Université Pierre et Marie Curie - Paris VI
École Doctorale de SCIENCES MECANIQUES, ACOUSTIQUE ET ELECTRONIQUE DE PARIS

THÈSE

Pour obtenir le grade de
Docteur
de
l'Université Pierre et Marie Curie - Paris VI
Spécialité : Informatique

Présentée et soutenue publiquement par

Pablo Augusto NEGRI

le 30 septembre 2008

Détection et Reconnaissance d'objets structurés :

Application aux Transports Intelligents

Thèse dirigée par ***Lionel PREVOST***

Réalisée à l'ISIR - Institut des Systèmes Intelligents et de Robotique
4 Place Jussieu
75252 Paris Cedex

Devant le jury :

Pr. Jack-Gérard POSTAIRE	Université des Sciences et Technologies de Lille, Rapporteur
Pr. Frederic JURIE	Université de Caen, Rapporteur
Dr. Lionel PREVOST, HdR	Université Paris VI, Directeur de la Thèse
Dr. Xavier CLADY	Université Paris VI, Examinateur
Pr. Laurent HEUTTE	Université de Rouen, Examinateur
Pr. Thierry ARTIERES	Université Paris VI, Examinateur

Remerciements

Je tiens tout d'abord à remercier le directeur de cette thèse, M. Lionel Prevost, pour m'avoir assisté, guidé et soutenu pendant ces années de thèse à l'ISIR. Je remercie également Xavier Clady qui a été mon compagnon de la route depuis mon arrivé au laboratoire et qui a aussi encadré tous mes travaux.

Je remercie fortement les commentaires de rapporteurs, Jack-Gerard Postaire et Frédéric Jurie, qui m'ont permis d'améliorer la version finale de cette thèse pour une meilleure clarté de la présentation de ce travail.

Mes remerciements vont également à Maurice Milgram, pour m'avoir fait confiance et m'avoir donné l'opportunité de réaliser cette thèse. Je lui remercie pour tous ses conseils et aussi pour m'avoir fait l'honneur de participer au Jury de soutenance.

Je remercie Jean-François Boissou et Fabien Hernandez de Peugeot Citroën PSA et Raphael Poulenard de LPREditor pour leur confiance, leur esprit critique et leur soutien pendant la réalisation des projets que nous avons réalisé ensemble.

Je tiens à remercie tous mes copains du laboratoire, ceux qui sont toujours là et ceux qui sont déjà partis.

Pour leurs encouragements et leur assistance aussi bien matérielle que morale qui m'ont permis de faire cette thèse dans de bonnes conditions, je remercie chaleureusement à ma famille, Rodo, Merce, Seba, Mariano, Ceci, " los Churin " et " los Ramos "; à mes amis à Paris, Pablo, Cecilia, Mariela, Ramiro, Bérénice, Juan, Candela, Olivia ; à mes amis d'Angers, Isabelle, François, Lisa, Emmanuelle ; à mes amis de la chorale, Flora, Chlöé, Jerôme et Clément ; et mes amies d'Argentine, Lucio et Gabriel.

Je dédie cette thèse à mes deux amours :Agustin et Priscila.

Résumé

Cette thèse est dédiée à l'étude de méthodes de vision artificielle pour la détection et la reconnaissance d'objets structurés, plus précisément les véhicules automobiles. Le corps de la thèse est divisé en deux parties principales.

La première partie est vouée à la détection de véhicules sur des scènes routières à l'aide d'un système embarqué de vision monoculaire. La stratégie utilisée se fonde sur une cascade de classifieurs de type Adaboost, inspirée des travaux de Viola et Jones. Deux familles de descripteurs sont évaluées : les filtres de Haar et les histogrammes de gradients orientés. Les premiers sont associés à une fonction de classification discriminante et les seconds, à une fonction de classification générative. A partir de leur concaténation, nous avons obtenu un nouveau détecteur Mixte. Les résultats ont démontré qu'il est plus performant et conserve les avantages de chaque approche. Nous avons proposé aussi des méthodes pour classifier les véhicules détectés en trois classes : voiture de tourisme, camionnettes et camions.

La deuxième partie est consacrée à la reconnaissance du type d'un véhicule (constructeur, modèle) à partir de sa vue de face. L'application principale visée est le contrôle d'accès dans des parkings, péages d'autoroutes, etc. Nous proposons un nouveau système de reconnaissance multi-classes qui utilise un descripteur visuel local, à base de pixels de contour orientés. La classification est obtenue à partir d'une méthode de votes, robuste aux occultations partielles. Plusieurs tests sur une base d'images prises en conditions réelles ont été réalisés : le taux de reconnaissance obtenu pour 20 classes de véhicules dépasse les 94 %.

Mots-Clés

Reconnaissance de formes, Apprentissage Automatique, Détection de véhicules, Filtres de Haar, Histogrammes de Gradient Orienté, Cascade de Classifieurs, Adaboost, Reconnaissance de type de Véhicule, Pixels de Contour Orientés, Méthode de Votes.

Detection and Recognition of structured objects : Application to Intelligent Transport Systems

Abstract

This dissertation aims the research of computer vision algorithm for object detection and recognition, in particular vehicles.

The first part addresses the on-road vehicle detection problem through an on-board monocular vision system. The detector uses a cascade of boosted classifiers, inspired in Viola and Jones' works. Two features are evaluated : Haar filters and Histograms of Oriented Gradient. The former are associated to discriminative classifiers, while the latter to generative classifiers. A new detector is obtained from their fusion, improving the results and the performance of the cascade architecture. Moreover, we propose a classification of detected vehicles in three classes : passenger cars, vans and trucks.

The second part analyses algorithms for vehicle make and model recognition. The principal application involves access control to parking, tollgates, etc. We propose a new voting algorithm based on oriented-contour points. The system obtains a classification rate above 94 % for a 20 different models dataset. Results also show the method is robust to partial occlusions.

Keywords

Pattern Recognition, Machine Learning, Vehicle Detection, Haar-like filters, Histogram of Oriented Gradient, Cascade of Classifiers, Adaboost, Vehicle Make and Model Recognition, Oriented-Contour Points, Voting Algorithm.

Table des matières

Table des figures

Liste des tableaux

INTRODUCTION GENERALE

La reconnaissance de formes est entrée en douceur dans le quotidien des gens. Qu'ils s'en aperçoivent ou pas, cette branche de l'apprentissage automatique est utilisée pour reconnaître leurs voix, leurs traits, leurs mouvements, ainsi que pour interpréter d'autres signaux du monde physique. Parmi ces applications, nous trouvons celles associées aux deux aspects de la sécurité des transports : la sécurité routière et la sécurité publique. Les investissements économiques de grandes entreprises et les arrêtés législatifs de gouvernements n'y connaissent pas de limites, étant donné qu'il s'agit de préserver la vie dans le premier cas, et la propriété privée, dans le second.

Cette thèse est dédiée à la recherche de méthodes de reconnaissance de formes dans ces domaines en ayant comme vedette principale : **les véhicules**.

Sécurité routière

L'Argentine, mon pays d'origine, détient un triste record au niveau de ses statistiques de sécurité routière. Ainsi, sept milles personnes meurent chaque année, ce qui représente plus de 2.5 % du total des morts en Argentine et la quatrième cause de mortalité après les maladies cardio-vasculaires, le cancer et les maladies cérébrales. D'un point de vue économique, les coûts estimés des accidents de circulation varient entre un et deux pour cent du Produit Intérieur Brut, suivant qu'on inclut ou non les coûts indirects. Cette problématique est donc un sujet extrêmement grave et le nombre d'accidentés de la route (blessés ou morts) comme les pertes matérielles permettent de le considérer comme une véritable catastrophe.

La réponse politique s'est concrétisée en 2006 par la présentation au Sénat d'un Projet National de Sécurité Routière (Plan Nacional de Seguridad Vial) signé par le président de la Nation. Il établit, comme politique d'Etat, une stratégie de lutte contre l'insécurité routière à court et moyen terme et crée un Centre National de Sécurité Routière (Sistema Nacional de Seguridad Vial). Ce centre est responsable de l'exécution uniforme, constante et organisée des politiques de transport, de trafic et de sécurité routière.

Il est intéressant d'analyser les statistiques présentées dans le Projet, faites sur la base des données mondiales collectées pendant l'année 2003. Les résultats placent l'Argentine à la treizième place pour le nombre de morts par dizaine de milliers de véhicules, à la quinzième pour le nombre de morts par centaine de milliers d'habitants. La France est

dix-septième pour le premier indice et vingt-deuxième pour le second.

Nous allons utiliser les données officielles [1] afin d'actualiser ces indices pour l'année 2006, dans le tableau . Cette actualisation permettra de situer la France, au-delà de la trentième place et l'Argentine dans le *Top Ten*.

	Argentine	France
Nb tués	7859	4709
Nb blessés	96374	102125
Parc automobile (milliers)	11826	30400
Population (millions)	39.3	61.4
Victimes / 10000 véhicules	6.64	1.54
Victimes / 100000 habitants	19.99	7.66

TABLE 1 – Statistiques des accidents routiers en Argentine et en France pour l'année 2006. Note : le nombre de tués est calculé dans les trente jours suivant l'accident).

Le Centre d'Expérimentation et de Sécurité Routière (Centro de Experimentatión y Seguridad Vial - CESVI), institution créée en Argentine par les compagnies d'assurances et dédiée à la recherche appliquée à l'industrie automobile, réalise des statistiques actualisées sur les causes d'accidents. Il a l'énorme avantage de s'appuyer sur les informations fournies par les expertises réalisées pour chaque accident routier.

Les statistiques présentées dans la figure 1 illustrent différentes analyses effectuées sur les accidents routiers. Le plus marquant est celui qui montre que le facteur humain est à l'origine de 90 % des accidents. Les statistiques montrent aussi qu'il y a plus d'accidents sur les routes droites que dans les courbes ; probablement à cause de la vitesse plus grande ou des dépassements potentiels.

L'analyse des statistiques nous permet de constater que les accidents se produisent pendant des conditions que nous pouvons appeler *nominales* : accidents diurnes sur des routes droites sur des chaussées sèches. Cette contradiction révèle que les conducteurs améliorent naturellement sa conduite pendant les conditions que nous allons appeler *non nominales* : la pluie, le brouillard, la visibilité réduite, etc. Ainsi, ils réduisent leur vitesse et respectent les distances inter-véhicules entre autres. Contrairement, pendant les conditions nominales, un excès de confiance peut amener les automobilistes à un déficit d'attention et à commettre des erreurs tragiques.

La figure 2 présente les erreurs humaines les plus couramment commises lors d'accidents. Le dépassement des véhicules (invasion de voie) représente ainsi la principale cause d'accidents. La fatigue et la distraction sont deux autres facteurs. Plusieurs de ces erreurs peuvent êtres dues à un déficit dans l'éducation. Aussi, le Projet envisage une formation

1. L'Observatoire National Interministériel de Sécurité Routière pour la France, et le Registro Nacional de Antecedentes de Transito pour l'Argentine

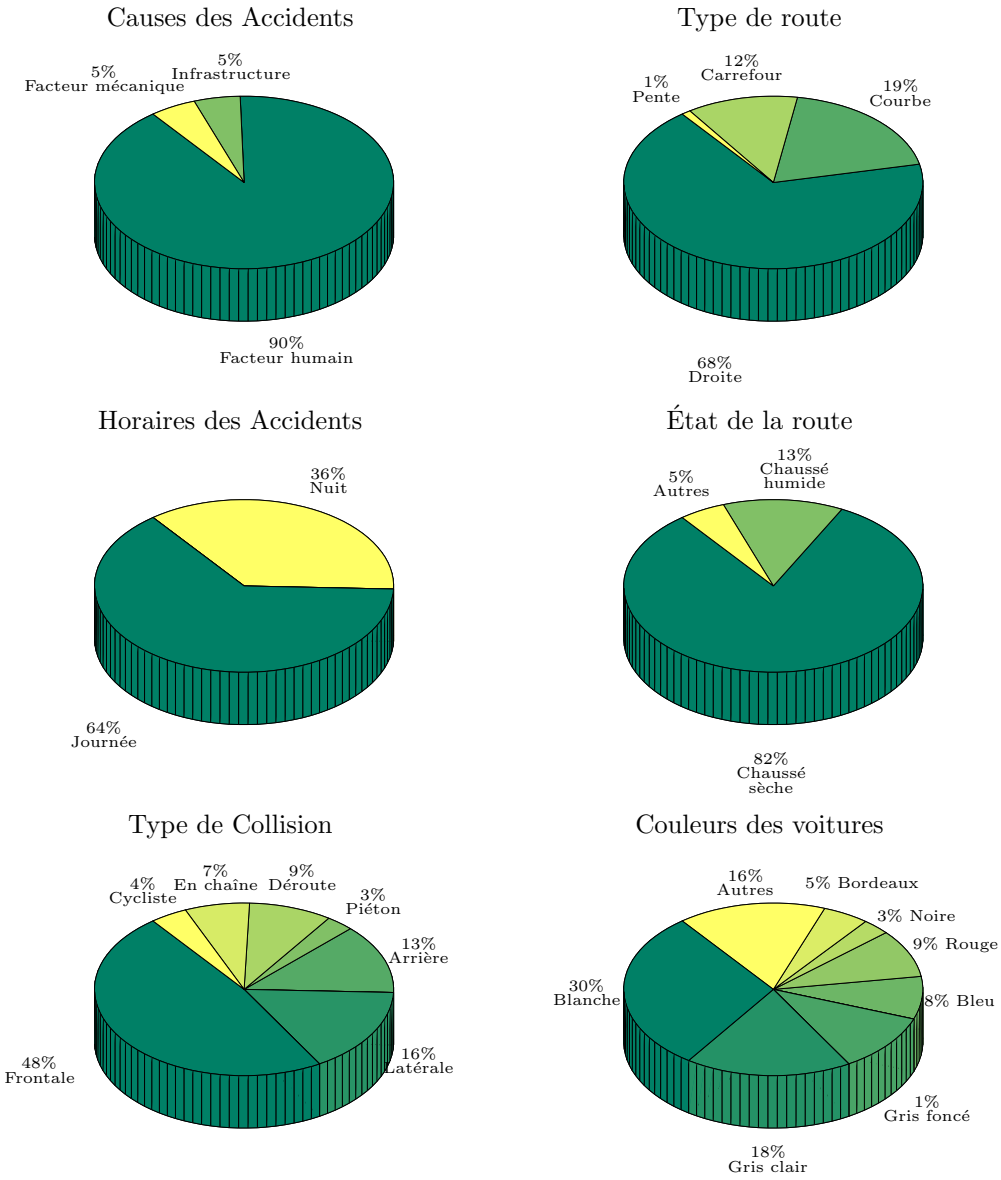


FIGURE 1 – Statistiques sur les accidents de voitures avec morts ou blessés graves.

au code de la route à l'école primaire, secondaire ainsi qu'à l'université. En complément, l'utilisation de systèmes d'aide à la conduite peuvent être proposés. Ces systèmes, embarqués dans les véhicules, pourraient avertir le conducteur d'une situation potentiellement dangereuse, à partir d'une analyse de l'environnement du véhicule : positions et vitesses des autres véhicules, obstacles, etc. Ils seraient capables de détecter des vitesses et distances inter-véhicules inappropriées, des invasions de voie et des dépassements déconseillés. Réagir à ces situations en évitant les inattentions des automobilistes couvre une grande partie des causes d'accidents montrées dans la figure 2.

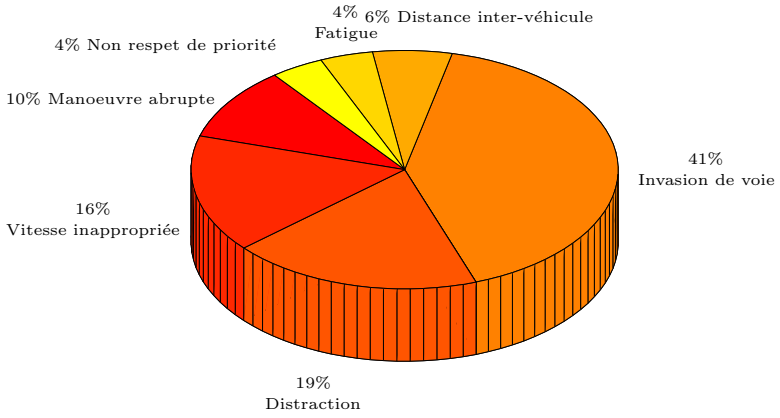


FIGURE 2 – Causes des accidents (facteur humain).

Sécurité publique ou individuelle

Aujourd'hui, en France, le bilan établi par le Ministère de l'Intérieur sur l'application de la loi du 23 janvier 2006 relative à la sécurité nationale, permet de constater un nombre important d'avancées. Les articles 1 et 2 de cette loi autorisent le recours à la vidéosurveillance comme outil de prévention des actes de terrorisme, et désormais, il existe des normes techniques qui permettent l'exploitation de l'image lors d'enquêtes judiciaires. Il s'agit de la vidéosurveillance, mais aussi des communications électroniques ou téléphoniques, du traitement des données nominatives des passagers aériens, et de la lecture automatisée des plaques d'immatriculation.

Ces mesures dotent la France de nouveaux instruments efficaces et d'ores et déjà en application. L'article 8, de la loi précédente, prévoit aussi l'utilisation d'un dispositif de Lecture Automatisée des Plaques d'Immatriculation, dit LAPI.

La Lecture Automatique de Plaques d'Immatriculation est un tout nouveau système de surveillance permettant notamment d'identifier les véhicules volés, de rechercher des véhicules suspects ou encore de surveiller des passages aux frontières et aux péages. Il peut être utilisé en station fixe ou embarqué sur un véhicule (voir figure 3). Le système de Lecture Automatisée des Plaques d'Immatriculations (LAPI) est en cours d'expérimentation depuis 2007, sur l'avis de la Commission nationale pour l'informatique et les libertés [2] (CNIL), à partir de six véhicules de la police équipés, dans les villes de Marseille et Saint-Denis.

Depuis le début de cette expérimentation, plus d'1,2 million de plaques ont été lues et comparées au Fichier des Véhicules Volés, 65 véhicules mis sous surveillance ont été détectés et 76 interpellations ont été effectuées [3]. Ce bilan encourage les autorités à géné-

2. `http://www.cnil.fr/index.php?id=2204\&news[uid]=438\&c`
3. `http://www.interieur.gouv.fr/misill/sections/a\_l\_interieur/le\_ministre/interventions/bilan-application-loi-23-01-2006/view`

raliser le programme sur l'ensemble du territoire. Dans d'autres pays d'Europe ce type des systèmes a déjà été adopté par les services de sécurité, comme par exemple en Angleterre, et sera prochainement mis en oeuvre aux États Unis.

FIGURE 3 – Installation des caméras sur le toit des véhicules de police.

Nous allons finir cette introduction avec une application qui concerne les deux précedentes et qui est très connue des automobilistes en France : les radars automatisés de contrôle de vitesse.

Inaugurés en 2003 pour le Ministre de l'Intérieur, Nicolas Sarkozy, ils déclenchent automatiquement un dispositif de prises de vues numériques en cas d'excès de vitesse. Les images sont traitées par un système LAPI qui permet d'identifier le titulaire de la carte grise à travers le fichier national des immatriculations.

Une vérification postérieure effectuée par un opérateur n'a pas empêché certaines erreurs du système : un tracteur qui roulait à plus de 100 km/h, une Citroën de 1930 flashée à 140 km/h, un automobiliste qui reçoit un procès verbal alors qu'il se trouvait chez lui au moment de l'infraction, etc [4]. Ce type d'erreurs peut saper la crédibilité du public en ces outils de sanction, par ailleurs très efficaces.

Organisation du mémoire

Nous avons présenté précédemment, le fort engagement des différents gouvernements pour ces deux aspects de la sécurité : la sécurité routière et la sécurité publique.

La vision artificielle et la reconnaissance de formes s'avèrent dans ce contexte applicatif, des outils avantageux qui seront abordées tout au long de ce manuscrit

La sécurité routière et la sécurité publique ont été traitées dans le cadre de deux contrats industriels entre l'Institut des Systèmes Intelligents et de Robotique (ISIR, ex LISIF-PARC) d'une part et deux sociétés : PSA Peugeot Citroën (entre janvier 2006 et mai 2008), pour le premier, et LPREditor (entre septembre 2004 et décembre 2005), pour le deuxième.

4. `http://www.radars-auto.com/humour\_radar\_rates.php`

Projet PSA

Le sujet du premier projet s'intéressait à l'optimisation des solutions embarquées de perception de l'environnement des véhicules, dans un but d'aide à la conduite et de confort. Il consistait à étudier différents algorithmes d'analyse d'images pour la détection de véhicules, en vue de la conception de systèmes anticollisions. Dans le cahier de charges ils étaient définis les points suivants pour la réalisation du projet :

- Les images en niveaux de gris de la route étaient fournies par un système de vision monoculaire embarqué.
- Une détection était définie comme l'information binaire indiquant la présence ou non d'un véhicule obstacle dans une région de l'image.
- La plage de distances considérée variait de 15 à 60 mètres.
- La base d'images, fournie par PSA Peugeot Citroën, était constituée de plus de 1000 scènes différentes (nous considérons que 2 scènes séparées par 1 minute sont différentes). La détection devait se faire sans utilisation de l'information temporelle.
- Il demandait une étude du comportement du système face à des situations diverses d'éclairage ou dans conditions climatiques peu favorables.
- De plus, il fallait fournir une information additionnelle concernant le type de véhicule détecté (véhicule de tourisme, camionnette, camion, etc)

Projet LPREditor

Le deuxième projet était consacré à la conception d'un classifieur à partir d'une unique image noir et blanc de l'avant d'un véhicule. Dans plusieurs applications d'identification de véhicules ou de contrôle des flux, mettant en jeu un capteur d'image, il est utile de déterminer le type des véhicules présents. Ce type de solution intéressait à l'entreprise LPREditor qui se dédie au développement des systèmes LAPI. L'association d'un système de reconnaissance de marque et modèle d'une voiture et d'un système LAPI peut augmenter la robustesse de ce dernier et mieux individualiser un véhicule afin d'éviter les erreurs.

Première partie

Projet PEUGEOT - Méthode de dopage pour la détection et classification de véhicules

Introduction

La progression du parc automobile est accompagnée par une demande croissante de systèmes d'aide à la conduite, promettant une augmentation du confort et de la sécurité [65]. Dans cette optique, diverses recherches ont été menées par la communauté dite des Transports Intelligents. Ceci se traduit par l'installation de dispositifs de hautes technologies et d'autres systèmes de commande dans les véhicules ainsi que sur la route. Nous pouvons classer ces systèmes dans différentes catégories dont les *Advanced Vehicle Highway Systems*(AVHS), *Advanced Safety Vehicle* (ASV) et *Advanced Driver Assistance Systems* (ADAS).

Concrètement, un système d'aide à la conduite peut aider les conducteurs dans différentes situations :

- Convoi de véhicules : cela consiste à asservir un ou plusieurs véhicules esclaves sur un véhicule de tête nommé maître ou leader. Cette application permettra d'optimiser le débit de circulation sur une autoroute.
- *Stop&Go* : il s'agit d'une situation similaire à la précédente, mais appliquée en environnement urbain et à faibles vitesses. Le véhicule porteur est conduit automatiquement dans un embouteillage ; quand la vitesse augmente, le conducteur reprend la main.
- Angle aveugle de perception : étant donné que les rétroviseurs ne parviennent pas à refléter tous les angles de l'environnement, des systèmes de sécurité peuvent avertir le conducteur sur des situations qui posent un risque de collision (d'autres véhicules en phase de dépassement).
- Supervision des manoeuvres : c'est le cas le plus complexe car le système doit détecter les véhicules présents dans la scène et réaliser leur suivi, qu'ils circulent dans la même direction que le véhicule porteur ou dans la direction contraire.

La route automatisée

Les deux premières situations entrent dans un scénario du type *Automated Highway System* (AHS). Il consiste en une approche où les systèmes de coopération route-véhicule, les systèmes de communication inter-véhicules, ainsi que les véhicules autonomes assistent

[65] S. Han, E. Ahn, and N. Kwak. Detection of multiple vehicles in image sequences for driving assistance system. In *International Conference on Computational Science and Its Applications*, volume 3480, pages 1122–1128, 2005.

le conducteur pour le décharger de certaines tâches, en plus d'accroître la sécurité. Tous les véhicules sont équipés d'une liaison radio afin de s'informer mutuellement sur leurs positions relatives par rapport à la route. L' *Advanced Cruise-Assist Highway System Research Association* (AHSRA), crée au Japon en 1996, est une association qui conduit un programme de recherche dans ce domaine pour incorporer plus de technologie à l'autoroute. En 1997, le consortium *National Automated Highway System* (NAHSC) a organisé aux États Unis une démonstration pour montrer la faisabilité des systèmes d'autoroute automatisée avec la participation des laboratoires de recherches et d'importants industriels. Nous pouvons citer le nouveau projet européen *Cooperative Vehicle-Infrastructure Systems* (CIVS) qui vise au développement d'un système complet de communication entre les véhicules et l'infrastructure routière.

Les véhicules intelligents

Cependant, un scénario plus réaliste est la situation de conduite en trafic mixte, avec des véhicules équipés coexistant avec des voitures non équipées. Dans ce cas, une solution basée essentiellement sur un système radio n'est pas viable. Les véhicules devront être équipés d'autres systèmes de perception de leur environnement. Face aux situations critiques, le système sera en mesure d'informer le conducteur ou de prendre temporairement le contrôle du véhicule [121]. Il est aussi envisageable de contrôler le véhicule lors de tâches monotones, dans le but d'augmenter le confort du conducteur. Au Japon, les recherches ont démarré sur la base des véhicules prototypes construits en Europe durant les années 80 et 90. Un effort conjoint entre le *Ministry of International Trade and Industry (Japon)* (MITI), Nissan et Fujitsu a donné naissance au premier grand projet japonais dans le domaine (appelé *Personal Vehicle System*). Plus récemment, un prototype *Smartway* a été planifié pour être opérationnel en 2003 et sur la chaîne de production en 2015.

En 1997 le gouvernement des États Unis lançait la *Intelligent Vehicle Initiative* (IVI) dont l'objectif est d'accélérer le développement, la disponibilité et l'utilisation des systèmes embarqués d'aide à la conduite. Nous y trouvons, par exemple, le groupe CMU Navlab, ayant une très longue histoire dans le développement des véhicules intelligents, et qui a produit une série de 11 prototypes opérationnels, du Navlab 1 jusqu'au Navlab 11. D'autre part, le département des transports de ce pays a lancé un projet avec General Motors pour développer et tester des systèmes d'évitement d'obstacles.

La recherche dans le domaine des véhicules intelligents, a débuté en Europe en 1986 avec le projet PROMETHEUS qui regroupait les constructeurs automobiles européens et de nombreux laboratoires de recherche. Un autre projet, *VIsion Technology Application (VITA)*, était la collaboration entre le laboratoire de M. von Seelen et le constructeur Daimler-Benz. Nous pouvons citer plusieurs prototypes de voitures intelligentes issus de ce projet : le véhicule VaMoRs, VaMp, MOB-LAB. Ce dernier utilise un système de

[121] U. Regensburger and V. Graefe. Visual recognition of obstacles on roads. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 980–987, Septembre 1994.

détection des obstacles et des lignes blanches appelé GOLD [17]. L'équipe du projet italien ARGO a conçu un système de sécurité actif agissant comme un co-pilote automatique. Ce système, implémenté dans un Lancia Thema 2000, permet une conduite autonome sur grands axes et autoroutes. Plus récemment, le laboratoire LITIS-PSI [5] s'est lancé dans un projet de véhicule intelligent similaire à partir d'un Renault Laguna II. Par ailleurs, Daimler-Chrysler a développé un système de vision stéréo embarqué pour la détection des piétons et des véhicules obstacles. Le système, appelé *Urban Traffic Assitant* (UTA), a été testé sur des véhicules du type Mercedes-Benz Classe E.

En France, le projet de recherche ARCOS (Action de Recherche pour une Conduite Sécurisée) débutait en 2001 et associait 58 équipes françaises de recherche. Les recherches s'articulaient sur quatre fonctions de préventions d'accidents : gérer les distances entre véhicules, prévenir les collisions, prévenir des sorties de route ou de voie et alerter les véhicules en amont d'incidents ou d'accidents.

Le projet concernait le développement d'un système véhicule-conducteur-infrastructure s'appuyant sur quatre fonctions ou couches principales (voir figure 4) :

1. Environnement coopératif, utilisation des satellites, caméras, etc.
2. Localisation submétrique, à travers des bases de données routières.
3. Détection et perception, systèmes extéroceptives de perception pour de véhicules intelligents.
4. Communication, moyens de communication inter-véhicules et véhicule-infrastructure.

L'installation de différents types de capteurs et la fusion des informations qu'ils obtiennent, permet au véhicule d'obtenir une description de son environnement. Nous pouvons citer, entre autres, les radars, les lasers, les systèmes d'ultrasons et les caméras embarquées. Nous allons nous concentrer sur ces dernières qui ont retenu l'attention des constructeurs automobiles et des laboratoires de recherche ces derniers années [40, 16,

5. Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes

[17] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino Lo Bianco, and A. Piazzi. Visual perception of obstacles and vehicles for platooning. *IEEE Transactions on Intelligent Transportation Systems*, 1(3) :164–176, 2000.

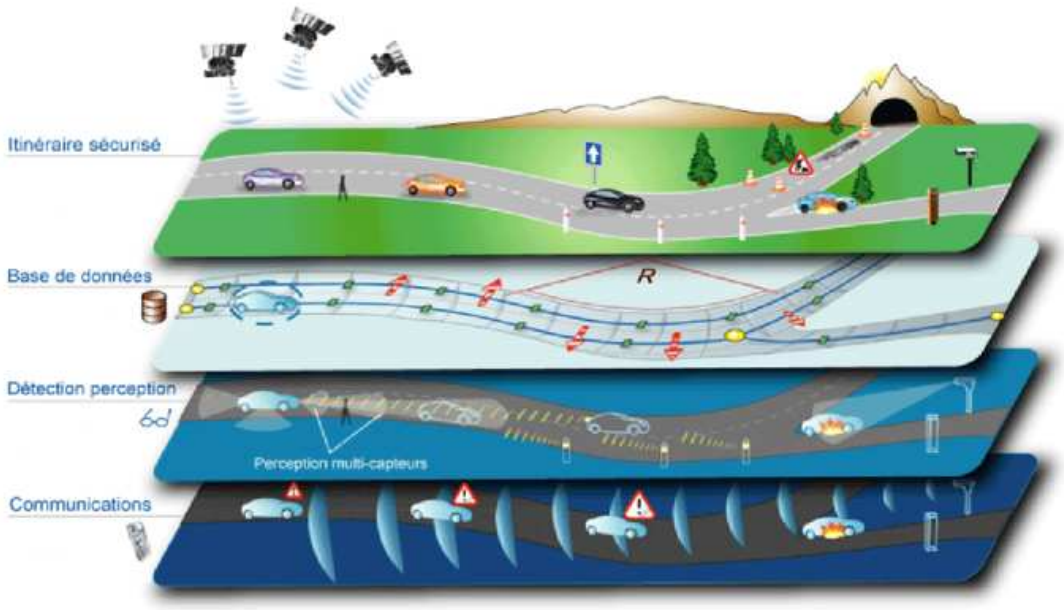


FIGURE 4 – Système ARCOS d'aide à la conduite.

129, 150, 37, 139, 121]. Un système de vision installé dans le véhicule porteur, fournit une description en temps réel de la localisation et de la taille des autres véhicules dans l'environnement, ainsi que de la route, des panneaux de signalisation et des autres usagers de la voirie. Il peut ainsi venir à l'aide d'un système, par exemple composé de radar, qui n'est pas capable d'identifier la nature d'un obstacle.

Dans les chapitres suivants, nous allons décrire divers systèmes de vision embarquée conçus pour la détection de véhicules dans la route.

---

[40] C. Demonceaux, A. Potelle, and D. Kachi-Akkouche. A multi-cameras 3d volumetric method for outdoor scenes : a road traffic monitoring application. *IEEE Transactions on Vehicular Technology*, 53(6) :1649–1656, November 2004.

[16] M. Betke, E. Haritaoglu, and L.S. Davis. Multiple vehicle detection and tracking in hard real time. In *Intelligent Vehicles Symposium*, pages 351–356, 1996.

[129] M.A. Sotelo, M.A. Garcia, and R. Flores. Vision based intelligent system for autonomous and assisted downtown driving. In *International Workshop on Computer Aided Systems Theory*, volume 2809, pages 326–336. Springer, 2003.

[150] T. Xiong and C. Debrunner. Stochastic car tracking with line- and color-based features. *Advances and Trends in Research and Development of Intelligent Transportation Systems : An Introduction to the Special Issue*, 5(4) :324–328, Decembre 2004.

[37] F. Dellaert. Canss : A candidate selection and search algorithm to initialize car tracking. Technical report, Robotics Institute, Carnegie Mellon University, 1997.

[139] N. Trujillo, R. Chapuis, F. Chausse, and C. Blanc. On road simultaneous vehicle recognition and localization by model based focused vision. In *IAPR Conference on Machine Vision Applications*, pages 388–391, Mai 2005.

[121] U. Regensburger and V. Graefe. Visual recognition of obstacles on roads. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 980–987, Septembre 1994.

Chapitre 1

Détection de véhicules : État de l'art

En vision artificielle et traitement d'images, les systèmes de détection d'objets exigent, en général, des algorithmes complexes. Une recherche exhaustive dans une image complète peut être envisagée pour des applications de type *off line* ou d'autres où le temps réel n'est pas une contrainte.

Les systèmes de détection des obstacles embarqués, de leur coté, doivent être conçus pour évaluer leur environnement à une vitesse telle que, dans le cas du déclenchement d'une alarme, le conducteur puisse avoir un temps de réaction. Nous parlons évidemment, de temps de réponse de l'ordre d'une fraction de seconde.

Pour accélérer au maximum les traitements sur les images fournies par les caméras embarquées, la plupart des méthodes dans la littérature considèrent deux étapes (voir figure 1.1) :

- la Génération des Hypothèses (GH) : le système rends de façon relativement rudimentaire et surtout rapide des positions potentielles de véhicules ou régions d'intérêt (ROI). Ceci restreint le champ de recherche pour l'étape suivante.
- la Validation des Hypothèses (VH) : les hypothèses issues de l'étape antérieure sont validées en utilisant des algorithmes plus complexes et les fausses sont éliminées.

Lors de notre revue de la littérature, nous avons pu constater que ces deux étapes peuvent se confondre en un seul algorithme, ce qui n'empêche pas l'utilisation de GH pour accélérer la détection.

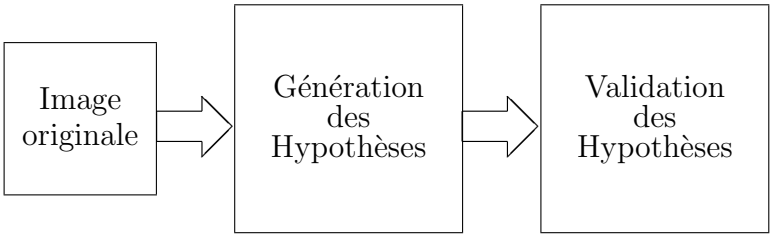


FIGURE 1.1 – Synopsis d'un système de détection de véhicules.

1.1 Génération des Hypothèses

Les approches proposées pour la Génération des Hypothèses dans la littérature peuvent être classées en trois catégories [133] :

– utilisant des primitives : ces méthodes utilisent une (ou des) information(s) *a priori* pour présumer de la localisation d'un véhicule dans une image. Les informations utilisées dans cette étape peuvent être la texture [19, 66], la symétrie horizontale [12, 158], la couleur [150, 62, 36, 68], l'ombre portée [119, 141, 103], les segments

---

[133] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection : A review. *IEEE Transanctions on Pattern Analalysis and Machine Intelligence*, 28(5) :694–711, 2006.

[19] T. Bucher, C. Curio, J. Edelbrunner, C. Igel, D. Kastrup, I. Leefken, G. Lorenz, A. Steinhage, and W von Seelen. Image processing and behavior planning for intelligent vehicles. *IEEE Transactions on Industrial Electronics*, 50(1) :62–75, Febrary 2003.

[66] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. Computer vision for driver assistance systems. In *SPIE Congress on Signal processing, sensor fusion, and target recognition*, volume 3364, pages 136–147, 1998.

[12] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet. A cooperative approach to vision-based vehicle detection. In *Proceedings on Intelligent Transportation Systems*, pages 207–212, Août 2001.

[158] T. Zielke and J. GroSS. Advances in computer vision-based obstacle detection for cars. In *In Proceedings 27th International Symposium on Automotive Technology and Automation*, pages 431–436, 1994.

[150] T. Xiong and C. Debrunner. Stochastic car tracking with line- and color-based features. *Advances and Trends in Research and Development of Intelligent Transportation Systems : An Introduction to the Special Issue*, 5(4) :324–328, Decembre 2004.

[62] D. Guo, T. Fraichard, M. Xie, and C. Laugier. Color modeling by spherical influence field in sensing drivingenvironment. In *IEEE Intelligent Vehicles Symposium*, pages 249–254, 2000.

[36] E. De Micheli, R. Prevete, G. Piccioli, and M. Campani. Color cues for traffic scene analysis. In *Intelligent Vehicles Symposium*, pages 466–471, 1995.

[68] W. Heisele, B. end Ritter. Obstacle detection based on color blob flow. In *IEEE Intelligent Vehicles Symposium*, pages 282–286, Septembre 1995.

[119] A. Prati, I. Mikic, R. Cucchiara, and M. M. Trivedi. Analysis and detection of shadows in video streams : A comparative evaluation. In *IEEE Computer Vision and Pattern Recognition Workshop on Empirical Evaluation Methods in Computer Vision*, volume 2, Decembre 2001.

[141] M.B. van Leeuwen and F.C.A. Groen. Vehicle detection with a mobile camera. Technical report, Computer Science Institute, University of amsterdam, The Netherlands, Octobre 2001.

[103] H. Mori and N. Charkai. Shadow and rythm as sign patterns of obstacle detection. In *International Symposium on Industrial Electronics*, pages 271–277, 1993.

[37, 99, 136], la route [90, 25, 80, 28, 9], les coins [14], etc.
- utilisant la stéréo-vision : ils existent essentiellement deux types de méthodes qui utilisent l'information stéréo pour la détection des véhicules ; le plan de disparités [49, 97, 48] et la transformée *Inverse Perspective Mapping* (IPM) [110, 156, 13].
- utilisant le mouvement : il s'agit des méthodes qui utilisent l'information de la séquence des images, c'est-à-dire le mouvement [40, 16, 127].

---

[37] F. Dellaert. Canss : A candidate selection and search algorithm to initialize car tracking. Technical report, Robotics Institute, Carnegie Mellon University, 1997.

[99] N. D. Matthews, P. E. An, D. Charnley, and C. J. Harris. Vehicle detection and recognition in greyscale imagery. *Control Engineering Practice*, 4(4) :473–479, Avril 1996.

[136] F. Thomanek, E.D. Dickmanns, and D Dickmanns. Multiple object recognition and scene interpretation for autonomous road vehicle guidance. In *IEEE Intelligent Vehicles Symposium*, pages 231–236, Octobre 1994.

[90] Q. Li, N. Zheng, and H. Cheng. Springrobot : A prototype autonomous vehicle and its algorithms for lane detection. *Intelligent Transportation Systems*, 5(4) :300–308, Decembre 2004.

[25] R. Chapuis, R. Aufrere, and F. Chausse. Accurate road following and reconstruction by computer vision. *IEEE Transactions on Intelligent Transportation Systems*, 3(4) :261–270, December 2002.

[80] T. Kato, Y. Ninomiya, and I. Masaki. An obstacle detection method by fusion of radar and motion stereo. *Intelligent Transportation Systems*, 3(3) :182–188, Septembre 2002.

[28] Sung Yug Choi and Jang Myung Lee. Applications of moving windows technique to autonomous vehicle navigation. *Image and Vision Computing*, 24(2) :120–130, 2006.

[9] M. Beauvais and S Lakshmanan. Clark : a heterogeneous sensor fusion method for finding lanes and obstacles. *Image and Vision Computing*, 18(5) :397–413, 2000.

[14] M. Bertozzi, A. Broggi, and S. Castelluccio. A real-time oriented system for vehicle detection. *Journal of Systems Architecture*, 43(1-5) :317–325, 1997.

[49] U. Franke and I. Kutzbach. Fast stereo based object detection for stop&go traffic. In *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, pages 339–344, Septembre 1996.

[97] R. Mandelbaum, L. McDowell, L. Bogoni, B. Reich, and M. Hansen. Real-time stereo processing, obstacle detection, and terrain estimation from vehicle-mounted stereo cameras. In *IEEE Workshop on Applications of Computer Vision*, pages 288–289, Octobre 1998.

[48] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *Proceedings IEEE Intelligent Vehicles Symposium 2000*, pages 273–278, 2000.

[110] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4) :353–363, 1993.

[156] G. Zhao and S. Yuta. Obstacle detection by vision system for an autonomous vehicle. In *Intelligent Vehicles Symposium*, pages 31–36, 1993.

[13] M. Bertozzi and A. Broggi. Vision-based vehicle guidance. *Computer*, 30(7) :49–55, 1997.

[40] C. Demonceaux, A. Potelle, and D. Kachi-Akkouche. A multi-cameras 3d volumetric method for outdoor scenes : a road traffic monitoring application. *IEEE Transactions on Vehicular Technology*, 53(6) :1649–1656, November 2004.

[16] M. Betke, E. Haritaoglu, and L.S. Davis. Multiple vehicle detection and tracking in hard real time. In *Intelligent Vehicles Symposium*, pages 351–356, 1996.

[127] M. Schmid. An approach to model-based 3-d recognition of vehicles in real time by machine vision. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, volume 3, pages 2064–2071, Septembre

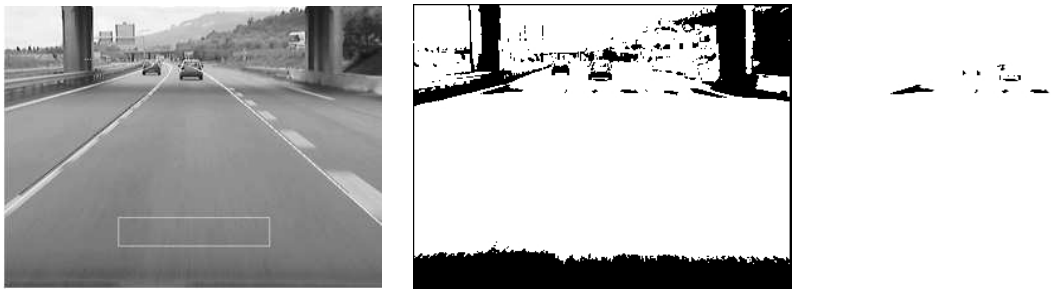
Les images de nos bases ne sont ni en stéréo ni sous forme de séquence (vidéo), elles ont été traitées en conséquence avec la première famille de méthodes. L'analyse du mouvement ne sera pas *a priori* utilisée dans notre système : nous allons travailler avec des images statiques pour réaliser la détection de véhicules.

Dans les sections suivantes, nous allons évoquer les indices de présence d'un véhicule et leur utilisation dans la littérature.

Ombres portées

La méthode de génération des hypothèses à partir des ombres portées considère la route comme une zone de couleur homogène. Une voiture qui se situe dans la route projette une ombre sur celle-ci, ce qui dans une image assombrit les pixels correspondants. Ces pixels peuvent être détectés en considérant qu'ils sont plus sombres que le modèle de couleur de la route.

Pour créer le modèle de la route, Clady [29] et Han [65] proposent d'utiliser un rectangle de la zone basse de l'image (supposée appartenir à la route). Ce rectangle sert pour trouver une moyenne des valeurs en niveaux de gris [29] ou une valeur moyenne en couleur dans l'espace HSV [65]. Après le seuillage, l'étape suivante consiste à appliquer un filtre pour éliminer toutes les ombres hors de la route, les ombres de petites tailles et celles trop grandes (voir figure 1.2). Une analyse temporelle de ces zones sur la séquence vidéo permet de filtrer aussi les fausses alarmes.



(a) (b) (c)

FIGURE 1.2 – Méthode des ombres portés.
(a) Image original, (b) image seuillée et (c) image après filtrage.

1994.

[29] X. Clady. *Contribution à la navigation autonome d'un véhicule automobile par vision.* PhD thesis, Université Blaise Pascal, France, 2003.

[65] S. Han, E. Ahn, and N. Kwak. Detection of multiple vehicles in image sequences for driving assistance system. In *International Conference on Computational Science and Its Applications*, volume 3480, pages 1122–1128, 2005.

Handmann [67, 66] propose lui une fusion du Codage des Orientations Locales (LOC) et une analyse des ombres. Pour les LOC, les primitives de l'image sont des lignes qui représentent un code binaire indiquant la direction de la variation des niveaux de gris (dans le voisinage du point considéré). La première étape consiste en l'analyse des niveaux de gris de la route pour trouver un seuil τ qui permet d'extraire les ombres. Ensuite, ils sélectionnent les LOC qui correspondent à une orientation horizontale et à une transition clair-à-obscur (si on traite l'image du bas vers le haut) et les regroupent. Ces pistes doivent respecter certaines contraintes de taille pour conformer des hypothèses pertinentes.

Une idée similaire est utilisée par Khammari [83] à partir d'un gradient horizontal négatif. Dans son travail, il construit une pyramide de trois échelles de l'image pour accélérer les calculs. Dans le troisième niveau (128x96 pixels) ils génèrent les hypothèses à partir d'un filtre de Sobel horizontal. Un seuillage adaptatif, qui prends en compte la taille des véhicules à partir de la géométrie de la route, est utilisé pour trouver les maxima locaux (voir figure 1.3). L'image binaire est finalement analysée pour extraire les segments horizontaux les plus longs, qui seront considérés comme les ombres portées, et éliminer les faux maxima en tenant compte de la perspective.

FIGURE 1.3 – Gradient horizontal négatif.
(a) Image du gradient, (b) Image binaire des maxima locaux.

Détection de la route

La méthode de détection de la route à partir de la couleur correspond à une approche similaire à celle de la méthode des ombres portées. Un modèle de couleur est aussi estimé et, les groupes de pixels qui, ayant une position sur la route, n'appartiennent pas à ce modèle, sont considérés comme obstacles. Différents espaces de couleur sont proposés dans

---

[67] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. An image processing system for driver assistance. In *IEEE International Conference on Intelligent Vehicles*, pages 481–486, 1998.

[66] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. Computer vision for driver assistance systems. In *SPIE Congress on Signal processing, sensor fusion, and target recognition*, volume 3364, pages 136–147, 1998.

[83] A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau. Vehicle detection combining gradient analysis and adaboost classification. In *IEEE International Conference on Intelligent Transportation Systems*, pages 66–71, Vienna, Austria, Septembre 2005.

les travaux de recherche : les niveaux de gris [29], le RGB [33, 123, 20], le HSV [15], les chrominances [123] ou le L*a*b [62]. Une fois obtenue l'image binaire, la dernière étape consiste à regrouper les pixels considérés comme non route en régions connectées qui vont former les ROI.

Symétrie

L'utilisation de la symétrie pour la génération des hypothèses considère qu'un véhicule est caractérisé par une forme rectangulaire artificielle, où ses contours peuvent être trouvés sans difficulté. Il est, en plus, placé dans une région spécifique de l'image, ce qui permet de filtrer les fausses alarmes de l'environnement.

Sotelo [129] restreint la recherche dans une zone rectangulaire qui couvre la partie centrale de l'image (voir figure 1.4) où les voitures ont une grande probabilité d'être présents. Ils obtiennent le plan de symétrie à partir de la couleur et des contours verticaux. Les voitures candidates sont suivies dans une séquence pour filtrer les fausses alarmes.

Les contours verticaux et horizontaux sont utilisés pour Broggi [17] afin de calculer trois plans de symétrie : un premier plan de symétrie vertical, un deuxième plan de symétrie horizontal, et un troisième à partir de leur fusion. Les régions de symétrie des contours sont vérifiées pour trouver la présence de deux coins représentant la partie inférieure de la boîte englobant le véhicule. Des contraintes de perspective et de taille (voir figure 1.5) sont utilisées pour accélérer l'algorithme et filtrer les fausses alarmes.

---

[29] X. Clady. *Contribution à la navigation autonome d'un véhicule automobile par vision*. PhD thesis, Université Blaise Pascal, France, 2003.

[33] J. Crisman and C. Thorpe. Color vision for road following. In *SPIE Conference on Mobile Robots*, volume 1007, page 175, 1990.

[123] J.C. Rojas and J.D. Crisman. Vehicle detection in color images. In *IEEE Conference on Intelligent Transportation System*, pages 403–408, Novembre 1997.

[20] S. D. Buluswar and B. A. Draper. Color machine vision for autonomous vehicles. *International Journal for Engineering Applications of Artificial Intelligence*, 1(2) :245–256, 1998.

[15] M. Betke, E. Haritaoglu, and L. S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12(2) :69–83, Août 2000.

[62] D. Guo, T. Fraichard, M. Xie, and C. Laugier. Color modeling by spherical influence field in sensing drivingenvironment. In *IEEE Intelligent Vehicles Symposium*, pages 249–254, 2000.

[129] M.A. Sotelo, M.A. Garcia, and R. Flores. Vision based intelligent system for autonomous and assisted downtown driving. In *International Workshop on Computer Aided Systems Theory*, volume 2809, pages 326–336. Springer, 2003.

[17] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino Lo Bianco, and A. Piazzi. Visual perception of obstacles and vehicles for platooning. *IEEE Transactions on Intelligent Transportation Systems*, 1(3) :164–176, 2000.

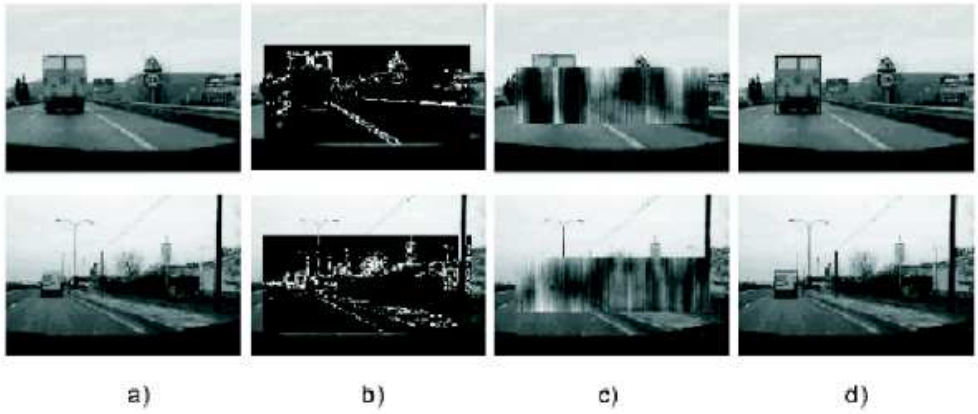
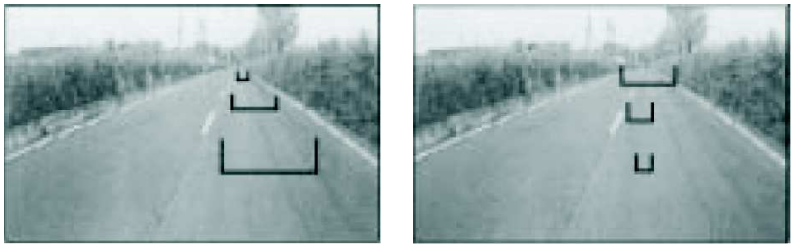


FIGURE 1.4 – Deux exemples de la détection de véhicules utilisant le plan de symétrie. (a) Image originale, (b) contours dans la région centrale de l'image, (c) plan de symetrie, (d) véhicules détectés.



(a) (b)

FIGURE 1.5 – Contraintes de perspective.
(a) Position et taille correctes et (b) limites inférieures incorrectes.

Gradient et Segments

Un exemple représentatif de la méthode de recherche des hypothèses en utilisant les segments de contours est celui de Sun [132]. Il propose une approche à multiples échelles en fusionnant une opération de diminution de la résolution (ré-échantillonnage) et une autre opération de lissage. La taille de l'image originale est réduite de moitié deux fois pour trouver trois résolutions, qui sont montrés dans la première colonne de la figure 1.6. La deuxième et troisième colonnes montrent, respectivement, les contours verticaux et horizontaux après le l'opération de lissage. Ces contours servent pour obtenir les profils des contours montrés dans la dernière colonne de la figure. Les pics et les vallées des profils fournissent une information robuste sur la présence d'un véhicule dans l'image. Les maxima locaux de la plus petite résolution sont projetés dans le niveau supérieur. Ensuite, les résultats sur ce niveau sont suivis dans le dernier niveau où les hypothèses finales sont générées. Les faux pics sont éliminés en utilisant certaines contraintes géométriques. L'approche multiple échelles permet au système d'être peu sensible au choix de paramètres

[132] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2) :125–137, Juin 2005.

FIGURE 1.6 – Méthode de génération à partir des profils de contours.

