michael J. Jones. "Robust Real-Time Face Detection", International Journal of Computer Vision, Vol 57, N°2, pp 137–154, 2004.

[2] Navneet, Dalal. Finding people in images and videos, thèse de doctorat. Institut National Polytechnique de Grenoble, 2006.

[3] Bugeau, Aurélie. (09/03/2016). Image numérique. http://dept-info.labri.fr/ENSEIGNEMENT/imageson/COURS/ImageNumerique-COURS1-2015.pdf

[4] Douze, Matthijs. (05/01/2016). Détection d'objets sur des images. https://www.canal-u.tv/video/inria/detection_d_objets_sur_des_images.18878

[5] Dembélé, Sounkalou (02/02/2016). Vision par Ordinateur 2D. http://members.femto-st.fr/sites/femto-st.fr.sounkalo-dembele/files/content/texts/coursVision2D.pdf.

[6] Lemmouchi, Mansoura. Identification des visages humains et réseaux de neurones, mémoire de magistère, université de Batna, Algérie, 2013.

[7] Morsli, Mohammed. « Chapitre N°1 », Introduction à la reconnaissance des formes (2ème partie), 2011, pp 1-3.

[8] Jitendra, Malik (2013). "The Three R's of Computer Vision", SUNY Buffalo. Department of Computer Science and Engineering, Californie, 07 mars, Université de Berkeley. https://www.youtube.com/watch?v=Mqg6eorYRIQ.

[9] Techniques-ingénieur. (18/04/2016). Hololens le casque de réalité augmentée signé Microsoft. http://www.techniques-ingenieur.fr/actualite/articles/hololens-le-casque-de-realite-augmentee-signe-microsoft-11436/.

[10] Leyrit, Laetitia. Reconnaissance d'objets en vision artificielle : application à la reconnaissance de piétons, thèse de doctorat. Université Blaise Pascal Clermont-Ferrand II, 2010.

[11] Fei Fei, Li. (22/02/2016). How we are teaching computers to understand pictures. https://www.youtube.com/watch?v=40riCqvRoMs.

[12] Grappa. (22/01/2016). Les domaines de l'intelligence artificielle. http://www.grappa.univ-lille3.fr/polys/intro/sortie003.html.

[13] egavves. (22/01/2016). A Brief History of Computer Vision (and not Time). http://www.egavves.com/a-brief-history-of-computer-vision/#sthash.rDR87XgZ.xuvEuWbg.dpuf.

[14] Cned - Académie-en-ligne. (09/03/2016). De l'œil au cerveau : quelques aspects de la vision. http://www.academie-en-ligne.fr/Ressources/7/SN12/AL7SN12TEPA0013-Sequence-08.pdf.

[15] Biri, Venceslas. (09/03/2016). Synthèse d'images I. http://docplayer.fr/storage/24/2461303/1464715906/XWBH3AwWxPPIfsKgix2yNg/2461303.pdf.

[16] Sadgal, Mohammed, El Fazziki, Aziz et Ait Outhman Abdellah. «Reconnaissance d'objets en imagerie aérienne ». RIST, Vol 14, N°02, pp 150-152, 2004.

[17] Mathwork. (02/04/2016). Train a cascade object detector. http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

[18] P. Kapsalas, K. Rapantzikos, A. Sofou, Y. Avrithi. (01/03/2016). Regions of interest for Accurate Object Detection. http://www.image.ece.ntua.gr/papers/535.pdf

[19] Davis, W.James, Sharma, Vinay. (01/03/2016). Robust Background Subtraction for Person Detection in Thermal Imagery. http://web.cse.ohio-state.edu/~jwdavis/Publications/otcbvs04.pdf.

[20] Sahli, Samir. Détection robuste et automatique de véhicules dans les images aériennes, thèse de doctorat. Université Laval Québec, Canada, 2013.

[21] Y.Ramadevi, T.Sridevi, B.Poornima, B.Kalyani. "Segmentation and Object Recognition Using Edge Detection Techniques", International Journal of Computer Science & Information Technology, Vol 2, N°6, pp 159-160, 2010.

[22] A. Fekir, N. Benamrane, A. Taleb-Ahmed. (16/02/2016). Détection et Suivi d'Objets dans une Séquence d'Images par Contours Actifs. http://ceur-ws.org/Vol-547/131.pdf.

[23] Sanjivani Shantaiya, Keshri Verma, Kamal Mehta. "A Survey on Approaches of Object Detection", International Journal of Computer Applications, Volume 65, No°18, March 2013.

[24] Maria C. Garcia-Alegre, David Martin et al. (03/03/2016). Real-Time Fusion of Visual Images and Laser Data Images for Safe Navigation in Outdoor Environments. http://www.intechopen.com/books/sensor-fusion-foundation-and-applications/real-time-fusion-of-visual-images-and-laser-data-images-for-safe-navigation-in-outdoor-environments.

[25] Jie Feng, Yichen Wei, Litian Tao et al. (03/03/2016). Salient Object Detection by Composition. http://research-srv.microsoft.com/en-us/people/yichenw/iccv11_salientobjectdetection.pdf

[26] D.M. Gavrila, V. Philomin. Real-time object detection for smart vehicules. (05/03/2016) http://www-cs.ccny.cuny.edu/~zhu/GC-Spring2010/Papers/Gavrila_ICCV99.pdf.

[27] Christopher Wren, Ali Azarbayejani, Trevor Darrell et al. "Pfinder: Real-Time Tracking of the Human Body", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 19, N°7, pp 780-785, 1997.

[28] Carlos Castillo, Carolina Chang. (09/03/2016). An Approach to Vision-Based Person Detection in Robotic Applications. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.63.6853.

[29] Dumitru Erhan, Christian Szegedy, Alexander Toshev et al. Scalable Object Detection using Deep Neural Networks. (09/03/2016). http://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Erhan_Scalable_Object_Detection_2014_CVPR_paper.pdf.

[30] Viola, Paul, Jones, Michael. (12/04/2016). Rapid object detection using a boosted cascade of simple features. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

[31] Padhraic Smyth. (16/04/2016). Machine learning-Linear classifiers and boosting. www.ics.uci.edu/~smyth/courses/cs271/topic12_learning_part2.ppt .

[32] univ-paris. (16/04/2016). Arbres de décision. https://samm.univ-paris1.fr/IMG/pdf/seance6.pdf .

[33] S.B. Kotsiantis, D. Kanellopoulos and P. E. Pintelas. " Local Boosting of Decision Stumps for Regression and Classification Problems". Journal of Computers, Vol 1, N°4, Juillet 2006.

[34] vision.caltech.edu. (27/04/2016). Faces 1999 (Front). http://www.vision.caltech.edu/Image_Datasets/faces/faces.tar.

[35] vision.caltech.edu. (25/03/2016). UIUC Image Database for Car Detection. http://cogcomp.cs.illinois.edu/Data/Car/

[36] agamenon.tsc.uah.es. (29/04/2016). Traffic Signs UAH Dataset. http://agamenon.tsc.uah.es/Investigacion/gram/traffic_signs.html.