(les seuils pour les contours et le choix de maximums dans les profils). La formation des premières hypothèses dans le niveau inférieure est très utile, en considérant que ce niveau contient seulement les contours les plus contrastés.

Betke [15] et Dellaert [38] utilisent aussi les profils des contours horizontaux et verticaux. La première utilise cette méthode pour définir la boîte englobant les véhicules qui sont à une distance importante du porteur. Les contours sont trouvés à partir du gradient de la figure, via un seuillage différencié : le seuil pour trouver les contours horizontaux est plus grande (relation de 7/5) par rapport au seuil de contours verticaux. Les travaux de Dellaert comportent un apprentissage à partir d'une base d'images étiquetées. Les profils verticaux et horizontaux composent l'entrée d'un classifieur du type régression non paramétrique qui calcule, pour chaque élément du profil, la probabilité *a posteriori* d'être un segment de la boîte englobant. Pour choisir les meilleures boîtes, il choisit celle qui minimise l'énergie calculée à partir des gradients verticaux et horizontaux.

Gepperth [57] a développé un algorithme qui utilise différemment les segments. Les contours sont trouvés via l'application d'un détecteur de contours du type Canny et une individualisation des segments. Le point milieu de chaque segment est le point de départ pour la recherche des limites verticales latérales afin de trouver un gabarit en "U".

Enfin, Matthews [100] emploie un filtre directionnel pour extraire les contours horizontaux et définir la position des véhicules à partir des régions où il trouve une importante concentration de ces contours. La méthode consiste à compter dans chaque colonne de

---

[15] M. Betke, E. Haritaoglu, and L. S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12(2) :69–83, Août 2000.

[38] F. Dellaert and C. Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on Intelligent Transportation Systems*, volume 3207, Octobre 1997.

[57] A. Gepperth, J. Edelbrunner, and T Bocher. Real-time detection and classification of cars in video sequences. In *IEEE Intelligent Vehicles Symposium*, pages 625–631, Juin 2005.

[100] Neil D. Matthews. *Visual Collision Avoidance*. PhD thesis, University of Southampton, Advanced Systems Research Group, Department of Electronics and Computer Science, Octobre 1994.

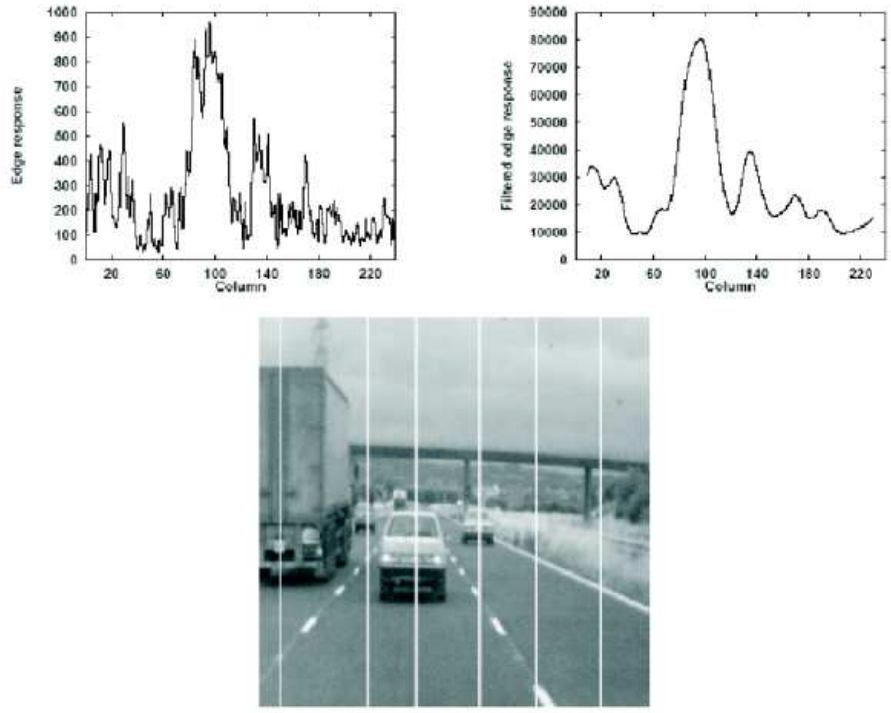


FIGURE 1.7 – Génération des hypothèses à partir des contours horizontaux.

l'image la quantité des contours horizontaux. Ensuite, une lissage de la réponse est appliqué et les maxima locaux sont détectés. Ces pics vont être considérés comme la position centrale des hypothèses (voir figure 1.7). La largeur du véhicule est ensuite définie à partir des segments horizontaux. La position verticale dans la colonne candidate est trouvée en utilisant un modèle de couleur de la route, pour identifier les ombres portées des véhicules. La hauteur du ROI est finalement déterminée suivant une relation proportionnelle avec la largeur.

Texture

La mesure de texture utilisée par Kalinke [78] pour trouver les zones candidates est l'entropie. L'entropie est utilisée comme une mesure de la quantité d'attention qu'il faut donner à une région de l'image. Plus le contenu dans une partie de l'image est incertain, plus l'information contenue est riche et elle requière donc une analyse détaillée. Par exemple, dans une image routière, la chaussée contient une texture très homogène ainsi que le feuillage des arbres. L'entropie (mesure du désordre) est déterminée par :

$$H(k) = -\sum_{k} p(x_k)\ log(p(x_k))$$

[78] T. Kalinke, C. Tzomakas, and W. v. Seelen. A texture-based object detection and an adaptive model-based classification. In *IEEE International Conference on Intelligent Vehicles 1998*, volume 1, pages 143–148, 1998.

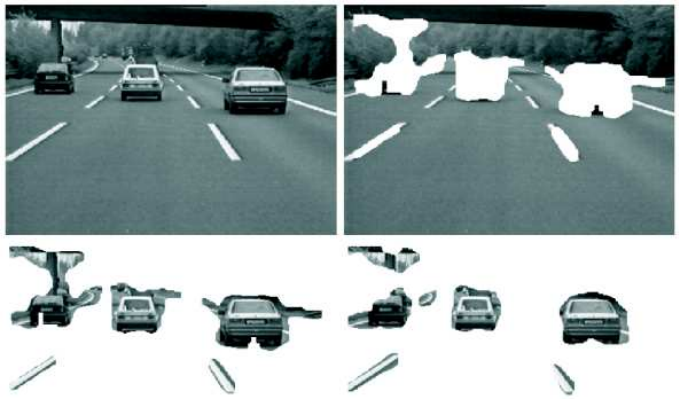


FIGURE 1.8 – Détection des objets structurés.

où $p(x_k)$ est une distribution de probabilité. Elle permet de trouver les zones non homogènes dans l'image. Le calcul de l'entropie est fait à partir des valeurs en niveaux de gris d'une petite fenêtre centrée sur le pixel d'intérêt. Ils proposent une autre méthode de génération des hypothèses à partir des matrices de co-occurrence. Les mesures d'énergie, contraste, entropie et corrélation sont combinées pour détecter les textures des véhicules. Les résultats sont meilleurs lorsqu'ils intègrent toutes ces informations, mais la quantité de calculs augmente en conséquence

Couleur

La couleur est très peu utilisée dans les systèmes de vision embarquée, du fait de la trop grande diversité de couleur des véhicules automobiles. Elle peut être cependant utile dans certaines occasions, notamment la nuit, dans des conditions de visibilité réduite ou lors de manœuvres particulières (freinage,...), pour détecter les feux arrières ou avant des véhicules [15, 131].

D'autres travaux [98, 5] utilisent aussi la couleur (en fait, les images achromatiques fournies par une caméra achromatique munie d'un filtre IR) pour détecter des amers sur des véhicules coopératifs.

---

[15] M. Betke, E. Haritaoglu, and L. S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12(2) :69–83, Août 2000.

[131] B. Steux. *Maps, un environnement logiciel dédié à la conception d'applications embarqués temps-réel. Utilisation pour la détection automatique de véhicules par fusion radar/vision.* PhD thesis, Ecole des Mines de Paris, Paris, France, Decembre 2001.

[98] F. Marmoiton. *Detection et suivi par vision monoculaire d'obstacles mobiles cooperatifs a partir d'un vehicule experimental automobile.* PhD thesis, Ecole Doctorale Sciences Pour l'Ingénieur de Clermont-Ferrand, Université Blaise Pascal, Clermont-Ferrand, France, Janvier 2000.

[5] R. Aufrère, F. Marmoiton, R. Chapuis, F. Collange, and J. P. Dérutin. Détection de route et suivi de véhicules par vision pour l'acc. *Traitement du Signal*, 17 :233–247, 2000.

1.2 Validation des Hypothèses

Les méthodes de Validation des Hypothèses sont la dernière étape du système de détection des véhicules. Pour chaque ROI rendue par l'étape de génération des hypothèses ils vont donner une réponse binaire indiquant la présence d'un véhicule ou non.

Les approches utilisées pour réaliser la validation peuvent être classifiées en deux catégories principales :

1. utilisant les primitives (*Template-Based*)
2. utilisant l'apparence (*Appearance-Based*)

Les méthodes par Primitives utilisent des gabarits prédéfinis pour la classe véhicule et réalisent une corrélation entre ceux-ci et l'image, pour la validation : contours horizontaux et verticaux [130], régions, *templates* déformables [31, 152, 43] et rigides [151, 134]. Les méthodes basées sur l'apparence apprennent les caractéristiques de la classe véhicule et de la classe non-véhicule à partir d'un ensemble d'images. Chaque image utilisée pour l'apprentissage est représentée par un vecteur de descripteurs locaux [2] ou globaux. Puis, l'apprentissage d'un classifieur (NNs, *Support Vector Machine* (SVM), Bayes, Adaboost, etc) permet d'estimer la frontière de décision entre la classe véhicule et la classe non-véhicule.

1.2.1 Méthodes utilisant les primitives

Régions homogènes

Parodi [114] suggère un système de détection d'obstacles et de perception de l'environnement dans une zone urbaine. Il utilise un détecteur de contours et, à l'aide de la

---

[130] N. Srinivasa. Vision-based vehicle detection and tracking method for forward collision warning in automobiles. In *IEEE Intelligent Vehicle Symposium*, volume 2, pages 626–631, Juin 2002.

[31] J.M. Collado, C. Hilario, A. de la Escalera, and J.M. Armingol. Model based vehicle detection for intelligent vehicles. In *International Symposium on Intelligent Vehicles*, pages 572–577, 2004.

[152] A.H.S. Yung, N.H.C.and Lai. Detection of vehicle occlusion using a generalized deformable model. In *IEEE International Symposium on Circuits and Systems*, volume 4, pages 154–157, Mai 1998.

[43] M. Dubuisson, S. Lakshmanan, and A. Jain. Vehicle segmentation and classication using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3) :293–308, Mars 1996.

[151] H. Yang, J. Lou, H. Sun, W. Hu, and T. Tan. Efficient and robust vehicle localization. In *International Conference on Image Processing*, volume 2, pages 355–358, Octobre 2001.

[134] T. N. Tan and K. D. Baker. Efficient image gradient based vehicle localization. *IEEE Transactions on Image Processing*, 9 :1343–1356, 2000.

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, pages 113–130, Londres, Royaume Uni, 2002. Springer-Verlag.

[114] P. Parodi and G. Piccioli. A feature based recognition scheme for traffic scenes. In *IEEE Intelligent Vehicles Symposium,*, pages 229–234, Tokyo, Japon, 1995.

Transformée de Hough Généralisée, il détecte les segments qui sont utilisés pour définir la route ainsi que les bâtiments, la ligne de l'horizon, etc. Les véhicules sont considérés comme des obstacles et, une fois l'environnement défini, ils doivent remplir certaines contraintes :

1. se trouver sur la route,
2. la présence d'un grand nombre de segments horizontaux dans la région,
3. la présence de régions homogènes.

Une fois le véhicule détecté, l'algorithme cherche dans la boîte englobante, des primitives plus détaillées : la fenêtre arrière et la plaque minéralogique. La méthode qu'il utilise pour cette opération n'est pas explicitement décrite, mais il semble qu'il cherche aussi des régions homogènes.

Segments

Srinivasa [130] et Kalinke [78] valident une ROI si elle contient une configuration de type "U" des segments de contour. Le premier cherche le barycentre de la région pour réaliser, à partir de ce point, une recherche radiale des droites dans huit directions (voir figure 1.9). Si les droites passent par le barycentre et coupent les contours de la région en, au moins, quatre points, la région est validée.

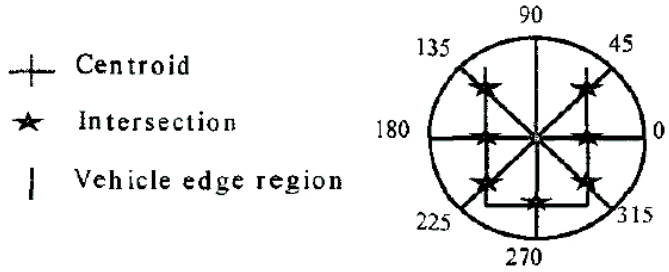


FIGURE 1.9 – Gabarit ou forme "U" d'un véhicule.

Regensburger [121] modélise les véhicules obstacles en fonction de la distance qui les sépare du véhicule porteur (figure 1.10). En général, les images des véhicules proches sont claires avec un bon contraste et beaucoup des détails visibles. Typiquement, l'image contient des régions homogènes délimitées par des contours bien définis. Ces contours sont appliqués sur un voisinage des limites de la ROI pour trouver le localisation exacte de la

[130] N. Srinivasa. Vision-based vehicle detection and tracking method for forward collision warning in automobiles. In *IEEE Intelligent Vehicle Symposium*, volume 2, pages 626–631, Juin 2002.

[78] T. Kalinke, C. Tzomakas, and W. v. Seelen. A texture-based object detection and an adaptive model-based classification. In *IEEE International Conference on Intelligent Vehicles 1998*, volume 1, pages 143–148, 1998.

[121] U. Regensburger and V. Graefe. Visual recognition of obstacles on roads. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 980–987, Septembre 1994.

voiture. Les contours horizontaux inférieurs définis pour les ombres portées permettent de compléter la modélisation.

Par ailleurs, les images d'objets distants ont un faible contraste et peu de détails. Un modèle en forme de U est utilisé : les contours gauche, droit et inférieur forment la première partie de la validation. Pour assurer une réponse positive sur ces véhicules, ils utilisent aussi les coins et la symétrie. Les fausses alarmes données par l'algorithme ne

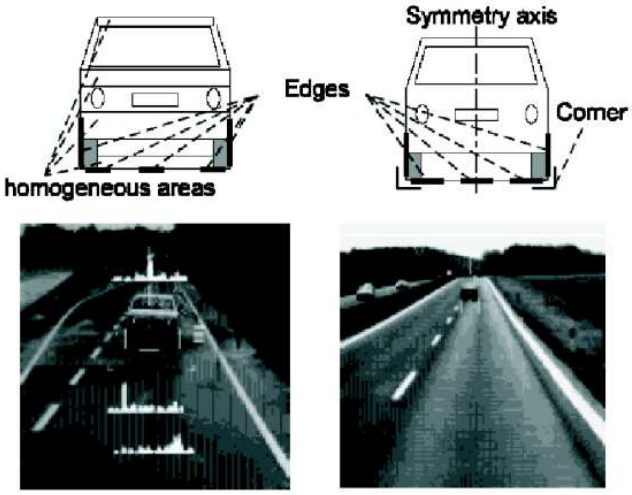


FIGURE 1.10 – Modèles de véhicules proches et distants.

sont pas rejetées immédiatement. Il faut que le système considère une ROI comme fausse alarme pendant un certain nombre d'images dans la séquence. Le problème du choix entre les deux modèles n'est pas décrit formellement dans le papier.

Graefe [61] emploie un ré-échantillonnage de la voie pour récupérer une image de la voiture candidate à une taille fixe, quelque soit sa distance au véhicule porteur. Une fois la route détectée, ils transforment l'image selon la perspective. Une quantité fixe n_1 des points est prise dans chaque ligne de la voie. Dans cette image ré-échantillonnée, ils vont chercher la voiture obstacle. Ensuite ils réalisent un deuxième ré-échantillonnage, en considérant l'hypothèse que l'objet appartient à un plan parallèle à celui de la caméra. Une grille carrée fixe, est appliquée sur l'objet détecté. Ils obtiennent donc, une imagette du véhicule candidat de taille fixe. Une méthode similaire à [121] est utilisée pour la validation du candidat.

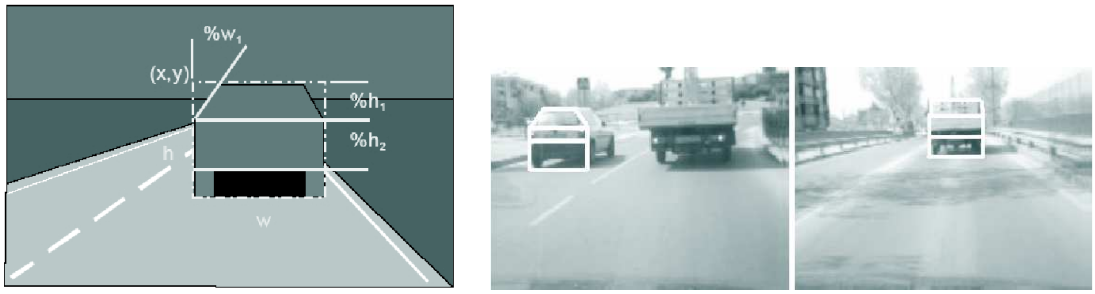
---

[61] V. Graefe and W. Efenberger. A novel approach for the detection of vehicles on freeways by real-time vision. In *IEEE Intelligent Vehicles Symposium*, Tokyo, Japon, Septembre 1996.

[121] U. Regensburger and V. Graefe. Visual recognition of obstacles on roads. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 980–987, Septembre 1994.

Gabarits déformables

Collado [31] a développé un modèle géométrique déformable pour reconnaître les voitures. Similairement à [43], il définit un schéma de segmentation basé sur un modèle gabarit.



(a) (b)

FIGURE 1.11 – Détection à partir du gabarit déformable.
(a) modèle géométrique du véhicule, (b) des résultats sur des images réelles.

Le gabarit d'un véhicule est défini via 7 paramètres (voir la figure 1.11) : la position (x,y), la largeur et la hauteur, la position des pare-brises, la position du pare-chocs et l'angle du toit. La fonction d'énergie du modèle géométrique inclut l'information de la forme, la symétrie du véhicule et des ombres portées. La symétrie est calculée à partir du gradient vertical et horizontal de l'image. La forme est définie à travers deux termes : le premier basé sur le gradient et le deuxième basé sur la distance des contours trouvés précédemment pour l'énergie de symétrie. Enfin, l'énergie des ombres d'un véhicule sera définie comme la moyenne des valeurs de gris de la partie inférieure du modèle. Un algorithme génétique permet de sélectionner les meilleurs paramètres. La figure 1.11 montre quelques résultats issus de son algorithme.

1.2.2 Méthodes utilisant l'Apparence

Dans cette sous-section, nous analysons les méthodes utilisées dans la littérature qui permettent de réaliser la classification d'une ROI en véhicule ou non-véhicule. A la différence des approches qui utilisent un gabarit de véhicule, ces méthodes apprennent l'apparence à partir d'une base d'images. Nous allons voir par la suite, les différentes espaces de paramètres employés dans la littérature pour réaliser le codage des images ainsi que les différents classifieurs choisis par les auteurs.

---

[31] J.M. Collado, C. Hilario, A. de la Escalera, and J.M. Armingol. Model based vehicle detection for intelligent vehicles. In *International Symposium on Intelligent Vehicles*, pages 572–577, 2004.

[43] M. Dubuisson, S. Lakshmanan, and A. Jain. Vehicle segmentation and classication using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3) :293–308, Mars 1996.

Niveaux de gris

La méthode de validation des hypothèses de Matthews [99] utilise les niveaux de gris des pixels de la ROI. Chaque ROI est ré-échantillonnée en une fenêtre de 20x20 pixels, qui est ensuite divisée en 4x4 vignettes, chacune de taille 5x5 pixels. Une Analyse en Composantes Principales (ACP) locale permet d'estimer pour chacune de ces vignettes les cinq meilleurs valeurs propres et vecteurs propres. Ce dernier vecteur de taille cinq est utilisé pour entraîner un réseau multi-couches qui estime la frontière de classification entre la classe véhicule et la classe route.

Wang [144] utilise une ACP pour projeter des régions de la voiture dans différents espaces de représentation. Cette étude considère le cas d'un système mobile de surveillance utilisé dans les parkings publics, qui détectent les véhicules stationnées, à partir de leurs vues frontale et arrière. Pour la détection, au lieu d'utiliser une approche considérant l'ensemble de l'image, ils cherchent à trouver les régions les plus caractéristiques, selon eux, d'une vue arrière de la voiture : le toit et les phares. Ils définissent quatre espaces de représentation qui vont servir pour projeter les exemples d'apprentissage et classifier les images. Dans le premier espace toutes ces sous-régions sont projetées sans information de sa position spatiale. Toutes les sous-régions sont également projetées dans un second espace incluant une information sur leur position. Pour le troisième espace, chaque sous-région est projetée dans son propre espace correspondant avec une information sur la position. La dernière représentation utilise l'analyse en composants indépendants (ICA) dans l'espace résiduel pour détecter les véhicules. Les vignettes résiduelles sont calculées à partir de la vignette originale et celle reconstruite avec l'ACP.

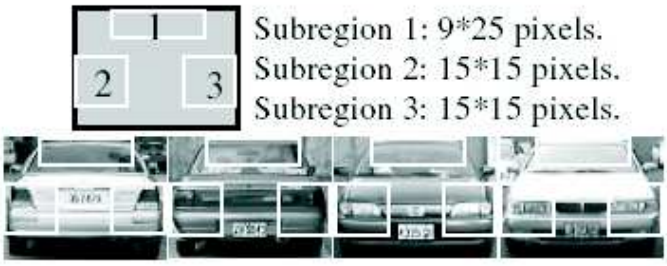


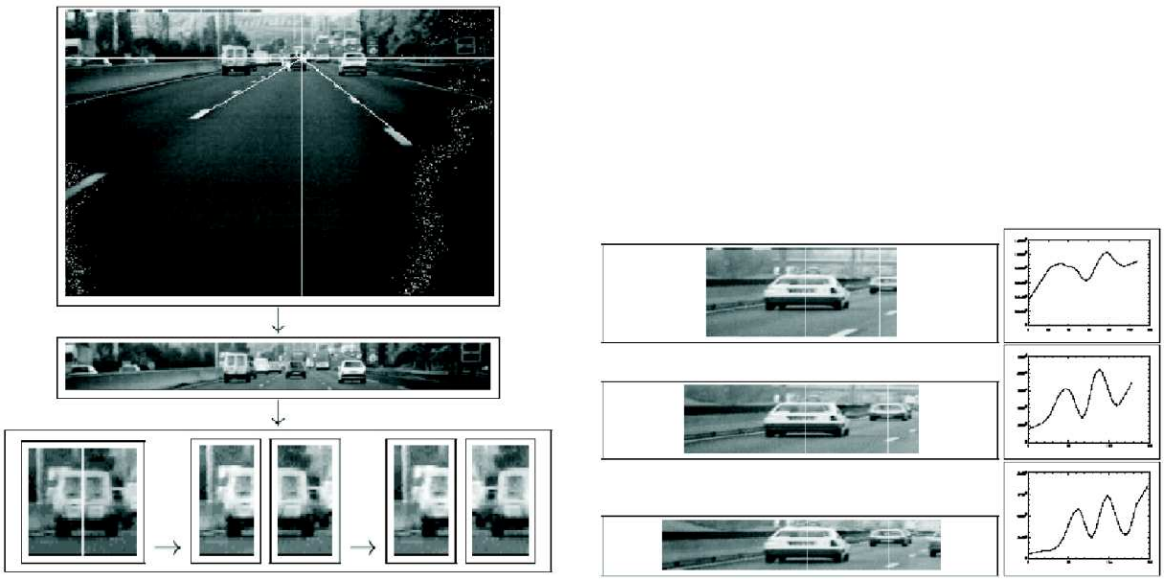
FIGURE 1.12 – Exemples de sous-régions.

[99] N. D. Matthews, P. E. An, D. Charnley, and C. J. Harris. Vehicle detection and recognition in greyscale imagery. *Control Engineering Practice*, 4(4) :473–479, Avril 1996.

[144] C.-C. Wang and J.-J. Lien. Automatic vehicle detection using statistical approach. In *Asian Conference on Computer Vision*, volume 3852, pages 171–182. Springer, 2006.

Symétrie

Le système présenté dans le travail de Kalinke [79] utilisent un réseau neuronal de type *Dynamic Neural Network* (DNN) qui estime l'existence d'un axe de symétrie dans une image. Ce réseau a été entraîné à partir d'images de voitures divisées en deux demi-images. A chaque partie deux opérateurs sont appliqués pour obtenir le vecteur d'entrée au réseau : un filtre de Laplace, pour éliminer les pixels de texture similaire, et un opérateur monotone pondéré, qui évalue la distribution des niveaux de gris des pixels. En combinant la sortie du réseau et un plan de correspondance liant tous les points de la demi-image gauche aux points de la demi-image droite, ils obtiennent un taux de similarité entre les deux moitiés de la vignette.



(a) (b)

FIGURE 1.13 – Détection à partir de la symétrie.
(a) Extraction des vignettes de la zone d'intérêt, (b) détection des axes de symétrie à partir de la courbe de symétrie.

Histogrammes de Gradient Orientés

Les histogrammes de gradients orientés (HoG) consistent en une organisation en histogramme des pixels d'un voisinage selon leur orientation et pondérés par leur magnitude.

[79] T. Kalinke and W. von Seelen. A neural network for symmetry-based object detection and tracking. In *DAGM-Symposium*, pages 37–44, 1996.

Dans des travaux récents [35, 157, 58], les HoG ont été utilisés avec succès pour la détection de piétons. De plus, un classifieur de type SVM [35] ou une combinaison de SVM et Adaboost [157, 58] est utilisé pour obtenir le détecteur.

Gepperth [57] utilise les contours orientés pour construire des histogrammes de gradient orienté (HoG) qu'ils appellent *Set of Orientation Energies* (SOE). Une ROI fournie par l'étage de génération est divisée en un nombre fixe de régions rectangulaires (champs récepteurs), où ils calculent le gradient. Les orientations du gradient sont ensuite quantifiées pour prendre des valeurs entières entre 0 et 3. Par la suite, la valeur de chaque bin de l'histogramme d'une région correspond à la somme des énergies (module du gradient) des pixels de chaque orientation (voir figure 1.14). La concaténation de tous les histogrammes de la grille représente l'entrée d'un réseau neuronal multi-couches avec une cellule de sortie.

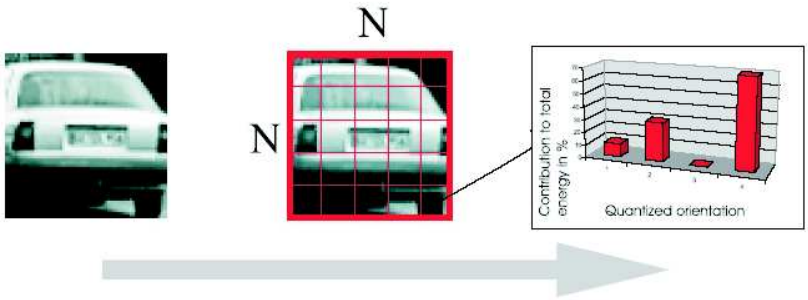


FIGURE 1.14 – Obtention du HoG pour un champ récepteur.

Ondelettes

La décomposition en ondelettes représente un autre espace de paramètres populairement utilisée dans la communauté, et ceci pour deux raisons : la représentation multi-échelles intrinsèque de celles-ci, ainsi que la possibilité, par sélection des ondelettes, d'obtenir une réduction de l'espace robuste. Les deux familles des ondelettes couramment employées sont celles de Gabor et les filtres rectangulaires aussi appelés filtres de Haar.

---

[35] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 2, pages 886–893, Juin 2005.

[157] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE International Converence on Computer Vision and Pattern Recognition*, pages 1491–1498, Washington, DC, États Unis, 2006.

[58] D. Geronimo, A. Lopez, D. Ponsa, and A.D. Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 418–425, 2007.

[57] A. Gepperth, J. Edelbrunner, and T Bocher. Real-time detection and classification of cars in video sequences. In *IEEE Intelligent Vehicles Symposium*, pages 625–631, Juin 2005.

Nous allons commencer pour analyser les travaux de Schneiderman [128] qui ont popularisé les ondelettes pour la détection des objets : visages et voitures. Son système permet de détecter ces objets à différentes orientations en utilisant plusieurs modèles (chacun dédié à une orientation). Son espace de paramètres est basé sur l'utilisation des ondelettes pour décomposer l'image originale en sous-bandes correspondant à la fréquence et à l'orientation dans une zone de celle-ci. Pour décomposer une image en ondelettes, ils utilisent deux filtres : un passe-bas d'ordre cinq et un passe-haut d'ordre trois.

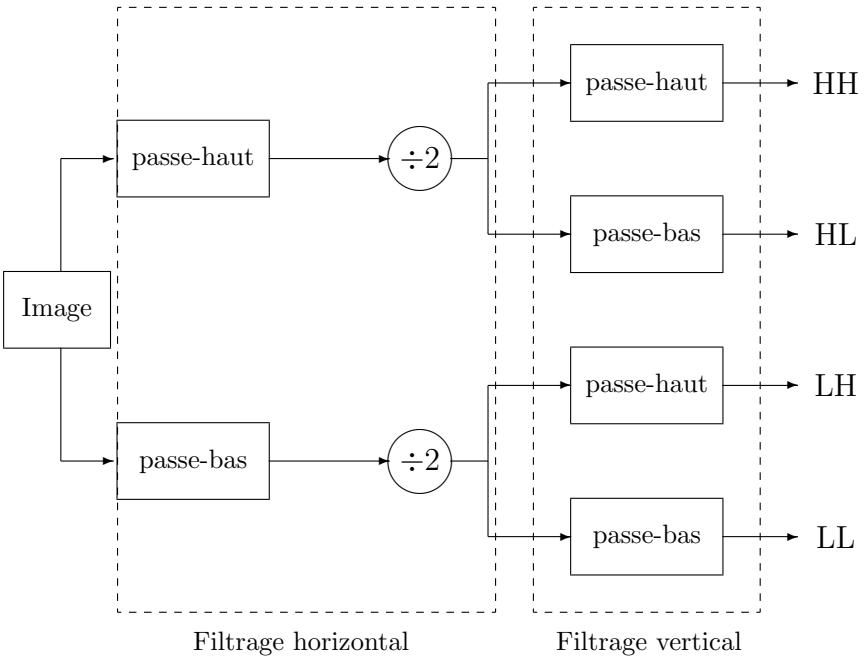


FIGURE 1.15 – Décomposition en ondelettes.

Dans la première étape (voir figure 1.15), l'image est filtrée dans la direction horizontale avec ces deux filtres. Ensuite, les deux sorties sont sous-échantillonnées par un facteur 2 dans la même direction. Ces signaux sont, à leur tour, filtrés avec les filtres précédents dans la direction verticale. La décomposition finale est composée de quatre sous-bandes : LL, HL, LH et HH. La bande LL (passe-bas horizontal et vertical) est simplement une version de plus basse résolution en basses fréquences. La bande LH (passe-bas dans la direction horizontale et passe-haut dans la vertical) représente les lignes et contours verticaux. En forme similaire, HL représente les contours horizontaux. Finalement, HH représente les contours diagonaux. La figure 1.16 montre un exemple de la décomposition en ondelettes pour l'image d'une voiture.

Ces quatre bandes sont le premier niveau de la décomposition. Les niveaux suivants

[128] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 746–751, 2000.

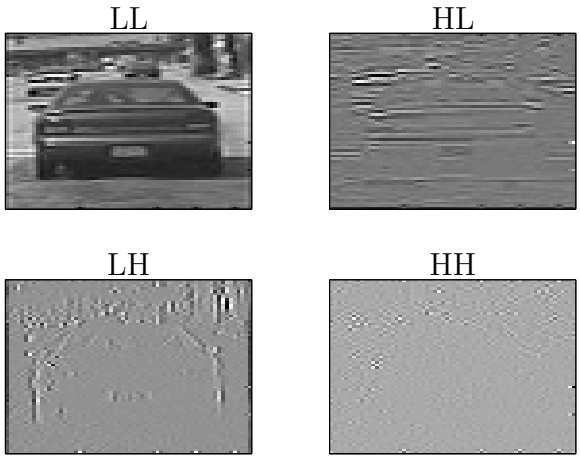


FIGURE 1.16 – Exemple de décomposition en ondelettes à partir des filtres donnés par Schneiderman.

sont obtenus en répétant sur la bande LL, la décomposition précédente. Dans leur représentation, ils ont trois niveaux (voir figure 1.17), donc 10 bandes au total.

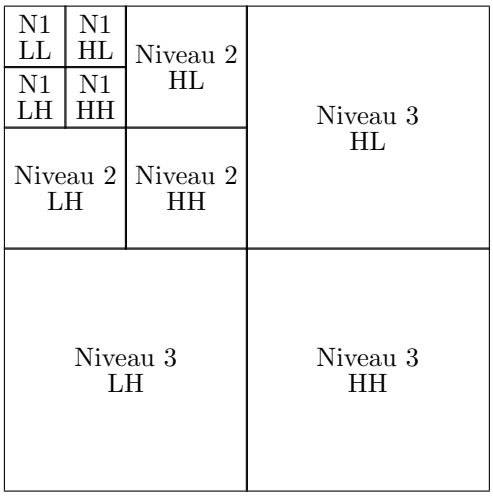


FIGURE 1.17 – Sous-bandes de la décomposition en ondelettes.

Les descripteurs sont définis comme la valeur du coefficient dans une sous-bande ainsi que des combinaisons des valeurs entre les bandes. Une fonction de densité de probabilité est calculée pour chacun à partir des valeurs présentes dans la base d'apprentissage. Sont ainsi modélisées les distributions des valeurs dans les exemples positifs et négatifs. L'algorithme Adaboost lui permet de choisir les descripteurs les plus discriminants pour réaliser la classification.

Wen [146] utilise une décomposition à cinq niveaux à partir des ondelettes de Haar. Pour une vignette de 32x32 pixels, ils obtiennent un vecteur de descripteurs de 1024

[146] X. Wen, H. Yuan, C. Yang, C. Song, B. Duan, and H. Zhao. Improved haar wavelet feature extraction approaches for vehicle detection. In *IEEE Intelligent Transportation Systems Conference*, pages 1050–1053, Septembre 2007.

éléments, après un filtrage. Ce vecteur est utilisé pour entraîner un classifieur de type SVM, qu'il emploie pour valider l'existence d'un véhicule dans une fenêtre.

Filtres de Gabor

Le système de détection de Cheng [27] a deux étapes : la première étape génère des hypothèses dans l'image à partir des segments horizontaux et verticaux à différentes échelles. Ensuite, ils valident ces ROIs à l'aide d'un vecteur de descripteurs de Gabor [132]. Pour l'apprentissage, une vignette est divisée en 9 sous-fenêtres et pour chacune ils calculent les quatre meilleurs descripteurs de Gabor à l'aide d'un algorithme de dopage. Le classifieur final est obtenu à partir d'un SVM qui est utilisé pour trouver l'hyperplan de séparation entre les exemples d'apprentissage positifs et négatifs.

Le classifieur introduit par Sun [132] est entraîné à l'aide d'une approche appelée *Evolutionary Gabor Filter Optimization* (EGFO). La méthode pour l'extraction des primitives consiste à re-échantillonner une vignette où se trouve un véhicule de la base d'apprentissage, à une taille fixe de 64x64 pixels. La première étape consiste à appliquer une fonction linéaire pour corriger les différences de luminance et une égalisation d'histogramme pour renforcer le contraste.

Ensuite, cette vignette est subdivisée en 9 vignettes superposées de 32x32 pixels. Le

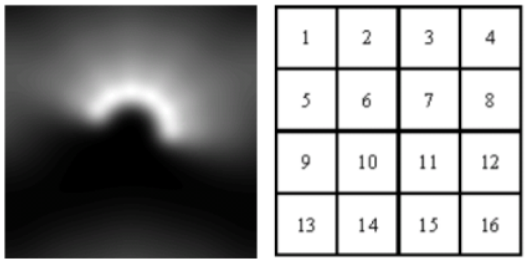


FIGURE 1.18 – Représentation des filtres de Gabor et les vignettes pour l'extraction des primitives.

recouvrement provoque une redondance qui permet de filtrer le bruit dû à l'arrière-plan autour du véhicule. Les filtres de Gabor sont appliqués à chaque vignette :

$$\Phi(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{\tilde{x}^2}{\sigma_x^2}+\frac{\tilde{y}^2}{\sigma_y^2})} \; e^{2\pi j W\tilde{x}}$$

Avec : $\tilde{x} = x\ cos\theta + y\ sin\theta$, $\tilde{y} = -x\ sin\theta + y\ cos\theta$, W, la fréquence radiale, $\theta \in [0,\pi]$. Un filtre Φ est entièrement défini avec les quatre paramètres :$\{\theta, W, \sigma_x, \sigma_y\}$.

---

[27] H. Cheng, N. Zheng, and C. Sun. Boosted gabor features applied to vehicle detection. In *IEEE International Conference on Pattern Recognition*, pages 662–666, 2006.

[132] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2) :125–137, Juin 2005.

Ils utilisent un algorithme génétique pour choisir un ensemble des filtres pour obtenir la meilleur performance possible. L'objectif du AG est de sélectionner la combinaison de paramètres adéquate pour chaque filtre de l'ensemble.

Filtres rectangulaires

Les filtres rectangulaires ou filtres de Haar, donnent une information sur la distribution des niveaux de gris entre deux régions voisines dans l'image. Nous allons en voir une description plus détaillée dans le chapitre suivant.

Dans [112], Papageorgiou utilise une analyse statistique des valeurs moyennes des filtres pour sélectionner 39 coefficients pour le visage et 29 coefficients pour les piétons les plus significatifs. Ensuite, il utilise un classifieur SVM pour apprendre les relations entre les coefficients des ondelettes qui définissent la classe objet.

Dans un travail postérieur, Papageorgiou [113] calcule les coefficients des filtres pour 2 échelles : 32x32 et 16x16. Il obtient 3.030 descripteurs qui utilisent pour entraîner un SVM qui va classifier un exemple en classe véhicule ou non véhicule.

Viola et Jones [143, 142], ont utilisé AdaBoost, dans sa version discrète, pour sélectionner les descripteurs et construire un classifieur fort (voir le chapitre suivant). Lienhart [93] a présenté un ensemble plus complet de filtres rectangulaires, montré dans la figure 1.19, ainsi qu'une méthode rapide pour calculer les filtres diagonaux. Cet ensemble compte avec une quantité totale de 117.941 filtres, établie pour une vignette de 24x24 pixels. Une analyse comparative entre Discrete AdaBoost, Real AdaBoost et Gentle AdaBoost est ensuite réalisée pour la détection de visages.

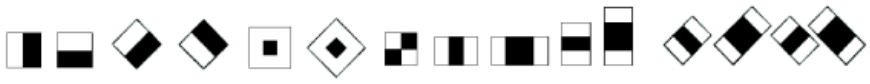


FIGURE 1.19 – Ensemble complet de filtres de Haar.

Bai [6] utilise les ondelettes de Haar et AdaBoost pour l'estimation du flux routier

[112] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *IEEE International Conference on Computer Vision*, pages 555–562, Janvier 1998.

[113] C.P. Papageorgious and T. Poggio. A trainable object detection system : Car detection in static images. Technical report, MIT AI Memo 1673 (CBCL Memo 180), 1999.

[143] P. A. Viola and M. J. Jones. Robust real-time face detection. In *International Conference on Computer Vision*, volume 2, page 747, Juillet 2001.

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

[93] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM Symposium for Pattern Recognition*, pages 297–304, 2003.

[6] L. Bai, W. Tompkinson, and Y. Wang. Computer vision techniques for traffic flow computation. *Pattern Analysis and Applications*, 7(4) :365–372, 2004.

Auteur	Méthode	B^+	B^-	error (%)	DC (%)	FA (%)
Sun [132]	EGFO	1051	1051	6.36	-	-
Gepperth [57]ψ	NN	50000	50000	3.8	-	-
Kato [81]	MC-MQDF	5000	5000	-	97.7	-
Khammari [83]ψ	-	1500	11000	-	90	0.01
Trujillo [139]ψ	-	500	-	-	96	-
Betke [15]ψ	-	-	-	-	80.9	-
Han [65]ψ	ASF+EOV	-	-	7.6 / 8.4	-	-

TABLE 1.1 – Comparaision des résultats présentes dans la bibliographie.

à partir d'une caméra statique. Demirkur [39] utilise les filtres de Haar comme espace de paramètres pour la détection de visages ainsi que le Gentle AdaBoost pour réaliser la classification. Wu [148] utilise les ondelettes de Haar et un classifieur du type ACP pour la détection de véhicules dans les images statiques. Ils comparent sa méthode à d'autres classifieurs (Fisher Linear Discriminant (FLD) et SVM) et obtiennent de meilleurs résultats.

1.3 Bilan

1.3.1 Résultats

Dans le tableau 1.1, nous montrons les résultats publiés par différents auteurs. B^+ et B^- sont respectivement le nombre d'exemples dans les bases d'apprentissage positive et négative. Les auteurs marqués avec ψ utilisent le suivi des hypothèses dans la séquence pour filtrer les fausses alarmes.

Comme nous pouvons le voir, il est très difficile de comparer réellement ces méthodes, car elles ont souvent été testées sur des bases d'images différentes et utilisent des critères d'évaluation différents. Sans compter que beaucoup de méthodes n'ont pas été évaluées sur des bases de données de taille suffisante, en particulier les méthodes les plus anciennes (notamment celles utilisant les primitives), ou que les résultats obtenus n'ont pas été publiés.

Dans la figure 1.20 nous avons représentée une synopsis des algorithmes et méthodes utilisés dans chacune des étapes de la détection.

---

[39] C. Demirkir and B. Sankur. Face detection using look-up table based gentle adaboost. In *International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 339–345, 2005.

[148] J. Wu and X. Zhang. A pca classifier and its application in vehicle detection. In *IEEE International Joint Conference on Neural Networks*, volume 1, pages 600–604, 2001.

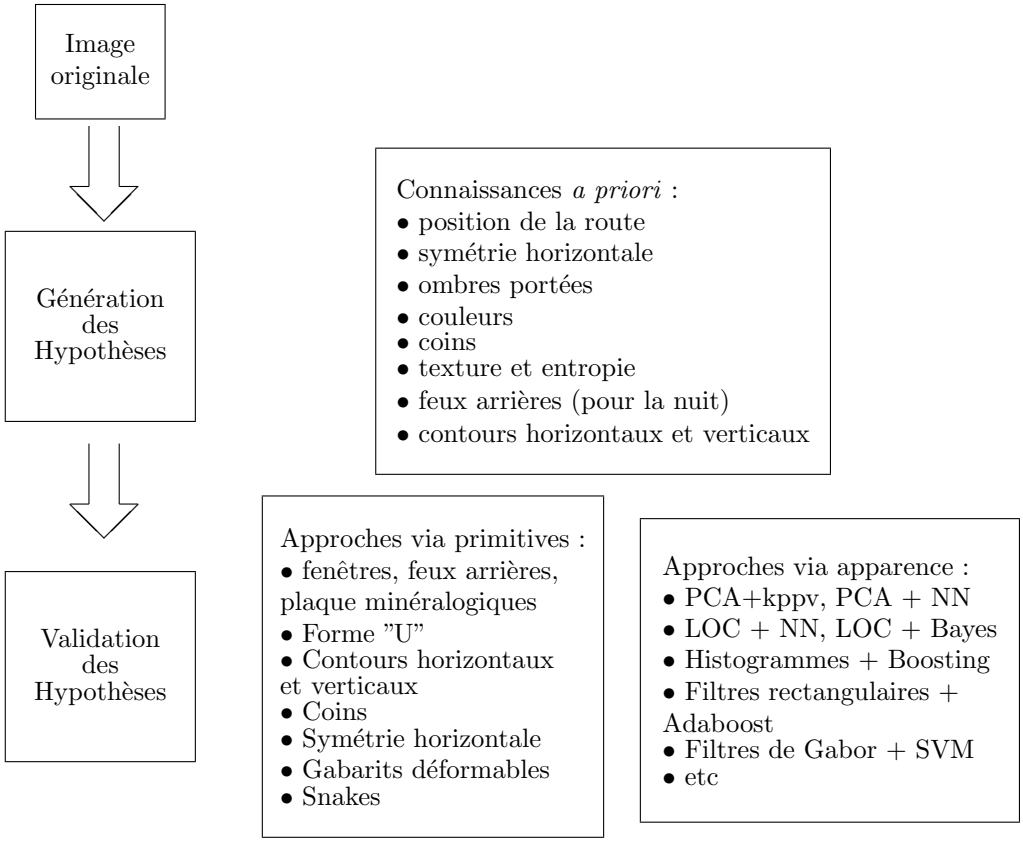


FIGURE 1.20 – Méthodes pour la détection de véhicules.

1.3.2 Discussion

La précédente revue de la littérature permet de conclure les points suivants :

- Les algorithmes de génération des hypothèses ont l'inconvénient de ne pas bien fonctionner pour tous les scénarios présentés. Il s'avère donc nécessaire, de réaliser la fusion de plusieurs de ces méthodes pour couvrir les différents cas, ce qui conduit à une complexité croissante d'un algorithme qui était sensé être simple. La zone de recherche peut être restreinte simplement à travers un analyse statistique qui nous permet de définir des zones d'intérêt dans l'image qui dépendent de la taille des voitures cherchées.
- Parmi les espaces de paramètres présentés précédemment, deux sont actuellement utilisés couramment par la communauté de reconnaissance de formes et appliqués sur plusieurs problématiques différentes : les filtres rectangulaires et les histogrammes de gradient orienté. Les premiers se sont montrés performants sur des applications en temps réel. Les deuxièmes ont donné naissance à d'autres familles de descripteurs utilisés en détection des points caractéristiques : Scale of Invariant Features Trans-

form (SIFT) [96], Speeded Up Robust Features (SURF) [8], etc. Nous allons utiliser ces deux descripteurs pour définir notre espace de paramètres.

- Les méthodes de validation qui utilisent les primitives ou gabarits servent principalement à la recherche d'un type de véhicule. Dans notre cas, nous avons la contrainte de devoir faire une détection de plusieurs types différents de véhicules. Nous allons proposer différentes architectures de détection et classification qui traitent séparément les types ou les réunissent dans un classifieur.
- Le problème des modèles génératifs est la définition des frontières de classification : ils n'arrivent pas à éliminer les fausses alarmes proches du modèle. A cet effet, nous allons utiliser aussi des fonctions de classification de type discriminantes.
- Parmi les algorithmes d'apprentissage étudiés auparavant (SVM, réseaux neuronaux, etc), la méthode de dopage s'est montrée très performante dans les systèmes de détection en temps réel, en raison à l'emploi d'une architecture en cascade. Elle a aussi la capacité à s'adapter aux deux approches proposées : générative et discriminante.
- Récemment, les méthodes d'apparence se sont multipliées. Elles s'appuient sur les dernières avancées de la recherche en matière d'apprentissage automatique (classifieurs statistiques ou neuronaux, méthodes de dopage, ...), obtiennent de bons résultats et fonctionnent quasiment en temps réel.

Le chapitre suivant est dédié à une description de la méthode de dopage à travers l'analyse des travaux de Viola et Jones sur la détection des visages. Ensuite, nous allons présenter notre système de détection des véhicules.

---

[96] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, pages 91–110, 2004.

[8] H. Bay, T. Tuytelaars, and L.J. Van Gool. Surf : Speeded up robust features. In *European Conference on Computer Vision*, volume 3951, pages 404–417, 2006.

Chapitre 2

Méthode de dopage et Algorithme de Viola et Jones

2.1 Introduction

Un joueur, frustré par ses pertes dans les courses de chevaux et envieux des gains de ses amis, décide de laisser parier pour lui un groupe de ces copains. Sa stratégie serait de jouer une somme d'argent fixe dans chaque course. Chaque ami recevrait une quantité de cette somme en fonction de ses succès. Cette façon de jouer, lui permettrait de gagner au long de la saison, une somme dans son ensemble proche de celle qu'il aurait obtenue s'il avait parié toujours comme son ami le plus chanceux [51].

Ce chapitre s'intéresse à l'analyse d'une méthode d'apprentissage largement utilisée à ce jour en reconnaissance de formes : la méthode de dopage (ou *boosting* en anglais) et à son utilisation dans l'algorithme de Viola et Jones .

Cette méthode, proposée par Freund et Schapire [51], convertit un algorithme *faible* d'apprentissage du type Probably Approximately Correct [140], qui a une performance légèrement supérieure que le hasard, en un autre avec un maximum d'efficacité. Elle cherche à maximiser la marge interclasse à partir d'une minimisation de l'erreur de classification.

Cet algorithme, plus tard appelé AdaBoost par ses auteurs [50], est utilisé par Viola

[51] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[140] L. G. Valiant. A theory of the learnable. *ACM Symposium on Theory of Computing*, 27(11) :1134–1142, 1984.

[50] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

et Jones [143, 142] pour la détection des objets. Il permet de sélectionner un nombre limité de descripteurs pour créer le modèle d'apparence de l'objet. En réalité, Adaboost ne sélectionne pas les descripteurs, mais leurs fonctions de classification faibles associées (cf. section 2.3). Ils proposent un schéma attentionel original appliqué à la détection de visages. Celui-ci repose sur une cascade de classifieurs de complexité croissante : chaque étage restreint de plus en plus la zone de recherche en rejetant une portion importante des zones ne contenant pas de visages. Cette méthode utilise des descripteurs dits de Haar appelés aussi filtres rectangulaires. L'utilisation d'une image intégrale pour le calcul des descripteurs de Haar, combinée à la cascade, permet un fonctionnement de l'ensemble en temps réel.

Par la suite, nous allons faire une analyse des trois principales contributions des travaux de Viola et Jones : les filtres rectangulaires et son calcul à travers l'image intégrale, la méthode d'apprentissage et l'architecture en cascade.

2.2 Filtres rectangulaires

Les filtres rectangulaires ou filtres de Haar permettent de coder la structure originale de la distribution des intensités d'un objet (visages, piétons, voitures, etc. [111, 112, 113]).

2.2.1 Définition

Les filtres consistent en un ensemble de relations d'inégalité entre la moyenne des intensités de différentes régions de l'objet. La figure 2.1 montre trois types de filtres de Haar en 2D. Ils capturent les changements d'intensité dans les directions horizontale, verticale et diagonale. La valeur du descripteur issu d'un filtre à 2 rectangles est la différence entre la somme des niveaux de gris des pixels du carré blanc et la somme des niveaux de gris des pixels du carré gris dans une région de l'image.

Viola et Jones utilisent trois types de filtres de la figure 2.1, à deux, trois et quatre rectangles. Les rectangles ont différentes tailles, par exemple, de 2x2 jusqu'à 24x24 pixels et ont des formes carrés ou rectangulaires. L'application de l'ensemble à toutes les positions

[143] P. A. Viola and M. J. Jones. Robust real-time face detection. In *International Conference on Computer Vision*, volume 2, page 747, Juillet 2001.

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

[111] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *IEEE Computer Vision and Pattern Recognition*, page 193, 1997.

[112] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *IEEE International Conference on Computer Vision*, pages 555–562, Janvier 1998.

[113] C.P. Papageorgious and T. Poggio. A trainable object detection system : Car detection in static images. Technical report, MIT AI Memo 1673 (CBCL Memo 180), 1999.

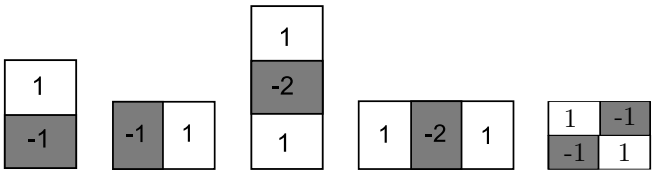


FIGURE 2.1 – Trois types de filtres rectangulaires en 2 dimensions.

possibles d'une vignette de 24x24 pixels, permet d'obtenir un vecteur exhaustif composé de 45.396 éléments.

Ils ont proposé la méthode d'Image Intégrale pour calculer ces descripteurs. Cette méthode était précédemment utilisée en synthèse d'images, en particulier dans les jeux vidéos, afin de générer rapidement des images dans des résolutions différentes et changeantes.

2.2.2 Image intégrale

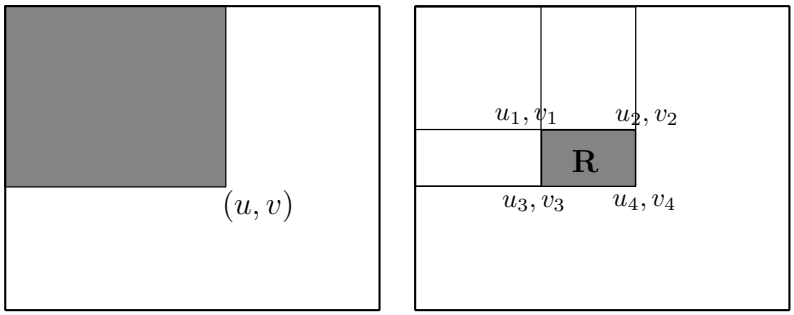


FIGURE 2.2 – Image intégrale et calcul de la somme des niveaux de gris dans un rectangle. (a) La valeur de l'image intégrale à la position (u, v), (b) quatre références à l'Image Intégrale permet le calcul de la somme des niveaux de gris dans R.

L'image intégrale est une représentation intermédiaire de l'image d'entrée permettant de réduire le temps de calcul des filtres. Soit une image en niveaux de gris $i(u, v)$ (où (u, v) sont les coordonnées dans l'image) l'image intégrale est définie comme :

$$ii(u, v) = \sum_{\substack{u'<u \\ v'<v}} i(u', v')$$

La somme des niveaux de gris des pixels dans une région rectangulaire de i peut être calculée rapidement à partir des 4 références à l'image intégrale (fig. 2.2) :

$$R = ii(u_4, v_4) + ii(u_1, v_1) - (ii(u_2, v_2) + ii(u_3, v_3))$$

Par conséquent, la différence entre deux rectangles adjacents est obtenue à travers six références à l'image intégrale $ii(u, v)$. Pour un filtre à trois rectangles, on a besoin de huit références.

2.2.3 Normalisation

L'image intégrale permet aussi de calculer rapidement les valeurs de moyenne et l'écart type des niveaux de gris d'une vignette pour réaliser une correction de la luminance. La nouvelle image est obtenue à partir d'une pseudo distance de Mahalanobis [93] :

$$I' = \frac{I - \mu}{2\sigma}$$

La moyenne est calculée à partir du dernier élément de l'image intégrale divisé par le nombre des éléments de la vignette et l'écart type peut se calculer à partir de :

$$\sigma = \sqrt{m^2 - \frac{1}{N}\sum x^2}$$

où m est la moyenne de la sous-fenêtre. Le terme $\sum\ x^2$ peut être obtenu avec une image intégrale des intensités au carré.

2.3 Sélection de descripteurs

La taille du vecteur de descripteurs des filtres rectangulaires f_i pour une vignette, est très supérieure au nombre de ces pixels. Nous pouvons penser cette information comme un énorme sac, où sont mélangés les filtres rectangulaires (45396 descripteurs au total). L'utilisation de l'ensemble complet pour réaliser la classification est inadéquat du point de vue du temps de calcul et de la robustesse, étant donné que certains de ces paramètres ne contiennent pas d'information pertinente (bruit).

Il est donc convenable de trouver une méthode permettant de récupérer du sac les descripteurs qui nous permettrons de modéliser la classe véhicule et, en même temps, la discriminer de la classe non véhicule. Des méthodes statistiques [29] ont été utilisées à ce propos mais ils ne concernent que le choix des descripteurs, ensuite, il faut appliquer un classifieur.

La méthode de dopage proposée par [51] permet de réaliser ces deux tâches simultanément. Cet algorithme, appelé AdaBoost pour ***Adaptive Boosting***, évolue au fur et à mesure qu'il choisit un descripteur dans le sac. Ce choix est fait en fonction de l'erreur commise par les fonctions de classification associées à chaque descripteur. Les fonctions

---

[93] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM Symposium for Pattern Recognition*, pages 297–304, 2003.

[29] X. Clady. *Contribution à la navigation autonome d'un véhicule automobile par vision*. PhD thesis, Université Blaise Pascal, France, 2003.

[51] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

FIGURE 2.3 – Le sac de descripteurs (espace de paramètres) est trop lourd pour le classifieur.

FIGURE 2.4 – Adaboost choisit de manière itérative un descripteur.
Mais, il ne le fait pas au hasard, comme dans le Bingo.

de classification g_j, proposées par Viola et Jones pour les filtres rectangulaires, consistent en une évaluation de la valeur de sortie du filtre par rapport à un seuil :

$$g_j = \begin{cases} 1 & \text{si } p_j f_j < p_j \theta_j \\ 0 & \text{sinon} \end{cases} \tag{2.1}$$

où f_j nous donne la valeur de sortie du filtre, θ_j est le seuil et p_j est la parité. Pour chaque descripteur j, AdaBoost adapte une distribution des probabilités qui lui permet de déterminer le seuil θ_j optimal pour la classification.

Chacune de ces fonctions faibles a une précision modérée dans la classification : elle prend une décision binaire à partir de la valeur de sortie du filtre. Mais la combinaison de plusieurs fonctions permet de construire un classifieur de performance plus grande, appelé classifieur $fort$. Cette combinaison est représentée par la somme pondérée suivante :

$$G = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t g_t \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t = S \\ 0 & \text{sinon} \end{cases} \tag{2.2}$$

La fonction de classification $forte$ G est composée de T fonctions de classification $faibles$ g_t, où α_t est un coefficient de pondération pour chacune des g_t. S est le seuil de la fonction de classification G qui minimise l'erreur de classification.

Nous transcrivons le pseudo-code dans le tableau 2.1, l'algorithme de dopage appelé *Discrete AdaBoost* [142].

Le paramètre β_t est choisi en fonction de l'erreur ϵ_t et il est utilisé pour actualiser le vecteur des poids w. Cette loi réduit la probabilité assignée aux exemples bien classifiés par la fonction faible et incrémente la probabilité des exemples où la prédiction est erronée. Ces exemples vont avoir une incidence plus forte dans l'estimation de l'erreur ϵ_t dans

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

AdaBoost Discret

1. Soient N exemples $(x_1,y_1),...,(x_N,y_N)$
   avec x_i le vecteur de descripteurs $\in \Re$ de l'exemple i
   et $y_i \in \{1, 0\}$ (positifs et négatifs)
2. Initialiser $w_i = \frac{1}{N}$, $i = 1, ..., N$
3. Pour t=1,...,T
   Normaliser les valeurs du vecteur de poids w_i
   $$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{N}} w_j$$
   Pour chaque descripteur j, entraîner à partir des w_i
   un classifieur g_j dont l'erreur est définie par :
   $$\epsilon_j = \sum_{i=1} \omega_i |g_j(x_i) - y_i|$$
   Choisir le classifieur g_t avec l'erreur ϵ_t la plus faible
   Actualiser les poids : $\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$
   où $e_i = 0$ si $g_t(x_i) = y_i$, $e_i = 1$ sinon,
   avec $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
4. sortie : $G = \sum_{t=1}^{T} \alpha_t g_t \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t$
   avec $\alpha_t = log(\frac{1}{\beta_t})$

TABLE 2.1 – Pseudo-code de l'algorithme Adaboost Discret

l'itération suivante (et donc le choix de la prochaine fonction faible). L'erreur de g_t dans la nouvelle distribution w_{t+1} est exactement 1/2.

La fonction de discrimination obtenue est une somme pondérée des fonctions faibles générées par l'algorithme. Le facteur β_t s'est réduit avec ϵ_t, ce qui incrémente la différence entre les distributions w_t et w_{t+1}. Un β_t plus petit, signifie que la fonction faible est plus précise et va avoir une influence plus importante dans le classifieur G à travers la pondération α_t.

L'erreur de G est bornée par [51] :

$$\epsilon_G \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

[51] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

2.4 Détection en Cascade

Le troisième apport du travail de Viola et Jones [142] consiste en l'introduction d'un apprentissage basé sur la méthode de dopage qui permet d'obtenir une fonction de classification avec une architecture en cascade, qu'ils appellent *Cascade Attentionelle*.

2.4.1 Principe de la *Cascade Attentionelle*

La cascade est la combinaison successive des classifieurs forts, dont la complexité est croissante tout au long de la structure (voir figure 2.5). Au départ, les classifieurs simples, éliminent la plupart des fausses alarmes, avant que les classifieurs plus complexes [1] soient utilisés pour éliminer des fenêtres plus difficiles. Ainsi, l'*attention* du système se concentre, au fur et à mesure des étages de la cascade, sur des zones de plus en plus réduites et pertinentes.

Cette façon d'organiser la détection permet d'incrémenter la performance totale du système (taux de détections correctes et taux de fausses alarmes), en même temps qu'elle augmente la vitesse en se focalisant sur des régions susceptibles de contenir un véhicule.

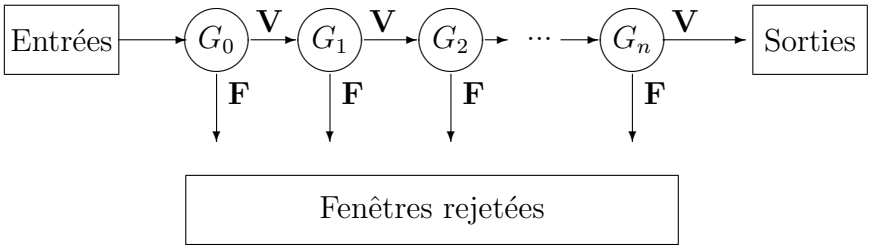


FIGURE 2.5 – Cascade de classifieurs forts.

Cela peut se démontrer intuitivement si nous prenons un premier classifieur G_0 composé d'un faible nombre des fonctions g_t (trois, quatre ou cinq au total) et l'appliquons sur plusieurs régions d'une image. Il est entraîné pour éliminer la moitié des régions négatives. Les régions qui ont subsisté subissent l'application du deuxième classifieur G_1, composé lui aussi d'un faible nombre de fonctions faibles. G_1 est spécialisé dans le traitement des échantillons plus proches de la classe véhicule, qui ont été validées par G_0, et élimine, à son tour, la moitié des fausses alarmes.

Nous pouvons considérer que les régions restantes ont de plus forte probabilité, encore, d'appartenir à la classe véhicule. Pour éliminer la moitié des fausses alarmes, les prochains

1. La complexité d'un classifieur G_i est définie comme le nombre de classifieurs faibles g_t qui le compose.

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

classifieurs seront composés d'un nombre de plus en plus important de g_t, avec pour conséquence l'augmentation du temps de calcul par région. Mais, étant donné que les étapes antérieures ont éliminé successivement un grand nombre des régions négatives, cette augmentation n'a pas d'incidence, du fait du très faible nombre de régions qui restent à évaluer.

Cette architecture s'est ainsi montrée appropriée pour la détection d'objets installés dans des systèmes embarqués [3, 59].

2.4.2 Apprentissage de la *Cascade Attentionelle*

Nous allons entraîner un ensemble de K fonctions de classification $G_1, ..., G_i, ..., G_K$, où chaque G_i est une fonction *forte* entraîné en utilisant l'algorithme Adaboost à l'étape i de la cascade.

Le nombre de classifieurs faibles pour chaque G_i n'est pas fixe : au lieu d'arrêter la boucle itérative d'AdaBoost pour un nombre maximum de descripteurs T, nous fixons deux paramètres de performance pour la fonction G_i : le taux minimum de détections correctes DC_{min} et le taux maximal de fausses alarmes acceptables FA_{max}.

Le choix de DC_{min} et FA_{max} modifie le comportement de la cascade, ainsi que son architecture. Lors de l'apprentissage de la cascade, le seuil S_i du classifieur G_i (équation 2.2) est décrémenté jusqu'à ce que G_i obtient un taux de détections correctes d'au moins DC_{min} sur la base de validation. Plus proche de cent pour cent est DC_{min}, plus petit est S_i ; le modèle du véhicule obtenu a alors une meilleure performance sur les exemples plus difficiles. Cependant, un nombre plus important d'exemples négatifs sera considéré comme positifs (fausses alarmes). La valeur de DC_{min} nous donne aussi le taux de détection théorique de la cascade attentionnelle : $D_{CA} = (DC_{min})^K$, où K est le nombre d'étages de la cascade. Par exemple, pour un $DC_{min} = 99.5$ %, le taux de détections théorique pour une cascade à seize étages obtient un taux de détections de 92.3 %.

Le processus d'apprentissage itératif s'arrête quand G_i atteint un taux de fausses alarmes maximal FA_{max} dans la base négative. Étant donné qu'idéalement, la cascade doit rejeter au moins la moitié des fausses alarmes, FA_{max} peut prendre une valeur égale à 50 %. Si nous donnons à FA_{max} des valeurs plus faibles, G_i a besoin de plus d'itérations (donc plus de fonctions faibles), pour obtenir ce taux de rejet.

---

[3] B. Alefs. Embedded vehicle detection by boosting. In *Intelligent Transportation Systems Conference*, pages 536–541, Vienna, 2006.

[59] H. Ghorayeb, B. Steux, and C. Laurgeau. Boosted algorithms for visual object detection on graphics processing units. In *Asian Conference on Computer Vision*, pages 254–263, Janvier 2006.

Apprentissage de la Cascade

1. Choisir DC_{min} et FA_{max}
2. Obtenir P et N_0 (exemples positifs et négatifs)
3. $i = 0$
4. Tant que $i < K - 1$

   $i = i + 1$

   $n_i = 0$, $f = 1$

   Tant que $f > FA_{max}$

   $n_i = n_i + 1$

   Utiliser P et N_i pour entraîner un classifieur G_i avec T = n_i

   Décrémenter le seuil de G_i jusqu'à atteindre DC_{min} dans la base de validation positive

   Évaluer G_i dans la base N_i pour obtenir f

   Le classifieur G_i est rajouté au détecteur en cascade.

   Évaluer le détecteur en cascade sur la base des images négatives et placer les fausses détections dans N_{i+1}

TABLE 2.2 – Apprentissage de la Cascade Attentionelle

2.5 Évolutions de la méthode de dopage

Depuis son introduction dans la communauté, la méthode de dopage a évolué et plusieurs modifications ont été introduites. Ces transformations cherchent à réduire le nombre des fausses alarmes, accélérer la détection, simplifier l'architecture des détecteurs, etc. Dans ce but, les travaux de recherche s'appuient sur des variantes d'Adaboost, ainsi que sur de nouvelles familles de descripteurs ou des améliorations dans l'architecture de la cascade. Nous allons présenter par la suite, quelques adaptations effectuées à la méthode de dopage pour la détection de véhicule ou de visage.

2.5.1 Variantes d'Adaboost

Schapire & Singer [126] ont repris l'algorithme d'Adaboost et ont suggéré plusieurs modifications pour améliorer le *boosting* en utilisant, entre autres, un taux de confiance donné par une valeur réelle nommée *confidence rated* au lieu de la réponse binaire du Adaboost Discret. Friedman [52] l'a appelé plus tard Adaboost Réel, nom retenu par la communauté scientifique due à sa réponse réelle.

Le tableau 2.3 montre le pseudo-code de cette algorithme.

AdaBoost Réel

1. Soient N exemples $(x_1,y_1),...,(x_N,y_N)$
   avec x_i le vecteur de descripteurs $\in \Re$ de l'exemple i et
   $y_i \in \{1,-1\}$ (positifs et négatifs)
2. Initialiser $w_i = \frac{1}{N}$, $i = 1,...,N$
3. Pour t=1,...,T
   Pour chaque descripteur j, entraîner à partir des w_i
   un classifieur g_j dont l'erreur est définie par :
   $$Z_j = \sum_{i=1} \omega_i e^{-y_i g_j(x_i)}$$
   Choisir le classifieur g_t avec l'erreur Z_t la plus faible
   Actualiser les poids : $\omega_{t+1,i} = \frac{\omega_{t,i} e^{-y_i g_t(x_i)}}{Z_t}$
4. la sortie : $G = sign(\sum_{t=1}^{T} g_t(x))$

TABLE 2.3 – Pseudo-code d'Adaboost Réel.

A chaque itération de l'algorithme, de nouvelles fonctions faibles g_j sont calculées pour chaque descripteur, à partir de la distribution w_i actualisée. Nous définissons les probabilités $P_w(y = 1|j)$ et $P_w(y = -1|j)$ conditionnelles des exemples positifs et négatifs respectivement. L'erreur Z_j est minimisée quand :

$$g_j = \frac{1}{2} ln(\frac{P_w(y = 1|j) + \epsilon}{P_w(y = -1|j) + \epsilon})$$

où ϵ évite la division par zéro et peut prendre la valeur 1/N.

Le GentleBoost [52] propose de calculer la fonction faible à partir d'une approximation aux moindres carrés, qui conduit au choix suivant :

$$g_j = P_w(y = 1|j) - P_w(y = -1|j)$$

[126] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3) :297–336, 1999.

[52] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : a statistical view of boosting. *The Annals of Statistics*, 28(2) :337–374, April 2000.

La méthode FloatBoost [92], ajoute une nouvelle étape à l'apprentissage du classifieur fort. Après avoir ajouté le dernier g_T, elle élimine de l'ensemble, les g_i qui n'ont plus un apport au niveau de la classification de sorte à minimiser l'erreur totale du classifieur.

2.5.2 Nouveaux descripteurs

Le travail de Khammari [83] propose une nouvelle famille de descripteurs appelés **points de contrôle** qui emploient les niveaux de gris de pixels pour la détection de véhicules. Chaque descripteur est défini par deux ensembles de points de contrôle : un ensemble positif $\{p_1, ..., p_i\}$ et un ensemble négatif $\{n_1, ..., n_j\}$, où chaque p et n correspondent à l'intensité d'un pixel dans l'image. Les primitives examinent les intensités des pixels dans $\{p_1, ..., p_i\}$ et $\{n_1, ..., n_j\}$ pour toutes les résolutions, et répondent positivement si :

$$\forall p \in \{p_1, ..., p_i\} \ \wedge \ \forall n \in \{n_1, ..., n_j\}, \ val(p) - val(n) \geq \theta$$

avec $\theta \in \Re$. La figure 2.6 montre un exemple des points de contrôles choisis par l'algorithme d'Adaboost dans les trois résolutions.

FIGURE 2.6 – Exemple de points de contrôle.

Étant donné le grand nombre de combinaisons de pixels qui peuvent être obtenues à partir de cette famille de primitives, un algorithme génétique a été mis en place pour faire le choix des ensembles. A chaque nouvelle génération, l'algorithme génétique réalise une mutation pour améliorer le choix de l'ensemble des primitives, en changeant le nombre des points de contrôle, le seuil et la résolution.

---

[92] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *European Conference on Computer Vision*, pages 67–81, London, 2002.

[83] A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau. Vehicle detection combining gradient analysis and adaboost classification. In *IEEE International Conference on Intelligent Transportation Systems*, pages 66–71, Vienna, Austria, Septembre 2005.

Alefs [3] utilise AdaBoost pour l'apprentissage d'un détecteur des véhicules et fait une comparaison entre plusieurs descripteurs :

1. filtres de Haar, | A - B | / (A + B), où A et B sont les moyennes des valeurs d'intensités dans les deux rectangles des filtres.
2. filtres orientés : ils gardent le signe de la soustraction (A - B) / (A + B).
3. filtres orientés 2 : chaque descripteur précédent est décliné en tenant compte de l'axe vertical de symétrie : une paire de descripteurs {A1,B1} et {A2,B2} est alors obtenu. Le descripteur final est calculé comme : A = A1 + A2, B = B1 + B2.
4. histogrammes d'orientation de contours : ils s'inspirent des descripteurs de Gepperth [57]. 8 classes par histogramme sont considérées, et couvrent les 360 degrés. Le critère de discrimination est donné par le produit scalaire entre les histogrammes des exemples positifs et négatifs. Une extension détermine l'angle entre les histogrammes des régions (rectangles) de type filtres de Haar.

Ils utilisent l'architecture en cascade et ont limité la quantité des fonctions faibles par étage à 50, pour éviter les cas de non-convergence.

Les travaux de Murphy [76] et Wang [145] proposent une fusion de différents filtrages : filtres de contours, filtres de coins et filtres Laplacians pour le premier et filtres d'intensité, différences de Gaussiennes et filtres de Gabor pour le second.

2.5.3 Optimisation de la cascade

Withopf [147] emploie le Gentle Adaboost et les descripteurs de Haar pour créer un détecteur de véhicules. L'architecture en cascade est optimisée à l'aide des fonctions faibles de classification du type *Nested* (voir aussi le travail de Huang [69]) où à chaque étage est récupéré le résultat de l'étage antérieur. Il affirme que les descripteurs du deuxième étage d'une cascade sont très semblables aux descripteurs du premier étage, étant donné que les négatifs n'ont pas varié beaucoup. Avec l'information de l'étage précédent, les descripteurs sont différents et le taux de faux positifs se réduit parce que l'algorithme utilise, d'une

[3] B. Alefs. Embedded vehicle detection by boosting. In *Intelligent Transportation Systems Conference*, pages 536–541, Vienna, 2006.

[57] A. Gepperth, J. Edelbrunner, and T Bocher. Real-time detection and classification of cars in video sequences. In *IEEE Intelligent Vehicles Symposium*, pages 625–631, Juin 2005.

[76] A. Torralba K. Murphy and W. T. Freeman. Using the forest to see the trees : a graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems*, 2003.

[145] H. Wang, P. Li, and T. Zhang. Boosted gaussian classifier with integral histogram for face detection. *International Journal of Pattern recognition and Artificial Intelligence*, 21(7) :1127–1139, 2007.

[147] D. Withopf and B. Jahne. Improved training algorithm for tree-like classifiers and its application to vehicle detection. In *IEEE Intelligent Transportation Systems Conference*, pages 642–647, Septembre 2007.

[69] C. Huang, H. Ai, B. Wu, and S. Lao. Boosting nested cascade detector for multi-view face detection. In *IEEE International Conference on Pattern Recognition*, volume 2, pages 415–418, 2004.

certaine façon, les descripteurs précédents. Étant donné que la quantité de descripteurs est plus faible, par rapport à une cascade conventionnelle, l'algorithme est plus rapide.

Ponsa [117] utilise le Real AdaBoost et les filtres de Haar pour l'apprentissage d'un détecteur de véhicules. Il a fait deux améliorations dans l'algorithme pour minimiser le nombre de descripteurs évalués par fenêtre et accélérer la classification. La première amélioration consiste à vérifier l'évolution de la somme pondéré du classifieur *fort* pour estimer le résultat et arrêter le calcul, ce qui lui permet de gagner en temps de réponse. L'autre amélioration se situe au niveau de l'apprentissage, en dédoublant les classifieurs forts et, chacun de ces nouveaux classifieurs et entraîné à partir des exemples négatifs qui ont été facilement classifiés. Un étage de la cascade est donc composé de plusieurs classifieurs forts qui sont appliqués sucesivement. Cette manipulation leur permet de diminuer le nombre de descripteurs requis pour évaluer chaque région.

Dans d'autres domaines de la détection d'objets, telle que la détection de visage, sont proposés des optimisations dans l'apprentissage de la cascade [18] ou leur fusion avec un arbre de décision [71].

2.6 Discussion

Dans ce chapitre, nous avons introduit la méthode de dopage pour l'apprentissage d'un classifieur. L'espace de paramètres ainsi que l'architecture en cascade, utilisés par Viola et Jones, ont été aussi exposés. Nous pouvons conclure que :

- L'avantage de descripteurs utilisés par Viola et Jones consiste à pouvoir les calculer en temps réel à travers l'image intégrale. Ils permettent aussi de gérer la détection d'objets en plusieurs échelles.
- L'algorithme Adaboost nous permet de sélectionner, parmi un grand nombre, les descripteurs les plus discriminants pour la classification.
- La fusion des fonctions de classification *faibles* associés à ces descripteurs forment un classifieur de bonne performance.
- La cascade apporte deux bénéfices principaux : minimise du temps de calcul, et affine les frontières de classification à travers le *bootstrapping* implicitement réalisé en introduisant des exemples de plus en plus difficiles dans l'apprentissage de chaque nouvel étage.

---

[117] D. Ponsa and A. Lopez. Cascade of classifiers for vehicle detection. In *Advanced Concepts for Intelligent Vision Systems*, volume 4678, pages 980–989. Springer, 2007.

[18] S. Charles Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1-3) :65–86, 2008.

[71] K. Ichikawa, T. Mita, and O. Hori. Component-based robust face detection using adaboost and decision tree. In *International Conference on Automatic Face and Gesture Recognition*, pages 413–420, 10-12 April 2006.

Dans le chapitre suivant, nous allons utiliser cette méthode pour construire notre détecteur de véhicules.

Chapitre 3

Conception du détecteur et Implémentation

3.1 Introduction

Dans ce chapitre, nous allons proposer la création de détecteurs similaires à celui de Viola et Jones et construits à partir de deux types de descripteurs : les filtres rectangulaires (par abus de langage, nous les désignerons par descripteurs de Haar) et les histogrammes de gradient orienté (*Histograms of Gradients*, HoG).

Les deux descripteurs choisis, les filtres de Haar et les histogrammes de gradients orientés, ont une nature similaire. La différence réside en une approche en régions pour les premiers et une approche locale pour les seconds. Les filtres prennent en compte la (dis)similarité entre régions. Pour leur part, les histogrammes sont obtenus par calcul du gradient qui décrit localement la variation des niveaux de gris. Nous détaillons dans la section 3.2 ces deux familles de descripteurs.

Aux premiers, nous allons associer une fonction de classification discriminante et aux seconds une fonction de classification générative. Dans toutes les approches de la littérature, les HoG sont associés à des classifieurs discriminants. Soit chaque *classe* (*bin* de l'histogramme) est considérée comme une dimension du vecteur de paramètres [101, 35, 10, 58] associé à un algorithme de classification discriminant. Soit l'ensemble des *classes* de chaque

[101] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, pages 69–82, 2004.

[35] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 2, pages 886–893, Juin 2005.

[10] J. Begard, N. Allezard, and P. Sayd. Real-time humans detection in urban scenes. In *British Machine Vision Conference*, University of Warwick, UK, 2007.

[58] D. Geronimo, A. Lopez, D. Ponsa, and A.D. Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 418–425, 2007.

descripteur SIFT [157] est associé à un classifieur discriminant. De plus, nous allons proposer un nouveau détecteur mixte qui combine dans son architecture les deux types de descripteurs : Haar (discriminants) et HoG (génératifs). Certains papiers proposent aussi la fusion entre différents descripteurs (Haar+HoG [58] ou Haar+Gabor [132]), mais ils l'abordent selon un point de vue strictement discriminant.

Enfin, les architectures décrites dans la section 3.4 et obtenues automatiquement par Adaboost pendant l'apprentissage du détecteur Haar+HoG nous permettront de confirmer la complémentarité des classifieurs génératifs et discriminants [107], notamment dans les résultats du chapitre 4. Cette complémentarité avait été constatée auparavant de façon intuitive à partir de la combinaison séquentielle de ces deux types de classifieurs [53, 120].

3.2 Espace de représentation

La méthodologie de détection consiste en une recherche dans l'image à l'aide d'une fenêtre glissante qui évalue à chaque position la présence d'un véhicule. Nous allons donc avoir une décision binaire qui correspond à une classification à deux classes : véhicule et non-véhicule.

Les exemples positifs (vignettes avec véhicules) ou négatifs (vignettes sans véhicules), se distribuent dans un espace à N dimensions qui dépend des paramètres choisis pour représenter l'information. Dans l'espace initial (défini par les niveaux de gris de l'image), ces exemples peuvent être mélangés. En sélectionnant l'espace de représentation et le classifieur adéquats, on peut espérer les séparer (voir figure 3.1). Dans ce travail, deux types de paramètres ont été évalués : les filtres de Haar et les histogrammes de gradient orienté. Chaque descripteur est associé à une fonction de classification différente :

---

[157] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE International Converence on Computer Vision and Pattern Recognition*, pages 1491–1498, Washington, DC, États Unis, 2006.

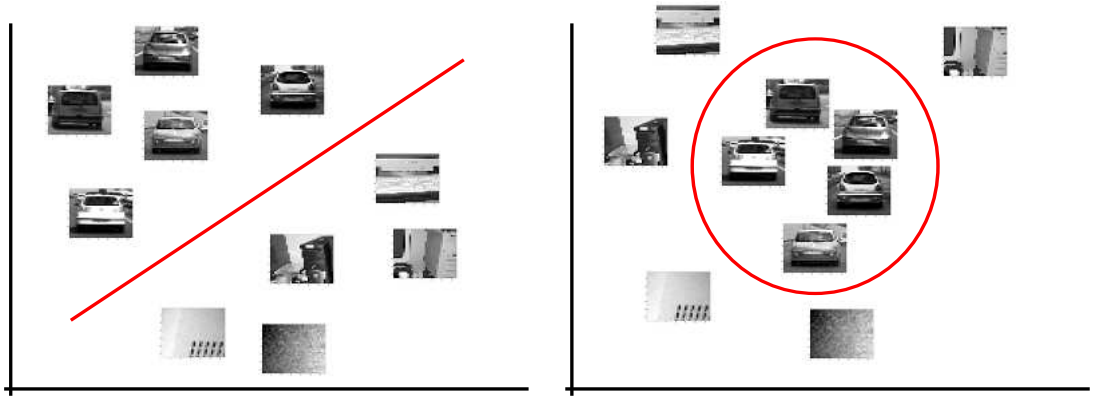
[58] D. Geronimo, A. Lopez, D. Ponsa, and A.D. Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 418–425, 2007.

[132] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2) :125–137, Juin 2005.

[107] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. In MIT Press, editor, *Conference on Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[53] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminative models for object category detection. In *IEEE International Conference on Computer Vision*, volume 2, pages 1363–1370, 2005.

[120] L. Prevost, L. Oudot, A. Moises, C. Michel-Sendis, and M. Milgram. Hybrid generative/discriminative classifier for unconstrained character recognition. *Pattern Recognition Letters*, 26(12) :1840–1848, 2005.



(a) (b)

FIGURE 3.1 – Vignettes des exemples positifs et négatifs dans des espaces de représentation 2D et fonction de classification.
(a) Fonction de classification discriminante, (b) générative.

- les fonctions de classification discriminantes trouvent un espace de projection et une frontière qui permet de classifier directement les exemples d'entrée. Dans notre approche, une entrée est projetée dans l'espace de classification à partir des valeurs des descripteurs de Haar. Adaboost trouve cet espace et la frontière (hyperplan) de classification associée.
- les fonctions de classification génératives que nous utilisons, au lieu d'estimer les densités de probabilités conditionnelles des classes et d'utiliser la règle de Bayes pour classifier, calculent la similarité entre une entrée inconnue et un modèle de la classe véhicule estimé a priori sur les exemples de la base d'apprentissage positive. L'algorithme Adaboost nous permet de raffiner ce modèle en sélectionnant les descripteurs les plus représentatifs pour réaliser la classification. Les descripteurs choisis ici sont les histogrammes de gradient orienté.

3.2.1 Descripteurs de Haar

La figure 3.2 montre l'ensemble des filtres de Haar utilisé dans nos travaux. Les filtres choisis sont ceux à *deux* et à *trois* rectangles. D'autres filtres, diagonaux, etc, ne nous apportent pas plus d'information : les voitures sont quasiment rectangulaires et ces quatre filtres suffisent pour définir ce gabarit.

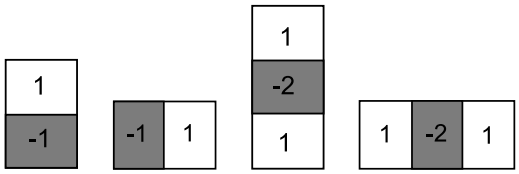


FIGURE 3.2 – Ensemble de filtres de Haar.

L'image 3.3 illustre le filtrage d'une image avec deux rectangles en deux échelles 1x2

pixels et 2x4 pixels. Les points plus clairs correspondent aux valeurs les plus grandes des descripteurs (en valeur absolue). Les figures montrent que les rectangles choisis mettent en valeur les contours verticaux et horizontaux de l'image à différentes échelles. L'image 3.4 montre les valeurs moyennes des descripteurs calculées à partir de tous les exemples positifs de la base d'apprentissage. On peut remarquer que les positions des contours verticaux et horizontaux sont assez stables dans l'ensemble des exemples d'apprentissage. Ils apparaissent donc avec une forte intensité dans les imagettes moyennes. De plus, les régions des feux et des fenêtres obtiennent des valeurs faibles des descripteurs étant donnée l'absence des contours : ce sont des régions homogènes. Les valeurs faibles autour de "la voiture" sont dues au bruit induit par le fond, non constant d'image en image.

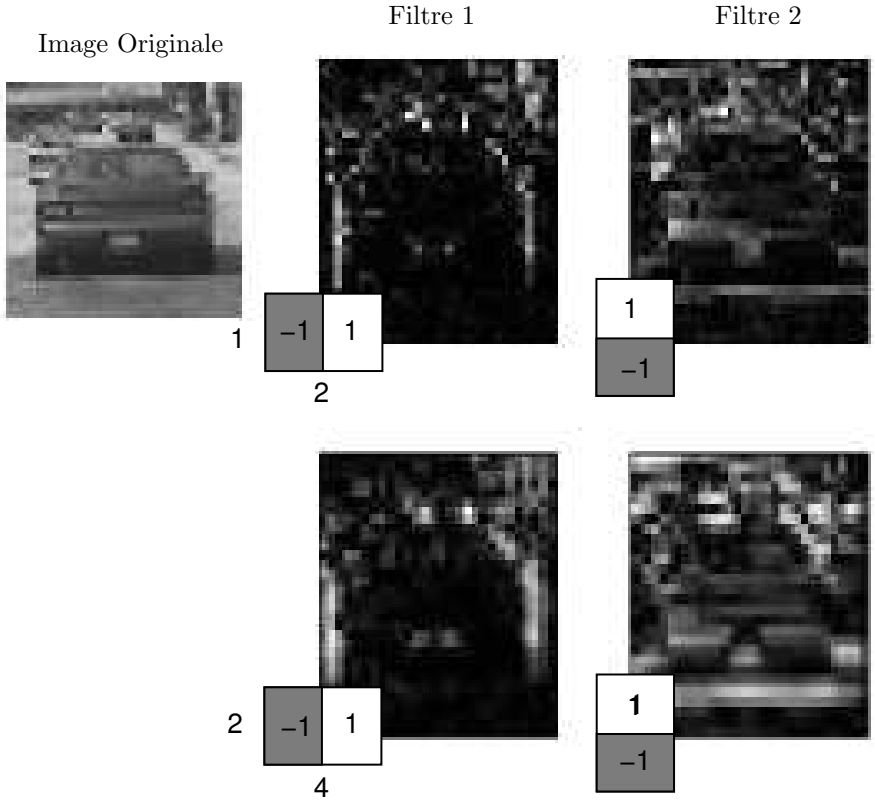


FIGURE 3.3 – Vignette positive originale et décomposition par deux filtres de Haar à différentes échelles.

Nous pouvons apprécier dans les deux figures, qu'en incrémentant la taille des filtres nous perdons des détails de la figure originale, bien que les contours principaux soient conservés.

Chaque descripteur j est donc issu de l'application d'un des filtres dans une partie de l'imagette. Nous pouvons le définir comme une fonction $f_j(x_j, y_j, s_j, r_j)$, où (x_j, y_j) est la position dans la vignette, r_j est le type de filtre rectangulaire (voir fig. 3.2) et s_j représente l'échelle (nous en avons 5 au total : 1x2, 2x4, 4x8, 8x16 et 16x32).

Afin de calculer tous les descripteurs de Haar pour une vignette de taille 32x32 pixels,

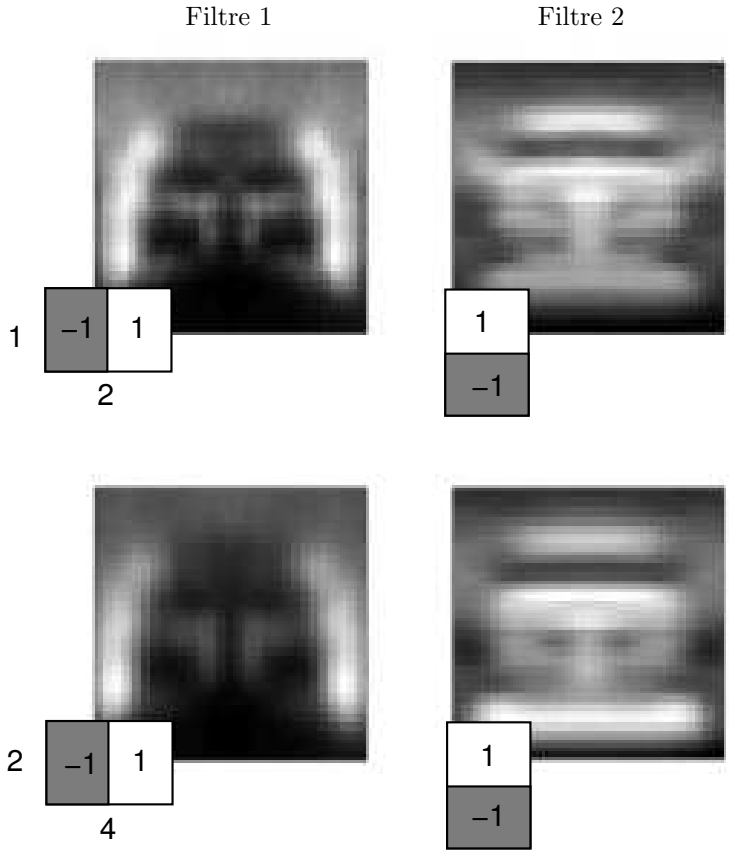


FIGURE 3.4 – Valeurs moyennes des descripteurs de Haar calculées sur la base de véhicules.

nous procédons de la façon suivante :

La taille des filtres rectangulaires est donnée par la variable dx.

- La valeur initiale de $dx = 1$ pixel.
- Le premier filtre : $haar_{2y}$ a une taille (dx,2*dx) (qui corresponde au premier filtre de la figure 3.2). Nous parcourons la vignette de la position x = 0 jusqu'à x = 32 - dx, dans les sens horizontal et de y = 0 jusqu'à y = 32 - 2*dx, dans le sens vertical, avec un pas d'un pixel.
- Le deuxième filtre : $haar_{2x}$ a une taille (2*dx,dx). Nous parcourons la vignette de x = 0 jusqu'à x = 32 - 2*dx et de y = 0 jusqu'à y = 32 - dx.
- Le troisième filtre $haar_{3y}$ a une taille (dx,3*dx) (il corresponde au troisième filtre de la figure 3.2). Nous parcourons la vignette de la position x = 0 jusqu'à x = 23 - dx et de y = 0 jusqu'à y = 32 - 3*dx.
- Pour le quatrième filtre $haar_{3x}$ qui a une taille (3*dx,dx), nous parcourons la vignette de x = 0 jusqu'à x = 32 - 3*dx et de y = 0 jusqu'à y = 32 - dx.
- Ensuite, nous multiplions dx par deux : $dx = dx * 2$.

Nous répétons l'opération 4 fois de plus. Au total nous avons 5 échelles (le valeur de dx plus grand est de 16 pixels).

Pour une vignette de taille 32x32 pixels, l'espace de paramètres de Haar est donc de dimension 11378. Le tableau 3.1 détaille le nombre de filtres pour chaque échelle.

	dx				
Filtre	1	2	4	8	16
$haar_{2x}$	992	899	725	425	17
$haar_{2y}$	992	899	725	425	17
$haar_{3x}$	960	837	609	225	0
$haar_{3y}$	960	837	609	225	0

TABLE 3.1 – Quantité de filtres rectangulaires pour les différentes valeurs de dx et les différentes types de filtres

Une vignette positive ou négative est projeté dans l'espace de représentation à partir de la liste complète de descripteurs de Haar (voir figure 3.5). Adaboost va choisir au moment de l'apprentissage, les descripteurs de la liste les plus discriminants (ceux qui mieux séparent les exemples positifs de négatifs de la base d'apprentissage) à travers sa fonction de classification faible (section 3.3.1).

$\{ f_1 \quad f_2 \quad f_3 \quad ... \quad f_{NHaar} \}$

FIGURE 3.5 – Vecteur de descripteurs de Haar pour une vignette avec un véhicule.

3.2.2 Histogrammes de Gradients Orientés

Ce descripteur local utilise le module et l'orientation du gradient autour d'un point d'intérêt ou à l'intérieur d'une région de l'image.

Pour le calcul du gradient, nous utilisons des filtres de Sobel de taille 3x3 sur l'image d'entrée en niveaux des gris, l'un pour la direction x, S_x, et l'autre pour la direction y, S_y. Le module du gradient est alors défini par :

$$M(u,v) = |S_x * i(u,v)| + |S_y * i(u,v)|$$

et son orientation par :

$$\phi(u,v) = tan^{-1}(\frac{S_x * i(u,v)}{S_y * i(u,v)})$$

Les orientations des pixels sont quantifiées afin qu'elles prennent des valeurs entières entre 0 et $N_{orr} - 1$ (ici N_{orr} est égal à 4). Nous utilisons le modulo π au lieu du modulo 2π, pour les valeurs des orientations, étant donné que les véhicules peuvent être de différents couleurs.

Chacun des histogrammes est construit de la façon suivante :

- nous parcourons tous les pixels de la région rectangulaire,
- pour chaque pixel d'orientation o, nous incrémentons la classe correspondante de l'histogramme de la valeur du module du gradient en ce point,
- une fois évalués tous les points, nous normalisons l'histogramme pour que la somme des intensités des classes soit égale à 1.

Chaque descripteur j est défini comme une fonction : $h_j(x_j, y_j, s_j, r_j)$, où (x_j, y_j) est la position dans la vignette, r_j le type de rectangle et s_j l'échelle. Les types de rectangles dépendent de la relation entre la largeur et la hauteur qui peut être (2x2), (2x3), (3x2). Nous considérons au total quatre échelles : $s : \{2, 4, 8\}$.

Les rectangles (ou carrés) où nous calculons les histogrammes de gradient orienté pour une vignette de taille 32x32 pixels sont obtenus de la façon suivante :

La taille des rectangles est donnée par la variable dx.

- La valeur initiale de $dx = 2$ pixel.
- Le premier rectangle : hog_1 a une taille (2*dx,2*dx). Nous parcourons la vignette de la position x = 1 jusqu'à x = 31 - 2*dx, dans les sens horizontal et de y = 1 jusqu'à y = 31 - 2*dx, dans le sens vertical, avec un pas d'un pixel. Nous commençons dans la position 1 au lieu de la position égale à zéro afin d'éviter des effets de bords du calcul du gradient.
- Le deuxième rectangle : hog_{3x} a une taille (3*dx,2*dx). Nous parcourons la vignette de x = 1 jusqu'à x = 31 - 2*dx et de y = 1 jusqu'à y = 32 - 2*dx.
- Le troisième rectangle hog_{3y} a une taille (2*dx,3*dx). Nous parcourons la vignette de la position x = 1 jusqu'à x = 31 - 2*dx et de y = 1 jusqu'à y = 31 - 3*dx.
- Ensuite, nous multiplions dx par deux : $dx = dx * 2$.

Nous répétons l'opération 2 fois de plus. Au total nous avons 3 échelles (le valeur de dx plus grand est de 8 pixels).

Pour une vignette de taille 32x32 pixels, l'espace de paramètres de HoG est donc de dimension 3917. Le tableau montre en détail la quantité de rectangles pour chaque échelle.

	dx		
Rectangle	2	4	8
hog_1	729	529	225
hog_{3x}	675	437	105
hog_{3y}	675	437	105

TABLE 3.2 – Quantité de rectangles pour les différentes valeurs de dx.

La figure 3.6 présente deux exemples des histogrammes normalisés obtenus à partir de deux régions d'une image. Nous observons que dans une des régions, les points de contours trouvés sont, en majorité, d'orientation horizontale (classe deux de l'histogramme). L'autre région, contient des pixels de contours de toutes les classes, avec cependant une majorité de contours verticaux.

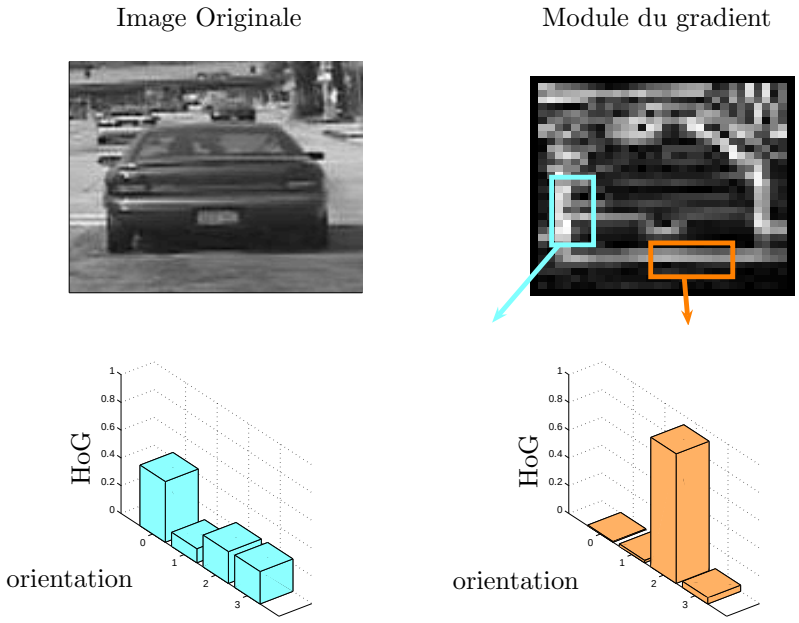


FIGURE 3.6 – Exemples de descripteurs de HoG extraits de l'image d'un véhicule.

Les vignettes sont ainsi projetées dans l'espace de représentation de HoG à partir de la liste de histogrammes (voir figure 3.7). Adaboost choisit plus tard ceux qui sont les plus similaires au modèle de véhicule (voir section 3.3.2). et, au même temps, rejettent le mieux les exemples négatifs de la base d'apprentissage.

$$\{\ h_1 \quad h_2 \quad h_3 \quad ... \quad h_{NHoG}\ \}$$

FIGURE 3.7 – Vecteur de descripteurs de Haar pour une vignette avec un véhicule.

Nous allons utiliser une représentation intermédiaire de l'image d'entrée, inspirée de l'image intégrale, qui permet le calcul rapide des histogrammes : l'histogramme intégral [118]. Nous allons ainsi obtenir un tableau tri-dimensionnel (la troisième dimension correspondant à l'orientation) qui nous permet de calculer l'accumulation des valeurs du module du gradient pour une orientation donnée dans une région à partir de quatre références à l'histogramme intégral. Le tableau *hh(u,v,o)* est calculé à partir d'une image en niveaux de gris i(u,v) et le module $M(u, v)$ et l'orientation $\phi(u, v)$ (déjà quantifié) du gradient.

[118] F. Porikli. Integral histogram : A fast way to extract histograms in cartesian spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.

$$hh(u, v, o') = \sum_{\substack{u'<u \\ v'<v}} M(u', v') | o' = \phi(u', v')$$

Nous pouvons donc, calculer l'accumulation des valeurs du module du gradient dans une région à partir de quatre références à l'histogramme intégral. Ensuite, l'histogramme complet est évalué avec $4N$ références à hh.

3.2.3 Espace mixte de représentation

Nous proposons dans ce travail une fusion des deux espaces de représentation précédents : filtres rectangulaires et histogrammes de gradient orientés. Une vignette sera donc projetée dans un espace *mixte* composé des deux vecteurs de descripteurs (voir figure 3.8). Pendant l'apprentissage du classifieur, Adaboost a la possibilité de choisir le descripteur le plus discriminant dans les deux familles. Le classifieur fort peut donc être composé d'une combinaison des fonctions faibles de classification appartenant aux deux espaces.

$$\{\ f_1 \quad f_2 \quad f_3 \quad ... \quad f_{NHaar}\ \} \cup \{\ h_1 \quad h_2 \quad h_3 \quad ... \quad h_{NHoG}\ \}$$

FIGURE 3.8 – Fusion des deux vecteurs de descripteurs pour une vignette avec un véhicule.

3.3 Fonctions de classification

Il nous faut à présent, définir les fonctions faibles g pour les deux types de descripteurs : Haar et HoG.

3.3.1 Classifieur de Haar discriminant

Nous définissons la fonction de classification faible associée au descripteur j, comme une réponse binaire g_j^{Haar} :

$$g_j^{Haar} = \begin{cases} 1 & \text{si } p_j f_j < p_j \theta_j \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

où f_j nous donne la valeur de sortie du filtre, θ_j est le seuil et p_j est la parité.

3.3.2 Classifieur de HoG génératif

Chaque histogramme h_i est composé de N classes qui sont, dans la littérature, l'objet de la constitution d'un classifieur discriminant. Les travaux qui utilisent des HoG discriminants à partir d'une seule classe de l'histogramme ajoutent, pour chaque classe sélectionnée par Adaboost, les autres classes comme autant de fonctions *faibles* supplémentaires [10]. Nous considérons que prendre individuellement la valeur de chaque classe pour prendre une décision de classification est méconnaître la composition d'un histogramme comme un ensemble.

C'est ainsi qu'est née l'idée du classifieur génératif qui calcule, pour chaque descripteur j, la distance entre un histogramme $h_j(x_j, y_j, s_j, r_j)$ de l'image d'entrée et l'histogramme correspondant du modèle $m_j(x_j, y_j, s_j, r_j)$. Effectivement, cette façon d'évaluer un descripteur permet de prendre en compte toutes les classes de l'histogramme au même temps.

Afin de définir ce modèle, qui va représenter la connaissance à priori de notre classe de véhicule, nous traitons tous les histogrammes h_j séparément. Chaque histogramme h_j est composé de N_{or} classes : $h_j = (c_0, c_1, c_2, c_3)$ ($N_{or} = 4$). Le modèle pour ce descripteur j va être un histogramme m_j calculé en prenant en compte tous les exemples d'apprentissage positifs :

$$m_j = (c_0^m, c_1^m, c_2^m, c_3^m)/c_k^m = median\{c_k^i\}_{i=1,\ldots,P}$$

où c_k^i représente la classe k de l'histogramme h_j calculé dans l'exemple i et P est le nombre total des exemples positifs de la base. Nous utilisons la fonction médiane parce qu'elle ne prend pas en compte les valeurs aberrantes, comme la moyenne.

Nous définissons alors le classifieur faible g_j^{HoG} associé au descripteur j tel que :

$$g_j^{HoG} = \begin{cases} 1 & \text{si } d(h_j, m_j) < \theta_j \\ 0 & \text{si non} \end{cases} \tag{3.2}$$

où $d(h_j, m_j)$ est la distance de Bhattacharyya [77] entre l'histogramme h_j et le modèle m_j, et θ_j est le seuil optimal sur la distance pour ce descripteur.

La distance de Bhattacharyya est définie comme :

$$d(h_j, m_j) = \sqrt{1 - h_j \bullet m_j}$$

où [$\bullet$] est le produit scalaire.

Cette distance, bornée entre [0,1], est largement utilisée pour mesurer la similarité entre deux histogrammes. Il en existe d'autres mais il est reconnu que celle-ci réalise un bon compromis entre complexité de calcul et performance.

---

[10] J. Begard, N. Allezard, and P. Sayd. Real-time humans detection in urban scenes. In *British Machine Vision Conference*, University of Warwick, UK, 2007.

[77] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1) :52–60, 1967.

3.3.3 Détermination du seuil optimal

Les fonctions de classification faibles g_{Haar} et g_{HoG} évaluent la valeur d'un descripteur par rapport à un seuil θ_j. Dans cette section, nous allons calculer le seuil optimal pour chacun de descripteurs.

Adaboost choisit à chaque itération, la fonction faible g_j qui obtient l'erreur la plus petite, donnée par l'équation :

$$\epsilon_j = \sum_{i=1} \omega_i |g_j(x_i) - y_i|$$

où w_i est le poids de l'exemple i.

Le terme $|g_j(x_i) - y_i|$ reçoit la valeur 1 si g_j réalise une mauvaise prédiction de la classe de l'exemple i. Ainsi, pour la parité négative $p_j = -1$, le terme de la soustraction est égal à 1 si :

- soit l'exemple étiqueté $(x_i, y_i = 1)$ et $g_j(x_i) = 0 \Rightarrow f_j(x_i) < \theta_j$: on est dans le cas d'un exemple positif et la valeur du descripteur j est plus petite que le seuil.
- soit l'exemple étiqueté $(x_i, y_i = 0)$ et $g_j(x_i) = 1 \Rightarrow f_j(x_i) > \theta_j$: on est maintenant, dans le cas d'une fausse alarme, où la valeur du descripteur de l'exemple négatif est plus grande que le seuil.

L'erreur est donnée par la somme de ces deux termes :

$$\epsilon_j = \sum_{\substack{f_j(x_i) > \theta_j \\ y_i = 0}} \omega_i + \sum_{\substack{f_j(x_i) < \theta_j \\ y_i = 1}} \omega_i \tag{3.3}$$

$$\epsilon_j = \sum_{f_j(x_i) > \theta_j} \omega_{n,i} + \sum_{f_j(x_i) < \theta_j} \omega_{p,i} \tag{3.4}$$

$$\epsilon_j = s_n + s_p \tag{3.5}$$

où $\omega_{n,i}$ les poids des exemples négatifs et $\omega_{p,i}$ les poids des exemples positif.

Une représentation graphique peut nous aider à mieux interpréter les équations précédentes. Pour un descripteur j, nous pouvons représenter dans un histogramme la distribution des valeurs dans la base des exemples positifs et négatifs (fig. 3.9) où chaque exemple est pondéré par les poids w_i. Ces histogrammes sont montrés dans la figure 3.9 où on a échantillonné les valeurs du descripteur j en 128 classes. Chacun de ces classes représente une valeur de seuil θ. Ils peuvent être aussi interprétés comme la probabilité $P(y = c|x)$ conditionnelle de la classe c. Les distributions positive et négative, pondérées pour la distribution des poids, seront respectivement $P_w(y = 1|x)$ et $P_w(y = 0|x)$.

La figure 3.9 montre aussi la représentation des équations (cumulées) 3.3. En faisant varier le θ, nous obtenons l'erreur e_θ associée. En prenant par exemple une classe θ, la valeur s_p de la somme des poids des exemples positifs avec $f_j < \theta$ est la valeur de l'ordonné à l'abscisse θ dans l'histogramme cumulé de la figure 3.9 (c). Également, on obtient s_n pour les exemples négatifs.

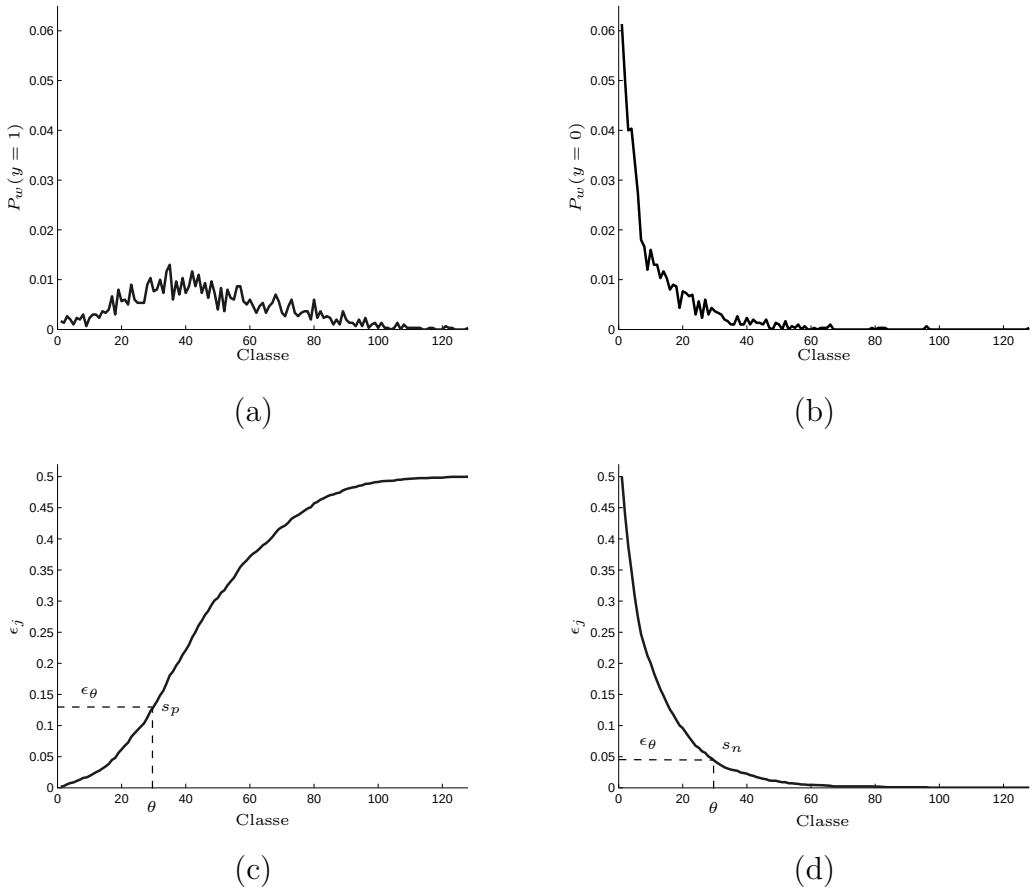


FIGURE 3.9 – Densité de probabilité des exemples d'apprentissage et fonctions de répartition.

(a) Densité de probabilité des exemples positives, (b) Densité de probabilité des exemples négatives, (c) Fonction de répartition des exemples positifs et (d) Fonction de répartition des exemples négatifs

Étant donné que nous cherchons à minimiser ϵ_j, cela veut dire, minimiser la somme $s_p + s_n$. Si on somme les deux histogrammes des erreurs cumulées on trouve la valeur de l'erreur pour chaque classe.

Dans l'exemple de la figure 3.10 on a trouvé le minimum de la somme dans θ_{min} : $e_{min} = 0.1537$. Cela correspond au bin 22, qui équivaut à une valeur : $\theta_{min} = 10.82$ pour ce descripteur. Ainsi, $\theta_j = \theta_{min}$.

Nous allons utiliser cette méthode pour définir les fonctions faibles associées aux descripteurs. Un raisonnement similaire peut être effectué pour la parité $p_j = 1$, ainsi que pour les distributions de probabilités obtenues du calcul de distances entre les histogrammes. Dans la section suivante, nous allons obtenir les détecteurs de véhicules à partir de la méthode de dopage.

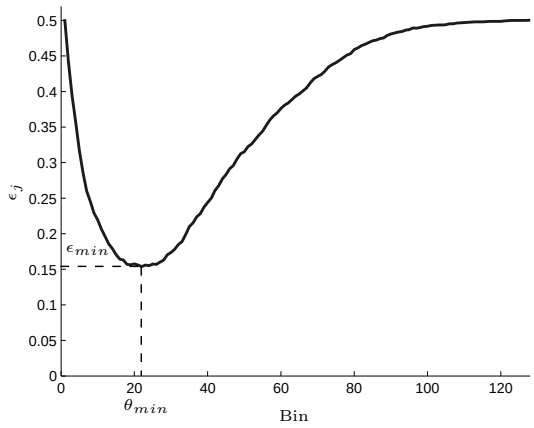


FIGURE 3.10 – Calcul du seuil qui minimise les erreurs.

3.4 Définition de la méthode

Dans cette section, nous allons présenter trois détecteurs d'architectures différentes : détecteur simple, détecteur en cascade et cascade contrôlée. Pour chaque architecture, nous obtenons un détecteur par espace de paramètres : Haar discriminant, HoG génératif et le détecteur Mixte qui représente l'union des deux précédents.

3.4.1 Détecteur simple

L'architecture du détecteur simple est composée d'un classifieur *fort* G, construit avec un nombre T de classifieurs *faibles* (voir figure 3.11).

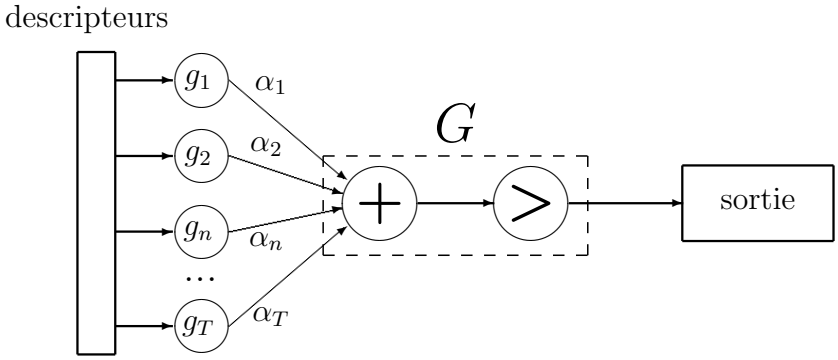


FIGURE 3.11 – Architecture du détecteur simple.

La sortie binaire du classifieur *fort* valide ou rejette une fenêtre dans l'image de test. Pour l'apprentissage, nous avons utilisé une base de véhicule de tourisme pour les exemples positifs. Les exemples négatifs ont été récoltés au hasard à partir d'une base ne contenant pas d'images de véhicules.

3.4.2 Détecteur en Cascade

L'architecture du détecteur en cascade est celle présentée dans la section 2.4. Les bases d'apprentissage sont les mêmes que celles utilisées dans l'entraînement des détecteurs simples, mais la base des exemples négatifs a été séparée dans 20 dossiers (20 est le nombre maximum choisi d'étages dans la cascade). La base négative N_i, utilisée pour entraîner le classifieur *fort* G_i de l'étage i, est formée d'exemples négatifs ayant été mal classés (c'est-à-dire considérés comme des véhicules) par les étages précédents.

Nous avons défini trois critères d'arrêt pour l'apprentissage de la cascade :

1. l'algorithme est arrivé à un nombre T_{max} d'itérations pour l'apprentissage du classifieur fort G_i sans obtenir un taux de fausses alarmes plus petit que FA_{max} (T_{max} est égal à 200 pour notre algorithme). Nous noterons dans nos résultats, certaines cascades *Non conv*, pour indiquer qu'il n'y a pas eu convergence d'Adaboost dans le dernier étage.
2. le nombre d'exemples négatifs trouvés dans les bases est plus petit que la moitié du nombre des exemples positifs (noté *Nég. non atteint*).
3. nous sommes arrivés à un nombre fixe K d'étage maximum, ici égal à 20.

Nous obtenons trois détecteurs en utilisant les trois espaces de paramètres : les filtres de Haar, les HoG et leur fusion (classifieur Mixte). Deux versions différentes pour chaque détecteur sont obtenues en faisant varier la quantité des exemples négatifs utilisée pour l'apprentissage : 2000 et 4000.

La figure 3.12 montre les descripteurs choisis dans les trois premiers étages pour les trois types de détecteurs. Les rectangle rouges représentent les descripteurs de type HoG et les rectangles verts les filtres rectangulaires.

La première conclusion que nous pouvons tirer de cette figure est que la quantité de descripteurs choisis par les détecteurs de type Hog et Mixte est beaucoup plus faible que le détecteur de type Haar. De plus, les détecteurs de type HoG ont été majoritairement sélectionnés dans ces premiers étages pour le détecteur Mixte.

3.4.3 Cascade controlée

Pour contourner le cas de non-convergence pendant l'apprentissage d'une cascade (ce qui arrive fréquemment, cf. les résultats du tableau 4.2), nous allons intervenir en modifiant le critère d'arrêt de la fonction G_i. Nous voulons éviter que l'algorithme Adaboost arrive au nombre maximal d'itérations T_{max} sans atteindre le FA_{max}, ce que nous amène à stopper l'apprentissage de la cascade.

L'apprentissage de la cascade contrôlée arrête les itérations d'un classifieur fort G_i, si le nombre de fonctions faibles ajoutées à G_i corresponds à un maximum défini pour l'étage i. Le classifieur G_i est alors conservé en l'état, puis nous passons à l'apprentissage de la fonction G_{i+1} du prochain étage.

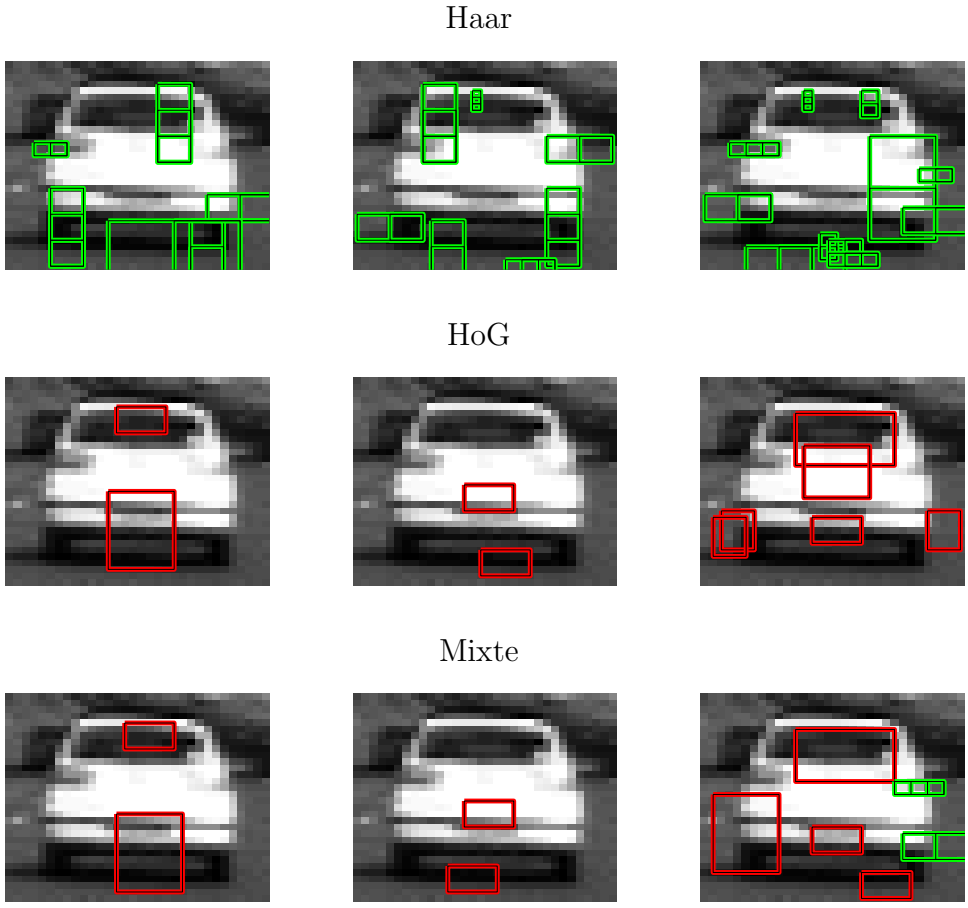


FIGURE 3.12 – Descripteurs choisis par Adaboost pour les trois premiers étages.

Pour fixer le nombre de descripteurs à chaque étage de la cascade, nous pouvons utiliser une loi croissante à notre convenance : linéaire, quadratique, exponentielle, etc. Pour obtenir ces lois, il suffit de fixer quelques points : un nombre très réduit de fonctions faibles dans les premiers étages et la quantité maximale pour le dernier.

Au cours de nos tests, nous avons utilisé la loi exponentielle. Elle nous permet d'obtenir une architecture selon laquelle, les premières fonctions fortes G_i de la cascade sont composés d'un faible nombre des g_i, pour, dans les derniers étages, ce nombre accroît considérablement pour éliminer les exemples négatifs plus difficiles.

Même s'il n'y a pas de convergence, cette méthodologie ne trahit pas le concept de la cascade. Bien que que le taux FA_{max} ne soit pas atteint dans les derniers étages, ils arrivent toujours à éliminer de fausses alarmes tout en conservant les exemples positifs

3.5 Bilan

Nous avons proposé dans ce chapitre deux espaces de représentation, les descripteurs de Haar et les histogrammes de gradient orienté, pour la détection de véhicules en uti-

lisant une approche similaire à celle de Viola & Jones. Les premiers sont associés à des classifieurs *faibles* de type discriminants et les seconds à des classifieurs de type génératifs. Un troisième détecteur est obtenu à partir dŠune fusion de ces deux espaces.

Nous allons étudié, dans le chapitre suivant, le comportement de différentes architectures : le détecteur simple et le détecteur en cascade. Nous verrons aussi que le détecteur réalisant la fusion des deux descripteurs combine les avantages des précédents détecteurs de Haar et de HoG : un taux de détections correctes élevé et un faible nombre de fausses alarmes.

Cette étude démontrera que les méthodes de dopages (*boosting*) sélectionnent automatiquement les classifieurs génératifs en premier, puis les discriminants. Auparavant, ceci était fait, comme nous l'avons dit, de façon intuitive en combinant séquentiellement les deux types de classifieurs [53, 120].

---

[53] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminative models for object category detection. In *IEEE International Conference on Computer Vision*, volume 2, pages 1363–1370, 2005.

[120] L. Prevost, L. Oudot, A. Moises, C. Michel-Sendis, and M. Milgram. Hybrid generative/discriminative classifier for unconstrained character recognition. *Pattern Recognition Letters*, 26(12) :1840–1848, 2005.

Chapitre 4

Évaluation du détecteur

Dans ce chapitre, nous décrivons le contenu des bases d'images utilisées pour l'apprentissage et les tests. Puis, nous détaillons la méthode d'évaluation des détecteurs. Nous analysons ensuite les résultats obtenus, pour enfin appliquer les détecteurs sur des scénarios plus difficiles.

4.1 Bases de données

Notre base de données est constituée d'images de route prises à partir d'une caméra embarquée installée à la hauteur du rétroviseur de la voiture.

FIGURE 4.1 – Exemples d'images de la base des données.

La caméra a une résolution de 640*480 pixels en niveaux de gris et un champ horizontal de 40 à 50 dégréés. Elle est placée à 1,25 mètres de hauteur, en retrait de 1,80 mètres par rapport à l'avant du capot moteur, qui est, lui même à une hauteur de 90 centimètres.

Les images de type JPG avec une qualité 100 percent, ont été prises à différents

moments de la journée et dans plusieurs environnements : routes nationales, autoroutes, périphériques, et en zones urbaines. Pour la plupart d'acquisitions, il a été défini une ROI pour le calcul automatique du gain (CAG) à une fenêtre de 24 à 616 à l'horizontale et 220 à 240 pour la verticale en se concentrant sur la route.

La base complète est constituée de plus de 1700 images où nous trouvons, principalement, la vue arrière d'un ou plusieurs véhicules, comptant au total 2600 voitures de tourisme.

4.1.1 Étiquetage des exemples positifs

L'étiquetage des images des véhicules est fait en deux temps. Dans un premier temps, nous encadrons chaque image d'un véhicule par un rectangle qui contient l'image complète de la voiture. Ensuite nous marquons six points selon le critère suivant :

- trois paires de points : une paire supérieure, une paire inférieure et une paire au milieu de la voiture (voir figure 4.2).
- les paires des points sont choisis sur des points symétriques sur la voiture.
- le paire inférieure est choisie de façon à marquer le point de contact entre les roues et la route.

Ces points servent à obtenir la vignette canonique de chaque voiture. Les points $(1, 2)$ donnent les limites supérieures de la voiture, les points $(4, 5)$ les limites inférieures et les points $(3, 6)$ permettent de calculer la position horizontale du véhicule. La vignette est finalement obtenue à partir d'un rectangle 20 % plus grand que les limites fixées par les points.

FIGURE 4.2 – Étiquetage d'un véhicule et la vignette canonique obtenue.

A partir d'une vignette de 32x32 pixels, notre plus petite échelle, nous allons ré-échantillonner les vignettes avec les résolutions multiples de 32, les plus proches de celles de l'image originale. Nous avons constaté que ré-échantillonner toutes les images à une taille 32x32 pixels nous fait perdre de l'information.

4.1.2 Constitution des bases d'images

Nous considérons les quatre bases suivantes :

⋆ **Base Positive** : la base positive est composée de 1521 vignettes. Cette quantité est doublée en synthétisant une image miroir par symétrie axiale. Du total des 3042 vignettes, deux tiers sont utilisés pour la Base d'Apprentissage Positive et le reste pour la Base de Validation Positive. Cette dernière nous sert à régler le seuil S du classifieur *fort* (voir équation 2.2). Des exemples de la base positive sont présentés dans la figure 4.3.

⋆ **Base de Test** : composée de 558 images de scènes réelles, différentes de celles d'apprentissage, contenant 1106 véhicules.

⋆ **Base Négative** : les exemples qui la composent sont tirés aléatoirement dans un ensemble de plus de 13000 images quelconques (ne contenant pas de véhicules).

⋆ **Base Négative Routière** : Nous cherchons des vignettes ne contenant pas de véhicules dans les images routières de la base de véhicules, en éliminant les rectangles correspondants aux voitures.

Base	**Images**	**Véhicules**
Positive	1200	3042
Test	558	1106
Négative	13000	-

(a) (b) (c)

FIGURE 4.3 – Exemples positifs et négatifs utilisés pour l'apprentissage.
(a) exemples positifs, (b) exemples négatifs utilisés dans les premiers étages, (c) exemples négatifs utilisés dans les derniers étages.

La première colonne de la figure 4.3 montre les exemples positifs de véhicules. Les autres colonnes montrent des exemples négatifs (non-véhicules) trouvés pour l'algorithme dans la base des images. Des exemples simples, utilisés pour l'apprentissage des premiers étages, sont dans la deuxième colonne. Dans la dernière colonne nous pouvons voir des exemples utilisés dans les derniers étages, qui sont plus proches à l'apparence d'un véhicule.

Étant donnée que le plus petit véhicule utilisé pour l'apprentissage correspond à une vignette de 32x32 pixels, nous considérons cette taille comme la taille minimale de détection de l'objet dans l'image. Cette taille correspond à une voiture située à plus de 80 mètres du véhicule porteur, pour ce capteur.

4.2 Détection par fenêtres glissantes

La détection des véhicules dans une image sera effectuée en utilisant une recherche par fenêtre glissante.

Cette stratégie consiste à positionner une fenêtre, à l'origine (x=0, y=0) de la matrice image. Nous extrayons la vignette (partie de l'image encadrée par la fenêtre), et appliquons le détecteur de véhicules. Si la fenêtre est validé (le détecteur a répondu positivement), nous conservons la position de celle-ci.

Ensuite, nous déplaçons la fenêtre de la position initiale dans le sens des x dŠun certain nombre de pixels. Nous extrayons à nouveau la vignette correspondante afin de faire la classification. Nous répétons cette opération et balayons la fenêtre sur toute la largeur de l'image. Puis, nous revenons à la position initiale et déplaçons la fenêtre dans le sens des y dŠun certain nombre de pixels et reprenons les tests. L'image est ainsi évaluée dans toutes les positions données par le déplacement de la fenêtre glissante.

Afin de faire une recherche des objets de tailles plus grandes, nous appliquons un facteur multiplicatif à la taille de la fenêtre originale, que nous appelons facteur d'échelle et qui doit être supérieur à un. Nous appelons les différentes tailles de la fenêtre glissante : échelles de recherche. Nous obtenons ainsi une nouvelle fenêtre plus grande que nous utilisons pour balayer l'image comme décrit précédemment. Le pas de glissement dans le sens des x et y va dépendre de la taille de la fenêtre. Il n'est pas nécessaire d'avoir un pas faible pour des fenêtres de grande taille étant donné que la variation dans la valeur de descripteurs (filtres rectangulaires ou histogrammes de gradient orienté) est très faible.

Notre fenêtre glissante originale est de 32x32 pixels et représente notre plus petite échelle de recherche. Le pas de glissement pour cette échelle est de 2 pixels. La plus grande échelle de recherche est de 224x224 pixels avec un pas de glissement de 7 pixels. Les valeurs intermédiaires que peuvent prendre les échelles de recherche dépendent du facteur d'échelle.

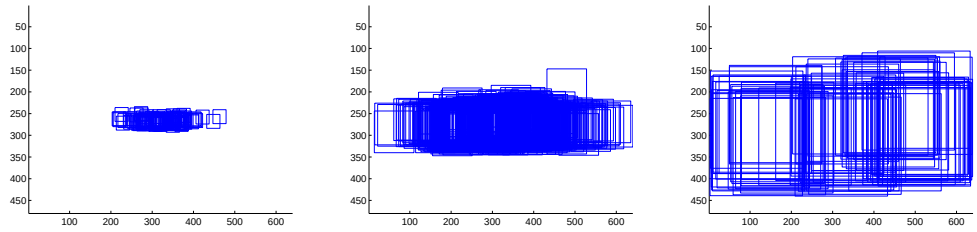
4.3 Restriction de la zone de recherche

Nous allons restreindre la zone de recherche de la fenêtre glissante à travers une analyse statistique des positions des voitures sur la route. Nous appliquons ainsi implicitement des contraintes de perspective similaires à celles de Broggi [17], à partir de la taille de la fenêtre

[17] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino Lo Bianco, and A. Piazzi. Visual perception of obstacles and vehicles for platooning. *IEEE Transactions on Intelligent Transportation Systems*,

de recherche. Il existe des zones dans l'image, où est impossible de trouver un véhicule d'une certaine taille.

Pour illustrer cette affirmation, nous superposons dans une matrice, tous les rectangles qui définissent un véhicule de la Base Positive d'une taille de 32x32, 96x96 et 224x224 pixels.



32x32 96x96 224x224

FIGURE 4.4 – Positions des véhicules dans la base positive par échelles.

La figure 4.5 montre une statistique des positions occupées par les véhicules dans la matrice image. L'axe de graduation de la figure représente le taux d'occupation de tous les rectangles englobant de véhicules dans la base d'images (de 0 à 40 %). Cette statistique nous permet d'obtenir une distribution estimée des positions d'un véhicule dans une image routière obtenue par notre système d'acquisition.

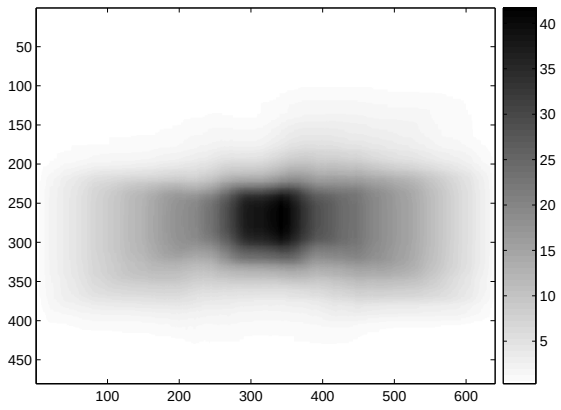


FIGURE 4.5 – Statistique sur les positions occupées par les véhicules dans les images routières de la base.

Nous pouvons donc restreindre la zone de recherche, en fixant des limites verticales pour chacune des échelles étudiées. Les rectangles de la figure 4.4, ainsi que les rectangles de recherche dans l'image de test, sont définis comme $r = (x, y, h, w)$, à partir du coin supérieur gauche et la longueur et largeur (elles sont égales, nous travaillons avec des carrés). L'analyse de la position verticale des rectangles de taille 32x32 de la figure 4.4 donne les valeurs suivantes :

1(3) :164–176, 2000.

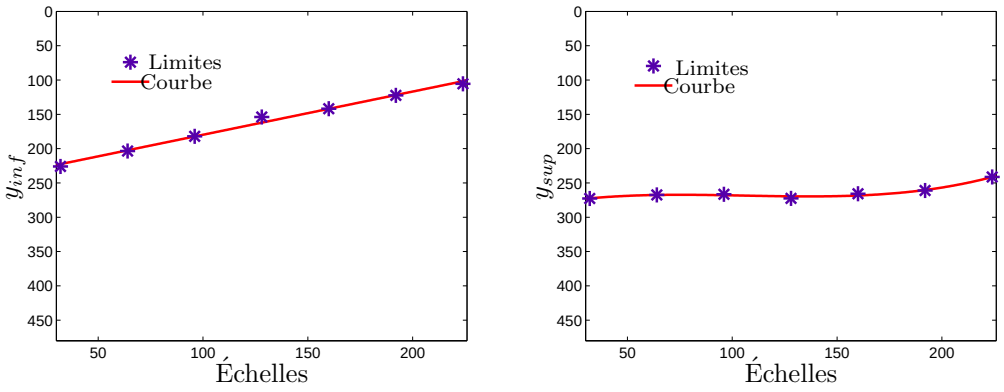
- $y_{min} = 234$
- $y_{max} = 259$
- moyenne, $\bar{y} = 249{,}3$
- écart type, $\sigma = 5{,}83$

Nous pouvons définir les limites verticales de recherche pour l'échelle 32x32 à partir des valeurs précédentes :

- $y_{inf}^{32} = \bar{y} - 4\sigma = 226$
- $y_{sup}^{32} = \bar{y} + 4\sigma = 272.6$

Nous réalisons la même analyse pour les échelles 64, 96, 128, 160, 192 et 224 afin de trouver les limites verticales de chacune.

Pendant les tests, nous allons utiliser des facteurs d'échelles qui ne sont pas forcement multiples de 32 : 1.25, 1.5, etc. Pour obtenir des limites inférieures et supérieures pour ces cas, nous allons déterminer une loi permettant d'établir ces valeurs en fonction de la taille du rectangle. Nous plaçons les valeurs des limites inférieures et supérieures des échelles et approximons les distributions par une droite dans le premier cas et une fonction polynomiale quadratique pour le deuxième.



(a) (b)

FIGURE 4.6 – Lois polynomiales approximant les distributions des valeurs. (a) limites inférieures et (b) limites supérieures

Nous obtenons alors pour les limites inférieures :

$$f_{inf}(x) = p_1 * x + p_2$$

où x est la taille du rectangle, $p_1 = -0.62$ et $p_2 = 242.8$.

et pour les limites supérieures :

$$f_{sup}(x) = p_1 * x^3 + p_2 * x^2 + p_3 * x + p_4$$

où $p_1 = -2.05 * 10^{-5}$, $p_2 = 0.0065$, $p_3 = -0.629$ et $p_4 = 286.6$.

Ces valeurs dépendent bien entendu de la position de la caméra dans le véhicule équipé.

4.4 Critères d'évaluation

Les critères utilisés dans la littérature pour évaluer les algorithmes de détection sont les suivants :

- le taux de détections correctes (DC),
- le taux ou le nombre de fausses alarmes (FA),
- la courbe Caractéristique Opérationnelle de Réception (*Receiver Operating Characteristic*, ROC), qui donne le DC en fonction de FA.

4.4.1 Taux de détection et taux de fausses alarmes

Une fenêtre validée par le classifieurs est considérée comme DC si elle remplit un critère de coïncidence avec la boîte englobant la vérité terrain. Les deux facteurs utilisés sont une différence de taille et une différence de position, avec valeurs 1.5 et 0.3 respectivement (prises à partir de la fonction de la librairie OpenCV qui estime la superposition des rectangles). Dans le cas contraire, elle sera jugée comme une FA. Ces deux paramètres nous permettent d'estimer le comportement d'un classifieur.

Le DC correspond alors, au pourcentage de véhicules correctement détectés sur l'ensemble des véhicules présents dans les images de la base de test. Le critère de coïncidence peut valider plus d'une fenêtre pour le même exemple positif. Dans ce cas, une seule sera validée et les autres ne sont prises en compte (ne sont pas considérées comme faux négatifs) :

$$DC = \frac{Total\ Positifs}{Total\ Vehicules}$$

Le taux FA est calculé à partir du nombre total de faux positifs divisé par le nombre total de fenêtres négatives dans l'image.

$$FA = \frac{Total\ Faux\ Positifs}{Total\ Fenetres\ Negatives}$$

Pour évaluer la performance de la méthode d'apprentissage, nous allons effectuer une procédure de cross-validation. Nous entraînons trois réalisations de chaque détecteur sur des bases d'apprentissage différentes : la distribution de la base positive en base d'apprentissage et base de validation se fait aléatoirement, tout comme le choix des exemples négatifs des premiers étages. Les taux DC et FA affichés dans les tableaux de résultats correspondent à la moyenne des DCs et FAs des trois réalisations.

4.4.2 Courbe ROC

Une courbe ROC est paramétrée par la valeur du seuil responsable de la classification. Dans le cas des classifieurs *forts*, ce seuil est celui de l'équation 2.2 qui valide, ou non, une fenêtre. La variation de sa valeur modifie le DC et le FA obtenus sur la base de test, et permet de tracer la courbe ROC. Nous faisons varier le seuil du maximum possible, où

nous n'avons aucune fausse alarme et aucune détection, jusqu'à une valeur égale à zéro, où nous avons validé toutes les fenêtres comme positives. Cette courbe permet ainsi de mesurer la performance d'un classifieur.

4.4.3 Courbe ROC pour une cascade

Nous utilisons la méthode proposé par Viola et Jones [142] pour obtenir la courbe ROC de la cascade.

Pour le cas d'un détecteur en cascade, il existe plusieurs seuils à varier (un pour chaque classifieur *fort*). D'abord, le seuil du dernière étage est varié de - infini jusqu'à + infini. Une valeur de seuil égale à + infini obtienne, évidement, un taux de DC et de FA égale à zéro. De l'autre coté, varier le seuil jusqu'à - infini c'est comme si nous avions enlevé cette étage. Pour continuer à désigner la courbe ROC, pour des valeurs de DC et FA plus élevés, nous continuons à varier le seuil de l'étage suivant, de sa valeur originale jusqu'à - infini. Nous repentons cette opération pour tous les étages pour obtenir la courbe ROC complète.

Cette courbe est doublement paramétré, à la fois par le nombre d'étages considérés et par la valeur du seuil appliqué au classifieur fort du dernier étage. Un point de fonctionnement, qui peut être choisi selon un taux de détections correctes ou un taux de fausses alarmes désiré, indique aussi le numéro d'étage auquel il appartient.

4.5 Analyse du comportement et des performances

Dans cette section, nous analysons les résultats obtenus pour les trois types (Haar, HoG et Mixte) et les trois implémentations (simple, cascade et cascade contrôlée) des détecteurs décrits précédemment. Nous commençons avec une vignette de 32x32 pixels jusqu'à une taille de 224x224 pixels avec un facteur d'échelle de $s = 1.25$. Au total, 42798 vignettes sont évaluées dans chaque image. Le pas de déplacement des vignettes dans l'image dépend aussi du facteur d'échelle. Enfin, bien qu'il ne soit pas préparé pour le temps réel, le temps de traitement moyen par image est évalué sur un PC cadencé à $2.2GHz$ et est donné pour guider le lecteur dans la comparaison des différents types de détecteurs.

4.5.1 Détecteur simple

Pour chacun des détecteurs, nous faisons varier la valeur du nombre maximum de fonctions de classification faibles ($T = 50$, 100 et 150 fonctions faibles). Nous fixons le nombre d'exemples négatifs pour l'apprentissage à 5000. La figure 4.7 présente les courbes

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

ROC pour chaque détecteur simple (Haar, HoG et Mixte) obtenues en faisant varier le seuil S de chaque classifieur.

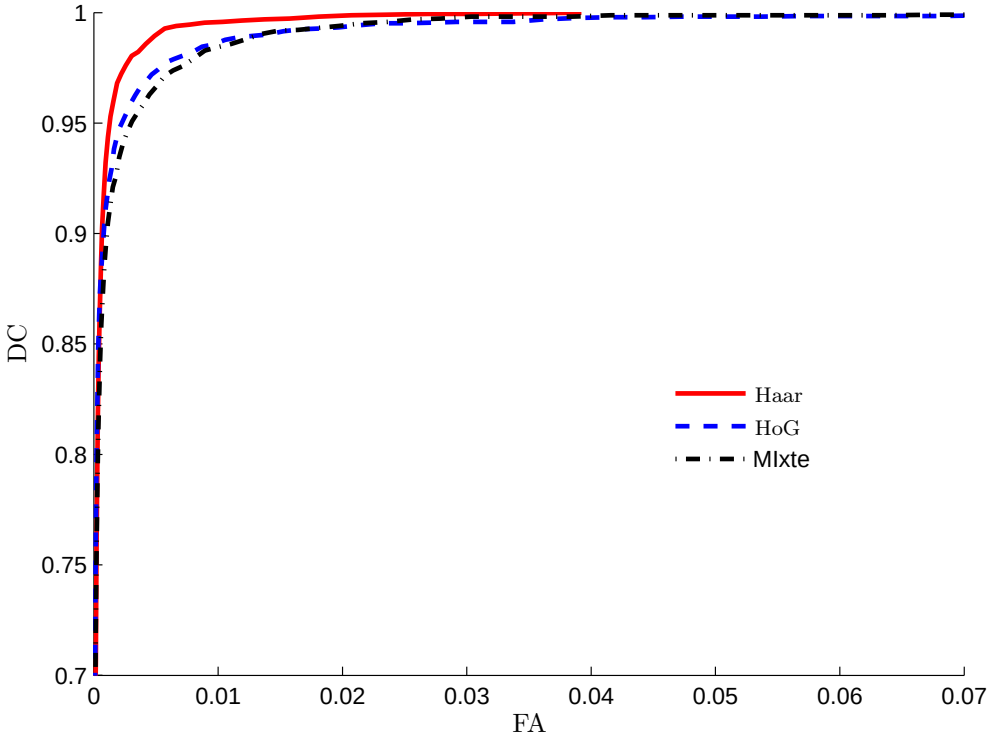


FIGURE 4.7 – Courbes ROC pour les trois détecteurs simples.

Les courbes ROC de la figure 4.7 montrent que le détecteur de Haar est plus performant que les deux autres. Le détecteur Mixte possède une performance similaire au détecteur HoG pour des valeur faibles de DC et à celui de Haar pour des valeurs de DC importantes.

Type	T	DC(%)	FA	Temps(sec)
Haar	50	100	0.0670	0.90
Haar	100	99.8	0.0332	1.73
Haar	150	99.7	0.0148	2.52
HoG	50	99.8	0.0571	1.19
HoG	100	99.7	0.0432	2.15
HoG	150	99.7	0.0313	3.22
Mixte	50	99.8	0.0399	0.87
Mixte	100	99.7	0.0190	1.68
Mixte	150	99.5	0.0209	2.47

TABLE 4.1 – Comportement des détecteurs simples

Le tableau 4.1 montre le comportement des détecteurs simples directement sorties de l'apprentissage avec AdaBoost. La valeur de seuil a été fixée de sorte d'obtenir un taux de détections correctes sur la Base de Validation Positive supérieure à 99.5%.

Pour le détecteur de Haar, l'augmentation des fonctions faibles lui permet d'éliminer jusqu'à cinq fois le nombre de fausses alarmes. En même temps, le taux de détections observe une légère baisse qui montre que le détecteur raffine ces frontières et dévient de plus en plus discriminant.

Le détecteur de HoG montre un comportement différent au précédent. D'abord, pour $T = 50$, il obtient un taux de FA plus faible que pour Haar. Cependant, le fait d'augmenter le nombre des fonctions faibles ne facilite pas l'élimination des FA, ce qui maintient aussi le taux de DC stable.

Le détecteur Mixte a un comportement intermédiaire entre les deux, en conservant un taux de détections important tout en éliminant un grand nombre de FA. La proportion de descripteurs HoG pour $T = 50$ est 52 %, pour $T = 100$ est de 42 % et pour $T = 150$ de 45 %.

Nous observons dans le tableau que le temps de calcul par image augmente naturellement avec la quantité de fonctions faibles. La différence entre le détecteur de Haar et celui de HoG réside dans le calcul de chaque descripteur. Pour Haar, nous avons au maximum huit références à l'image intégrale et une étape préalable de normalisation des valeurs à partir de l'écart type des niveaux de gris de la vignette. Un descripteur de HoG a besoin de 16 références à l'histogramme intégrale et une normalisation de ces valeurs. De plus, la fonction de discrimination pour le premier est basée sur un simple seuillage, et pour le deuxième, nous devons calculer la distance de Bhattacharyya.

Le détecteur simple possède deux inconvénients principales : il obtient un taux élevé de DC au détriment d'un grand nombre de fausses alarmes, et l'utilisation d'un grand nombre de descripteurs dans un seul étage devient irréaliste du point de vue d'une application en temps réel. L'architecture en cascade répond à la première question grâce au bootstraping. Les exemples négatifs qui seront choisis au long de l'apprentissage du détecteur sont de plus en plus proches des frontières, ce qui oblige à les raffiner pour les éliminer. Le taux de FA pour la cascade sera donc beaucoup plus faible. Pour la deuxième question, nous avons vu dans la section 2.4, que des classifieurs *forts* avec peu de fonctions faibles vont éliminer la plupart des régions négatives de l'image, au lieu d'être obligé à évaluer une centaine de descripteurs pour toutes les régions à tester.

4.5.2 Détecteur en Cascade

Le tableau 4.2 précise l'architecture de chacun des détecteurs en cascade ainsi que leurs comportements. Les critères de performance pour la fonction G_i sont les suivants : le taux minimum de détections correctes $DC_{min} = 0.995$ et le taux maximal de fausses alarmes acceptable, $FA_{max} = 0.40$ (nous avons effectué une analyse de sensibilité à ces paramètres dans l'annexe A).

Nous avons utilisé deux quantités d'exemples négatifs d'apprentissage pour étudier la sensibilité de l'algorithme à ce paramètre.

Type	#Nég	#étages	#Desc	DC(%)	FA	Temps	Arrêt
Haar	2000	17	720	88.87	0.00014	0.59	Non conv
Haar	4000	15	542	93.43	0.00015	0.58	Non conv
HoG	2000	7	111	99.57	0,01154	0.34	Non conv
HoG	4000	7	170	99.40	0,00718	0.39	Non conv
Mixte	2000	15	410	95.4	0.00030	0.28	Non conv
Mixte	4000	15	502	93.5	0.00012	0.26	Nég. non atteint

TABLE 4.2 – Comportement des détecteurs en Cascade

Le détecteur de Haar, qui a été entraîné avec 2000 exemples négatifs par étage, interrompe l'apprentissage dans le 18ème étage, n'arrivant pas à la convergence. Nous éliminons ce dernier et conservons l'architecture de la cascade jusqu'au 17ème étage. Le taux de DC est plutôt faible, 89 %, mais nous avons éliminé cent fois plus de FA par rapport au détecteur simple. Lorsque nous augmentons le nombre de vignettes négatives dans l'apprentissage, les processus ne convergent plus à un nombre d'étages plus faible. Dans la figure 4.8, nous donnons le nombre de descripteurs de chaque détecteur, par étage et selon le nombre de négatifs d'apprentissage.

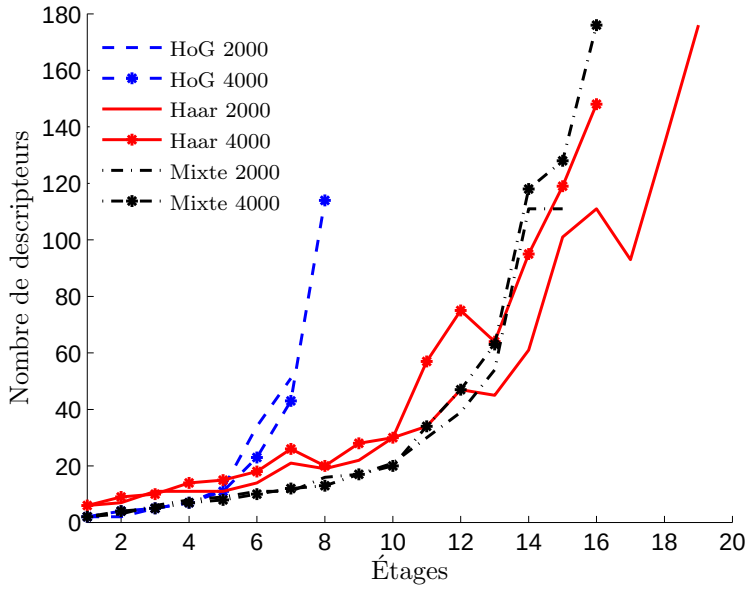


FIGURE 4.8 – Nombre de descripteurs par étage pour les trois détecteurs. Quantité de descripteurs sélectionnées à chaque étage pour les détecteurs à 2000 et à 4000 négatifs.

La différence dans le taux de DC est due au fait que le détecteur à 2000 négatifs a plus d'étages que celui à 4000 négatifs. Nous observons dans la figure 4.9 que les deux courbes ont la même pente. Cependant, le fait d'avoir arrêté l'apprentissage plus tôt donne au détecteur à 4000 négatifs un DC plus important.

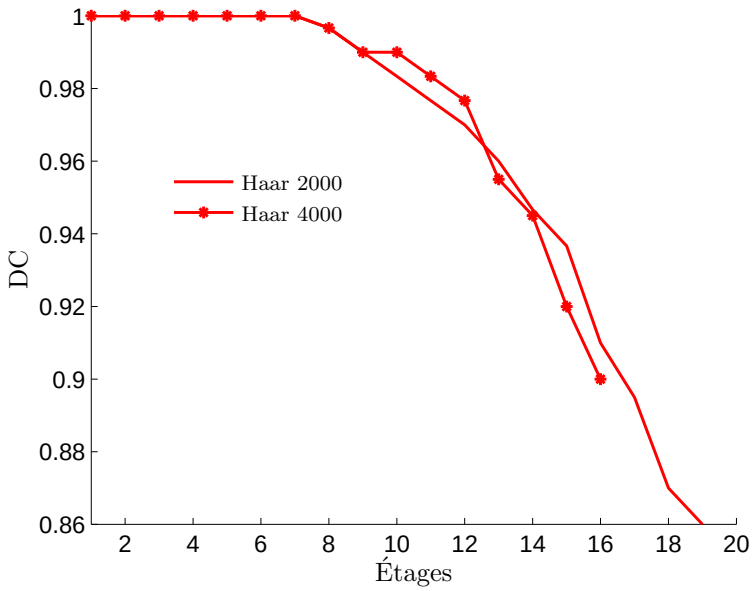


FIGURE 4.9 – Taux de détections correctes par étage pour le détecteur de Haar à 2000 et à 4000 négatifs.

Le tableau montre que le détecteur de HoG converge seulement dans les premiers étages. Dans la figure 4.8 nous constatons que le nombre de descripteurs de HoG augmente fortement, d'un étage à l'autre, avant la non convergence. C'est un comportement assez symptomatique des classifieurs génératifs : ils arrivent très bien à modéliser les échantillons positifs (d'où les taux de détections correctes élevés pour les détecteurs de HoG dans le tableau 4.2), mais ils ont besoin de complexifier fortement le modèle pour dessiner une frontière nette avec les échantillons négatifs proches aux échantillons positifs. Le grand nombre de FA de ces détecteurs est due au faible nombre des étages.

Il faut remarquer aussi que les fonctions de classication forte de HoG convergent avec un nombre nettement plus faible de descripteurs dans les premiers étages que ceux de Haar. En effet, il faut peu de descripteurs pour éliminer les vignettes négatives relativement éloignées du modèle. Inversement, le détecteur de Haar nécessite un plus grand nombre de descripteurs pour estimer correctement une frontière entre les classes.

A nouveau, nous pouvons constater que la combinaison des deux descripteurs dans le détecteur Mixte permet d'améliorer le comportement : il utilise les descripteurs de HoG pour éliminer les échantillons négatifs éloignés du modèle et ceux de Haar pour ceux qui sont proches de la frontière.

Le contrôle du processus d'apprentissage sur le nombre de descripteurs limité à chaque étage, proposé dans la section 3.4.3, nous permet d'obtenir le meme nombre d'étages pour chaque détecteur et donc de réaliser une comparaison plus pertinente entre les détecteurs.

4.5.3 Détecteur en cascade contrôlée

Dans cette section, nous allons réaliser une analyse comparative des comportements des différents détecteurs mais aussi de leurs performances.

Analyse comparative des comportements

La figure 4.10 présente la quantité de descripteurs par étage des trois détecteurs entraînés en utilisant 2000 exemples négatifs pour l'apprentissage. Les points qui se situent sous la loi exponentielle correspondent aux étages qui ont convergé avant d'arriver à la limite maximale du nombre de descripteurs.

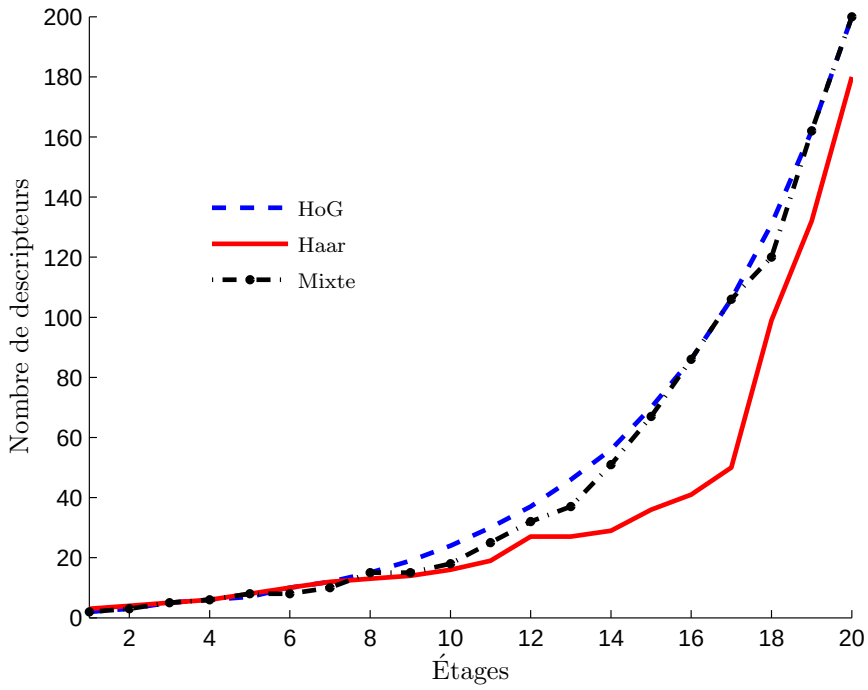


FIGURE 4.10 – Quantité de descripteurs par étage pour les trois détecteurs.

Le détecteur de type HoG converge avant d'arriver au maximum dans les premiers étages avec une très faible quantité de descripteurs. Par contre, dans les étages supérieurs, il sature, ne pouvant plus arriver à la convergence.

Le détecteur de Haar expose un comportement inverse : il n'arrive pas à la convergence dans les premiers étages, mais à partir du 7ème étage, l'apprentissage obtient un nombre de descripteurs plus faible que le maximum donné par la loi exponentielle.

Le détecteur Mixte a un comportement intermédiaire entre les deux précédents. Il compte avec un faible nombre de descripteurs dans les premiers étages (similairement au détecteur de HoG). Ensuite, au long de la cascade, il converge avant d'atteindre le nombre maximum de descripteurs de la loi, avec cependant un nombre plus important que le détecteur de Haar.

La figure 4.11, illustre l'évolution de la proportion de descripteurs de HoG par rapport au nombre total de descripteurs sélectionnés à chaque étage. Nous observons que, initialement, ce sont les descripteurs de HoG qui ont été choisis par Adaboost comme étant les plus discriminants. Dans les derniers étages, ce sont plutôt les descripteurs de Haar qui composent les fonctions *fortes*.

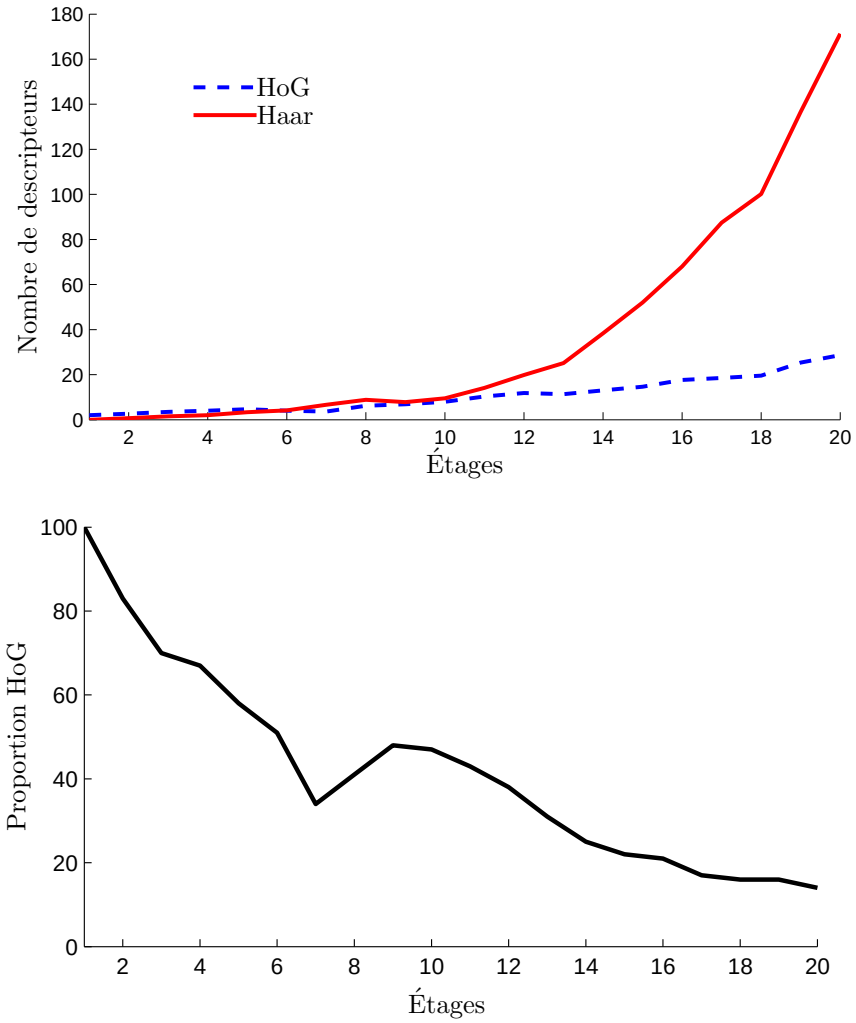


FIGURE 4.11 – Nombre et proportion de descripteurs de HoG à chaque étage du détecteur Mixte.

Ceci confirme les remarques énoncées précédemment pour les détecteurs en cascade sans contrôle à propos du comportement symptomatique des approches générative et discriminante. Les détecteurs de type génératif ont besoin d'un faible nombre de fonctions faibles pour tracer les frontières de classification. Ils arrivent ainsi à éliminer un grand nombre des régions dès les premiers étages. Par la suite, étant donné que le *bootstraping* ajoute des exemples négatifs d'apprentissage beaucoup plus proches au modèle du véhicule, Adaboost requière alors des descripteurs discriminants. Ces descripteurs sont capables de raffiner les frontières en éliminant les négatifs restants.

Nous pouvons constater ce comportement dans la figure 4.12. Les détecteurs de HoG

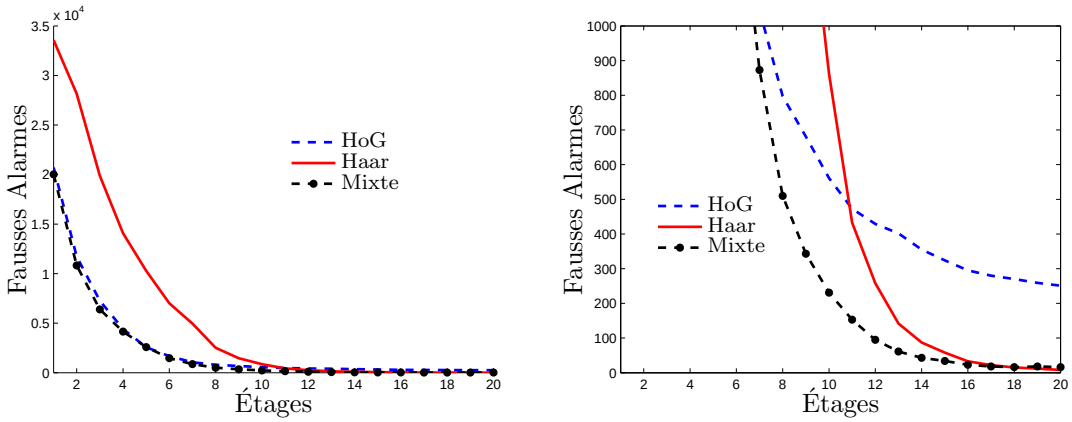


FIGURE 4.12 – Élimination des fausses alarmes au fur et à mesure des étages. Résultats pour les trois détecteurs à 2000 négatifs.

et Mixte, qui éliminent au départ plus de fausses alarmes que le détecteur de Haar, ont un comportement similaire jusqu'au sixième étage.

Ils arrivent plusieurs phénomènes intéressants à ce stade. D'abord, le détecteur Mixte compte avec 50 % de descripteurs de Haar et HoG. Il abandonne la pente de la courbe d'élimination de FA de HoG pour se mettre en parallèle avec la courbe de Haar (figure 4.12 (b)). De plus, dans la figure 4.13, nous observons que le détecteur de HoG commence à éliminer aussi des exemples positifs. Nous pouvons associer son niveau de complexité au nombre de FA.

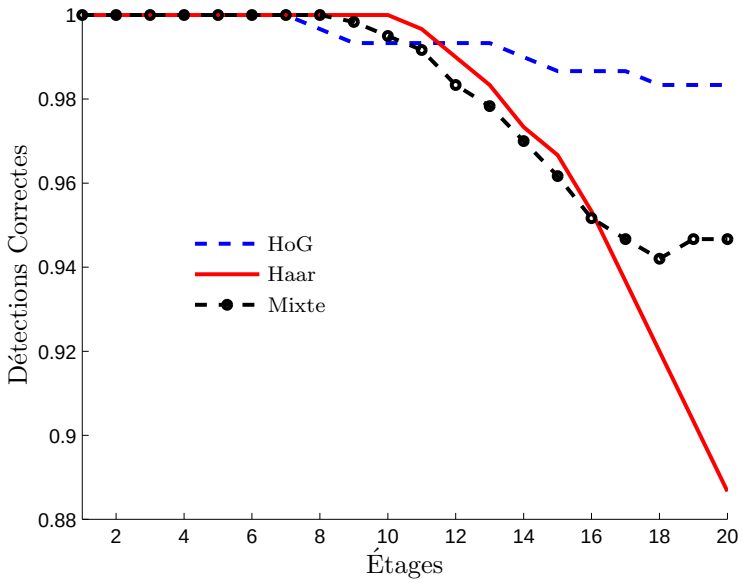


FIGURE 4.13 – Taux de détections correctes en fonction des étages de la cascade. Résultats pour les trois détecteurs à 2000 négatifs.

Quand le détecteur de Haar arrive à ce nombre de fausses alarmes, au dixième étage,

il commence aussi à éliminer des exemples positifs. Il rattrape, vers le sixième étage, le nombre de fausses alarmes du détecteur Mixte. Jusqu'à ce point, ils avaient une courbe de DC similaire, mais à partir de cela, le détecteur de Haar devient de plus en plus discriminant, en éliminant des fausses alarmes en même temps qu'il voit son taux de DC diminuer. Le détecteur Mixte, par contre, reprend un comportement comme celui de HoG et son taux de DC reste stable (la petite remontée dans les valeurs est due à que les DC ont été calculés comme la moyenne de trois détecteurs), ainsi que son nombre de FA.

Cela se traduit dans les résultats du tableau 4.5.3 qui montrent les comportements globaux des détecteurs en cascade contrôlée.

Type	# Neg	# Desc	DC (%)	FA	t (s)
Haar	2000	730	88.4	0.00024	0.64
Haar	4000	924	91.9	0.00039	0.64
HoG	2000	1025	98.4	0.01241	0.61
HoG	4000	1030	98.1	0.01026	0.60
Mixte	2000	773	93.4	0.00036	0.29
Mixte	4000	917	94.0	0.00042	0.31

TABLE 4.3 – Comportement des détecteurs en Cascade Controlée.

Le détecteur de Haar entraîné avec 2000 négatifs, est le plus discriminant du tableau. Il a un taux de FA très bas, au détriment d'un taux de DC aussi faible. L'augmentation des exemples négatifs d'apprentissage incrémente le nombre de descripteurs total de la cascade. Il améliore aussi sont taux de DC avec un nombre plus important de FA.

La cascade de HoG obtient un taux élevé de DC ainsi que des FA. Le fait d'augmenter le nombre des négatifs lui permet aussi d'éliminer un plus grand nombre de FA, mais, qui reste encore très important.

Finalement, le détecteur Mixte obtient un comportement supérieur aux deux autres, en particulier en terme du temps de réponse. Cela s'explique aisément en analysant la courbe 4.12 où le nombre des hypothèses rejetées par le détecteur Mixte est très important dès les premiers étages de la cascade. Son taux de DC est le plus élevé et celui de FA faible.

Analyse comparative des performances

Les courbes ROC doublement paramétrées (par le nombre d'étages considérés et la valeur du seuil au dernier étage) de la figure 4.14 montrent la performance des trois détecteurs. Nous observons que le détecteur Mixte obtient des performances supérieures aux deux autres.

Nous allons utiliser ces courbes ROC doublement paramétrées pour placer les trois types de détecteurs au même point de fonctionnement. Nous fixons le DC à 93.4%, valeur de DC pour le détecteur Mixte, et nous obtenons trois détecteurs en cascade à partir de la méthode développée dans la section 4.4. Le tableau 4.4 montre les résultats obtenus pour ces détecteurs.

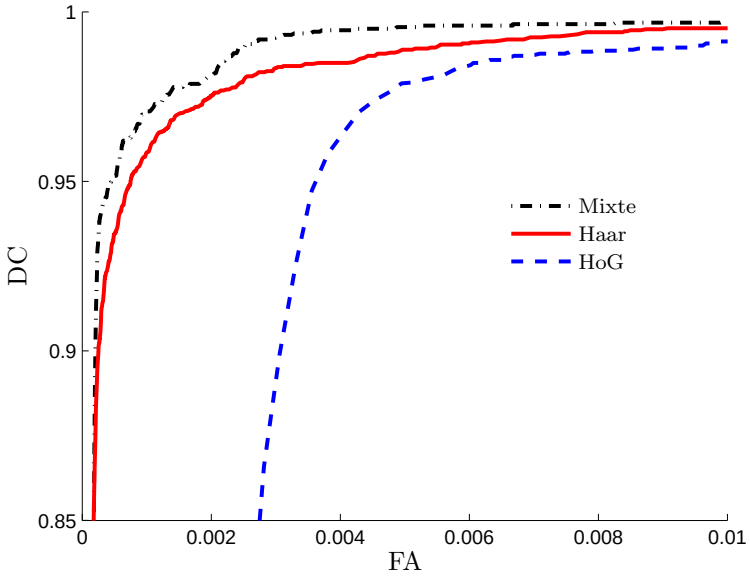


FIGURE 4.14 – Courbes ROC des détecteurs en cascade.

Type	#Nég	#étages	DC(%)	FA (%)	Temps (s)
Haar	2000	17	93.4	0.00049	0.60
HoG	2000	20	93.4	0,0035	0.61
Mixte	2000	17	93.4	0.00025	0.28

TABLE 4.4 – Performance des détecteurs pour un point de fonctionnement à 93.4 %.

L'ensemble de ces observations se reflètent très concrètement dans les images de scènes routières de la figure 4.15, où les carrés blancs signalent une détection (correcte ou fausse) obtenue. Nous observons que le détecteur de Haar ne détecte pas tous les véhicules mais ne produit que peu de fausses alarmes. En revanche, le détecteur de HoG réalise de nombreuses fausses alarmes, mais détecte quasiment tous les véhicules. Enfin, le détecteur Mixte réduit sensiblement le nombre de fausses alarmes tout en détectant les mêmes véhicules que celui de HoG.

Dans la section suivante, nous allons analyser les résultats des détecteurs appliqués sur des différents scénarios : urbains, suburbains, faible éclairage, soleil rasant, etc.

4.6 Différents Scénarios

Les résultats exposés dans la section précédente correspondent à la base de test que nous allons désigner nominale. Les images qui composent cette base ont été capturées avec un bon éclairage pendant des conditions climatiques favorables. Dans la première partie de cette section, nous allons classer ces images par rapport au scénario dans lequel elles ont été prises pour étudier le comportement des détecteurs.

FIGURE 4.15 – Résultats des trois détecteurs obtenus sur des images de scènes autoroutières.

Ensuite, nous allons utiliser une deuxième base de test composée d'images prises sous des situations adverses d'éclairage et conditions climatiques peu favorables. Nous appelons cette base, non-nominale.

4.6.1 Scénarios urbain et suburbain

Chaque image de la base de test est étiquetée à partir de trois scénarios :

- Scénario rural ou nationale, comptant avec 58 images enregistrées sur une route nationale.
- Scénario autoroute, composé de 314 images prises sur une autoroute ou sur le périphérique (sans bâtiments à vue).
- Scénario urbain, où nous trouvons 186 images d'une rue urbaine ou contenant des bâtiments.

Les détecteurs du tableau 4.4 sont utilisés pour réaliser une comparaison de la quantité des FA obtenues face aux différents scénarios de test. Dans la figure 4.16, nous avons placé le nombre de FA par détecteur et par scénario, en moyenne et en écart-type. La faible quantité de FA dans le scénario rural est expliquée par l'absence de structures artificielles comme des bâtiments, des panneaux, des ponts, etc. Ce type de structures contient des traits horizontaux ou des formes rectangulaires qui font confondre le détecteur. Nous voyons effectivement que le nombre moyen des FA augmente pour ces scénarios ainsi que

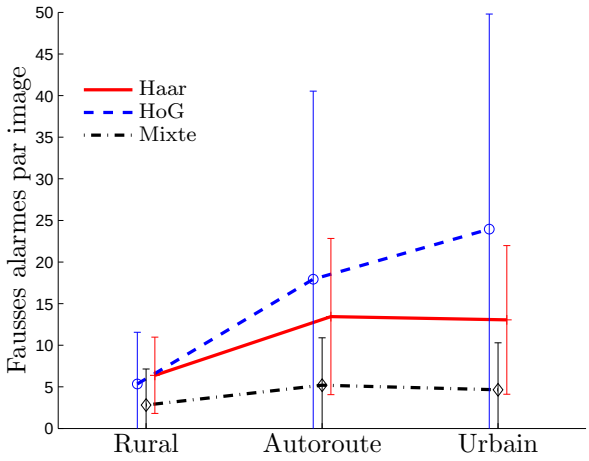


FIGURE 4.16 – Moyennes et écart-types des fausses alarmes par scénario.

l'écart -type, pour les trois détecteurs.

Nous notons qu'il existe une diminution des FA entre le scénario Autoroute et le scénario Urbain pour les détecteurs de Haar et Mixte. Par contre, pour le détecteur HoG, ce nombre augmente, ainsi que son écart type. En effet, ce détecteur n'arrive plus à éliminer les faux positifs dues aux structures artificielles. La figure 4.17 montre des exemples des trois détecteurs pour les différents scénarios.

4.6.2 Cas non-nominaux

Constitution de la base de données

La base non-nominale des images prises sous des conditions défavorables est classée en cinq scénarios différents :

1. Tunnel : images prises dans un tunnel.
2. Brouillard : images prises dans le brouillard ou sous une pluie fine.
3. Soleil : images avec du soleil rasant ou soleil de face.
4. Occlusions : images où un des véhicules est partiellement caché. Nous allons définir trois niveaux d'occlusion qui dépend du pourcentage du voiture occultée.
5. Ombres : les véhicules sont placés dans des zones ombragées.

Dans la figure 4.18, nous pouvons observer quelques exemples de chaque scénario.

La base non-nominale est composée de 391 images. Nous trouvons pour chaque scénario, le nombre d'exemples présenté dans le tableau 4.5.

Nous allons appliquer sur cette base le détecteur Mixte D_M à 2000 négatifs de la section précédente (tableau 4.5.3). Les résultats sur la base sont montrés dans le tableau 4.6 (avec un rappel des résultats sur la base nominale).

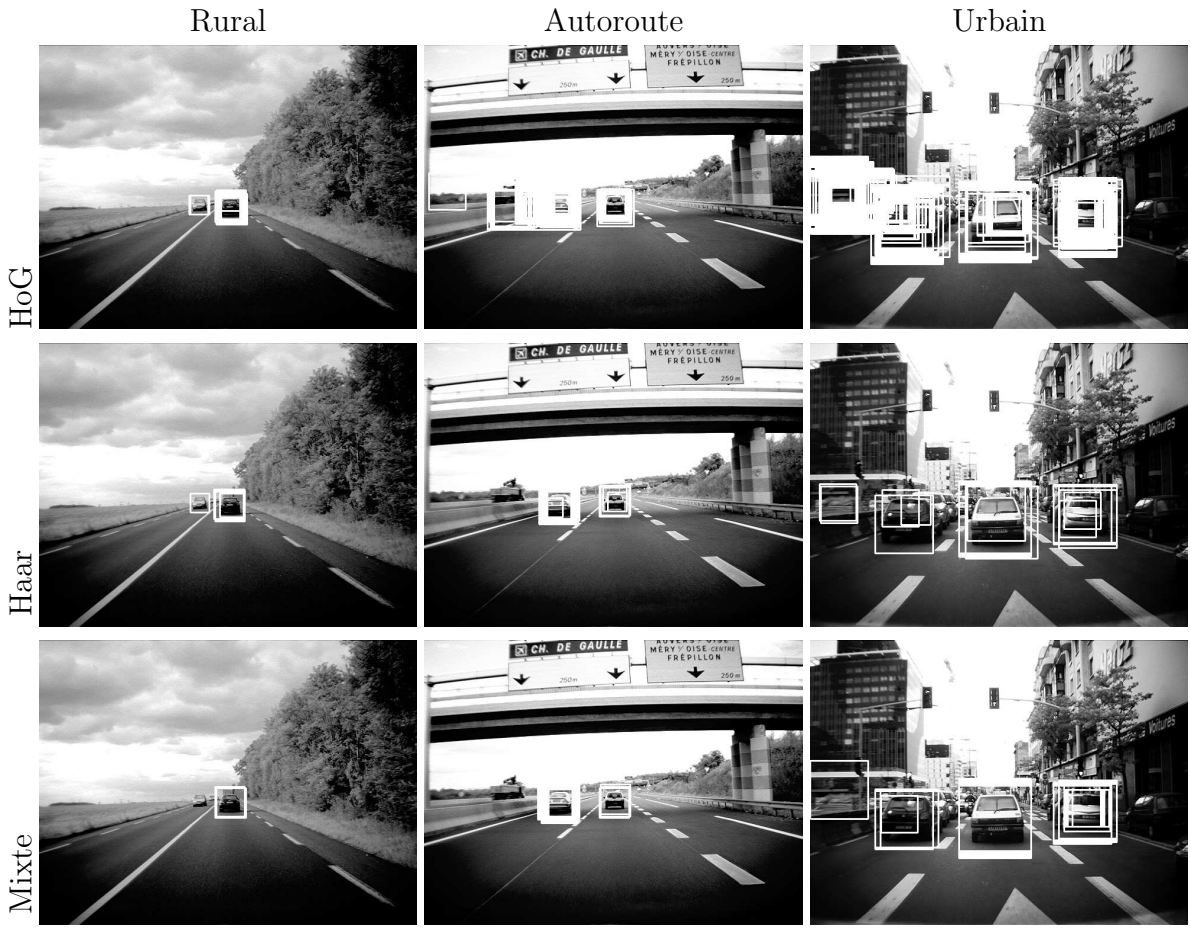


FIGURE 4.17 – Exemples des trois détecteurs sur différents scénarios.

FIGURE 4.18 – Exemples d'images de cas non-nominaux.

Les résultats indépendants de chaque scénario sont présentés dans l'histogramme de la figure 4.19. Le cas le plus difficile est celui du soleil avec seulement 45 % de détections. Nous pouvons observer dans les exemples de la figure 4.18, que ce type d'images change complètement l'apparence du véhicule, ainsi que de l'environnement due à la saturation

Scénario	Quantité exemples
Tunnel	33
Brouillard	88
Soleil	90
Occlusions	136
Ombres	44

TABLE 4.5 – Constitution de la base de test non nominale.

Cas	**DC**	**FA**	**FA moy**
Nominal	93.63	0.00016	4.3
Non Nominal	59.07	0.000137	5.87

TABLE 4.6 – Comportement du détecteur sur les deux bases de test.

en luminance.

Un autre résultat intéressant est que le comportement sous les tunnels est le même que dans le cas des ombres. D'après les images de la figure 4.18, nous voyons que ces deux cas sont très semblables. En effet, l'incidence physique sur l'image est la même. Nous n'avons pas eu relativement aux autres cas une grande perte de détections pour ces cas, étant donné que nous appliquons une étape de normalisation des images qui réduisent l'influence des ombres.

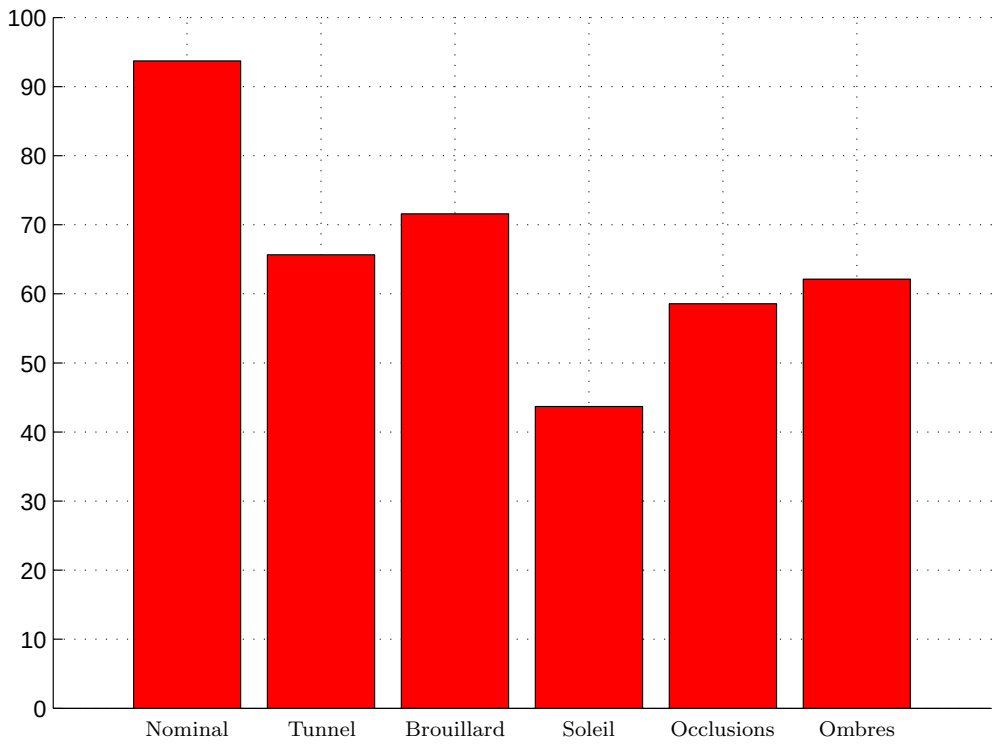


FIGURE 4.19 – Résultats sur les cas non-nominaux pour le détecteur appris sur les cas nominaux.

Apprentissage d'un détecteur avec des images non-nominales

Nous allons entraîner un nouveau détecteur Mixte en cascade contrôlée à 2000 négatifs, en utilisant une nouvelle base d'images de véhicules qui inclut des exemples de cas non-nominaux.

Nous disposons de 638 images de véhicules qui ont été prises dans des cas non nominaux. Nous doublons cette quantité tournant les images dans l'axe vertical. La nouvelle base, composée d'un total de 4318 exemples de véhicules, est donc utilisée pour obtenir le détecteur D_{Mnn}.

Nous appliquons ce nouveau détecteur dans les bases de test nominal et non-nominal et nous obtenons les résultats montrés dans le tableau 4.7.

Cas	DC	FA	FA moy
Nominal	98	0.00052	22.4
Non Nominal	78.1	0.00046	20

TABLE 4.7 – Comportement du détecteur appris sur tous les cas.

Pour comparer les deux types de détecteurs, nous fixons un point de fonctionnement dans le premier détecteur à 98 % de détections correctes dans la courbe ROC, c'est qui nous rend une nouvelle cascade avec les résultats suivants :

Cas	DC	FA	FA moy
Nominal	97.8	0.00085	36.4
Non Nominal	76.63	0.00075	32

TABLE 4.8 – Comportement du détecteur D_M avec un point de fonctionnement fixé à 98 %.

Nous apprécions que le taux de détections correctes obtenu est similaire, mais la quantité des fausses alarmes par image est un tiers plus grand que le détecteur entrainé avec tous les cas.

L'analyse détaillée des résultats sur chaque scénario est illustrée dans l'histogramme de la figure 4.20.

La courbe ROC de la figure 4.21 nous montre le comportement de chaque détecteur dans les deux cas différents. Nous observons que le détecteur D_{Mnn} a un meilleur comportement que D_M, que ce soit dans les cas nominaux ou dans les cas non-nominaux.

Enfin, nous pouvons conclure que le fait d'augmenter la base des positifs avec les cas non-nominaux nous permet d'améliorer la modélisation de la classe véhicule en obtenant un taux de détections importante, au même temps, nous éliminons un nombre plus grand de fausses alarmes.

La figure 4.22 présente des exemples de détections de deux détecteurs mis au même point de fonctionnement (taux de détections correctes égal à 95 %). Nous pouvons constater le nombre plus importante de fausses alarmes pour D_M que pour D_{Mnn}.

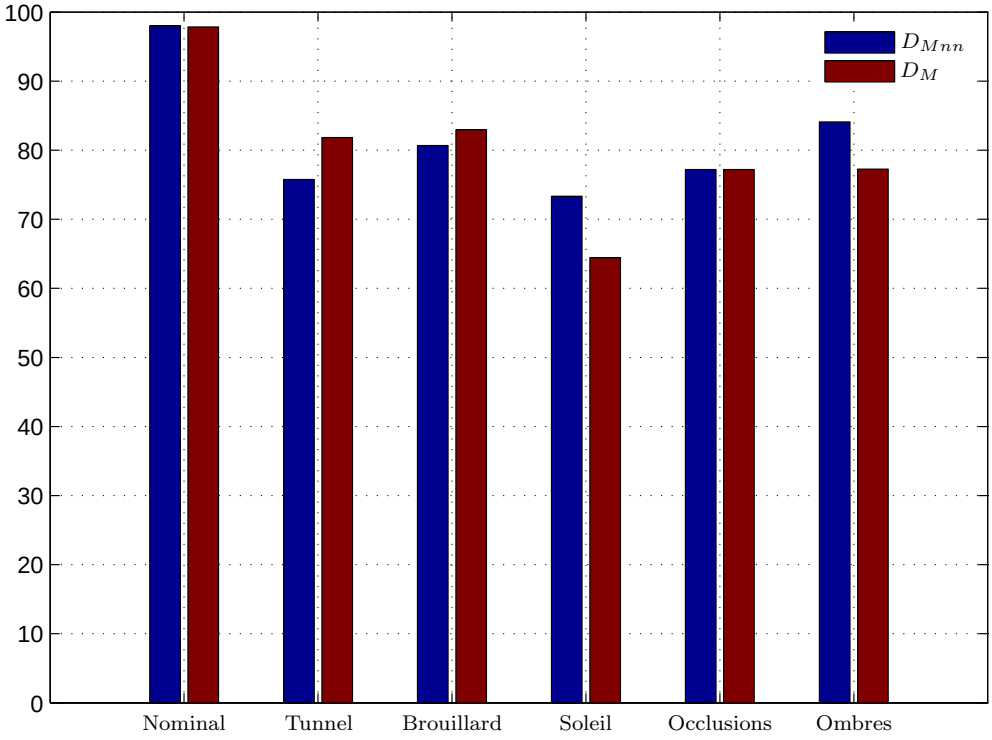


FIGURE 4.20 – Résultats sur les cas non-nominaux pour les deux détecteurs au même point de fonctionnement (98 %).

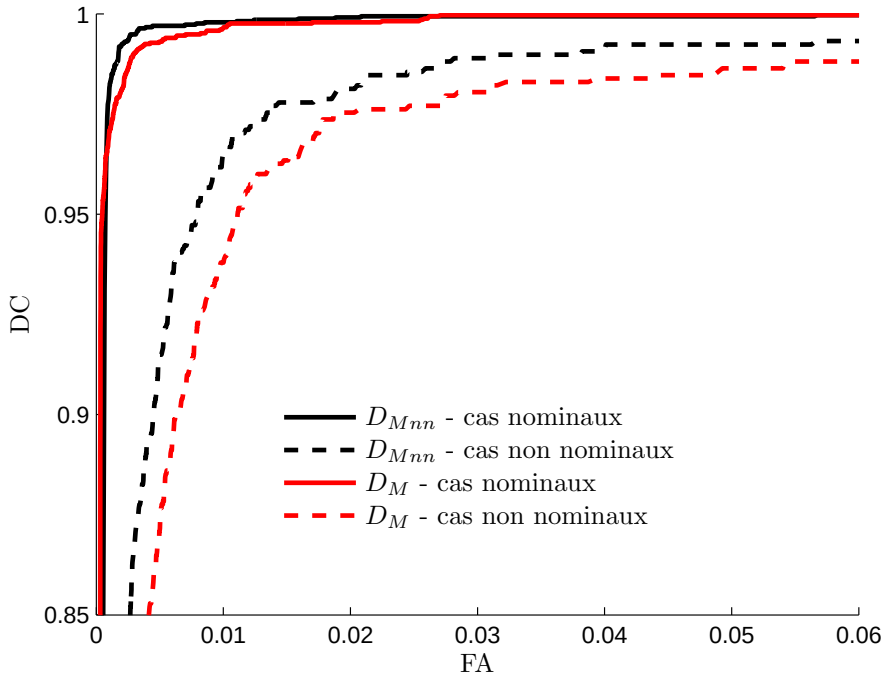


FIGURE 4.21 – Courbes ROC pour les deux détecteurs entrainés sur les cas nominaux et non nominaux.

Bilan

Cette section a étudié le comportement d'un classifieur en cascade sur des images prises dans des scénarios défavorables. Le détecteur Mixte s'est montré robuste aux images prises sur des situations adverses. Ajouter des exemples non-nominaux pendant l'apprentissage permet d'obtenir un détecteur qui obtient des meilleurs résultats sur les deux bases : nominale et non-nominale.

Certes nous observons une nette diminution des performances de notre détecteur dans les situations non-nominales. Cependant, la limite des capteurs, tels que la vision embarquée, correspond aux limites de la vision humaine. Dans quelques images, même pour une personne, il est difficile de détecter les véhicules. Pour ces cas, d'autres types de capteurs seront plus adéquats pour donner une réponse fiable.

FIGURE 4.22 – Exemples de résultats de détections des deux détecteurs.

Chapitre 5

Cascade Attentionnelle pour la classification du véhicule

5.1 Introduction

La classification des véhicules en voiture de tourisme, utilitaire ou camion, permet d'obtenir des statistiques pour estimer la taille, le poids, ainsi que la fréquence de présence de ces classes dans une voie. Taylor [135] a identifié différents domaines où l'information concernant la classification des véhicules s'avère nécessaire :

- ⋆ Législation et régulations pour des systèmes de trafic : il existe aujourd'hui en France des législations et restrictions sur la circulation de certains types de véhicules sur les autoroutes. La classification est nécessaire pour prouver la pertinence et l'efficacité de ces mesures et proposer des nouvelles mesures.
- ⋆ Études sur la sécurité : analyser le comportement des conducteurs et le type de véhicules concernés dans les accidents, etc.
- ⋆ Planification des routes, ponts, parkings, etc.
- ⋆ Systèmes de péage automatique : confirmer que les usagers paient le tarif correcte et éviter les fraudes.

Ces applications correspondent à une analyse *off line* des images pour l'extraction des conclusions et des statistiques. D'autres types d'applications utilisent la classification en

[135] M.A.P. Taylor and W. Young. *Traffic Analysis : New Technology and New Solutions.* Hodder Arnold, 1988.

temps réel dont les systèmes de surveillance routière [21, 63, 102, 64].

Bien que cette classification s'avère utile pour les systèmes anti-collision, elle n'est pas trop développée dans le domaine de la vision embarquée. En effet, la dynamique des véhicules varie d'une classe à une autre et le suivi d'un véhicule obstacle peut être mieux estimé si cette information est disponible.

Les travaux que nous allons aborder dans ce chapitre correspondent à l'étape consécutive à la détection d'un véhicule : l'identification de celui-ci dans un nombre de classes connues. La détection nous fournit des hypothèses sur la position et la taille des rectangles englobant un véhicule dans une image. La classification nous indique à laquelle des trois classes il appartient : véhicule de tourisme, utilitaires ou camions. Nous trouvons dans la littérature quelques méthodes de détection suivi de classification d'objets [125, 74, 94]. Nous avons déjà abordé la détection dans les chapitres précédents. La classification, quant à elle, se présente comme un problème complexe et nous avons de fortes contraintes pour la conception du système : il doit répondre en temps réel et les erreurs de classification doivent être minimisées. Les algorithmes que nous présentons par la suite, ont été identifiés selon deux types : les méthodes basées sur les modèles et les méthodes d'arbre de décision.

5.1.1 Méthodes basées sur les modèles

Ces méthodes modélisent chaque classe dans un espace de paramètres spécifique. La classification consiste à trouver le modèle le plus proche de l'image de test.

Kalinke [78] utilise la distance de Hausdorff [70] pour séparer deux classes de véhi-

---

[21] M.J.J. Burden and M.G.H. Bell. Vehicle classification using stereo vision. In *Image Processing and Its Applications, Sixth International Conference on*, volume 2, pages 881–885, Dublin, Juilliet 1997.

[63] S. Gupte, O. Masoud, R.F.K. Martin, and N.P. Papanikolopoulos. Detection and classification of vehicles. *IEEE Intelligent Transportation Systems*, 3(1) :37–47, Mars 2002.

[102] S. Mohottala, M. Kagesawa, and K. Ikeuchi. Vehicle class recognition using 3d cg models. In *Intelligent Transportation Systems*, 2003.

[64] D. Han, M. J. Leotta, D. B. Cooper, and J. L. Mundy. Vehicle class recognition from video-based on 3d curve probes. In *IEEE International Conference on Computer Communications and Networks*, pages 285–292, 2005.

[125] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Pattern Analysis and Machine Intelligence*, 20(1) :23–38, Janvier 1998.

[74] M. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, MERL, 2003.

[94] Yen-Yu Lin and Tyng-Luh Liu. Robust face detection with multi-class boosting. In *IEEE Computer Vision and Pattern Recognition*, volume 1, pages 680–687, Washington, DC, États Unis, 2005.

[78] T. Kalinke, C. Tzomakas, and W. v. Seelen. A texture-based object detection and an adaptive model-based classification. In *IEEE International Conference on Intelligent Vehicles 1998*, volume 1, pages 143–148, 1998.

[70] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the haus-

cules : voitures de tourisme et camions. Deux modèles différents sont générés : le premier a été construit empiriquement et le deuxième a été estimé statistiquement en utilisant 50 types des voitures et camions. La distance de Hausdorff est utilisée pour mesurer la similarité entre les contours de type LOC [60] des modèles et ceux de l'image. Un réseau de neurones, placé dans l'étape finale de la classification, utilise comme entrées, les distances de Hausdorff aux quatre modèles et les classifie en voitures, camions et non-objets (fond).

Kato [81] utilise une fonction de discrimination quadratique modifiée pour détecter trois classes des véhicules : voitures, motos et camions. Les exemples sont regroupés en *clusters* dans l'espace originale (niveaux de gris) et ces derniers vont représenter les classes véhicule et non-véhicule. Chacun de ces *clusters* est considéré comme une distribution normale, avec une moyenne et une matrice de covariances. La fonction de discrimination permet de classifier un exemple de test comme étant positif ou négatif, à partir de la distance au plus proche *cluster*. Ils ne différencient pas une détection comme étant une voiture, une moto ou un camion, bien que la méthode puisse servir à cela.

La constitution d'un modèle de classe peut être aussi envisagé à partir de l'utilisation, par exemple, des descripteurs SIFT [96, 95] ou la création d'un vocabulaire visuel : les Sacs De Mots [75, 109, 87] (*Bag of words*).

Les algorithmes qui utilisent les SIFT comme espace de paramètres, commencent par une recherche des points d'intérêt et l'extraction de leurs descripteurs locaux. La reconnaissance des objets est réalisée via l'application d'un apprentissage supervisé : l'ensemble de descripteurs qui décrit un objet est appris à partir d'une base de données. Ensuite, la détection ou la reconnaissance de cet objet dans une image peut être fait à l'aide de la Transformée de Hough généralisée [46]. Ce type de descripteurs s'est montré robuste aux

---

dorff distance. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(9) :850–864, Septembre 1993.

[60] C. Goerick, D. Noll, and M. Werner. Artificial neural networks in real-time car detection and tracking applications. *Pattern Recognition Letters*, 17(4) :335–343, Avril 1996.

[81] T. Kato, Y. Ninomiya, and I. Masaki. Preceding vehicle recognition based on learning from sample images. *IEEE Transactions on Intelligent Transportations Systems*, 3(4) :252–260, Decembre 2002.

[96] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, pages 91–110, 2004.

[95] D. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[75] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *IEEE International Conference on Computer Vision*, volume 1, pages 604–610, 2005.

[109] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*, volume 4, pages 490–503. Springer, 2006.

[87] D. Larlus and F. Jurie. Latent mixture vocabularies for object categorization. In *British Machine Vision Conference*, 2006.

[46] F. Estrada, A. Jepson, and D. Fleet. Local features tutorial. Technical report, Université de Toronto, 2004. Disponible dans http ://www.cs.toronto.edu/ jepson/csc2503/tutSIFT04.pdf.

changements d'échelle, aux occultations et aux rotations.

Les Sacs de Mots sont basés sur une représentation locale adaptée aux images des classes qu'ils appellent mots. Les options pour le codage des images sont : les SIFT, les filtres [88] ou les vignettes d'intensités [1].

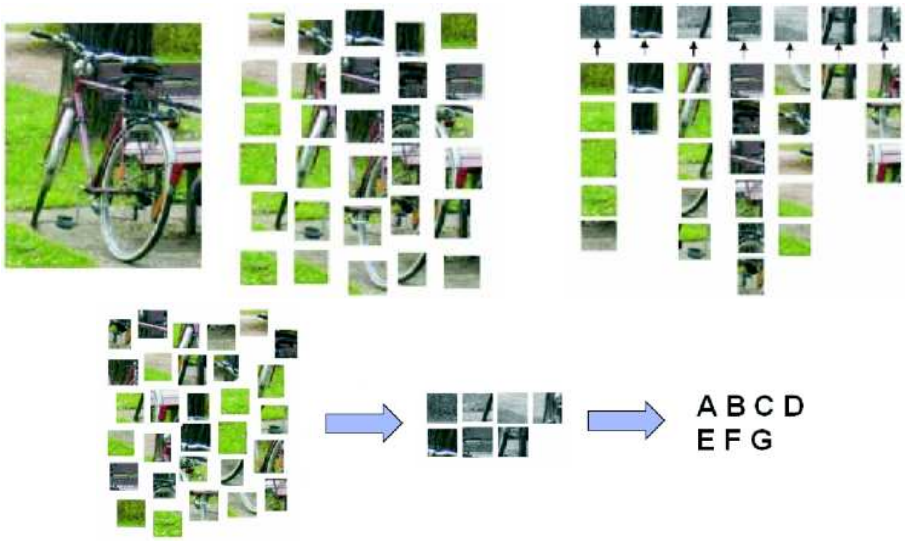


FIGURE 5.1 – Extraction et sélection des vignettes dans une image pour l'obtention des mots visuels composant un objet.

La figure 5.1 montre l'extraction des vignettes, réparties uniformément dans l'image, qui sont transformées en mots. Le vocabulaire est construit à partir des statistiques de présence de ces mots dans les images de la même classe dans la base d'apprentissage. Csurka [34] s'inspire de cette représentation pour obtenir un détecteur d'objets de différentes classes. Une fois obtenues les vignettes d'objets de classes, elles sont utilisées comme vecteurs de descripteurs pour réaliser une classification de type un contre tous. L'auteur utilise deux types de classifieurs : Support Vector Machine (SVM) et un classifieur entraîné par la méthode Adaboost. Le principal inconvénient de cette méthode est le temps de calcul qui, pour le moment, reste loin du temps réel.

5.1.2 Méthodes basées sur les arbres de décision

Ces méthodes combinent l'étape de détection et celle de classification et chaque image d'entrée est traitée par plusieurs classifieurs. Nous allons exposer par la suite les tra-

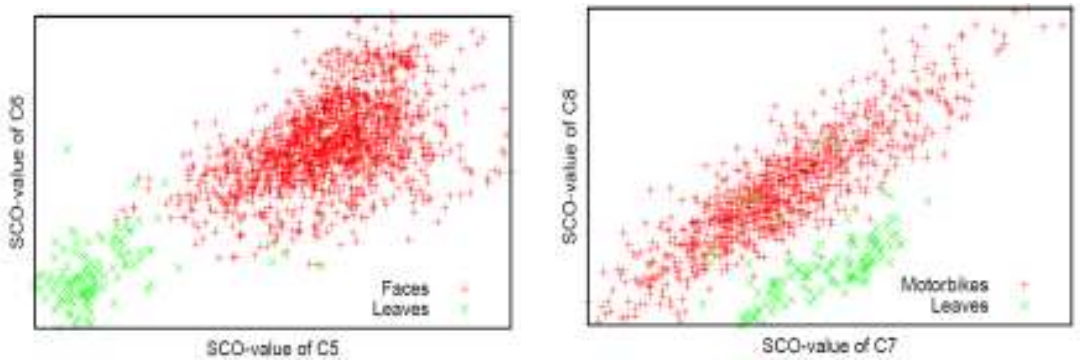
[88] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1) :29–44, 2001.

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11) :1475–1490, 2004.

[34] G. Csurka, C. R. Dance, F. Perronnin, and J. Willamowski. Generic visual categorization using weak geometry. In *Toward Category-Level Object Recognition*, pages 207–224, 2006.

vaux qui, proches de notre objectif, visent à un traitement en temps réel, à partir d'une architecture similaire à la cascade de Viola et Jones.

L'algorithme d'apprentissage d' Isukapalli [72] construit un arbre de décision (*decision tree classifier*, DTC, figure 5.3) où à chaque noeud est placé un classifieur obtenu par la méthode de dopage, qui est capable de détecter les M classes (nommé MC-VJ, pour $multi - classViolaetJones$). La sortie de chaque classifieur MC-VJ est appelée valeur SCO (*sum of classifier output*). L'idée est que les valeurs SCO des objets d'une même classe sont proches et que chacune des classes peut former un *cluster* dans cet espace. La figure 5.2 montre deux exemples des valeurs de deux SCO dans un espace à deux objets. La figure (a) montre l'espace obtenu pour les visages et les feuilles à partir des valeurs



(a) (b)

FIGURE 5.2 – Deux exemples de classes dans l'espace proposé par l'approche de Isukapalli. La figure (a) correspond à les classes visages-feuilles et la figure (b) à les classes motos-feuilles (b).

SCO de deux classifieurs, C_5 et C_6. Ils peuvent, notamment, séparer les visages des feuilles, pour l'espace à gauche, et les motos et les feuilles, dans l'image de droite. Similairement ils obtiennent un autre espace pour séparer les motos et les visages.

La figure 5.3 montre l'arbre de décision d'Isukapalli. Une image d'entrée est évaluée par les MC-VJ et la programmation dynamique la conduit par la branche correspondante, en prenant en compte les valeurs SCO obtenues dans l'étape précédente. A la fin de l'arbre, seront utilisés des détecteurs spécialisés sur une des classes (ici appelés SC-VJ pour *single class Viola et Jones*).

Les arbres de classifieurs sont aussi employés dans les travaux de Li [91] en regroupant les différentes vues des visages à partir de leur orientation. Au total, 11 classifieurs sont entraînés avec $FloatBoost$. Ces classifieurs forment un détecteur pyramidal du type *coarse-to-fine*. Les premiers classifieurs regroupent la plupart ou toutes les classes. Les derniers

[72] R. Isukapalli and A.M. Elgammal. Learning policies for efficiently identifying objects of many classes. In *IAPR International Conference on Pattern Recognition*, pages III : 356–361, 2006.

[91] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9) :1112–1123, 2004.

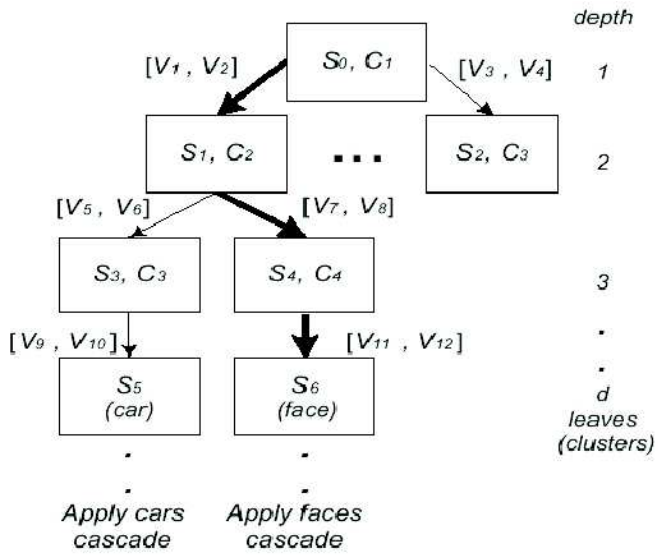
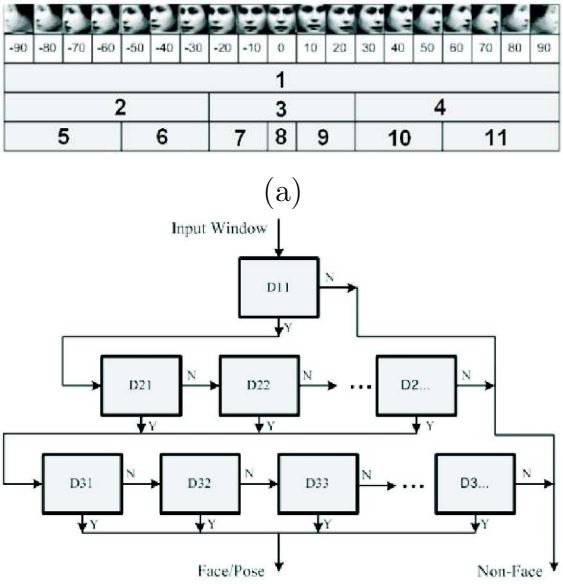


FIGURE 5.3 – Arbre de décision de Isukapalli.

sont plus spécialisés et regroupent les classes qui sont les plus proches (ou qui peuvent être fusionnées en une seule classe). Dans la figure 5.4, nous voyons la pyramide créée par



(a)

(b)

FIGURE 5.4 – Schéma en pyramide.
(a) les classes qui regroupent les 11 classifieurs, (b) détection en cascade.

les auteurs comme exemple. Le premier classifieur regroupe toutes les orientations. Les trois suivants, se concentrent sur trois orientations : [-90,-30], [-30,+30] et [+30,+90]. Dans la dernière couche de la pyramide les intervalles des orientations considérés par chaque classifieur sont plus réduits. La méthode de détection est montrée dans la figure 5.4. Une vignette d'entrée est évaluée par la couche supérieure, qui lui permet de passer dans la couche suivante si elle est validée comme positive.

Torralba et al. [137] regroupent aussi les M classes pour construire un détecteur en pyramide, au lieu d'entraîner indépendamment M classifieurs binaires. Pour le cas de trois

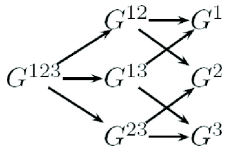


FIGURE 5.5 – Arbre de décision de Torralba.

classes d'objets, ils définissent les fonctions de classifications suivantes :

$$\begin{aligned} H(v,1) &= G^{1,2,3}(v) + G^{1,2}(v) + G^{1,3} + G^{1}(v) \\ H(v,2) &= G^{1,2,3}(v) + G^{1,2}(v) + G^{2,3} + G^{2}(v) \\ H(v,2) &= G^{1,2,3}(v) + G^{1,3}(v) + G^{2,3} + G^{3}(v) \end{aligned}$$

où chaque $G^{S(n)}(v)$ corresponds à un classifieur boosté :

$$G^{S(n)} = \sum_{m=1}^{M_n} g_m^n(v)$$

L'indice n représente le noeud dans l'arbre de la figure 5.5. $S(n)$ est l'ensemble des classes qui partagent le noeud n.

Chaque fonction de classification forte est construite de la façon suivante :

- Dans une itération d'Adaboost sont mises en concurrence les fonctions faibles appartenant à chaque ensemble $S(n)$ (les six ensembles de la figure 5.5),
- La fonction faible g_t^n de l'ensemble S(n), qui commet l'erreur la plus petite pour tous les n, est retenue dans le classifieur boosté $G^{S(n)}$.

Dans cette méthodologie, un ensemble $S(n)$ peut rester vide, si aucune fonctions faible n'a été choisie. Ils ont basé son algorithme sur la version de dopage *GentleBoost* et utilisent un dictionnaire de 21 objets, disponible en ligne [1].

La figure 5.6 expose un exemple d'obtention des fonctions de classification pour trois classes (cercles) et négatifs (croix). Chaque frontière des fonctions G^n est défini à partir d'une fonction faible. Les classifieurs finales $H(1)$, $H(2)$ et $H(3)$ montrent dans la partie plus claire, les espaces considérés comme positifs pour chacune des classes.

1. http ://web.mit.edu/torralba/www/multiclass.html

[137] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features : Efficient boosting procedures for multiclass object detection. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 762–769, 2004.

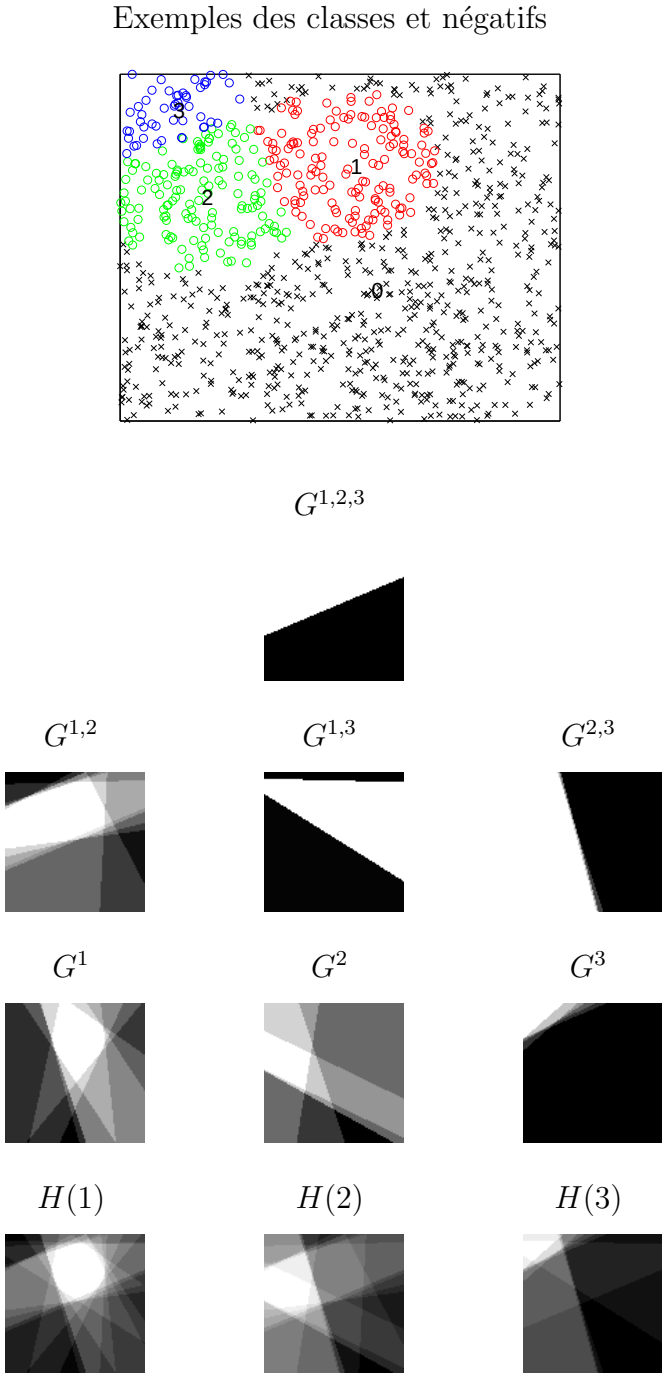


FIGURE 5.6 – Conception du classifieur de Torralba.

5.1.3 Bilan

La revue littéraire nous permet d'avancer quelques conclusions :

- Les classifieurs de Kato et Kalinke utilisent l'ensemble complet de descripteurs de chaque vignette pour faire la classification (niveaux de gris pour le premier et contours LOC pour le deuxième). Une étape préliminaire de génération des hypothèses les permet de se concentrer sur les hypothèses les plus probables.
- L'idée de Isupakalli est originale, mais les classes qui ont servi pour les tests sont très

différentes. Au contraire, les classes de véhicules sont assez proches et ont beaucoup de caractéristiques communes.

- L'arbre de classifieurs de Li se présente comme une solution viable. Nous pourrions regrouper tous les véhicules dans un premier détecteurs, pour spécialiser des autres sur chacune des classes.
- Le travail de Torralba prouve qu'il est possible de tracer des frontières particulières à chaque classe dans l'espace déterminé par Adaboost. Nous allons utiliser plus tard cette idée pour la génération d'un nouveau détecteur.

Les solutions que nous allons proposer dans les chapitres suivants sont inspirées par les méthodes précédentes. Nous allons développer deux classifieurs qui sont composés d'une étape de détection suivie de la classification, ainsi qu'un autre classifieur qui a une architecture arborescente. Finalement, le dernier classifieur utilise l'information obtenue pendant l'étape de détection.

La section suivante présente la base de données des classes.

5.2 Base de données

Nous avons trois classes de véhicules (voir figure 5.7) :

1. voitures de tourisme,
2. utilitaires, camionnettes et 4x4,
3. petit et gros camions.

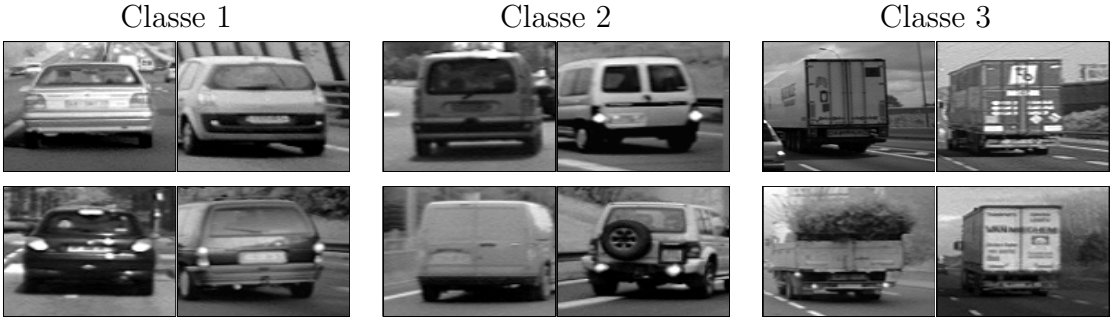


FIGURE 5.7 – Exemples des trois classes de véhicules.

La base de donnés des classes n'est pas homogène, nous avons un nombre plus important d'exemples de la classe 1 que des autres (voir le tableau 5.1).

Les exemples virtuels sont crées pour équilibrer les bases, étant donnée que les résultats obtenus sur la base originale ne sont pas satisfaisants. Si la base d'apprentissage positive n'est pas assez vaste (en nombre de classes et en variation des caractéristiques) elle n'est pas assez représentative de la classe pour valider les véhicules de la base de test.

Pour obtenir les images virtuelles quatre manipulations sont appliquées à chaque image de véhicule des classes 2 et 3 :

	Nombre d'exemples		
	Base Original	Base Virtuelle	Base de Test
classe 1	1521	3042	1106
classe 2	336	2564	278
classe 3	264	2104	137

TABLE 5.1 – Constitution des bases.

a) changement de la luminance : nous utilisons une loi gamma (non linéaire) pour changer les niveaux de gris des pixels,

b) filtrage : un filtre gaussien est appliqué dans l'image pour ensuite sous-échantillonner l'image à l'échelle suivante (multiple de 32) plus petite,

c) changement du cadrage : nous utilisons les six points de la vérité terrain d'une autre façon, la position horizontale est calculée à partir du point le plus à droite et du point le plus à gauche. De plus, le cadrage se fait avec un rectangle 15 % plus grand.

d) nous effectuons aussi une rotation suivant l'axe vertical des vignettes, en profitant de la symétrie des véhicules, pour doubler la quantité.

Le fait d'avoir augmenté la base d'apprentissage va nous permettre d'obtenir des détecteurs appropriés. Cependant, ces exemples virtuels n'ont pas introduit une diversité réelle dans les exemples positifs. Il serait plus pertinent d'augmenter la base à partir d'exemples réels.

5.3 Détection multi-classes

Le choix du type de détecteur, qui fournit les hypothèses à l'étape de classification, s'avère très important. D'abord, il doit minimiser les fausses alarmes et, ensuite, la boîte englobant l'hypothèse doit être ajustée le mieux possible à la voiture, pour optimiser la classification.

La conception du détecteur se révèle comme un problème complexe pour notre problématique. Préalablement, nous allons étudier la corrélation qui existe entre nos classes de véhicules.

5.3.1 Détecteurs individuels

Nous entraînons trois détecteurs Mixtes D_1, D_2 et D_3, un pour chacune des classes. La base de Test, sur laquelle nous allons les évaluer, est composée de 1522 véhicules distribués de la façon montrée dans le tableau 5.1.

Le tableau 5.2 présente les résultats croisés des détecteurs. Nous observons qu'un détecteur qui a utilisé les exemples de la classe 1 pour l'apprentissage, détecte aussi 78 % des exemples de la classe 2 et 18 % des exemples de la classe 3. Dans la deuxième ligne, le détecteur de la classe 2 obtient un bon taux de détections pour les autres deux classes.

Détecteur	DC C1	DC C2	DC C3	FA
D_1	**93.23**	78.3	18.23	0.000115
D_2	58.6	**90.3**	49.16	0.000056
D_3	33.8	80.1	**93.4**	0.000181

TABLE 5.2 – Comportement des trois détecteurs.

La dernière ligne montre les résultats du détecteur de la classe 3. En conclusion, la classe 2 a des caractéristiques communes aux deux autres. Les classes 1 et 3 sont plus éloignées dans l'espace de paramètres et, donc, plus faciles à discriminer.

Détecteur	DC C1	DC C2	DC C3	FA
D_1	**96**	88.7	30.6	0.000355
D_2	88	**96**	88	0.000829
D_3	46	84.7	**96**	0.000292

TABLE 5.3 – Perfomances des trois détecteurs au même point de fonctionnement : 96 % DC.

Nous avons mis les détecteurs D_1, D_2 et D_3 au même point de fonctionnement (96 %) de DC de sa classe et nous analysons dans le tableau 5.3 les résultats obtenus. Le point de fonctionnement obtenu pour D_1 et D_3 nous montre que le taux de détections des autres classes augmente légèrement pour les deux. De plus, pour le premier, le taux de fausses alarmes est doublé mais, pour D_3 s'est réduit. Le comportement pour le détecteur D_2 est bien différent que les autres deux. Il a vu augmenter fortement le taux de détections correctes des autres classes une fois qu'il est devenue mois discriminant. Cela prouve l'importante corrélation qu'il existe entre cette classe et les autres deux et anticipe la difficilté qu'auront les classifieurs pour faire la discrimination entre classes.

5.3.2 Détecteur multi-classes

Nous allons obtenir deux détecteurs Mixtes en groupant toutes les classes dans la base positive. Le premier détecteur est un détecteur en cascade non contrôlée et l'autre est un détecteur en cascade contrôlée.

Détecteur	Descripteurs	DC	DC C1	DC C2	DC C3	FA
Non Controlé	2230	95.2	95.8	96	89.1	**0.00961**
Controlé	1032	**96.5**	**96.8**	**96.8**	**93.4**	0.00134

TABLE 5.4 – Comportement des deux détecteurs.

Les résultats de la table 5.4 montrent que les deux détecteurs ont un taux de détections élevé, mais en même temps, un nombre des fausses alarmes très grand. Ceci est un résultat normal. Réunir ensemble des classes hétérogènes dans la base positive oblige Adaboost à élargir les frontières du classifieur pour devenir moins discriminant. En pratique, Adaboost décrémente le seuil du classifieur fort qui fait augmenter en même temps le nombre de

fausses alarmes. Sauf quelques cas ponctuels [3], les travaux de la littérature évitent de mélanger différentes classes de véhicules ou de montrer le nombre de fausses alarmes résultant [117].

Détecteur	**Descripteurs**	**DC**	**DC C1**	**DC C2**	**DC C3**	FA
Non Controlé	1630	96	97	96.8	**90**	**0.00119**
Controlé	1032	**96.1**	**98**	**97**	94	0.00129

TABLE 5.5 – Performances des deux détecteurs obtenus à partir des courbes ROC avec un taux de DC de 96 %.

Nous pouvons comparer ces détecteurs à partir de la courbe ROC doublement paramétrée. Le point de fonctionnement est fixé à un nombre de détections correctes de 96 %. Nous pouvons voir dans le tableau 5.5 les résultats de ces nouveaux détecteurs, ainsi que sa constitution (nombre de descripteurs). Les nouveaux détecteurs ont une architecture avec un nombre plus faible de descripteurs qu'auparavant (ils s'arrêtent avant d'arriver au dernier étage). Le fait de devenir plus discriminant pour la cascade contrôlée a entraîné une perte plus importante des exemples de la classe 3. Nous pouvons attribuer cette perte à une mauvaise généralisation de cette classe due au faible nombre d'exemples d'apprentissage.

Dans la section suivante, nous allons proposer quatre architectures de détection et classification.

5.4 Classification du type de véhicule

Cette section présente les architectures que nous avons utilisées pour faire la classification de véhicules. La première solution correspond à l'application directe des détecteurs individuels. La deuxième et la troisième sont des cascades arborescentes, avec une étape de détection et une autre de classification. La dernière utilise l'espace de projection du détecteur pour réaliser la classification.

Nous allons évaluer chaque ensemble de classifieurs sur la vérité terrain (VT) de la base de test. Nous pouvons considérer la VT comme un cas de détection idéale, où un détecteur nous a rendu les positions exactes des véhicules dans les images et qui correspondent aux rectangles englobant étiquetés par l'utilisateur.

Nous obtenons pour chaque architecture deux mesures de performance de la classification : la micro-précision, qui mesure le taux de classifications correctes sur le total

[3] B. Alefs. Embedded vehicle detection by boosting. In *Intelligent Transportation Systems Conference*, pages 536–541, Vienna, 2006.

[117] D. Ponsa and A. Lopez. Cascade of classifiers for vehicle detection. In *Advanced Concepts for Intelligent Vision Systems*, volume 4678, pages 980–989. Springer, 2007.

des exemples, et la macro-précision, qui mesure la moyenne des taux de classifications correctes par classe.

5.4.1 Classification par les Détecteurs Individuels

Les détecteurs D_1, D_2 et D_3 de la section 5.3, spécialisés sur chacune des classes, sont appliqués en parallèle sur une image selon le schéma de la figure 5.8.

L'étape de classification prend en compte la sortie du dernier étage de chaque cascade G_n^1, G_n^2 et G_n^3. Nous utilisons une fonction sigmoïde qui nous indique une probabilité pour chaque détecteur, à partir de ces valeurs. Cette probabilité est donnée par la fonction suivante :

$$P(positive|G_n(x)) = \frac{1}{1 + exp(AG_n(x) + B)}$$

où $G_n(x)$ est la sortie du dernier classifieur fort, et A et B des constantes réelles. Ces deux constantes sont estimées à la fin de l'apprentissage du classifieur fort $G_n(x)$. Les sorties de l'application de $G_n(x)$ aux exemples de la base de validation positive et de la base négative vont ajuster la fonction sigmoïde à l'aide de l'algorithme de Platt [26], pour que les premiers prennent la valeur 1 et les seconds la valeur 0.

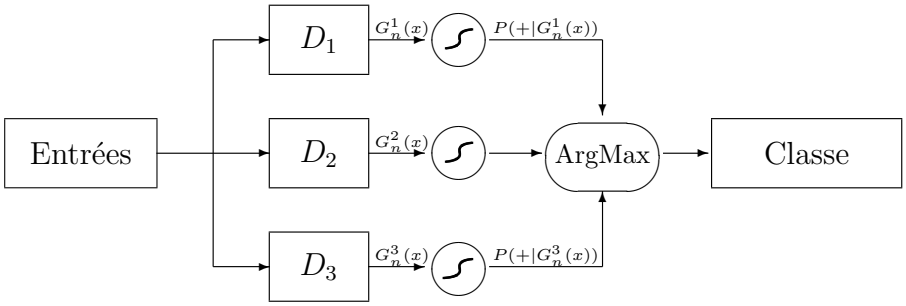


FIGURE 5.8 – Architecture du classifieur basé sur les détecteurs individuels

Pour évaluer l'architecture sur la VT nous calculons pour chaque rectangle englobant la probabilité des trois détecteurs. Celle qui a la plus grande valeur, est considérée comme la classe gagnante :

$$c(\mathbf{x}) = Argmax(P(pos|G_n^1(x)), P(pos|G_n^2(x)), P(pos|G_n^3(x)))$$

Nous présentons les résultats obtenus sur la VT dans le tableau 5.6.

Le taux de classifications correctes des classes 2 et 3 est assez bas : 61.1 % et 56.9 %. Ce comportement était à prévoir étant donné que nous avons utilisé des détecteurs pour réaliser une classification. Effectivement, ils n'ont pas été entraînés pour faire une

[26] T. Chateau, V. Gay Belille, F. Chausse, and J.T. Lapreste. Real-time tracking with classifiers. In *Workshop on Dynamical Vision*, pages 218–231, 2006.

Détecteurs Individuels	
Micro-précision	79.4
Macro-précision	68.3

Matrice de confusion

	Nb véhicules				Taux		
	c1	c2	c3		c1	c2	c3
c1	961	135	10	→	86.8	12.2	0.9
c2	101	170	7	→	36.3	61.1	2.5
c3	15	44	78	→	10.9	32.1	56.9

TABLE 5.6 – Performance de la méthode de "Détecteurs Individuels" sur la Vérité Terrain

discrimination inter-classes : pour l'entraînement d'une classe, nous n'avons pas utilisés des exemples des autres classes comme négatifs.

De plus, le principal inconvenient de cette architecture est le temps de calcul, étant donné que nous parcourons avec les trois détecteurs en parallèle notre zone de recherche dans l'image.

5.4.2 Classification Pyramidale

Nous allons nous inspirer du schéma en pyramide que Li [91] utilise pour la détection des visages à plusieurs orientations.

Dans notre architecture, un premier détecteur, entraîné avec toutes les classes confondues dans la base positive, élimine un maximum de fausses alarmes pour fournir les hypothèses à une deuxième étape de détection et classification. Ce détecteur, que nous appelons $D^P_{1.2.3}$, est un détecteur Mixte en cascade contrôlée et il est composé de 10 étages. Le choix de ce paramètre s'appuie sur la courbe d'évolution du taux de détections correctes et de fausses alarmes au long des étages présentée dans la section 4.5. Un nombre plus important d'étages pourrait entraîner une élimination des exemples positifs.

Pendant la deuxième étape, ces hypothèses sont traitées par des détecteurs en cascade spécialisés sur chacune de classes, que nous appelons D^P_1, D^P_2 et D^P_3. Ils sont entraînés avec les exemples de la classe correspondante dans sa base positive, et dans la base négative sont placés des exemples des autres classes, ainsi que des exemples d'images sans véhicules.

Quand le premier détecteur D^P_1 ne valide pas une fenêtre comme étant de classe 1, elle est évaluée par le classifieur suivant D^P_2. Il l'envoie au dernier détecteur D^P_3 s'il la considère comme étant négative (voir figure 5.9). Chaque hypothèse garde l'étiquette correspondante au détecteur qui l'a validé.

Afin d'évaluer la classification sur la VT, nous appliquons successivement les détecteurs D^P_1, D^P_2 et D^P_3 sur les rectangles englobants. Si l'un d'entre eux valide une vérité

[91] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9) :1112–1123, 2004.

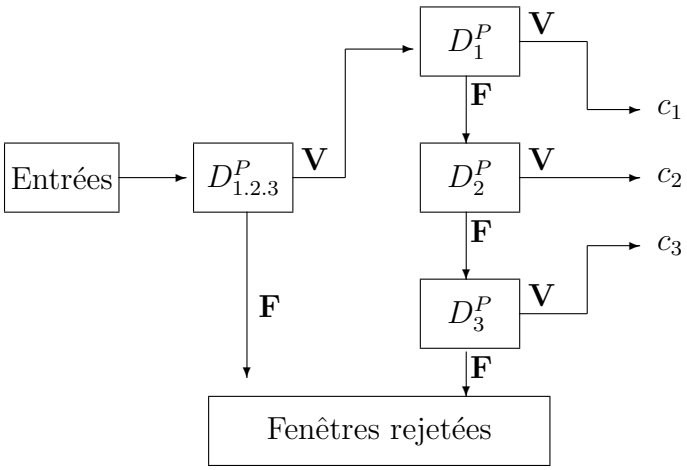


FIGURE 5.9 – Architecture Pyramidale.

terrain, nous la considérons comme de cette classe. Il peut arriver qu'une vérité terrain soit rejetée par les trois détecteurs. Dans ce cas, nous considérons qu'il s'agit d'une erreur de classification et nous l'ajoutons à la colonne des exemples rejetés du tableau 5.7 : colonne **R**.

Classification Pyramidale	
Micro-précision	66.6
Macro-précision	61.7

Matrice de confusion

	Nb véhicules					Taux			
	c1	c2	c3	R		c1	c2	c3	R
c1	782	7	10	307	→	70.7	0.6	0.9	27.7
c2	42	142	4	86	→	15.1	52.5	1.4	30.9
c3	2	15	85	35	→	1.4	10.9	62.0	25.5

TABLE 5.7 – Performance de l'architecture Pyramidale.

Dans le tableau 5.7, nous constatons que des VT ont été rejetées par les détecteurs et représentent 30 % par classe. En général, la méthode n'a pas donnée de bons taux de classification sur la VT. Nous avons constaté que la majorité de ces cas correspondent aux rectangle englobants qui étaient trop serrés sur le véhicule : la vignette contient moins de contours verticaux, qui sont aussi importants pour le modèle du véhicule.

5.4.3 Classifieurs en Parallèle

Nous allons changer l'architecture précédente pour celle de la figure 5.10. Elle est composée d'un détecteur en cascade multiclasse $D_{1.2.3}$ de 20 étages, chargé d'éliminer les fausses alarmes et de rendre les hypothèses à l'étape suivante. Nous avons augmenté la

quantité d'étages, étant donné qu'il n'y a pas, dans cette architecture, une deuxième étape de détection et d'élimination des négatifs.

Chaque hypothèse est évaluée par des classifieurs simples dédiés à la classification inter-classe. Ils ont été spécialisés sur chacune des classes et vont associer à l'hypothèse une classe selon ces sorties. Par exemple, si nous voulons obtenir un classifieur C_1 dédié à la classe 1, nous utilisons les exemples de cette classe comme positifs. Les contre-exemples, qui composent la base négative, sont ceux des classes 2 et 3. De la même façon, nous obtenons C_2 et C_3. Les classifieurs C_i consistent en une fonction forte construite avec une quantité des fonctions faibles fixe à $T = 50$.

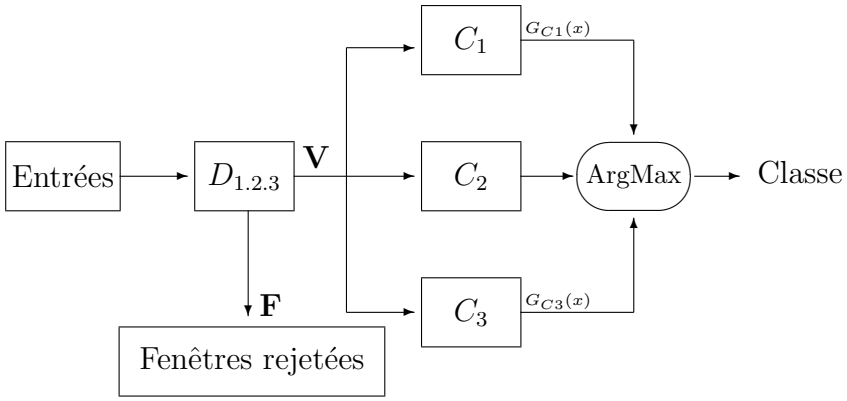


FIGURE 5.10 – Architecture du classifieur en parallèle.

La classification consiste à évaluer sur l'hypothèse $\mathbf{x}$ les trois C_i. Celui qui obtient la sortie normalisée $G_{Ci}(\mathbf{x})$ la plus importante (voir figure 5.10), donne la classe gagnante :

$$c(\mathbf{x}) = Argmax(G_{C1}, G_{C2}, G_{C3})$$

L'évaluation des classifieurs sur la vérité terrain nous a donné une micro-précision égale à **86.06 %** et une macro-précision égal à **79.09 %**.

La classification peut être améliorée en ajoutant trois nouveaux classifieurs : $C_{1.2}$, $C_{1.3}$, $C_{2.3}$ [137]. Nous prenons l'idée de Torralba qui permet d'augmenter la distance entre les classes dans l'espace de classification. Les nouveaux classifieurs sont entraînés avec deux classes comme positives et la troisième comme négative. L'espace de classification de chaque classe est construit à partir de la somme des sorties de trois classifieurs : C_1, $C_{1.2}$ et $C_{1.3}$ pour la classe 1, et ainsi de même pour les autres. Nous définissons la sortie d'un classifieur comme la valeur de la somme pondérée de ces fonctions faibles.

Pour réaliser la classification, nous allons considérer trois triplets :

$$c(x) = Argmax(G_{C1} + G_{C1.2} + G_{C1.3}, G_{C2} + G_{C1.2} + G_{C2.3}, G_{C3} + G_{C1.3} + G_{C2.3})$$

[137] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features : Efficient boosting procedures for multiclass object detection. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 762–769, 2004.

Les résultats sur la vérité terrain sont montrés dans le tableau 5.8. Nous pouvons constater une amélioration par rapport aux résultats obtenus à partir de l'utilisation de seulement trois classifieurs dans la Micro-précision et la Macro-précision.

Classifieurs en Parallèle	
Micro-précision	91.58
Macro-précision	82.91

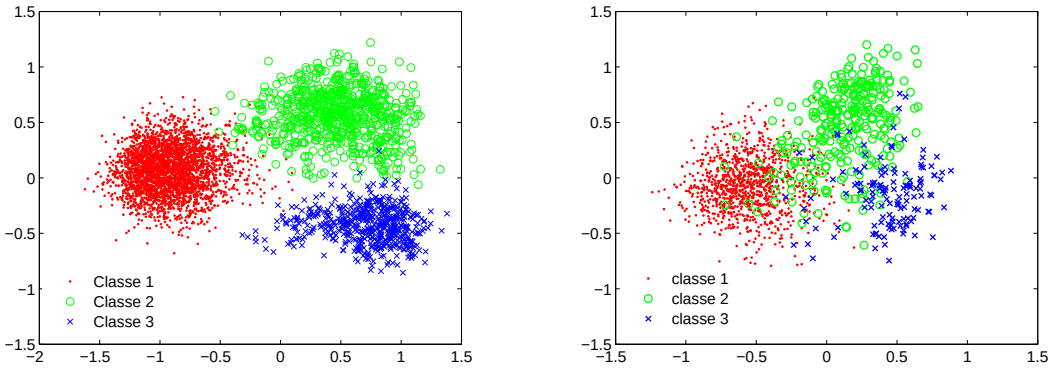
Matrice de confusion

	Nb véhicules				Taux		
	c1	c2	c3		c1	c2	c3
c1	1069	11	26	→	96.7	1.0	2.4
c2	30	228	20	→	10.8	82.0	7.2
c3	5	36	96	→	3.6	26.3	70.1

TABLE 5.8 – Performance sur la vérité terrain des classifieurs en parallèle.

Analyse de l'espace de classification

Nous appliquons les triples des classifieurs sur la base de véhicules d'apprentissage. A l'aide d'une Analyse en Composantes Principales (ACP), nous projetons ces valeurs dans un espace 2D (voir figure 5.11 (a)).



(a) (b)

FIGURE 5.11 – Projection 2D des valeurs de sortie de classifieurs C_i à l'aide d'un ACP. (a) les sorties d'exemples d'apprentissage et (b) la sortie des exemples de test.

La figure 5.11 (a) nous montre que les classes sont assez séparables dans l'espace d'apprentissage. Une simple somme nous permet d'obtenir des résultats acceptables dans la base de test. Pourtant, la figure (b) qui montre la projection des exemples de la vérité terrain dans cet espace, prouve qu'ils sont assez proches et que la généralisation n'est pas parfaite.

Classifieur en Parallèle + NN	
Micro-précision	90.33
Macro-précision	82.28

Matrice de confusion

	Nb véhicules				Taux		
	c1	c2	c3		c1	c2	c3
c1	989	26	24	→	95,2	2,5	2,3
c2	33	144	33	→	15,6	68,5	15,9
c3	4	7	57	→	6,1	10,8	83,1

TABLE 5.9 – Performance de classifieurs en parallèle et classification à l'aide d'un réseau de neurones.

En ce sens, un réseau de neurones (NN) peut être envisagé pour optimiser la classification. Les entrées de ce réseau sont les sorties des six classifieurs. Par cross-validation, nous obtenons une architecture optimale d'un réseau à six entrées, deux cellules cachées et trois sorties. Pour ne pas biaiser les résultats, nous avons utilisé dans l'entraînement du NN, une quantité fixe des exemples de la base de test pour chacune des classes. Nous avons choisi utiliser 68 exemples, qui représente la moitié des échantillons de test de la classe 3. Le reste est employé pour le test.

Le tableau 5.9 nous permet de constater que le résultat pour la classe 3 est meilleur, mais nous remarquons une plus faible discrimination entre la classe 1 et la classe 2. Cela est peut être dû au fait que la classe 2 contient des véhicules 4x4 qui sont similaires à quelques exemples de la classe 1.

5.4.4 Classification Multi-Modèle

Les étages des détecteurs en cascade projettent les données dans un espace, où ils sont capables de décider, à l'aide d'une fonction de classification, s'ils sont positifs ou non. Ces espaces, définis à partir de l'ensemble des fonctions faibles du classifieur fort, sont différents d'un étage à l'autre. Effectivement, l'utilisation du *bootstraping* fournit à l'apprentissage d'un étage des exemples négatifs validés par les étages précédents.

Nous pouvons donc penser qu'un exemple de test $\mathbf{x}$ est projeté successivement, tout au long de la cascade, dans des sous-espaces différents. L'espace total sert pour discriminer les positifs (véhicules) des négatifs (non-véhicules), mais nous voulons l'analyser pour établir s'il est possible de faire aussi une discrimination entre les classes.

Isupalki [72] et Torralba [137] ont montré que si nous entraînons un classifieur sur dif-

[72] R. Isukapalli and A.M. Elgammal. Learning policies for efficiently identifying objects of many classes. In *IAPR International Conference on Pattern Recognition*, pages III : 356–361, 2006.

[137] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features : Efficient boosting procedures for

férentes classes, ses fonctions faibles répondent différemment selon la classe. Pour Isupalki, bien que ses classes soient plutôt différentes (par exemple il compare des feuilles avec des motos), les valeurs de sorties des détecteurs forts sont différentes suivant la classe. Cela est dû à qu'un ensemble des fonctions faibles répond pour une classe et un autre pour l'autre. Évidement, ces deux ensembles peuvent partager quelques fonctions faibles, mais, au final, la somme pondérée d'un ensemble donne des valeurs distinctes. Torralba montre dans le tableau de la figure 5.12 l'ensemble complet des fonctions faibles composant son classifieur, dans la premier ligne. Les lignes suivantes associent à chaque descripteur la classe à laquelle ils répondent le mieux. Les descripteurs sont triés de gauche à droite, des plus génériques (partagés par plusieurs classes) aux plus spécifiques.

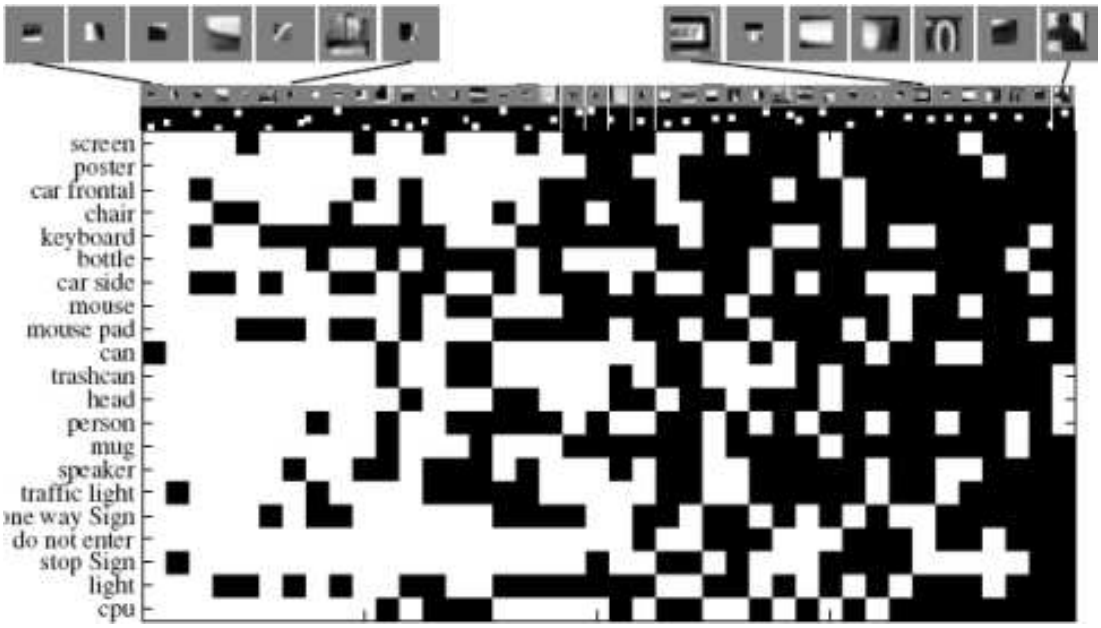


FIGURE 5.12 – Fonctions faibles du classifieur de Torralba et comment sont elles partagés pour les 21 classes considérées.

La méthode proposée concatène dans un vecteur f_i, toutes les réponses binaires de chaque fonction faible du détecteur en cascade (voir figure 5.13). Ce vecteur est utilisé pour projeter les exemples d'apprentissage dans un espace de classification.

La décision à partir de ce vecteur est construite à l'aide d'un réseau neuronal. Chaque composante du vecteur f_i est considérée comme une entrée du système qui aurait trois sorties, qui correspondent aux trois classes. Une optimisation du réseau nous a permis de choisir 20 cellules cachées.

---

multiclass object detection. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 762–769, 2004.

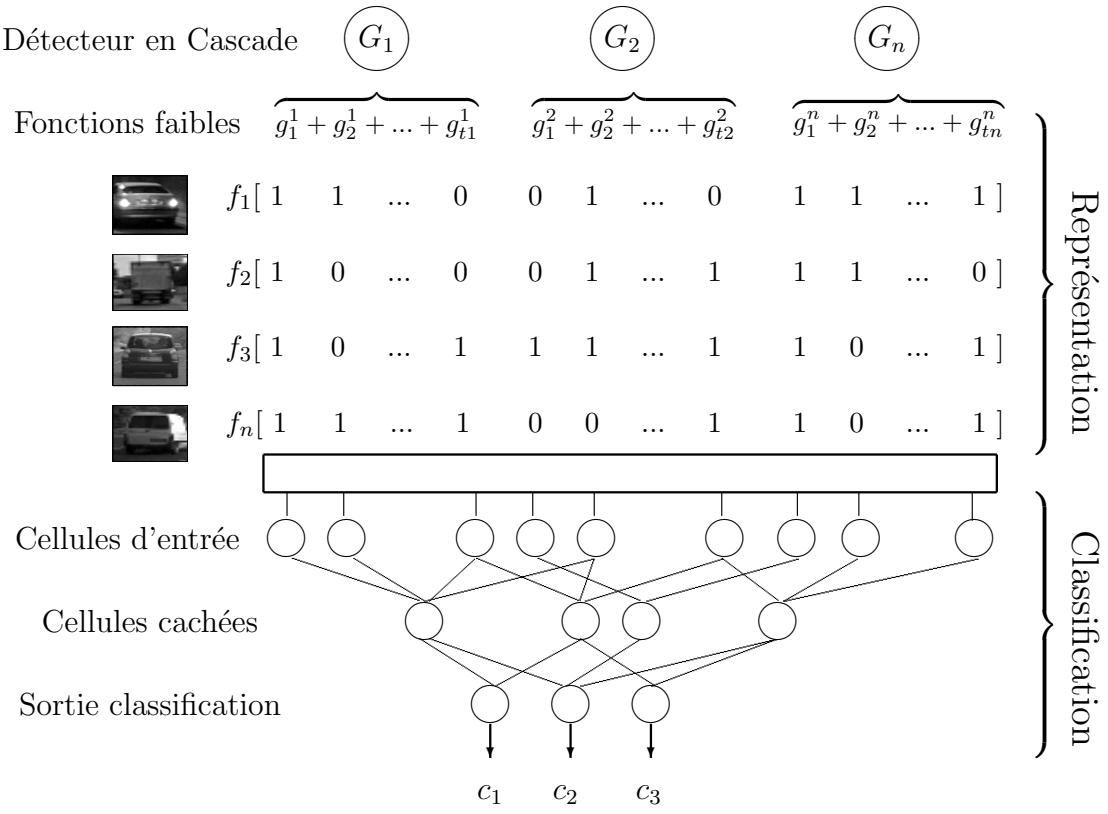


FIGURE 5.13 – Architecture du classifieur multi-modèles.

Détecteur multi-modèle

Nous avons supposé précédemment que, dans chaque sous-espace généré par les fonctions fortes du détecteur, les classes se groupent séparément. Ce groupement, peut être renforcé si nous utilisons une nouvelle approche Multi-Modèle dans l'apprentissage de la cascade. L'idée s'appuie sur le fait que les fonctions faibles pourraient répondre pour une classe si elles ont été apprises (valeur du seuil) à partir des valeurs de descripteurs des exemples de cette classe. En pratique, cela revienne à réaliser un apprentissage des détecteurs D_1, D_2, D_3 et D_{123} simultanément dans la même cascade (voir pseudo code du tableau 5.10).

Nous obtenons des fonctions de classification faibles g_{HoG}^n avec $n = [1; 2; 3; 123]$, qui calculent leur seuil optimal en utilisant le modèle de la classe n et les distributions des poids correspondante à celle-ci. Pour le cas de l'ensemble, le seuil est calculé avec le vecteur de poids complet et le modèle des trois classes mélangées (n=[123]). A chaque itération d'Adaboost, nous mettons en concurrence ces quatre familles de fonctions pour garder celle qui commet l'erreur la plus faible dans la classification entre les exemples positifs (toutes les classes) et les exemples négatifs.

De la même façon que pour les descripteurs de HoG, ceux de Haar peuvent être

Apprentissage du détecteur Multi-Modèle

1. Soient N exemples $(x_1,y_1),...,(x_N,y_N)$
   avec x_i les vecteurs de descripteurs $\in \Re$ des exemples i des trois classes
   et $y_i \in \{1,2,3,0\}$ (les trois classes et négatifs)
2. Initialiser $w_i = \frac{1}{N}$, $i = 1,...,N$,
   avec $w_{p,k}$ poids des positifs et $w_{n,k}$ poids des négatifs, et $t = 0$
3. Tant que $f > FA_{max}$ ET $t < T_{max}$
   Pour chaque descripteurs j, entrainer à partir des $w_{p,k}$ / y_k=c et
   $w_{n,k}$ / $y_k = 0$, un classifieur binaire $g^c_{j-HoG/Haar}$ dont l'erreur est définie par :
   $$\epsilon^c_{j-HoG/Haar} = \sum_{i=1}^{N} \omega_i |g^c_{j-HoG/Haar}(x_i) - y'_i|$$
   tel que $c = \{1,2,3,123\}$ et $y'_i = 0$ si $y_i = 0$ et $y'_i = 1$ sinon
   Choisir le classifieur $g^n_{t-HoG/Haar}$ avec l'erreur ϵ_t la plus faible
   Actualiser les poids : $\omega_{t+1,i} = \omega_{t,i}\beta_t^{1-e_i}$
   où $e_i = 0$ si $g_t(x_i) = y_i$, $e_i = 1$ sinon,
   avec $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
   Ajouter $g^n_{t-HoG/Haar}$ dans le classifieur fort G_n
   Décrémenter le seuil de G_n jusqu'à atteindre DC_{min}
   dans la base de validation
   Évaluer G_n dans la base des négatifs pour obtenir f
   $t = t + 1$

TABLE 5.10 – Pseudo code de l'apprentissage du détecteur Multi-Modèle.

spécialisés dans la réponse à une classe. Nous pouvons définir de nouvelles fonctions faibles g^n_{Haar} avec $n = [1;2;3;123]$.

La nature de l'algorithme Adaboost contribue au groupement des classes dans les sous-espaces. D'abord, il trouve les caractéristiques communes entre les exemples d'apprentissage positifs. Même dans les cas où il existe un ensemble de ces caractéristiques (ou des fonctions faibles) par classes, l'ensemble total, réunit dans le classifieur fort, va à valider les exemples comme positifs. D'autre part, la distribution des poids, lui permet de se concentrer sur des exemples de classes qui ne sont assez représentées dans le classifieur fort en lui donnant un poids plus important s'ils sont systématiquement mal classifiés. Idéalement, nous pouvons supposer qu'il se produit une sélection des fonctions faibles, qui passe successivement d'une classe à l'autre.

A partir de ce détecteur, nous obtenons les vecteurs de descripteurs f_i de chaque VT que nous classifions à l'aide du réseau neuronal (voir figure 5.13). Les résultats sont montrés dans le tableau 5.11.

Classifieur Multi-Modèle	
Micro-précision	88.23
Macro-précision	84.68

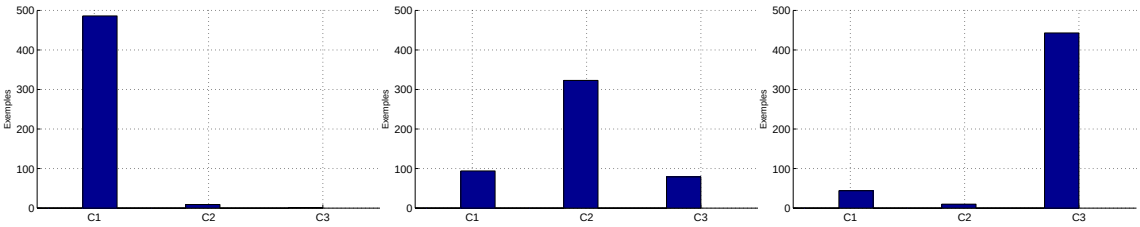
Matrice de confusion

	Nb véhicules				Taux		
	c1	c2	c3		c1	c2	c3
c1	1021	31	54	→	92.3	2.8	4.9
c2	49	196	33	→	17.6	70.5	11.9
c3	5	7	125	→	3.6	5.1	91.2

TABLE 5.11 – Performance du classifieur Multi-Modèle.

Vérification du groupement

Pour vérifier le groupement par classes dans l'espace complet, nous avons utilisé la librairie CLUTO [2] (pour Clustering Toolbox) qui permet de réaliser le regroupement des classes dans de grands espaces des données. Les paramètres qu'il faut assigner à cette librairie sont le nombre de classes désirée, la méthode de classification ainsi que toutes les données des exemples à regrouper. Les vecteurs des données sont représentés par les f_i, binaires, sortie des fonctions faibles de la cascade. Nous avons choisi d'utiliser une méthode non linéaire de regroupement basée sur les graphes. Les vecteurs de données proviennent de 500 exemples de chaque classe et chacun de ces éléments prend la valeur 1 si la fonctionne de classification faible est validée et 0 sinon. Le résultat obtenu est illustré par la figure 5.14, ou chaque histogramme montre le nombre des exemples assignés par classe. Ce résultat, ainsi que les résultats sur la vérité terrain sont très encourageants et valident notre hypothèse.



Classe 1 Classe 2 Classe 3

FIGURE 5.14 – Regroupement obtenu lors de l'utilisation de la librairie CLUTO.

5.4.5 Analyse Comparative

Le tableau 5.12 montre le récapitulatif des résultats des architectures sur la vérité terrain.

2. http ://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Architecture	Micro-précision	Macro-précision
Détecteurs individuels	79.6	68.3
Classifieur Pyramidal	66.6	61.7
Classifieurs Parallèle	**91.5**	82.9
Classifieur Multi-Modèle	88.2	**84.6**

TABLE 5.12 – Récapitulatif des résultats des architectures de classifications sur la VT.

Les deux derniers classifieurs obtiennent les meilleurs résultas avec un faible écart entre eux.

Les détecteurs individuels obtiennent un score faible mais assez respectable, étant donné qu'ils n'ont pas été conçus pour réaliser une discrimination inter-classes. Le plus bas score obtenu est celui du classifieur pyramidal, étant donné que beaucoup des véhicules ont été rejetés par les détecteurs avant de finaliser la classification. Les classifieurs en parallèle obtiennent la micro-précision la plus importante, notamment pour les bonnes classifications effectuées pour la classe 1. Cependant, les résultats pour la classe 3 ne sont pas les meilleurs. L'utilisation d'une décision à l'aide d'un réseau neuronal a permis de mieux séparer ces exemples, au détriment des exemples de la classe 2. Finalement, le classifieur Multi-Modèle obtient la macro-précision la plus importante, notamment à cause d'une discrimination plus importante entre les classes 1 et 3.

Les résultats précédents ont montré comment serait une classification à partir d'une détection idéale. Il reste à évaluer les architectures complètes, en ajoutant l'étape de la détection. Pour filtrer les détections avant d'appliquer la classification, nous allons utiliser deux approches que nous présentons dans la section suivante ainsi que les résultats finals obtenus.

5.5 Filtrage et cadrage des détections

Dans cette section nous allons évaluer deux méthodes pour encadrer une détection à partir des hypothèses fournies par le détecteur.

La sortie d'un détecteur est une liste de rectangles qui représentent des positions dans l'image validés comme véhicules possibles (voir figure 5.15). Ces rectangles peuvent être de différentes tailles et sont, en général, placés dans le voisinage de la vrai position du véhicule. Le problème que nous traitons dans cette section consiste à grouper les nuages d'hypothèses avant l'application d'un classifieur.

Nous présentons par la suite deux méthodes : la méthode utilisée dans la librairie *OpenCV* pour la détection des objets et une méthode basée sur l'algorithme *Mean Shift*.

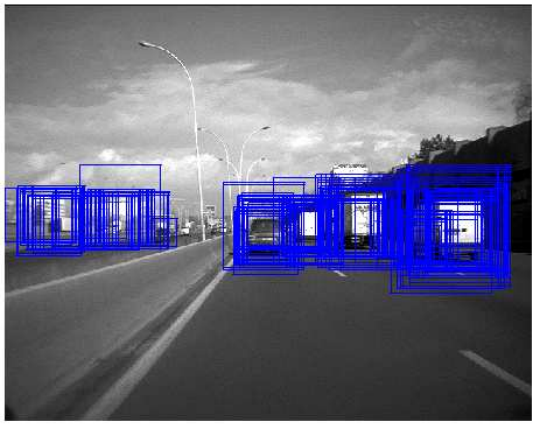
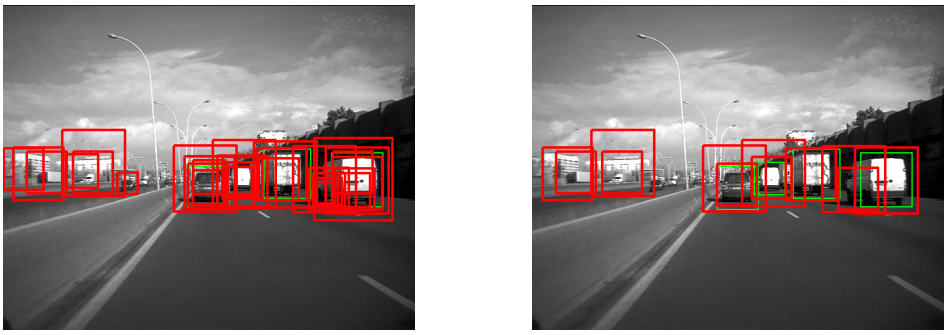


FIGURE 5.15 – Hypothèses données pour le détecteur.

5.5.1 Méthode *OpenCV*

La méthode utilisée dans le code de la librairie *OpenCV* est divisée en deux étapes. La première calcule une moyenne des hypothèses qui se trouvent suffisamment proches. L'algorithme valide comme voisins un couple de rectangles, si le deuxième se trouve à une distance inférieure à 20 % de la taille du premier et si leurs tailles ont une différence inférieure à 20 %. En regroupant tous les rectangles voisins, le rectangle moyen est trouvé en calculant la moyenne des positions et des tailles de l'ensemble. Dans la figure 5.16 (a) ces moyennes sont représentés par les rectangles rouges et les rectangles verts montrent la vérité terrain.



(a) (b)

FIGURE 5.16 – Les deux étapes d'encadrement de la méthode de *OpenCV*.
(a) calcul des moyennes, (b) filtrage

La deuxième étape est un filtrage des fenêtres pour éliminer celles qui se trouvent contenues dans d'autres à partir des critères de voisinage. Nous pouvons constater dans la figure 5.16 (b) le résultat final pour cette méthode. Il existe encore un nombre important

des fenêtres à évaluer avant l'étape de classification.

Nous allons proposer une autre méthode qui utilise une approche différente pour le calcul des positions moyennes des détections et qui permet de diminuer le nombre de fenêtres restantes.

5.5.2 *Mean Shift* pour l'encadrement des hypothèses

Nous allons exposer par la suite, une méthode qui utilise l'algorithme *Mean Shift* [32] pour calculer les positions moyennes des détections.

Les rectangles de la figure 5.15 sont réprésentés différement, à partir de leurs points centrales, comme nous pouvons l'apprécier dans la figure 5.17. Ces points sont ensuite placés dans un espace 3D, où la troisième dimension est l'axe correspondant à la taille de chaque rectangle.

FIGURE 5.17 – Points qui représentent les hypothèses donnés pour le détecteur.

Pour normaliser les valeurs de la troisième dimension, nous utilisons la formule suivante :

$$z = \frac{log(s/32)}{log(s)}$$

où s est le facteur d'échelle utilisé dans le test, qui multiplie la valeur 32 qui corresponde à la plus petite taille de fenêtre de recherche dans l'image. Nous effectuons cette opération afin de rapprocher les points correspondants aux rectangles des échelles supérieures aux points correspondants des échelles inférieures. Par exemple, la taille de rectangle 98 pixels

[32] D. Comaniciu and P. Meer. Mean shift : A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5) :603–619, 2002.

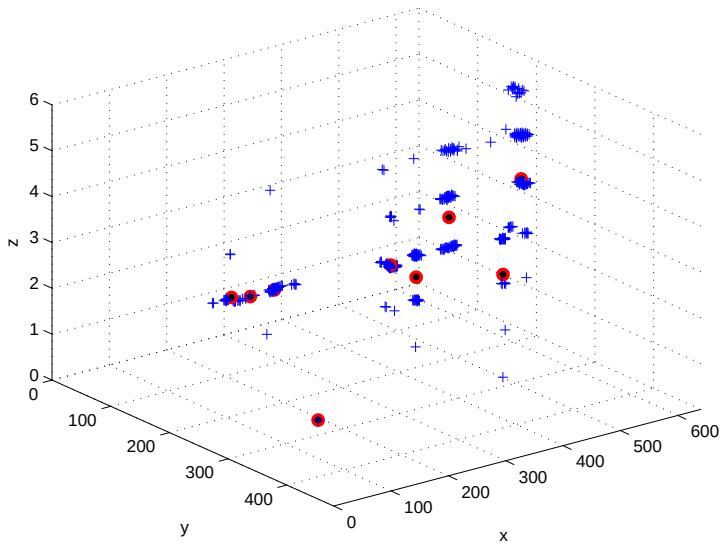


FIGURE 5.18 – Centres des nuages de points trouvés avec la méthode de *Mean Shift*.

est succédée par la taille 122 pixels (pour un facteur d'échelle 1.25). Cette séparation de 24 pixels n'est pas souhaité dans le calcul de *Mean Shift* étant donné que l'idée est regrouper ces deux fenêtres, si elles ont le centre coïncident.

Le *Mean Shift* est un algorithme qui calcule les moyennes pondérées des positions de façon itérative, jusqu'à convergence. Étant donnée que les rectangles de grande taille ont une influence trop importante dans le calcul de la position moyenne de la détection, surtout quand ils sont mélangés avec d'autres de plus petite taille, nous avons introduit un vecteur de pondération. Ce vecteur $\mathbf{w}$ correspond à l'inverse de la valeur de la taille de chaque rectangle au carré et va sanctionner ainsi la différence de taille dans un ensemble.

Le paramètre utilisé pour ce *Mean Shift* est une distance minimale de voisinage D (dans nos travaux, $D = 20$ pixels nous a donné les meilleurs résultats).

Le pseudo-code suivant décrit la méthode de *Mean Shift* pour le calcul des positions moyennes :

1. Soient N points $p_i = (x, y, z)$ dans l'espace 3D
2. Soit un vecteur de poids $\mathbf{w} = (w_i)$ où chaque w_i est calculé à partir de l'échelle de p_i.
3. Soit un vecteur T de taille N et $T(i) = 0$, $i = 1, ..., N$, qui indique si un point a été visité
4. Tant qu'il existe un p_i tel que $T(i) = 0$
   (a) Choisir un point p_j tel que $T(j) = 0$
   (b) Définir la position moyenne comme étant $p_m^t = p_j$
   (c) Trouver l'ensemble des points p_i qui se trouvent à une distance euclidienne de p_m^t plus petite que le seuil D

$$|p_m^t - p_i| < D, \forall i$$

(d) Calculer la nouvelle position moyenne p_m^{t+1}, pondérée par $\mathbf{w}$, de l'ensemble

(e) Si, $|p_m^{t+1} - p_m^t| > \epsilon_C$, $p_m^t = p_m^{t+1}$ et nous répétons (c) et (d).

(f) $T(i) = 1 \; \forall \; i$ de l'ensemble de la moyenne

(g) La moyenne p_m^{t+1} est conservée dans la liste des hypothèses.

Le paramètre ϵ_C est une constante de valeur faible qui indique que la position moyenne est arrivée à convergence.

Les points rouges de la figure 5.18 marquent la position des centres trouvés par *Mean Shift* pour cet exemple. Nous récupérons la position de la moyenne, des valeurs (x, y) de chaque point obtenu par *Mean Shift* et sa taille à partir de la troisième dimension. Dans la figure 5.19 nous observons les points de la méthode *Mean Shift* convertis en rectangles. Le nombre final de détections par véhicule est plus faible que celui qu'avait obtenu la méthode *OpenCV*, surtout les grandes fenêtres qui ont été absorbés par les positions moyennes. Cependant, ce bénéficie entraîne aussi un glissement de la détection finale par rapport à la vérité terrain (voir notamment l'exemple du camion).

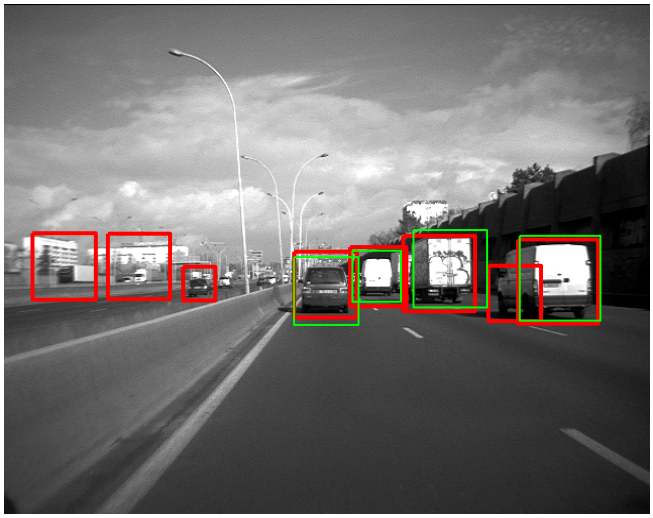


FIGURE 5.19 – Résultats de l'encadrement à partir de *Mean Shift* sur la vérité terrain.

Cette figure montre aussi quatre détections supplémentaires qui correspondent aux fausses alarmes, aussi présents avec *OpenCV*.

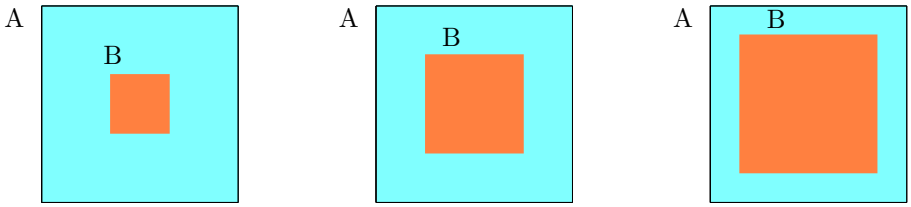
5.5.3 Comparaison des méthodes

Les rectangles obtenus par les détecteurs sont considérés comme étant des détections correctes s'ils remplissent un critère de recouvrement sur la vérité terrain. Ce critère, entre un rectangle de la vérité terrain A et un rectangle B issue de la détection, est donné par

[147] :

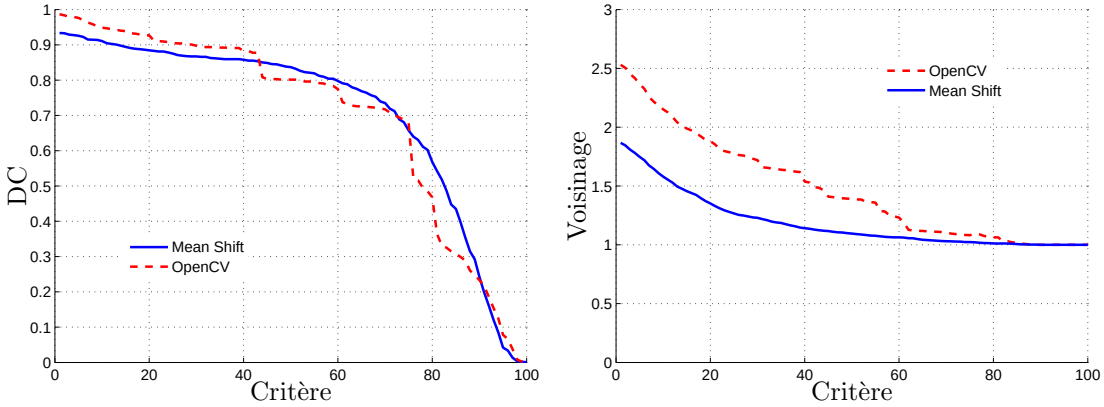
$$CR = 2\frac{A \cap B}{A + B}$$

où $A \cap B$ est la surface d'intersection entre A et B, et $A + B$ est la somme des surfaces de deux rectangles. La figure 5.20 montre trois exemples de recouvrement.



critère égal à 0.46 critère égal à 0.66 critère égal à 0.82

FIGURE 5.20 – Exemples de différents recouvrements.



(a) (b)

FIGURE 5.21 – Courbes comparatives entre la méthode OpenCV et le *Mean Shift*.

La courbe (a) de la figure 5.21 montre le taux de détections correctes obtenu par les méthodes en faisant varier la valeur du critère. Globalement, la méthode *Mean Shift* obtient de meilleurs résultats que la méthode de OpenCV. A partir de une valeur de 0.9, les courbes sont coïncidentes.

La courbe (b) de la figure, affiche la moyenne du nombre de fenêtres dans un voisinage de la vérité terrain en fonction de la valeur de recouvrement. La valeur de départ de cette courbe, pour un critère de recouvrement légèrement supérieure à zéro, correspond à la quantité totale (en moyenne) de détections qui intersectent le rectangle de la vérité terrain.

Pour une valeur du critère de recouvrement égal à 0.66, la méthode *OpenCV* a presque 1.5 détections par vérité terrain. Ce qui montre que deux hypothèses avec un bon critère de

[147] D. Withopf and B. Jahne. Improved training algorithm for tree-like classifiers and its application to vehicle detection. In *IEEE Intelligent Transportation Systems Conference*, pages 642–647, Septembre 2007.

recouvrement (qui signifie qu'elles sont très proches) n'ont pas été réunis par la méthode. Cela vérifie les résultats des figures 5.16 et 5.19 qui montrent un nombre plus important d'hypothèses par rapport à *Mean Shift*. Le résultat du *Mean Shift* semble plus efficace pour réaliser la classification, étant donné qu'il minimise les rectangles de sortie.

Par la suite, les résultats finaux de détection, issues des deux méthodes de regroupement, seront utilisés pour la classification du type de véhicule.

5.5.4 Analyses comparatives

Nous allons appliquer, dans un premier temps, la méthode *OpenCV* et la méthode de *Mean Shift* pour filtrer sur les détections obtenues à partir des quatre architectures précédemment proposées. Ensuite, les rectangles moyens obtenus sont évalués dans chaque architecture pour réaliser la classification du type de véhicule. La performance des méthodes de filtrage et des architecture de détection puis classification est présentée sur deux tableaux : le tableau 5.13 montre les résultats de l'étape de détection et dans le tableau 5.14sont développés les résultats de la classification. Nous avons choisi de calculer les résultats de détection présentés en utilisant un critère de recouvrement supérieure ou égal à 0.66. La performance en classification est mesurée en fonction de la micro précision et la macro précision, calculée sur les détections qui remplissent le critère de recouvrement sur la vérité terrain. Dans l'annexe B, le lecteur intéressé peut trouver les courbes de variations de ces résultats en fonction du critère.

Le premier tableau 5.13 montre les taux de détections pour chaque architecture de classification et chaque méthode, *OpenCV* et *Mean Shift*, ainsi que la détection sans filtrer les hypothèses Dans la première colonne du tableau 5.13, nous avons placé le taux de détections correctes (DC) de chaque architecture. La deuxième colonne représente le nombre total de fausses alarmes (FA) obtenues dans la base de test. Enfin, la colonne Det/VT indique le nombre des détections par véhicule (ou vérité terrain) qui remplissent le critère de recouvrement supérieure ou égal à 0.66.

Le premier fait que nous constatons de l'analyse de résultats est un taux de DC sur les détections regroupés plus faible que le taux calculé sur les détections sans filtrage. Nous avons donc perdu des détections pendant les calculs des rectangles moyens. Les raisons sont diverses, mais nous pouvons citer, par exemple le cas d'une détection bien positionnée, mais correspondante à un rectangle avec une taille plus grande que la vérité terrain, qui fait chuter le taux de recouvrement. Effectivement, le nuage de points correspondants aux centres de rectangles peut être bien positionné par rapport à la vérité terrain. Cependant, il peut exister de rectangles de grande taille parmi eux, qui ont une influence importante dans le calcul de la taille moyenne de la détection. La méthode *Mean Shift* essaie de minimiser cet effet à partir du vecteur de poids qui dépend de la taille des rectangles, et nous vérifions dans le tableau une meilleur performance de cette méthode à partir des valeurs de Det/VT. La méthode *OpenCV* n'a pas la même dynamique et élimine ces

	DC	FA	Det/VT
	Détecteurs individuels		
Sans filtrage	91.5	26739	19.7
Mean Shift	81.9	2734	1.5
OpenCV	80.4	3317	1.28
	Classifieur Pyramidal		
Sans filtrage	99	180079	68.62
Mean Shift	76.7	5692	1,03
OpenCV	77.1	5502	1,14
	Classifieurs en Parallèle		
Sans filtrage	96.9	89401	50.3
Mean Shift	76.4	2689	1.04
OpenCV	72.4	3733	1,11
	Classifieur Multi-Modèle		
Sans filtrage	98.8	193702	81.5
Mean Shift	48.9	4431	1
OpenCV	75.5	5342	1,15

TABLE 5.13 – Comportement en détection des quatre architectures proposées.

rectangles lors du filtrage des petits rectangles contenus dans des grands (ou à l'inverse à partir du nombre de voisinage). Un effet similaire est obtenu si le rectangle moyen est de plus petit taille que la vérité terrain. D'où l'intérêt de minimiser le nombre des hypothèses finales du détecteur pour qu'il discrimine le mieux possible la classe véhicule de la classe non-véhicule.

Les détecteurs individuels se montrent ainsi très discriminants avec un faible nombre des hypothèses par détection et de fausses alarmes sans filtrage. Avoir été entraînés pour une seule classe contre les négatifs, lui permet d'éliminer un grand nombre de ces dernières. Le nombre important de détections par vérité terrain pour les méthodes de filtrage répond à la méthode que nous avons utilisée pour la classification. Les hypothèses issues de chaque détecteur D_i sont filtrées, dans un premier temps, à partir de *OpenCV* ou *Mean Shift*. Dans une deuxième étape, les trois ensembles de rectangles moyens sont réunis et un nouveau filtrage élimine des détections coïncidentes pour garder celle qui a la sortie du classifieur la plus importante. Cependant, il y a toujours de détections qui ne sont pas exactement coïncidents et qui sont prises en compte pour la classification. Évidement, dans ces cas, elles font augmenter le taux Det/VT.

Le classifieur pyramidal obtient un nombre de fausses alarmes six fois plus grand que l'architecture précédente et quatre fois plus de détections par vérité terrain. La méthode *Mean Shift* minimise le nombre de fenêtres par vérité terrain pour ce critère de recouvrement. Le même résultat est obtenu dans le cas de l'architecture en parallèle. La quantité de fausses alarmes est plus faible pour cette architecture. Le détecteur de toutes les classes

	Clf/VT	**Micro-P**	**Macro-P**
	Détecteurs Individuels		
Sans filtrage	87	90.2	85
Mean Shift	66.7	58.6	55.1
OpenCV	75.1	69.8	62.7
	Classifieur Pyramidale		
Sans filtrage	77.8	79	70.9
Mean Shift	82	82.1	78.8
OpenCV	80.8	79.9	76.9
	Classifieurs en Parallèle		
Sans filtrage	77.4	75	71.5
Mean Shift	77.3	76.7	75.2
OpenCV	83.7	83.4	82.8
	Classifieur Multi-Modèle		
Sans filtrage	68.9	68.3	62.6
Mean Shift	63.7	63.7	61.1
OpenCV	74.3	74.5	71.1

TABLE 5.14 – Comportement en classification sur les hypothèses générés pendant la détection, avec un critère de recouvrement supérieure ou égal à 0.66

.

arrive à éliminer un nombre plus important de négatifs que l'architecture pyramidale. L'architecture qui nous donne le plus grand nombre de FA et de Det/VT (sans regrouper) est celle du classifieur multi-modèle. Comme nous l'avons expliqué précédemment, nous allons voir par la suite que cela nuit à la classification.

Le tableau 5.14 présente les résultats de classification par architectures pour les détections qui remplissent un critère de recouvrement supérieure ou égal à 0.66.

La première colonne Clf/VT calcule le taux de classifications correctes par véhicule entre toutes ces détections. En pratique, s'il y a deux détections pour un véhicule nous les classifions et comparons le résultat avec la classe du véhicule. Si les deux détections ont été bien classifiées le taux est égal à 100 % et si une seule est bien classifiée, le taux est 50 %. Ce paramètre peut représenter une mesure sur la qualité des détections sur un véhicule. Enfin, les deux dernières colonnes montrent la micro précision et la macro précision.

Le classifieur basé sur les Détecteurs Individuels obtient la meilleure performance en la Micro et Macro Précision pour les détections sans filtrage. Les Détecteurs ont définit un rectangle englobant qui est plus similaire aux vignettes qui ont servit d'apprentissage, par rapport aux rectangles utilisés pour englober la vérité terrain. Cependant, les taux de Micro et Macro Précision ont chuté une fois appliquées les méthodes de filtrage. Nous pouvons constater que le nombre de Det/VT reste important pour les deux méthodes et

cette quantité diversifie les résultats des classifications et fait descendre le taux Clf/VT.

Le classifieur pyramidal améliore les taux de Micro et Macro Précision. Pourtant, il a six fois plus de fausses alarmes et quatre fois plus Det/VT que l'architecture précedente. Cependant, ces détections se montrent d'une bonne précision quand nous analysons les valeurs des Clf/VT des méthodes de groupement. Le *Mean Shift*, obtient un bon taux de classifications dans la Micro et la Macro Précision qui sont, en effet, les meilleurs scores parmi les architectures.

Pour les classifieurs en parallèle la méthode de *OpenCV* montre une meilleur performance à celle de *Mean Shift*. Nous observons une baisse dans le taux Micro Précision mais le taux de Macro Précision est similaire aux aux valeurs obtenues sur la vérité terrain.

Nous allons analyser plus en détails le dernier classifieur étant donné les mauvais résultats obtenus. Nous constatons, d'abord, l'énorme quantité des FA sans regroupement. Toutefois, cette quantité, ainsi que le nombre de Det/VT est semblable au Classifieur en Pyramide. La différence réside dans la qualité des détections, mesurée à partir du taux Clf/VT, où nous perdons 10 %. Cela se voit refléter dans les taux de Micro et Macro Précision qui chutent d'un 10 %. Cependant, la Det/VT du *Mean Shift* est celle qui a la plus petite valeur avec une fenêtre par VT. Néanmoins, cette détection n'est pas de bonne qualité car le Clf/VT est très bas. Nous pouvons apprécier dans les deux exemples de cette architecture montrés dans la figure 5.22, que les hypothèses moyennes sont plus petites que la vérité terrain (les voitures) et elles sont finalement considérées comme étant fausses alarmes.

5.6 Bilan

Nous avons examiné dans ce chapitre différentes architectures de classification pour différencier les véhicules en trois classes : véhicule de tourisme, utilitaires et camions.

Dans un premier temps, les quatre architectures ont été évaluées sur la vérité terrain de la base de test, sans faire appel au détecteur. Ces tests consistent à extraire les rectangles englobant les véhicules, définis par l'utilisateur, pour les classifier. Globalement, les classifieurs obtenus à partir de l'algorithme Adaboost arrivent à bien discriminer les classes, soit avec un classifieur simple, soit avec une architecture en cascade.

Nous avons ensuite combiné l'opération de détection à celle de classification : les hypothèses issues de la détection ont été filtrées à l'aide de deux méthodes de filtrage qui cherchent a trouver le meilleur rectangle englobant le véhicule. La première méthode est celle employée par *OpenCV* pour la détection d'objets et la deuxième utilise l'algorithme *Mean Shift*. Le filtrage de *OpenCV* obtient, en général, de meilleurs résultats de classification que l'autre méthode mais au prix d'un nombre plus important de fausses alarmes. En effet, cette méthode réunit moins bien les hypothèses, notamment de grande taille avec

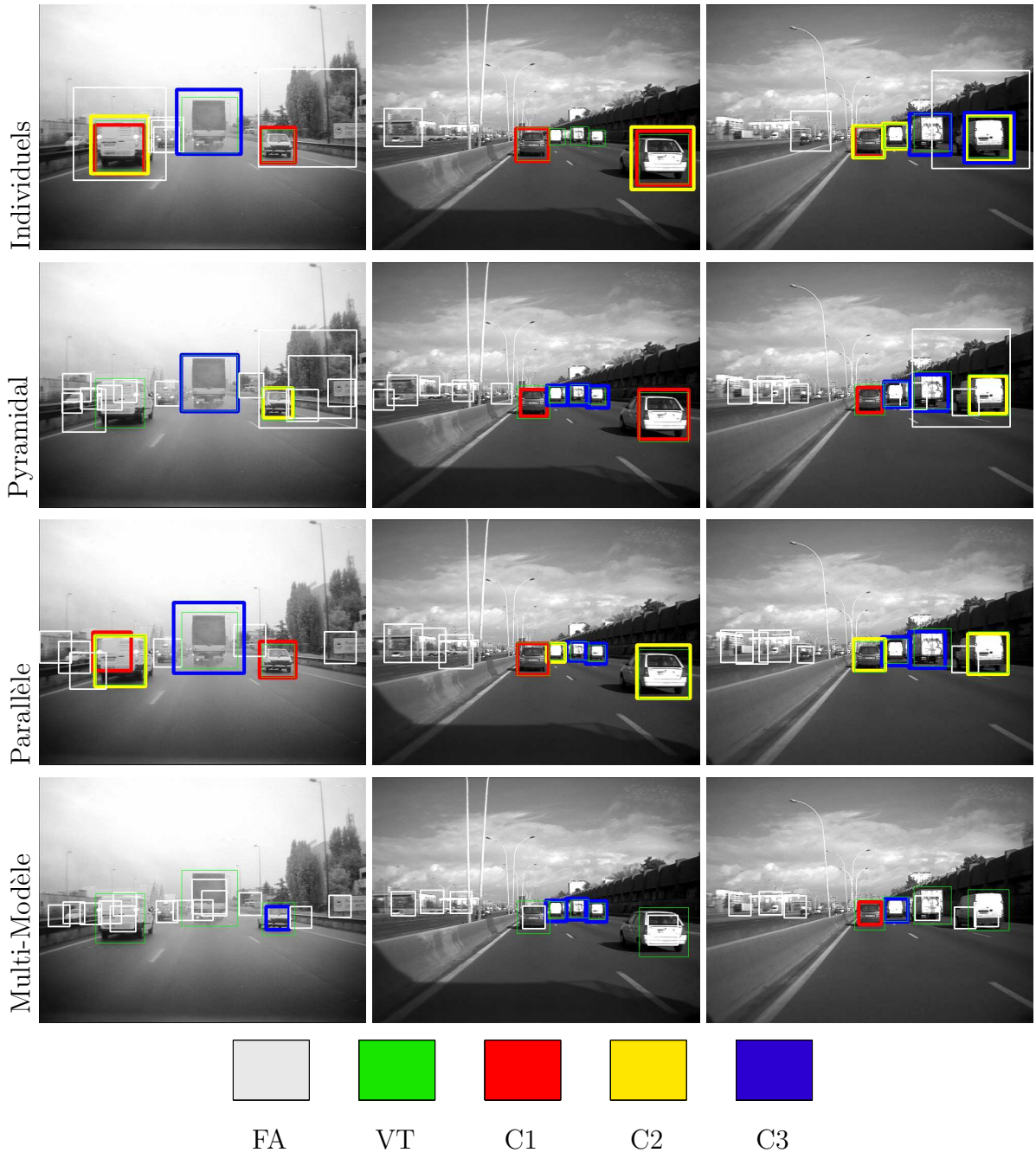


FIGURE 5.22 – Exemples de résultats des quatre architectures et filtrage à partir de *Mean Shift*.

celles de taille plus petite. De son coté, le filtrage de *Mean Shift* arrive à regrouper ce type de cas, mais en même temps, cela peut devenir une désavantage étant donné que les boites englobantes trouvées ont une taille plus grande (ou plus petite) que le véhicule. Malheureusement, si l'hypothèse moyenne a un mauvais cadrage les résultats de classification chutent drastiquement.

Les détecteurs individuels obtiennent de bons résultats sur le taux de détections et de fausses alarmes. Les hypothèses après le filtrage sont bien encadrées mais la méthode de classification ne considère pas une discrimination entre les classes et ces résultats sont

donc moins bons.

L'architecture pyramidale, ainsi que l'architecture en parallèle ont des résultats acceptables pour la détection, et une meilleur discrimination entre les classes, ce qui est logique puisqu'ils sont entraînés à discriminer.

La méthode de classification qui utilise l'espace de projection obtenu à partir d'un apprentissage de type Adaboost multi-modèle obtient des résultats satisfaisants sur la vérité terrain. Elle a un potentiel intéressant mais il faut encore améliorer l'étape de détection.

Chapitre 6

Conclusions

Nous avons présenté dans cette partie un algorithme de reconnaissance de formes, appliqué à la détection embarquée de véhicules en utilisant une approche similaire à celle de Viola et Jones. Différentes architectures de détecteurs en cascade ont aussi été proposées pour réaliser une classification des véhicules détectés.

6.1 Conclusions sur la Détection

Deux espaces de paramètres, les descripteurs de Haar et les histogrammes de gradient orienté, ont été utilisés pour la construction de détecteurs de véhicules en cascade. Les premiers sont associés à des classifieurs *faibles* de type discriminant et les seconds à des classifieurs de type génératif. Un troisième détecteur est obtenu à partir d'une fusion de ces deux espaces.

Dans les premiers chapitres, nous avons étudié le comportement de différentes architectures de détection : le détecteur simple et le détecteur en cascade. Pour optimiser la performance du détecteur en cascade, nous avons fixé le nombre maximum des descripteurs dans chaque étage de la cascade, c'est-à-dire dans chaque fonction de classification *forte*.

Le détecteur réalisant la concaténation des deux familles de descripteurs combine les avantages des précédents détecteurs dits de Haar et de HoG : un taux de détections correctes élevé et un faible nombre de fausses alarmes. L'architecture de ce détecteur, obtenue automatiquement à l'aide de la méthode de dopage, utilise en premier les classifieurs de type génératif pour éliminer les échantillons négatifs éloignés du modèle, puis il se sert des descripteurs discriminants pour dessiner des frontières nettes entre les exemples positifs et ceux négatifs proches, encore présents. Ce comportement correspond aux deux étapes de la détection des véhicules, où en premier lieu, une connaissance *a priori* d'une voiture permet de se concentrer rapidement sur des zones candidates pour ensuite, éliminer plus finement les fausses.

Nous avons aussi évalué le comportement du détecteur par scénarios : urbain versus suburbain, et cas nominaux versus non-nominaux (éclairage faible et conditions adverses).

Parmi les travaux qui pourraient être mis en place par la suite nous pouvons citer l'implémentation d'autres combinaisons de descripteurs. Dans cette étude, nous n'avons pas testé tous les descripteurs possibles pour la détection de véhicules. L'utilisation d'autres familles de descripteurs pourraient être pertinente. Plusieurs autres travaux dans la littérature [58, 132] montrent en effet que la combinaison de différentes familles peut améliorer les performances (taux de détection et de fausses alarmes).

La méthode de génération du modèle utilisée pour les HoG peut être améliorée, ainsi que le calcul de la distance. Notre modèle consiste en une médiane unimodale. Il pourrait être intéressant d'avoir une approche multi-modale, avec plusieurs centres. Par exemple, la méthode *coarse-to-fine* de Li, fait penser à une discrimination des véhicules à partir de la vue arrière. Ces vues changent si le véhicule obstacle se trouve exactement en face du véhicule porteur de la caméra ou bien s'il est possible d'apercevoir le coté de la voiture, dans les autres cas (situation de dépassement, de virage ou sur une autoroute à plusieurs voies). Nous proposons une architecture pyramidale qui serait spécialisée sur ces différentes vues ayant un modèle pour chacune. Ce type de détecteur pourrait améliorer le taux de détections, ainsi que la classification ultérieure. Le point faible est le besoin d'une base d'apprentissage pertinente avec un nombre suffisant d'exemples pour chaque cas. De même, la distance utilisée est celle de Bhattacharyya, d'autres distances entre histogrammes existent et demanderaient à être testées [24].

Enfin, l'utilisation d'un algorithme de suivi dans une séquence améliorerait les taux de détections en même temps qu'il éliminerait les fausses alarmes. En effet, il n'est pas nécessaire de détecter les véhicules dans toutes les images de la séquence, pourvu qu'ils soient tous détectés au moins une fois et relativement rapidement (afin d'initialiser un algorithme de suivi).

6.2 Conclusions sur la Classification

La classification de véhicules est abordée dans cette partie en utilisant quatre architectures de détecteurs en cascade. Nous avons évalué ces différentes architectures sur la vérité terrain puis sur les hypothèses fournies par l'étape de détection. Deux méthodes de filtrage des hypothèses sont utilisées à la suite de l'étape de détection. Nous analysons l'influence de cette étape dans une étude comparatif des résultats obtenus dans la classification.

---

[58] D. Geronimo, A. Lopez, D. Ponsa, and A.D. Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 418–425, 2007.

[132] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2) :125–137, Juin 2005.

[24] S.H. Cha and S. N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6) :1355–1370, 2002.

Nous constatons dans ce chapitre qu'il est difficile de détecter différentes classes d'objets à l'aide d'un seul détecteur. Le principal inconvénient réside dans le grand nombre des hypothèses générées qui a une incidence importante sur le résultat de la méthode de filtrage (rectangle englobant mal le véhicule qui est par conséquent mal classé). Nous proposons d'utiliser une architecture de type *Nested* [69], qui permet d'éliminer un nombre plus important de fausses alarmes, sans perte des détections positives.

L'utilisation de l'Adaboost Réel peut s'avérer intéressant pour les méthodes de classification qui emploient un réseau neuronal, par exemple le classifieur multi-modèle. En effet, les sorties réelles des fonctions faibles nommées "critères de confiance" dans [126], sont une information plus riche que les sorties binaires de l'Adaboost Discret.

Enfin, la classification multi-modèle peut être combinée avec les autres architectures, pyramidale ou en parallèle, afin d'utiliser ce nouvel espace de classification pour discriminer entre les classes. Effectivement, ces architectures sont compatibles. Il suffirait de mettre à la suite du détecteur multi-modèle l'une ou l'autre. Le vecteur soumis au réseau de neurones serait alors augmenté des sorties des classifieurs faibles de la partie ajoutée. Ces informations supplémentaires seraient d'autant plus pertinentes que les classifieurs en question ont été appris en concurrence (une classe contre les autres).

[69] C. Huang, H. Ai, B. Wu, and S. Lao. Boosting nested cascade detector for multi-view face detection. In *IEEE International Conference on Pattern Recognition*, volume 2, pages 415–418, 2004.

[126] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3) :297–336, 1999.

Deuxième partie

Projet LRPeditor - Méthode de votes pour la reconnaissance du type de véhicule

Introduction

La lecture automatique de plaques d'immatriculation (LAPI) est largement utilisée actuellement pour l'identification des véhicules. Ces systèmes, conçus d'avantage pour des applications concernant la sécurité peuvent être installés soit dans des postes fixes pour le contrôle d'accès aux parkings publics, routes privées, passages de frontière, station-services, etc., soit dans des postes mobiles sur des voitures des services spéciaux de sécurité et de police. Les systèmes ont comme objectif de lire la plaque d'immatriculation pour la comparer à une base de données de clients, dans le premier cas ou des voitures recherchées, dans le deuxième.

Une autre application envisagée concerne la gestion de la circulation. Un système LAPI peut suivre un véhicule dans une ville pour optimiser le trafic local, par exemple, à partir d'un réglage adaptatif du temps vert de feux rouges. Cette tâche peut être réalisée d'une façon *off-line*, à travers d'une analyse des données statistiques ou *on-line* à travers d'une reconnaissance en temps réel.

En pratique, les systèmes LAPI doivent faire face à un grand nombre de difficultés : mauvaise qualité de l'image, mauvais éclairage ou présence des reflets, occultations, fausses plaques, etc. Dans certains cas, si le système n'est pas capable de lire la plaque d'immatriculation, il peut attirer l'attention d'un opérateur humain.

Il serait aussi possible de faire des validations avec une information partielle sur la base de clients (dans les cas d'un nombre limité d'utilisateurs) ou d'utiliser d'autres détails du véhicule, à travers d'un système du type VMMR (*Vehicle Make and Model Recognition*), pour affiner la recherche.

Cette dernière solution a attiré l'attention des entreprises qui se dédient à l'installation des systèmes LAPI dans des parkings publics et privés : LPREditor [1], Eyedea [2], AsiaVision [3], etc. La combinaison d'un système LAPI et d'un système VMMR, accroît la robustesse de l'ensemble et permet de valider les réponses individuelles, et évite certaines fraudes, telles que les changements des plaques minéralogiques.

Dans cette deuxième partie du mémoire, nous allons proposer un système de reconnaissance du type de véhicule (marque et modèle) à partir de l'information fournie par une caméra.

1. http ://www.lpreditor.com
2. http ://www.eyedea.cz
3. http ://www.asiavision.com.hk/

Chapitre 1

Reconnaissance de véhicules : État de l'art

Les systèmes de vision pour la reconnaissance de véhicules font partie des Systèmes de Transports Intelligents. Trois de ses principales applications doivent être distinguées.

La première concerne les caméras embarquées, dédiées à la détection des véhicules dits obstacles se présentant au devant du véhicule équipé, qui a été déjà présentée dans le chapitre précédent. Les véhicules localisés peuvent être classifiés en : voitures de tourisme, camions, moto, etc. [78, 81, 55].

La deuxième application vise la télésurveillance des autoroutes et d'autres voies de circulation. Celle-ci permet une meilleur gestion du trafic routier, et alerte les services concernés en cas d'incidents ou d'accidents[42]. Un modèle géométrique peut être utilisé pour classer le véhicule en catégories (véhicules de tourisme, véhicules utilitaires, véhicules lourds, etc.) grâce aux modèles paramétriques ou déformables, 2D ou 3D [65, 63, 149, 47,

---

[78] T. Kalinke, C. Tzomakas, and W. v. Seelen. A texture-based object detection and an adaptive model-based classification. In *IEEE International Conference on Intelligent Vehicles 1998*, volume 1, pages 143–148, 1998.

[81] T. Kato, Y. Ninomiya, and I. Masaki. Preceding vehicle recognition based on learning from sample images. *IEEE Transactions on Intelligent Transportations Systems*, 3(4) :252–260, Decembre 2002.

[55] T. Gandhi and M. M. Trivedi. Video based surround vehicle detection, classification and logging from moving platforms : Issues and approaches. In *IEEE Intelligent Vehicles Symposium*, pages 1067–1071, Istambul, Juin 2007.

[42] J. Douret and R. Benosman. A multi-cameras 3d volumetric method for outdoor scenes : a road traffic monitoring application. In *IAPR International Conference on Pattern Recognition*, pages III : 334–337, 2004.

43].

Enfin, la troisième s'adresse aux systèmes ayant pour objet d'identifier les véhicules, à travers l'utilisation des systèmes LAPI et VMMR. Par la suite, nous allons exposer les travaux de la littérature réalisés pour la conception des systèmes VMMR.

1.1 Systèmes VMMR

La reconnaissance du type de véhicule est un domaine de recherche assez récent, qui commence à intéresser différents laboratoires dans le monde. Nous allons décrire quelques travaux pour nous concentrer sur ceux qui sont, à notre avis, les plus représentatifs de la méthode. Le VMMR peut être abordé à partir de deux types de méthodes d'apprentissage : structurelles et d'apparence.

1.1.1 Approches Structurelles

Les approches structurelles transforment l'objet à reconnaître en un ensemble de composants (segments, phares, logo, etc) qui sont extraits individuellement, pour les combiner dans un vecteur qui décrit l'objet d'étude. Torres [138] utilise cette méthode et présente un système de reconnaissance en utilisant les contours des images de la vue arrière des véhicules. Chaque classe est définie à partir d'un gabarit composé des segments [122]. Le processus de classification calcule la distance de Hausdorff minimale entre l'entrée (test) et les gabarits des classes à partir d'une méthode appelée *Line segment Hausdorff Distance* (LHD) [56]. Il atteint un faible taux de reconnaissance de 59.7 %, ce qui n'est pas étonnant. Pendant nos essais, nous avons constaté que les segments peuvent se répéter d'une

[65] S. Han, E. Ahn, and N. Kwak. Detection of multiple vehicles in image sequences for driving assistance system. In *International Conference on Computational Science and Its Applications*, volume 3480, pages 1122–1128, 2005.

[63] S. Gupte, O. Masoud, R.F.K. Martin, and N.P. Papanikolopoulos. Detection and classification of vehicles. *IEEE Intelligent Transportation Systems*, 3(1) :37–47, Mars 2002.

[149] W. Wu, Z. QiSen, and W. Mingjun. A method of vehicle classification using models and neural networks. In *Vehicular Technology Conference*, volume 4, pages 3022–3026, 2001.

[47] J. M. Ferryman, A. D. Worrall, G. D. Sullivan, and K. D. Baker. A generic deformable model for vehicle recognition. In *British Machine Vision Conference*, pages 127–136, 1995.

[43] M. Dubuisson, S. Lakshmanan, and A. Jain. Vehicle segmentation and classication using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3) :293–308, Mars 1996.

[138] D. A. Torres. More local structure information for make-model recognition. University of California, San Diego, 2005.

[122] K. Reumann and A.P. Witkam. Optimizing curve segmentation in computer graphics. In *International Computing Symposium*, pages 467–472, 1974.

[56] Y. Gao and M. K.H. Leung. Face recognition using line edge map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6) :764–779, 2002.

classe à une autre. Leur position n'est jamais fixe en raison des différences de recalage, ce qui fait accroître énormément la confusion entre les classes.

1.1.2 Approches par Apparence

Les deuxièmes méthodes, fondées sur l'apparence, utilisent des algorithmes d'apprentissage supervisé pour apprendre les caractéristiques de chaque classe.

Kazemi [82] projette chaque image de véhicule dans trois espaces de paramètres correspondant à trois variantes de la transformée de Fourier. Les trois descripteurs utilisés sont : la transformée de Fourier, la transformée en ondelettes et la transformée en *courbelettes*. La méthode de classification consiste à trouver les k plus proches voisins de l'exemple de test parmi les exemples de référence, dans l'espace de projection. La base de données est composée de 300 images correspondants à cinq classes de véhicules en vue arrière. L'ensemble d'apprentissage contient 230 images normalisées à une taille de 128x128 pixels. Le reste est utilisé pour les tests. Les *courbelettes* ont obtenu les meilleurs résultats de classification.

Le vecteur de descripteurs qu'utilise Munroe [104] pour segmenter l'image d'un véhicule est composé des pixels de contours, calculés à partir de la méthode de Canny [22]. Les tests ont été réalisés sur un ensemble de cinq classes de véhicules vus de face composé de 150 exemples (30 exemples par classe). Parmi les classifieurs considérés, les réseaux neuronaux obtiennent le meilleur comportement avec un taux de classifications correctes de 99.5 %. Un classifieur de type kNN est aussi évalué et donne des résultats satisfaisants avec un taux de reconnaissance de 97.5 %.

Petrovic [116, 115] a appliqué différentes méthodes d'extraction de primitives sur une base d'images de véhicules vus de face prises dans un parking. Pour chaque image, il extrait une région d'intérêt (en anglais *Region of Interest* ou RoI) à partir de la position de la plaque minéralogique. Ensuite, cette vignette subit un sous-échantillonnage qui la convertit à une taille de 50x120 pixels.

Nous allons présenter le descripteur qui a obtenu les meilleurs résultats en tests : les *Squared Mapped Gradients*. Les gradients (s_x, s_y) sont obtenus à partir d'un filtrage de

[82] F. M. Kazemi, S. Samadi, H. Pourreza, and M. R. Akbarzadeh. Vehicle recognition based on fourier, wavelet and curvelet transforms - a comparative study. *International Journal of Computer Sciences and Network Security*, 7(2) :130–135, 2007.

[104] D. T. Munroe and M. G. Madden. Multi-class and single-class classification approaches to vehicle model recognition from images. In *Conference on Artificial Intelligence and Cognitive Science*, Septembre 2005.

[22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligent*, 8(6) :679–698, 1986.

[116] V.S. Petrovic and T.F. Cootes. Analysis of features for rigid structure vehicle type recognition. In *British Machine Vision Conference*, volume 2, pages 587–596, Septembre 2004.

[115] V. S. Petrovic and T. F. Cootes. Vehicle type recognition with match refinement. In *IAPR International Conference on Pattern Recognition*, volume 3, pages 95–98, 2004.

Sobel et, pour chaque pixel, il associe la paire de descripteurs :

$$(g_x^{SM}, g_y^{SM}) = (\frac{s_x^2 - s_y^2}{s_x^2 + s_y^2}, \frac{2 s_x s_y}{s_x^2 + s_y^2})$$

Les paires de descripteurs sont ensuite concaténées dans un vecteur $\mathbf{f}$.

Petrovic calcule un vecteur de poids $\mathbf{w}$ qui pondère chaque élément de $\mathbf{f}$. Pour calculer les valeurs de pondération, il commence par obtenir la moyenne du module du gradient de chaque pixel sur tous les exemples de la base de référence. Ces valeurs sont placées dans un vecteur $\mathbf{v}$. Ensuite, il détermine les valeurs des écart type de la paire de descripteurs en utilisant aussi la base de référence, et le place dans le vecteur $\mathbf{s}$. Le vecteur $\mathbf{w}$ est alors obtenue à partir de la multiplication normalisée de $\mathbf{v}$ et $\mathbf{s}$:

$$\mathbf{w} = \frac{\mathbf{v}.\mathbf{s}}{|\mathbf{v}.\mathbf{s}|}$$

Les poids importants correspondent aux pixels qui ont une grande énergie (module du gradient) sur de plusieurs exemples de référence, et ces valeurs de descripteurs différent significativement.

(a) (b) (c) (d)

FIGURE 1.1 – Segmentation des vignettes et vecteur de poids proposés par Petrovic. (a) matrice des poids $\mathbf{w}$ pour le descripteur g_x^{SM}, (b) image canonique, (c) descripteur g_x, (d) descripteur g_y.

La distance entre un exemple de test et une référence est calculée à l'aide d'un produit scalaire : $d = 1 - \mathbf{f}_{test}^T \mathbf{f}_r$. L'entrée est classifiée à partir de la référence qui a obtenu la distance la plus faible. La base d'images de référence contient 105 exemples, un par classe, et la base de test 1027 images. Les résultats obtenus pour les *Squared Mapped Gradients* sont de 97 % de classifications correctes.

Le travail de thèse de Dlagnekov [41] correspond à une application de reconnaissance de type embarquée qui fusionne un système LAPI à un système de VMMR. Les images des vues arrières des voitures ont été prises à l'aide des caméras embarquées. Différents descripteurs locaux sont utilisés pour réaliser la classification : les niveaux de gris, les *context shapes* [11], et les *Scale Invariant Feature Transforms* (SIFT). La classification

[41] L. Dlagnekov. *Video-based car surveillance : License plate make and model recognition.* PhD thesis, University of California, San Diego, 2005.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4) :509–522, 2002.

à partir des niveaux de gris, utilisait un sous espace obtenu à l'aide d'une ACP. Pour améliorer les résultats, ils ont éliminé les trois vecteurs propres qui correspondent aux trois plus grandes valeurs propres. Ils capturent plutôt les variations de luminance. Cette étape est nécessaire étant donné que les voitures d'une même classe peuvent avoir des couleurs différentes.

Ce sont les descripteurs de type SIFT qui obtiennent les meilleurs résultats. Lowe [96, 95] a développé cette nouvelle famille de descripteurs locaux, appelée *Scale Invariant Feature Transform* (SIFT), qui est utilisée pour la reconnaissance d'objets. Cette méthode détermine les points d'intérêt d'un objet et, pour chacun des points elle obtient un vecteur de caractéristiques.

Pour trouver les points d'intérêt, elle utilise un algorithme de différences de gaussiennes (DOG). D'abord, cette méthode obtient une fonction $L(x, y, \sigma)$ de la convolution entre l'image d'entrée $I(x, y)$ et un filtre gaussien $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

où

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

La différence de gaussiennes est obtenue en appliquant la soustraction de deux filtres séparés par un facteur de multiplication k :

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

La figure 1.2 montre le calcul des DOG pour deux octaves obtenues à partir de deux tailles du filtre gaussien. Le nombre de k_i est fixé pour obtenir toujours le même nombre de DOG par octave.

Ensuite, les pixels des matrices de DOGs sont évalués pour trouver les minima et maxima locaux. Chaque pixel est comparé avec ses 26 voisins dans une région de 3x3 dans sa propre DOG et les DOGs adjacentes. Ce point est conservé comme étant un point d'intérêt si son intensité est plus grande (ou plus petite) que celle de tous ses voisins.

La prochaine étape implique l'extraction d'un vecteur de caractéristiques de ces points d'intérêt. Pour chaque image $L(x, y, \sigma)$, sont calculés le module du gradient et son orientation.

Dans la figure 1.3, nous pouvons observer une image des orientations et modules du gradient autour d'un point d'intérêt (à gauche). L'histogramme qui décrit ce point contient

[96] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, pages 91–110, 2004.

[95] D. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

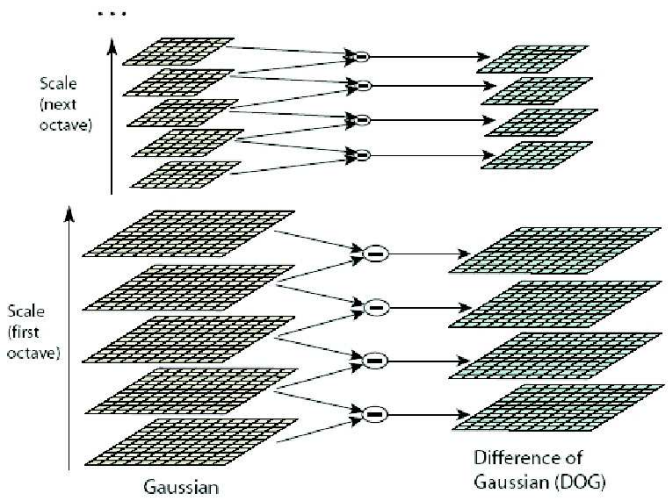


FIGURE 1.2 – Obtention des matrices de différences de gaussiennes.

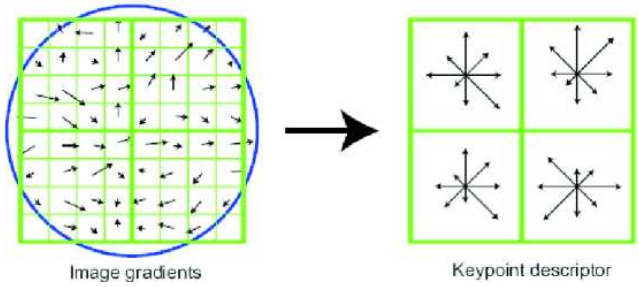


FIGURE 1.3 – Histogrammes des orientations.

un *bin* pour chaque orientation. La valeur des *bins* est calculée en sommant le module du gradient des pixels dans le bloque ayant cette orientation (dans la figure 1.3 il y en a huit). La figure de droite représente les histogrammes de chaque bloque et la taille de flèches indique la valeur du *bin* dans cette direction. Une région de voisinage autour d'un point d'intérêt de taille 8x8 pixels, est convertie en 4 blocs égaux, et un histogramme pour chacun. Le vecteur des caractéristiques est composé de 4 (classes)x8=32 éléments. Pour réduire les effets dus aux changements de luminance, les valeurs des histogrammes sont normalisées.

Ces descripteurs sont appliqués par Dlagnekov pour extraire les caractéristiques des images des véhicules. Pendant l'apprentissage, l'algorithme trouve les points d'intérêt de chaque image de la base de référence. Au cours des tests, le système calcule les descripteurs de SIFT de l'image d'entrée et cherche à faire une correspondance entre ceux-ci et les descripteurs appartenant aux références.

En général, seuls deux ou trois points d'intérêt suffisent pour classifier la classe de véhicule, mais un point faible du travail de Dlagnekov consiste à avoir une petite base de test avec une ou deux images et un nombre très limitée des classes.

Zafar [153] utilise aussi les descripteurs SIFT, mais propose une méthode de correspon-

[153] I. Zafar, B.S. Acar, and E.A. Edirisinghe. Vehicle make & model identification using scale invariant

dance plus exhaustive. Elle calcule une mesure de similarité ou distance entre les points d'intérêt de la référence et les descripteurs SIFT dans les mêmes positions dans l'image de test. Pour que l'algorithme soit robuste aux décalages de position, la recherche locale dans l'image de test est élargie à une fenêtre de 5x5 pixels. La distance est obtenue à l'aide d'un produit scalaire des vecteurs de descripteurs SIFT. Pour chaque descripteur de la référence est conservé le produit scalaire le plus faible (distance plus proche) et la mesure de similarité est donc calculée comme la moyenne de toutes ces valeurs. Zafar [154] propose dans un autre article, les niveaux de gris comme espace de paramètres et une classification à l'aide d'un Analyse Discriminante Linéaire à deux dimensions [89]. Cette représentation permet de travailler avec de grandes matrices et obtient un résultat similaire à l'Analyse Discriminante Linéaire traditionnelle.

1.2 Discussion

L'évaluation des travaux de la littérature nous permet d'avancer des conclusions intéressantes :

- Les différentes méthodes de classification se basent sur une évaluation locale des descripteurs. Il est donc pertinent d'obtenir, dans l'étape initiale, une image canonique stable qui corrige des défauts de la prise de vue pour pouvoir faire la comparaison entre les exemples de référence et les exemples de test.
- Les descripteurs qui ont obtenu les meilleurs résultats sont ceux basés sur les gradients orientés, soit à travers des valeurs ponctuelles, soit par l'obtention d'un histogramme. Ce type de descripteurs est robuste à la variation des couleurs des véhicules de la même classe et se montre toujours très discriminant pour la classification.
- L'introduction du vecteur des poids par Petrovic permet de mieux séparer les classes en donnant une influence importante aux points les plus représentatif d'une classe.
- La plupart des méthodes réalisent la classification à partir d'une base de référence. Pour obtenir des bons résultats, cette base doit être suffisamment représentative. Cependant, pour la commercialisation d'un système de type clé en main, la constitution de la base de référence dévient un vrai problème, étant donné que les images acquises du système peuvent varier d'un client à l'autre. Il est donc convenable de contourner cette contrainte à l'aide d'un système de type génératif qui compte avec un modèle pour chacune des classes et qui soit robuste aux possibles changements de point de vue.

---

transforms. In *Visualization, Imaging, and Image Processing*, Août 2007.

[154] I. Zafar, E. A. Edirisinghe, S. Acar, and H. E. Bez. Two-dimensional statistical linear discriminant analysis for real-time robust vehicle-type recognition. In *SPIE Real-Time Image Processing*, volume 6496, Février 2007.

[89] M. Li and B.Z. Yuan. 2d-lda : A statistical linear discriminant analysis for image matrix. *Pattern Recognition Letters*, 26(5) :527–532, Avril 2005.

- Finalement, la faible quantité d'exemples d'apprentissage (parfois un seul par classe) limite l'utilisation des méthodes que nous avons étudiée précédemment, comme la méthode de dopage. Nous proposons une méthode qui peut modéliser une classe en utilisant une seule image.

1.3 Genèse de notre système

1.3.1 Objectifs

Le système a pour objectif l'analyse de la vue frontal d'un véhicule et la recherche dans une base de classes du correspondant le plus proche.

Les contraintes pour le développement d'un tel système sont :

– La prise des images : elles peuvent être prises de différents point de vue.
– La présence de la barrière : la figure 1.4 montre des images réelles prises pour un système de lecture de plaques où nous constatons la présence de la barrière qui cache une partie de la calandre de la voiture.
– Le temps de réponse du système : il doit réagir proche du temps réel (moins d'un second).

(a) (b)

FIGURE 1.4 – Deux exemples de véhicules à l'entrée d'un parking.
Note : la barrière occulte une partie de la calandre (Source : LPREditor).

1.3.2 Superposition des contours

En général, les calandres des différentes voitures ont des caractéristiques très variées d'un modèle à l'autre. Cependant, chaque modèle de véhicule a un gabarit constant, défini par le constructeur. D'ailleurs, ces mêmes constructeurs donnent à un modèle de véhicule différentes couleurs, brillances, etc. Bien que cette différentiation soit très convenable du point de vue commercial (accroit les ventes), elle introduit quelques restrictions pour les procédures de reconnaissance de formes basées sur les algorithmes rétiniens (valeurs en niveaux de gris des pixels).

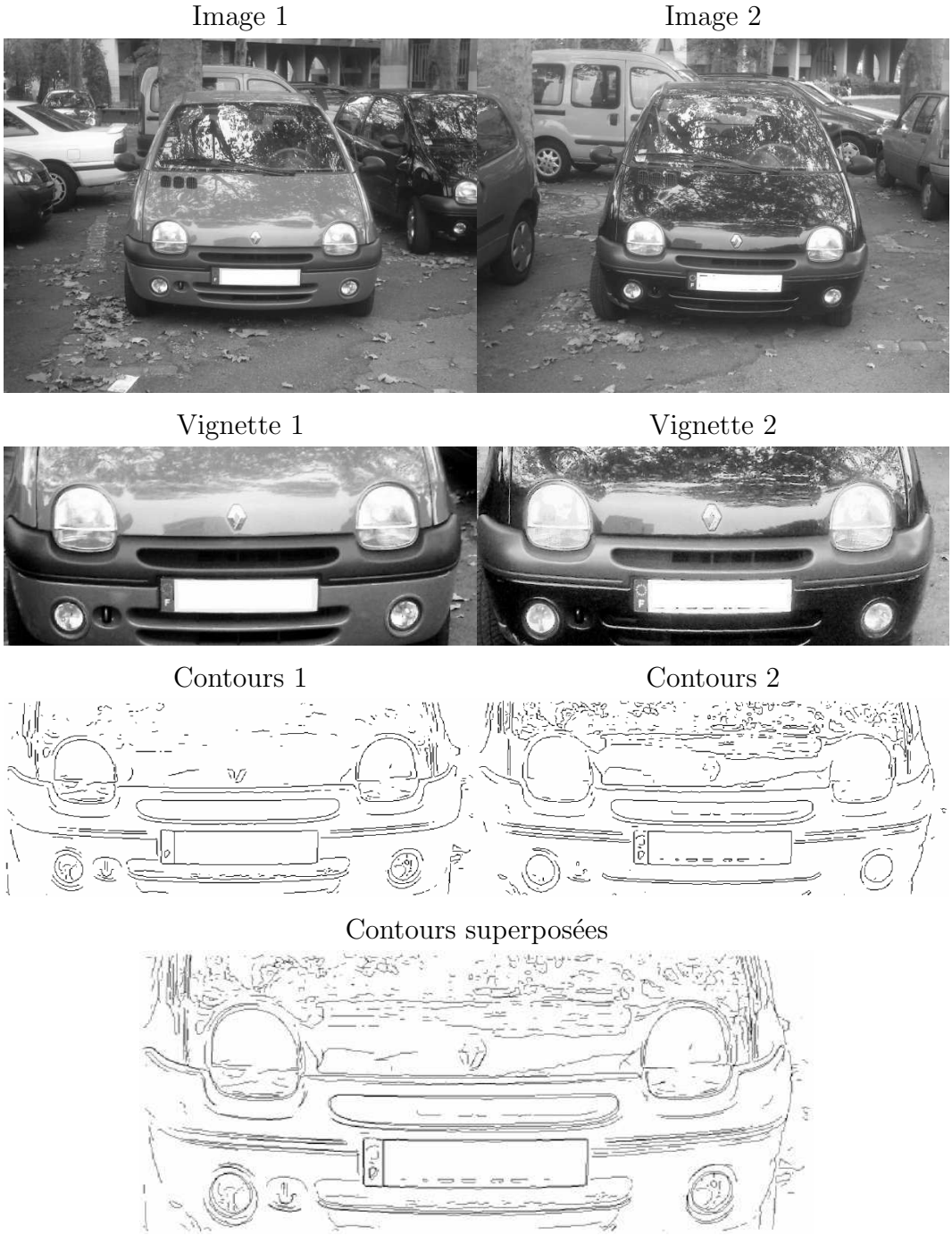


FIGURE 1.5 – Contours superposés de deux exemples de véhicules du même type.

Nous pouvons faire un analyse des contours obtenus à partir du gradient de l'image. La figure 1.5 montre deux véhicules du même modèle mais de couleurs différentes. Pour chacune des images, nous obtenons une vignette à partir d'une méthode qui consiste à placer la plaque d'immatriculation dans un positions connue (plus de détails dans la section 2.1.2). La ligne suivante montre les images de contours obtenues à l'aide de l'algorithme de Canny-Deriche. Enfin, nous réalisons une superposition simple des contours qui illustre effectivement que les contours sont quasiment coïncidents, même s'ils correspondent à des véhicules de couleurs différentes.

La génération du modèle de chaque classe à partir des pixels de contours pourrait s'effectuer à partir du calcul d'une image de contours moyenne. La comparaison des contours d'une image de test et des contours du modèle permettra d'obtenir un indice de ressemblance utile à la classification.

Nous présentons par la suite une méthode de votes, qui nous permet d'obtenir une mesure de similarité entre deux images de contours.

1.3.3 Définition d'une mesure de similarité

Nous présentons dans cette partie une mesure de similarité pour décider si les véhicules appartiennent ou non à la même classe. Cette mesure compare la répartition spatiale de deux matrices de pixels de contours. D'autres mesures, comme la distance Euclidienne, la distance de Hausdorff, peuvent être utilisées. Cependant, seules les méthodes de votes, en utilisant un algorithme spécialement conçu, nous permettent d'éviter un calcul de distances pixel par pixel.

Les algorithmes de votes sont largement utilisés dans le domaine de la reconnaissance d'objets et ils se montrent robustes face aux occultations [7, 54, 85]. Comme nous avons vu précédemment, nous allons être confrontés à ce problème en raison de la présence de la barrière qui cache une partie de la vue frontal du véhicule (voir la figure 1.4).

Pour estimer la ressemblance entre le modèle d'une classe et un exemple de test, l'algorithme proposé compte le nombre des pixels de contour de ce dernier qui se trouvent dans le voisinage des pixels de contour du premier.

L'algorithme suivant, représenté dans la figure 1.6, calcule cette mesure :

1. Nous travaillons avec tous les pixels de contour de la matrice $\mathbf{E}$ de l'exemple de test. Dans l'exemple de la figure 1.6, nous montrons deux pixels $\mathbf{p}_{i1}$ et $\mathbf{p}_{i2}$,
2. Ces points sont projetés sur le modèle $\mathbf{M}^k$, points $\mathbf{p}'_{m1}$ et $\mathbf{p}'_{m2}$,
3. Le point $\mathbf{p}'_{m1}$ qui est dans le voisinage d'un point de $\mathbf{M}^k$, représenté par le cercle centré à $\mathbf{p}_m$ et de rayon r origine le point $\mathbf{p}_v$ dans la matrice des votes $\mathbf{V}$,
4. La similarité ou quantité des votes v, est obtenue en comptant le nombre de points $\mathbf{p}_v$ dans $\mathbf{V}$

$$v = \sum_{x,y} \mathbf{V}(x, y) \tag{1.1}$$

---

[7] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Readings in computer vision : issues, problems, principles, and paradigms*, pages 714–725, 1987.

[54] P.F. Fung, W.S. Lee, and I. King. Randomized generalized hough transform for 2-d grayscale object detection. In *IAPR International Conference on Pattern Recognition*, volume 2, pages 511–515, 1996.

[85] A. Kimura and T. Watanabe. An extension of the generalized hough transform to realize affine-invariant two-dimensional (2d) shape detection. In *IAPR International Conference on Pattern Recognition*, page 10065, Washington, DC, États Unis, 2002.

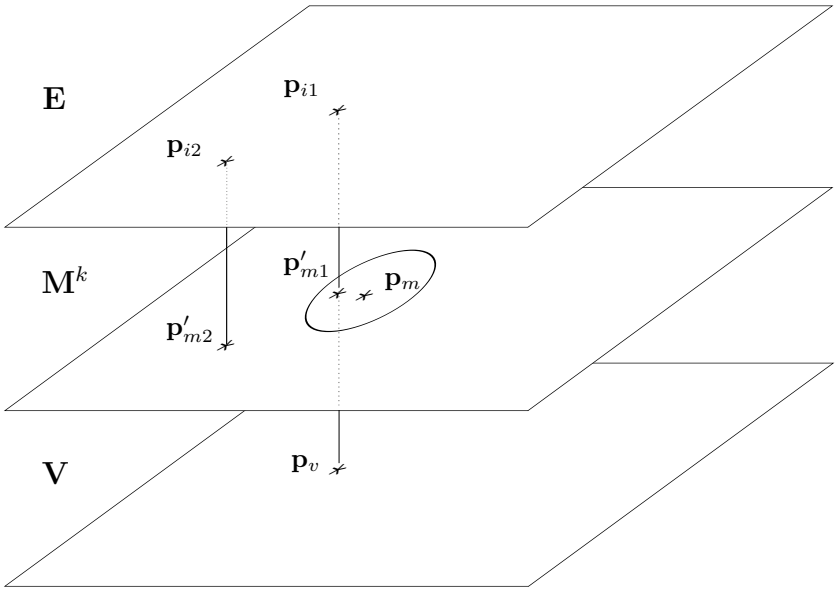


FIGURE 1.6 – Exemple de l'algorithme de votes pour calculer une mesure de similarité.

Une manipulation spéciale nous permet de calculer les votes v de façon rapide. D'abord, nous transformons la matrice $\mathbf{M}^k$ en une matrice des régions $\mathbf{R}_{filled}$ représentée par les cercles de voisinage de rayon r remplis de 1 et 0 ailleurs. Ensuite, l'intersection (une opération ET) entre les points de la matrice $\mathbf{E}$ et la matrice $\mathbf{R}_{filled}$ nous permet d'obtenir $\mathbf{V}$. Cette opération n'a pas besoin d'un calcul itératif des distances ce qui signifie un gain du temps de calcul. L'équation 1.1 devient :

$$v = \sum_{x,y} \mathbf{E} \quad \cap \quad \mathbf{R}_{filled} \tag{1.2}$$

Nous allons utiliser ces méthodes dans les chapitres suivants pour construire le système de classification.

1.4 Contributions

Les contributions de notre travail de recherche pour cette problématique peuvent être résumées dans quatre points :

1. L'obtention de la RoI à l'aide d'une transformation affine.
2. La génération du modèle de classe à partir d'une image moyenne.
3. Une méthode de classification basée sur les votes.
4. La conception de matrices de régions pondérées.

Ces contributions sont développées en détail dans le chapitre suivant.

Dans le chapitre suivant, nous allons proposer un système qu'utilise les descripteurs de contours orientés et une nouvelle mesure de classification calculée à partir des votes.

Chapitre 2

Modélisation du type de véhicule

Pour trouver le modèle de chaque classe, nous utilisons une image moyenne calculée entre les exemples de la classe qui sont placés dans une base d'apprentissage. Ce modèle va contenir les pixels de contour orientés les plus représentatifs de cette classe. Les pixels choisis nous permettront d'obtenir une représentation en régions pour appliquer une méthode de calcul rapide des votes. Ce calcul est optimisé grâce à l'association d'un poids pour chaque pixel du modèle qui permet de mieux discriminer entre les classes de la base.

2.1 Mise en forme des données et codage

Notre algorithme de classification est fondé sur la répartition spatiale des pixels de contour. Évidemment, pour arriver à obtenir un résultat satisfaisant, nous devons recaler les images originales de façon efficace.

Pour définir une région d'intérêt (RoI) dans l'image, dans notre cas sur la calandre, nous utilisons la plaque minéralogique en tant que référence. La taille de la plaque minéralogique correspond à un patron défini par la communauté européenne et fixé dans plusieurs pays. Notre expérience nous a montré que les voitures contenant les anciennes plaques minéralogiques, de différentes tailles, ne représentent que moins de 5 %. L'avantage d'utiliser la plaque minéralogique est que celle-ci va être toujours visible. En effet, étant donné que notre algorithme complète un système LAPI, nous supposons que le matériel (caméras) est réglé pour l'avoir toujours en vue, en évitant la barrière. Nous ne nous occupons pas de la détection de la plaque : nous faisons l'hypothèse que le système LAPI nous donne sa position. Cependant, dans ce domaine, le lecteur pourra s'intéresser aux

travaux de Cano [23], Kwasnicka [86], Kim [84] et Zayed [155].

Dans notre approximation, la plaque d'immatriculation représente le plan de la calandre de la voiture. Les quatre coins de la plaque sont utilisés comme paramètres d'une transformation affine qui projette les images dans un repère lié à celle-ci.

Le traitement appliqué sur chaque image est toujours le même, que ce soit une image de la base d'apprentissage employée pour créer les modèles ou une image de la base de test qui doit être classifiée.

2.1.1 Transformation affine

La transformation affine nous permet de conserver la structure affine entre les deux espaces en préservant le parallélisme des segments du parallélogramme de la plaque minéralogique (fig. 2.1). Les coins de la plaque minéralogique représentent le plan de la calandre dans notre modèle.

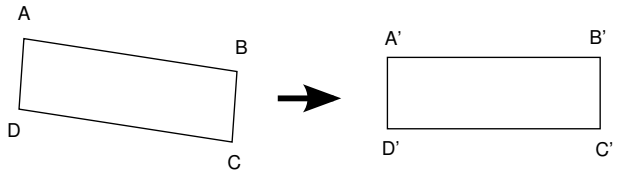


FIGURE 2.1 – Transformation affine.

A partir des points de l'image (u,v), nous voulons obtenir, après la transformation, des nouveaux points (x,y) :

$$\begin{cases} x = a \cdot u + b \cdot v + c \\ y = d \cdot u + e \cdot v + f \end{cases}$$

Sous forme matricielle :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

En forme générale :

---

[23] J. Cano and J. C. Pérez-Cortés. Vehicle license plate segmentation in natural images. In *Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science, pages 142–149, Juin 2003.

[86] H. Kwasnicka and B. Wawrzyniak. License plate localization and recognition in camera pictures. In *Artificial Intelligence Methods*, Gliwice, Pologne, Novembre 2002.

[84] K. I. Kim, K. Jung, and J. H. Kim. Color texture-based object detection : An application to license plate localization. In *Pattern Recognition with Support Vector Machines*, volume 2388/2002, pages 293–309, Août 2002.

[155] M. Zayed and J. Boonaert. Detection et suivi d'une plaque d'immatriculation pour l'attelage virtuel. In *18 Journées des Jeunes Chercheurs en Robotique*, Douai, France, Septembre 2004.

$$\mathbf{x} = \mathbf{M_{af}} \cdot \mathbf{u}$$

Nous faisons la transformation affine à partir de {A,B,C,D}, un ensemble de 4 points de la plaque.

A partir des valeurs de l'ensemble {A,B,C,D} et les valeurs souhaités {A',B',C',D'}, (cf. figure 2.1), nous trouvons la matrice homogène de transformation affine $\mathbf{M_{af}}$.

$$\begin{pmatrix} A'_x & B'_x & C'_x & D'_x \\ A'_y & B'_y & C'_y & D'_y \\ 1 & 1 & 1 & 1 \end{pmatrix} = \mathbf{M_{af}} \cdot \begin{pmatrix} A_x & B_x & C_x & D_x \\ A_y & B_y & C_y & D_y \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Nous avons donc :

$$\mathbf{x} = \mathbf{M_{af}} \cdot \mathbf{u} \tag{2.1}$$

$$\mathbf{M_{af}} = \mathbf{x} \cdot \mathbf{u}^{-1} \tag{2.2}$$

où U^{-1} est la pseudo-inverse de U.

Pour transformer les pixels de l'image originale à la vignette, nous avons besoin d'abord, des quatre coins de la plaque minéralogique : {A,B,C,D}. Ensuite, nous définissons la taille de la vignette (W_i, H_i) et la position désirée des coins de la plaque dans celle-ci : {A', B',C',D'}.

Tous les points de l'image originale sont placés dans une matrice de la forme :

$$\begin{pmatrix} x_1 & x_2 & ... & x_n \\ y_1 & y_2 & ... & y_n \\ 1 & 1 & ... & 1 \end{pmatrix}$$

où $n = W_o x H_o$, (W_o, H_o) représentent la taille de l'image originale.

Cette matrice est transformée à l'aide de $\mathbf{M_{af}}$, pour obtenir les nouvelles positions :

$$\begin{pmatrix} x'_1 & x'_2 & ... & x'_n \\ y'_1 & y'_2 & ... & y'_n \\ 1 & 1 & ... & 1 \end{pmatrix} = \mathbf{M_{af}} \cdot \begin{pmatrix} x_1 & x_2 & ... & x_n \\ y_1 & y_2 & ... & y_n \\ 1 & 1 & ... & 1 \end{pmatrix}$$

Nous éliminons les pixels dont la position est hors les limites de la vignette (W_i, H_i).

Pour obtenir finalement la vignette, nous donnons aux nouvelles positions, les niveaux de gris du pixel correspondant de l'image originale :

$$\mathbf{P}(x'_i, y'_i) = \mathbf{I}(x_i, y_i) \quad i = 1, ..., m$$

où $m = W_p x H_p$.

2.1.2 Encadrement de la région d'intérêt

Pour l'application de la transformation affine, nous devons choisir la position des quatre coins de la plaque dans une vignette de taille fixe. D'abord, nous fixons la largeur de la plaque d'immatriculation à 200 pixels. Nous avons réalisé ce choix après une analyse statistique de la largeur de la plaque dans la base des images de véhicules. La figure 2.2 montre la courbe des valeurs cumulées de largeur des plaques minéralogiques en pixels. Nous voyons qu'à 200 pixels, la courbe arrive à une stabilisation des valeurs sur la base d'images. Nous pouvons considérer une valeur plus petite, mais cela vaudrait dire que les images seront sous-échantillonnées avec une perte d'information.

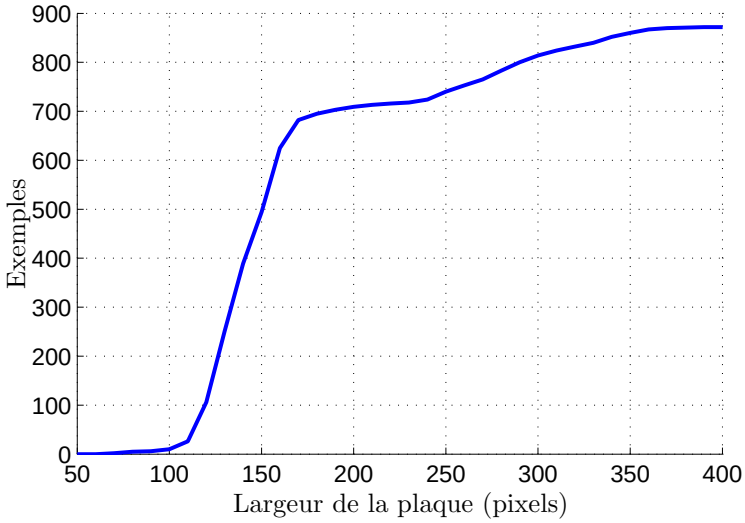


FIGURE 2.2 – Histogramme cumulé des largeurs de la plaque minéralogique dans les bases des véhicules.

Ensuite, nous utilisons la région d'intérêt proposée par Petrovic [116], ce qui nous donne une vignette de 220 pixels de hauteur et 520 pixels de largeur. Ce choix a donné de meilleurs résultats que celui de Negri [105], surtout pour la classification avec occlusions (section 4.5).

Les positions désirées de la plaque minéralogique pour cette vignette, en respectant les proportions réelles en hauteur et en largeur, sont :

1. A' = (160,116)
2. B' = (360,116)
3. C' = (360,163)

---

[116] V.S. Petrovic and T.F. Cootes. Analysis of features for rigid structure vehicle type recognition. In *British Machine Vision Conference*, volume 2, pages 587–596, Septembre 2004.

[105] P. Negri, X. Clady, M. Milgram, and R. Poulenard. An oriented-contour point based voting algorithm for vehicle type classification. In *IAPR International Conference on Pattern Recognition*, volume 1, pages 574–577, Août 2006.

4. D' = (160,163)

La figure 2.3 illustre le résultat obtenu.

(a) (b)

FIGURE 2.3 – Obtention de la vignette à partir de l'image originale.
(a) image originale, (b) vignette obtenue après la transformation affine. La variable w correspond à la largeur de la plaque d'immatriculation (dans notre cas, $w = 200$).

2.1.3 Pixels de contour orientés

L'analyse de la bibliographie, ainsi que de tests réalisés dans la section suivante, nous ont montré que les pixels de contour orientés permettent d'obtenir des meilleurs résultats que les contours simples. Nous allons les utiliser pour faire le codage de notre vignette et extraire l'information spatiale nécessaire pour réaliser la classification.

A partir d'une image en niveaux de gris, nous allons calculer les contours à l'aide de la méthode du gradient. A cet effet, un filtre de Sobel de taille 3x3 [73] est utilisé pour obtenir les dérivées de l'image dans la direction x et la direction y, à partir de la convolution de celui-ci avec l'image :

$$M_x = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \quad M_y = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

Le module du gradient est alors défini :

$$G(x,y) = |M_x * I(x,y)| + |M_y * I(x,y)|$$

et son orientation :

$$\phi(x,y) = arctan(\frac{M_x * I(x,y)}{M_y * I(x,y)})$$

Pour obtenir les pixels de contour, nous allons utiliser une méthode qui garde un pourcentage de la totalité des points avec le plus haut module du gradient. Nous quantifions les orientations de ces points pour qu'ils prennent des valeurs entre 0 et $N-1$ (avec N égal à 4). Pour gérer les cas des voitures du même type mais de couleurs différentes (changement

[73] B. Jahne, H. Haubecker, and P. Geibler. *HandBook of Computer Vision and Applications*, volume 2. Academic Press, 1999.

de la polarité du contour), nous utilisons le module π au lieu du module 2π. Cela veut dire que nous avons quatre orientations au total : verticale, horizontale et deux diagonales. Le résultat est montré dans la figure 2.4 qui illustre les différentes orientations du gradient avec différentes couleurs.

(a) (b) (c)

FIGURE 2.4 – Contours des pixels orientés d'un véhicule.
(a) Image originale, (b) image après égalisation de l'histogrammme, (c) contours orientés (une couleur différente par orientation).

La figure 2.4 (c) présente les primitives avec lesquelles nous allons travailler par la suite. Chaque pixel de contour de l'image contient une valeur entière entre 0 et N-1. Les autres pixels ont une valeur quelconque (-1 par exemple) et ne sont pas pris en compte dans notre algorithme. Chaque pixel $\mathbf{p_i}$ du contour orienté peut être représenté par un vecteur :

$$\mathbf{p_i} = \begin{bmatrix} x_i \\ y_i \\ o_i \end{bmatrix} \tag{2.3}$$

où (x_i,y_i) est la position du point, et o_i est l'orientation du gradient en $\mathbf{p_i}$.

Dorénavant, la matrice de pixels de contour orientés d'un exemple $\mathbf{x}$ será notée $\mathbf{E}_x$.

2.2 Génération du modèle

2.2.1 Image moyenne

L'image moyenne de deux exemples de véhicules de la même classe peut être obtenue par le calcul des positions intermédiaires de ses pixels de contours.

La figure 2.5 montre l'obtention d'un point de contour moyen entre deux images de contours $\mathbf{I}$ et $\mathbf{J}$:

- Pour chaque point de contour $\mathbf{p}_i$ dans $\mathbf{I}$, trouver le point $\mathbf{p}_j$ le plus proche (distance euclidienne) dans $\mathbf{J}$.
- Le point moyen $\mathbf{p}_a$ dans $\mathbf{A}$ est défini à partir de la position moyenne entre $\mathbf{p}'_i$, la projection de $\mathbf{p}_i$ dans $\mathbf{J}$, et $\mathbf{p}_j$.

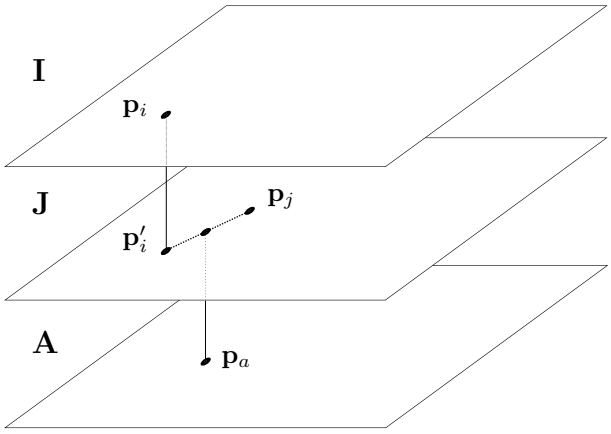


FIGURE 2.5 – Calcul des positions intermédiaires.

Nous allons illustrer cette idée dans l'application de l'algorithme sur un gabarit simple. La figure 2.6 montre un exemple du calcul de l'image moyenne calculée entre le même gabarit légèrement déplacé et tourné. La méthode de l'image moyenne nous donne deux résultats qui dépendent de l'image d'origine prise.

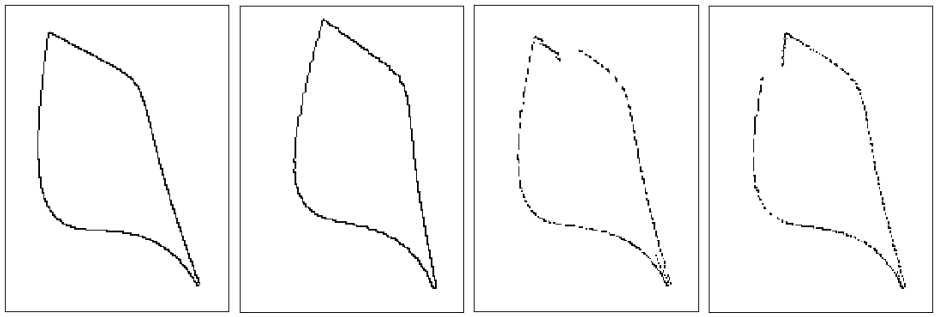


FIGURE 2.6 – Image moyenne à partir des contours simples.
(a) image moyenne de **I** vers **J**, (b) image moyenne de **J** vers **I**.

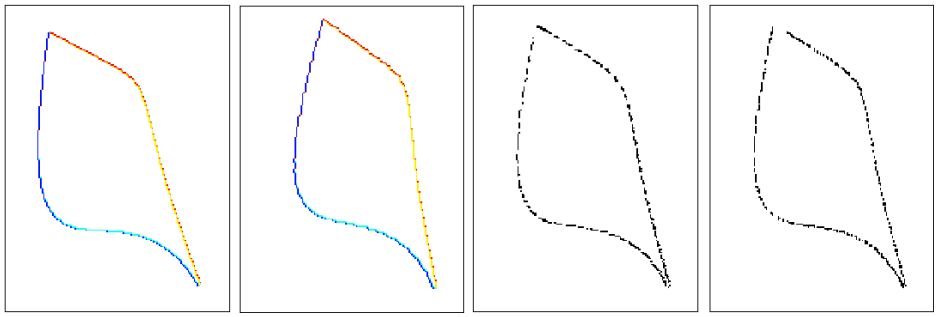
Pour mesurer la qualité des résultats nous allons calculer les distances entre les formes trouvées et les gabarits originaux. Pour chaque point du gabarit, nous trouvons la distance euclidienne au point plus proche de l'image moyenne. La distance moyenne entre les deux gabarits et les deux images moyennes nous donne 5.62 pixels.

Le même calcul effectué entre les deux images moyennes donne une distance moyenne de 1.34 pixels.

Nous allons évaluer maintenant le comportement du système face à une information additionnelle concernant l'orientation du gradient des pixels de contour. Pour chaque point de contour $\mathbf{p}_i$ dans **I**, nous allons trouver le point $\mathbf{p}_j$ le plus proche dans **J** qui possède la même orientation du gradient que $\mathbf{p}_i$.

Dans la figure 2.7 l'orientation du gradient des points de contour est quantifiée pour obtenir huit valeurs entières.

Visuellement, nous pouvons observer que les images moyennes obtenues sont plus proches des gabarits originaux et qu'il n'y a pas une grande variation entre les deux.



I **J** (a) (b)

FIGURE 2.7 – Image moyenne à partir de contours orientés.
(a) image moyenne de **I** vers **J**, (b) image moyenne de **J** vers **I**.

La moyenne des distances entre les gabarits et les moyennes est de 5.51, et la distance entre les images moyennes est de 0.55 pixels.

Ce résultat prouve que le choix d'ajouter l'information des orientations des contours nous permet de trouver des images moyennes plus stables et que le résultat ne dépend pas de l'image prise comme source.

2.2.2 Création de la matrice d'accumulation

L'objectif de cette section est de trouver les pixels de contour orientés les plus représentatifs pour une classe k. L'utilisation de la méthode de l'image moyenne nous permet d'identifier les pixels de contour qui se répètent dans les exemples d'apprentissage de cette classe. Notre méthode prend ces exemples par couples et cumule dans une matrice les points de contours de l'image moyenne.

La procédure suivante nous permettra de construire cette matrice :

1. Parcours de tous les exemples d'apprentissage de notre classe.
2. Traitement par couples : nous prenons tous les couples possibles sans répétition.
3. Alignement des images : nous faisons un alignement d'une image vers l'autre, pour corriger les erreurs commises au moment du recalage, à l'aide d'une transformation affine (annexe C).
4. Calcul de l'image moyenne : pour $\mathbf{E}_i$ et $\mathbf{E}_j$, deux matrices de contours orientés, nous calculons l'image moyenne $\mathbf{A}_{ij}$ que nous séparons en N matrices, une matrice pour chaque orientation. La figure 2.8 nous montre un exemple de ce processus.
5. Répétition du processus pour tous les couples possibles de l'ensemble d'apprentissage, en trouvant $\mathbf{A}_{13}$, $\mathbf{A}_{14}$,..., etc.

6. Obtention de la matrice d'accumulation de la classe k en faisant l'addition de tous les $\mathbf{A}_{ij}$ (voir figure 2.9) :

$$\mathbf{A}^k = \sum_{i,j/i \neq j} \mathbf{A}_{ij}$$

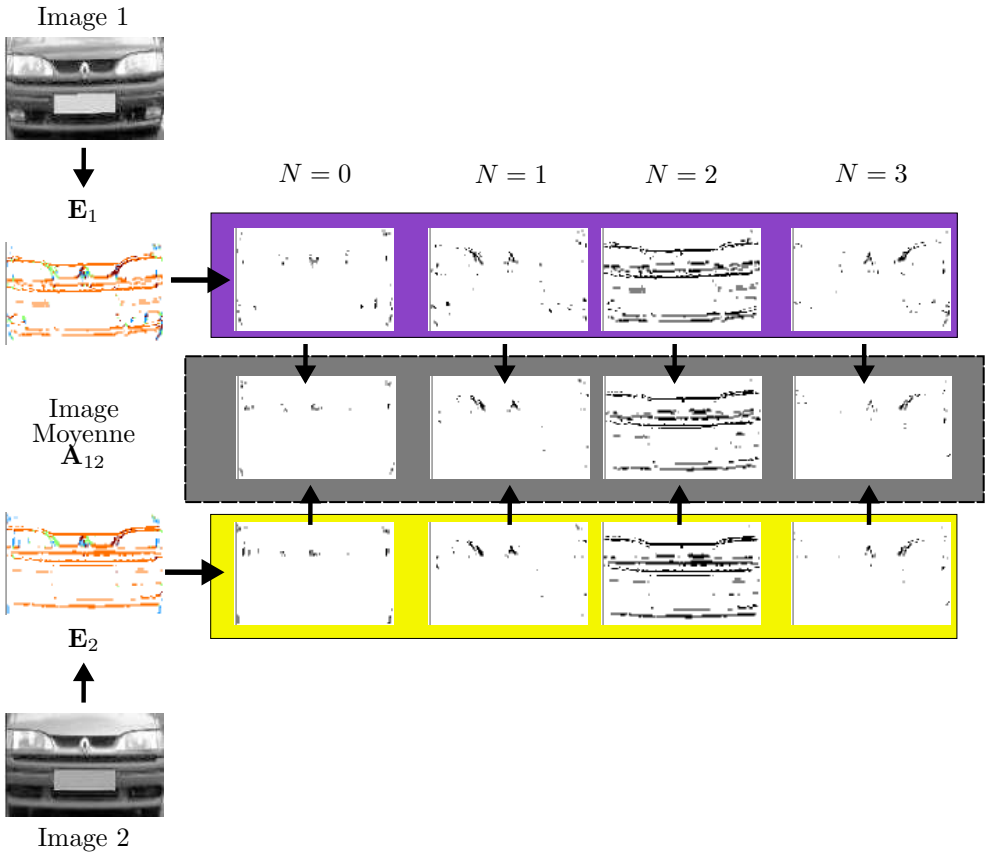


FIGURE 2.8 – Création de l'image moyenne $\mathbf{A}_{12}$ à partir de deux images avec un plan pour chaque orientation du gradient.

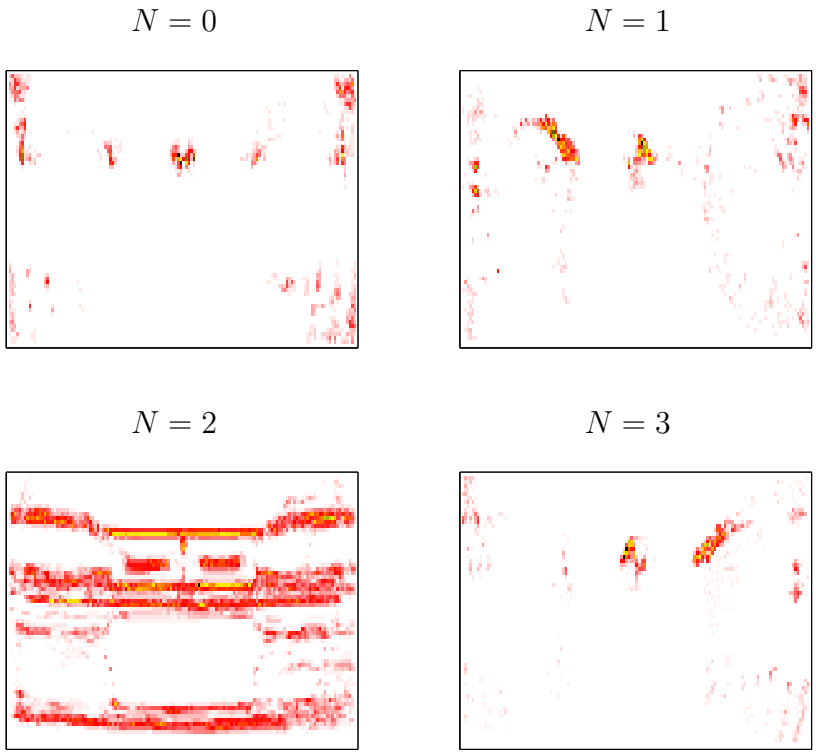


FIGURE 2.9 – Matrices d'accumulation des votes pour chaque orientation du gradient.

2.2.3 Sélection de points représentatifs

Dans cette section, nous utilisons la matrice d'accumulation de votes $\mathbf{A}^k$ pour trouver les points les plus représentatifs de la classe k. En effet, la matrice d'accumulation sera parcourue afin de trouver les pixels qui ont obtenu un maximum de votes. Ces points vont composer le modèle $\mathbf{M}^k$ de pixels de contour orientés de la classe k. La matrice multidimensionnelle $\mathbf{M}^k$ est définie avec une taille (W, H, N), où W et H sont la taille de la vignette et N le nombre d'orientations.

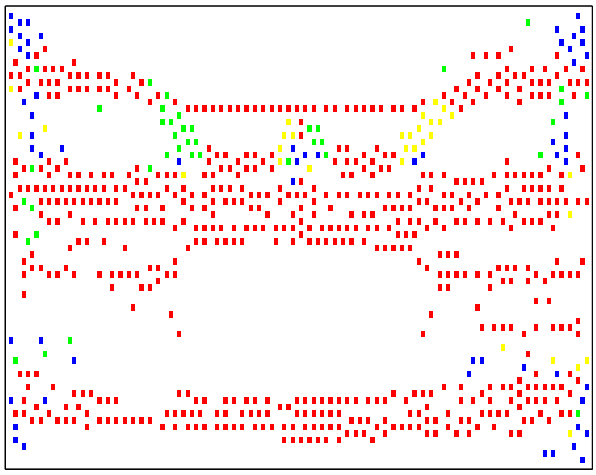


FIGURE 2.10 – Modèle pour la classe Renault 19.
Le modèle $\mathbf{M}^k$ est représentée en 2D où chaque point a une couleur différente pour chaque orientation du gradient.

Les points de $\mathbf{M}^k$ sont trouvés à l'aide d'une méthode itérative :

1. Une fonction f_{max} cherche pour chaque pixel de $\mathbf{A}^k$, l'orientation du gradient la plus votée :

   $$f_{max}(\mathbf{A}^k) = \{\mathbf{A}^k_{mx}, \mathbf{P}^k_{mx}\}$$

   Nous obtenons deux matrices de taille (WxH) : la matrice $\mathbf{A}^k_{mx}$ qui contient le nombre maximal des votes des orientations pour chaque pixel, et la matrice $\mathbf{P}^k_{mx}$ qui indique, pour chaque maximum, l'orientation à laquelle il correspond.
2. Nous trouvons le pixel avec le plus de votes dans $\mathbf{A}^k_{mx}$: $p_m(x_m, y_m)$.
3. Nous donnons la valeur 1 à l'élément de la matrice $\mathbf{M}^k$, généré par p_m :

   $$\mathbf{M}^k(x_m, y_m, \mathbf{P}^k_{mx}(x_m, y_m)) = 1$$

4. Nous effaçons un cercle de rayon r centré au point p_m pour écarter la position des prochains maxima.

5. Les quatre opérations antérieures sont répétées jusqu'à obtenir une quantité Q des points fixes (il est convenable que tous les modèles des classes contiennent la même quantité de points).

Quand il n'existe qu'un seul exemple dans la base d'apprentissage, nous ne pouvons pas créer la matrice $\mathbf{A}^k$ d'accumulation des votes. Nous devons, donc, utiliser une autre source d'information qui nous rend un poids pour chacun pixel de contour. Dans ce cas, nous utilisons la matrice du module du gradient $G(x, y)$ pour remplacer la matrice d'accumulation $\mathbf{A}^k_{mx}$.

La figure 2.10 montre le modèle de pixels de contour orientés créé pour la classe Renault 19.

La prochaine étape consiste à donner à chaque point, un poids discriminant par rapport aux autres modèles de classes.

2.2.4 Matrices de pondération

Les pixels de contour orientés du modèle obtenus dans la section précédente, vont être à l'origine des matrices des régions $\mathbf{R}_{filled}$. L'utilisation de ces matrices nous apportent deux avantages. D'une part, l'astuce pour trouver rapidement la mesure de similarité entre deux ensembles de points (équation 1.2 de la section 1.3.3). D'autre part, nous allons donner un poids discriminant à chaque point de la région, dans le même sens que Petrovic utilise son vecteur des poids $\mathbf{w}$.

Nous définissions une matrice de régions $\mathbf{R}_{filled}$, comme celle qui modélise le voisinage d'un pixel orienté du modèle avec des uns et zéro ailleurs.

La figure 2.11 montre les cartes des distances de Chamfer [124] D_i calculées pour chacune des orientations du modèle $\mathbf{M}^k$ de la classe k. L'application d'un seuil r sur les cartes des distances nous permet d'obtenir les matrices de régions de Chamfer R_i.

L'intérêt de la distance de Chamfer par rapport à une dilatation d'un disque de rayon r est qu'elle conserve pour chacun des pixels des régions, l'indice du plus proche élément différent de zero de $\mathbf{M}^k$. Une fonction de classification va utiliser cet indice par la suite.

Afin d'optimiser la méthode de votes, nous allons calculer un poids discriminant pour chaque point des matrices de régions $\mathbf{R}^k$. Les pixels du modèle k qui sont rarement présents dans les autres classes obtiennent des poids plus importants. Des poids faibles sont donnés aux points présents dans la plupart des autres classes. La matrice de régions $\mathbf{R}^k$ dévient la matrice des régions pondérées $\mathbf{W}^k_+$ obtenue à partir de l'équation suivante :

$$\mathbf{W}^k_+ = \frac{1}{K-1} \sum_{i,i \neq k} (\mathbf{R}^k - \mathbf{R}^i \cap \mathbf{R}^k)$$

[124] A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *Association for Computing Machinery*, 13(4) :471–494, 1966.

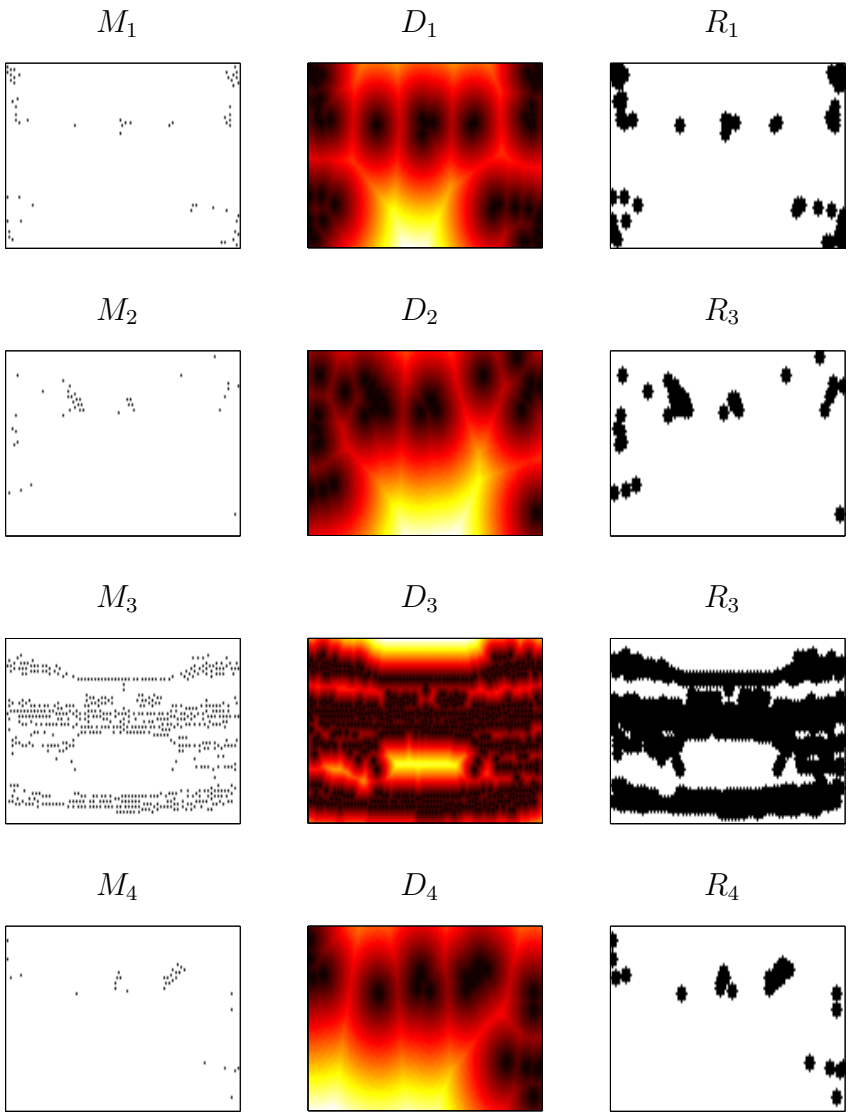


FIGURE 2.11 – Obtention des matrices de régions R_i.
A partir des points de M_i, nous obtenons la carte des distances de Chamfer D_i. Les matrices des régions de Chamfer R_i sont obtenues en seuillant dans D_i les distances inférieures à r.

où K est la quantité totale de classes.

De la même façon que nous récompensons les points qui tombent dans les voisinages de la classe k, nous pouvons aussi pénaliser ceux qui ne le font pas. En considérant alors des poids négatifs, nous pouvons construire une matrice de pondération $\mathbf{W}_-^k$:

$$\mathbf{W}_-^k = -\frac{1}{K-1}\sum_{i,i\neq k}(\mathbf{R}^i - \mathbf{R}^i \cap \mathbf{R}^k)$$

2.2.5 Listes des points triés

Enfin, nous présentons dans cette section une méthode pour optimiser le choix des pixels de contours orientés de l'exemple de test.

Une matrice $\mathbf{E}_t$, issue d'un calcul des contours d'un exemple t, contient un nombre important de points. Une grande partie correspondent aux contours de la calandre et d'autres proviennent des différentes sources de bruit ; ombres, reflets, etc. Afin de garder les premiers et minimiser les choix des deuxièmes pour réaliser la classification, nous allons évaluer statistiquement les positions des pixels orientés qui appartiennent aux calandres dans la base d'apprentissage.

Nous disposons déjà de beaucoup d'informations dans chaque modèle que nous utilisons pour calculer une matrice somme des $\mathbf{W}_+$ sur toutes les classes :

$$\mathbf{S} = \sum_{i=1}^{K} \mathbf{W}_+^i$$

Les points qui ont une valeur maximal dans $\mathbf{S}$ sont présents dans plusieurs modèles et ont une forte probabilité d'appartenir à la calandre.

Nous trions les pixels de $\mathbf{S}$ des plus grandes valeurs aux plus petites (différentes de zéros) et plaçons leurs positions dans une liste $\mathbf{t}$:

$$\mathbf{t} = \begin{pmatrix} x_0 & x_1 & \dots \\ y_0 & y_1 & \dots \\ o_0 & o_1 & \dots \end{pmatrix}$$

Pour extraire les points de contour orientés d'un exemple d'entrée $\mathbf{E}_t$ que nous devons classifier, nous utilisons $\mathbf{t}$ de la façon suivante : nous parcourons les éléments de la liste $\mathbf{t}$ pour vérifier s'ils sont présents dans $\mathbf{E}_t$, jusqu'à compléter un nombre T.

De façon similaire, nous sommes en position de trier aussi la liste de points de chacune des classes $\mathbf{M}^k$. Dans une liste $\mathbf{t}^k$, les premières places seront maintenant occupées par les points plus discriminants au moins discriminants de la classe k.

Pour obtenir la liste des points triés du modèle k, nous employons $\mathbf{W}_+^k$. Les points de $\mathbf{W}_+^k$ qui ont une valeur importante sont ceux qui sont peu ou pas présents dans les autres modèles.

2.3 Bilan

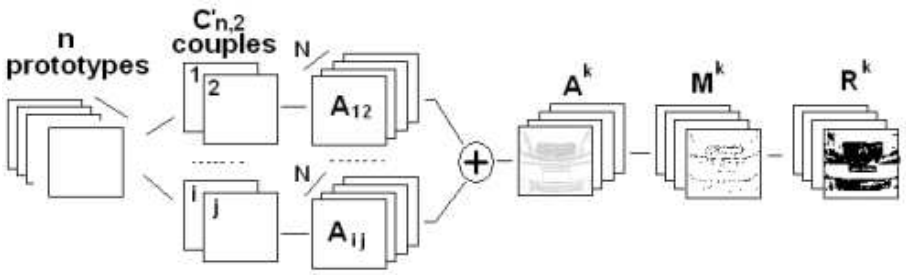


FIGURE 2.12 – Étapes pour la création du modèle de la classe k.

La figure 2.12 illustre les étapes pour la création du modèle de la classe k.

D'abord, les n prototypes ou exemples de la classe k sont traités en couples (sans répétition) pour obtenir les matrices d'accumulation $\mathbf{A}_{ij}$ issues du calcul des images moyennes. Ensuite, les valeurs maximales de $\mathbf{A}^k$, la somme des $\mathbf{A}_{ij}$, permet d'obtenir le modèle de la classe k, $\mathbf{M}^k$, qui représente la liste des points de contour orientés. Les points de $\mathbf{M}^k$ nous permettent de construire la carte des distances de Chamfer $\mathbf{C}$, seuillée avec une valeur de voisinage r pour obtenir les matrices $\mathbf{R}^k$. La matrice $\mathbf{W}_+$ est une matrice de régions de poids pondérés calculés à partir des points les plus discriminants de la classe k par rapport aux autres classes de la base, en utilisant $\mathbf{R}^k$. La matrice $\mathbf{W}_-$ pondère les points des autres classes qui ne sont pas présents dans la classe k. La liste de points $\mathbf{t}^k$ dispose dans les premiers places les points les plus discriminants de cette classe. Finalement, la liste $\mathbf{t}$ permet de sélectionner les points de la vignette qui ont une forte probabilité d'appartenir à la calandre.

Au moment de la classification, un exemple de test $\mathbf{E}_t$ est comparé à tous les modèles de classes de la base. Chaque modèle de la classe k est composé de l'ensemble : $\{\mathbf{M}^k, \mathbf{t}^k, \mathbf{W}_+, \mathbf{W}_-\}$.

Dans le chapitre suivant, nous allons présenter la fonction de classification qui nous permet de mesurer la similarité entre le test $\mathbf{E}_t$ et les modèles $\mathbf{M}^k$.

Chapitre 3

Classification

L'architecture de classification est similaire aux classifieur en parallèle de la section 5.4.3.

Nous remplaçons les classifieurs boostés pour des fonctions de similarité $g_i(\mathbf{x})$, $i = 1 \dots K$, où K est la quantité de classes dans la base et $\mathbf{x}$ l'exemple de test.

Le classifieur calcule K fonctions de similarité et détermine la fonction g_i avec le meilleur score pour classer l'entrée en conséquence (voir la figure 3.1 [45]).

$$c = ArgMax_i(g_i(\mathbf{x}))$$

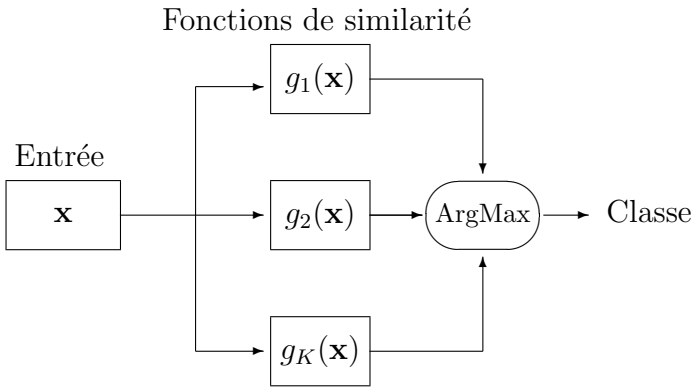


FIGURE 3.1 – Diagramme fonctionnel du classifieur génératif.

Deux types de fonctions de classification composent chaque g_i. La première fonction de classification obtient un score à travers le calcul de votes introduit dans la section 1.3.3. Nous définissons trois genres des votes : les votes positifs, les votes négatifs et les votes des classes vers l'entrée.

---

[45] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2001.

La deuxième fonction de classification calcule l'erreur commise pour une transformation homologue des pixels de l'entrée vers les pixels des modèles de classes.

Tout d'abord, nous allons limiter à une quantité T, le nombre de points de contour orientés de l'entrée $\mathbf{E}_x$ utilisés avec les fonctions de classification. Nous allons placer tous ces points dans une matrice $\mathbf{P}$ en trois dimensions : (W, H, N), où W et H représentent la taille de la vignette et N est le nombre d'orientations.

La méthodologie est simple : nous parcourons la liste de points triés $\mathbf{t}$ élément par élément. Si un de ces éléments correspond à un pixel de contour de $\mathbf{E}_x$, nous attribuons la valeur 1 à l'élément correspondant dans $\mathbf{P}$:

$$\mathbf{P}(x_i, y_i, o_i) = 1$$

où $o_i = \mathbf{E}(x_i, y_i)$.

L'opération est répétée jusqu'à avoir T points égaux à 1 dans $\mathbf{P}$. La valeur que prend T, qui donne la quantité de points de travail, est un compromis entre temps de calcul et un pourcentage désiré de classifications correctes.

3.1 Classification par votes

Pour réaliser la classification par votes, nous allons utiliser la mesure de similarité introduite dans la section 1.3.3 et l'équation 1.2 , qui calcule les votes grâce à une opération ET entre une matrice de pixels et une matrice de régions.

Votes positifs

Le score de votes positifs est incrémenté si un point de $\mathbf{P}$ est dans le voisinage d'un point de $\mathbf{M}^k$. Chacun de ces votes est pondéré par le tableau $\mathbf{W}^k_+$. Nous appliquons l'équation 1.2 (de la section 1.3.3) à $\mathbf{P}$ et $\mathbf{W}^k_+$:

$$v^k_+ = \sum_{x,y,o} \mathbf{P} \bullet \mathbf{W}^k_+$$

où $\bullet$ est une opération élément par élément.

Votes négatifs

Les votes négatifs prennent en compte les points du $\mathbf{P}$ qui ne sont pas dans le voisinage de $\mathbf{M}_k$. Nous pénalisons la classe k en accumulant ces points (pondérés par la matrice $\mathbf{W}^k_-$) :

$$v^k_- = \sum_{x,y,o} \mathbf{P} \bullet \mathbf{W}^k_-$$

Votes vers le test

Précédemment, chaque point de l'exemple de test votait pour ou contre chacun des modèles. Ici nous inversons la procédure et chaque point de chaque modèle de classe vote pour l'exemple de test. Cette procédure permet de renforcer l'algorithme de votes directs (test vers modèles) en travaillant à partir un nouveau ensemble de points, pris du modèle de la classe (les points utilisés pour les votes ne sont plus les mêmes). Sommairement, la méthode est la même que celle qui est détaillée dans l'étape précédente. Nous construisons la matrice des distances de Chamfer pour $\mathbf{P}$, les points de l'exemple de test. Après un seuillage des points qui se trouvent à une distance inférieure à r, nous obtenons la matrice des régions $\mathbf{R}$. Par la suite, nous prenons les premiers T points de la liste t^k et construisons la matrice $\mathbf{P}^k$ à partir des points du modèle $\mathbf{M}^k$. De plus, chaque point de la matrice $\mathbf{P}^k$ est pondérée par $\mathbf{W}_+^k$:

$$v_+^t = \sum_x \sum_y \sum_o \mathbf{R} \bullet \mathbf{P}^k \bullet \mathbf{W}_+^k$$

3.2 Classification par distance

L'algorithme est décomposé en deux parties. La première partie trouve des correspondances entre points de $\mathbf{P}$ et les points $\mathbf{M}^k$ du modèle de la classe k, pour calculer la matrice de transformation affine $\mathbf{M_{af}}^k$. La deuxième partie utilise cette matrice $\mathbf{M_{af}}^k$ pour calculer l'erreur de recalage qui correspond à la mesure de similarité.

Correspondance entre points

Nous allons utiliser le résultat du calcul du plan de distances de Chamfer $\mathbf{D}^k$, pour trouver les points de correspondance entre $\mathbf{P}$ et $\mathbf{M}^k$. En conservant les points de $\mathbf{P}$ qui sont à une distance plus petite que d_{ch} de $\mathbf{M}^k$, nous construisons une liste $\mathbf{p}$ et son correspondante $\mathbf{m}^k$. Pour chaque point (u_i, v_i) de $\mathbf{m}^k$ correspond un point (x_i, y_i) de $\mathbf{p}$.

$$\begin{pmatrix} u_0 & u_1 & ... & u_{n-1} \\ v_0 & v_1 & ... & v_{n-1} \\ 1 & 1 & ... & 1 \end{pmatrix} = \mathbf{M_{af}}^k * \begin{pmatrix} x_0 & x_1 & ... & x_{n-1} \\ y_0 & y_1 & ... & y_{n-1} \\ 1 & 1 & ... & 1 \end{pmatrix}$$

La matrice de transformation est calculée de la façon suivante :

$$\mathbf{M_{af}}^k = \mathbf{m}^k * \mathbf{p}^{-1}$$

où $\mathbf{p}^{-1}$ est la pseudo-inverse de $\mathbf{p}$.

Calcul de l'erreur

Nous définissons un vecteur $\mathbf{p}^k$, résultat de l'application de $\mathbf{M_{af}}^k$ au $\mathbf{p}$:

$$\mathbf{p}^k = \mathbf{M_{af}}^k * \mathbf{p}$$

L'erreur de recalage entre les points du $\mathbf{p}$ et les points $\mathbf{m}^k$ est calculé comme la moyenne des distances entre les deux ensemble des points :

$$e^k = \frac{1}{n} \sum_{i}^{n} |\mathbf{m}^k(i) - \mathbf{p}^k(i)|$$

Ce score peut être assimilé à une distance de Hausdorff modifiée [44].

3.3 Fonction de similarité

Nous allons fusionner les deux types de fonctions de classification obtenus dans la section précédente dans la fonction de similarité $g_i(\mathbf{x})$ (voir figure 3.2).

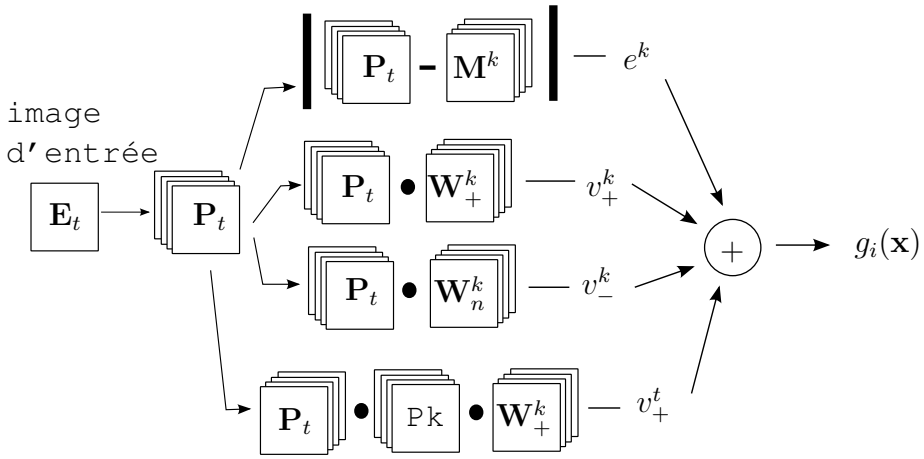


FIGURE 3.2 – Fonction de similarité : fusion des scores basés sur les votes et la distance.

Nous avons n classes au total, ce qui nous ramène à calculer n fonctions de classification $g_i(\mathbf{x})$ pour chaque exemple de test $\mathbf{x}$.

Normalisation des vecteurs de classification

Avant la fusion des vecteurs de classification, une pseudo-distance de Mahalanobis est utilisée pour normaliser les valeurs des fonctions de classification. Nous prenons par exemple les votes positifs : cette méthode de normalisation utilise les n votes positifs du test vers les classes comme un vecteur $\mathbf{v}_+$:

$$\mathbf{v}_+ = [v_+^1, v_+^2, , v_+^n]$$

[44] M.-P. Dubuisson and A.K. Jain. A modified hausdorff distance for object matching. In *IAPR International Conference on Computer Vision and Image Processing*, volume 1, pages 566–568, Octobre 1994.

Ce vecteur nous sert pour calculer une moyenne μ_+ et un écart type σ_+. Ensuite, chacun des v_+^i est normalisé à partir de (μ_+,σ_+) :

$$\bar{v}_+^k = \frac{v_+^k - \mu_+}{\sigma_+}$$

De la même façon, nous normalisons les autres sorties des fonctions de classification.

Afin de fusionner l'erreur de distances avec les votes, nous appliquons une fonction exponentielle négative (décroissante) pour attribuer un score maximum dans les cas d'une erreur minimale.

Cette normalisation nous permet d'écarter les sorties des fonctions de classification avec des valeurs faibles.

Fusion et Classification

Nous pouvons voir la fusion de ces scores comme une simple somme :

$$\begin{aligned} g_1(\mathbf{x}) &= v_+^1 + v_-^1 + v_{+1}^t + e^1 \\ g_2(\mathbf{x}) &= v_+^2 + v_-^2 + v_{+2}^t + e^2 \\ ... &= ... \\ g_n(\mathbf{x}) &= v_+^n + v_-^n + v_{+n}^t + e^n \end{aligned}$$

$$\begin{bmatrix} g_1 \\ g_2 \\ \dots \\ g_n \end{bmatrix} = \begin{bmatrix} v_+^1 \\ v_+^2 \\ \dots \\ v_+^n \end{bmatrix} + \begin{bmatrix} v_-^1 \\ v_-^2 \\ \dots \\ v_-^n \end{bmatrix} + \begin{bmatrix} v_{+1}^t \\ v_{+2}^t \\ \dots \\ v_{+n}^t \end{bmatrix} + \begin{bmatrix} e^1 \\ e^2 \\ \dots \\ e^n \end{bmatrix}$$

Sous forme vectorielle :

$$\mathbf{g} = \alpha_1 \ \mathbf{v}_+ + \alpha_2 \ \mathbf{v}_- + \alpha_3 \ \mathbf{v}_+^t + \alpha_4 \ \mathbf{e}_r \tag{3.1}$$

où nous avons ajouté les α_i qui pondèrent chaque élément de la somme. La valeur par défaut des α est 0.25, mais elle peut être déterminée par une fonction de classification, par exemple, à l'aide d'un réseau de neurones qui optimise la décision à partir des quatre entrées.

La classe c qui va être associée à l'exemple de test $\mathbf{x}$ est calculée à partir d'une simple recherche du maximum dans le vecteur des g_i :

$$\mathbf{x} \in c \ \ / \ \ c = G(\mathbf{x}) = ArgMax(\mathbf{g}(\mathbf{x}))$$

Le prochain chapitre montre les résultats que nous avons obtenu dans l'implémentation de ce système de classification.

3.4 Critère de rejet

Le système d'identification multi-classe développé répond toujours pour une des classes de la base. Nous présentons ici une approche qui permet d'obtenir une mesure de confiance pour notre système. Un exemple de test qui ne satisfait pas le critère pourra être soumis à un autre classificateur ou à un processus de fusion (par exemple, avec le système de détection de plaques ou avec un détecteur des logos) pour évaluer s'il s'agit d'un exemple mal classifié ou d'un exemple qui ne se trouve pas dans la base de données. La base d'images est complétée avec des exemples de véhicule qui n'appartiennent pas à notre base de classes connue.

Nous allons montrer par la suite, que la sortie de notre système de votes nous permettra de construire un nouveau classifieur qui sera capable de donner une mesure de confiance à la classification.

Analyse du comportement du classifieur

Dans la section 3.3, nous classifions l'exemple de test à travers la recherche du score maximum parmi les scores des fonctions de classifications placés dans le vecteur $\mathbf{g}(\mathbf{x})$.

L'analyse des valeurs du vecteur $\mathbf{g}(\mathbf{x})$ nous a permis de constater un comportement du classifieur face aux bonnes et mauvaises détections. En générale, quand la réponse du classifieur est correcte, la différence entre le deux plus grands scores dans $\mathbf{g}(\mathbf{x})$ est importante. Au contraire, si la réponse du classifieur est incorrecte, cette différence est plus faible.

Nous pouvons constater ce comportement dans la figure 3.3. La valeur de différence entre la plus grande et la deuxième plus grande fonctions de classification quand nous nous trouvons face à un véhicule bien classé correspond à l'histogramme rouge. Quand le véhicule est mal classé ou quand la classe ne se trouve dans la base, cette différence est plus faible, comme nous montre l'histogramme bleu.

Nous allons proposer une fonction de discrimination ou frontière de décision afin de déterminer la mesure de confiance. Chaque exemple de test sera projeté dans un espace où, dans les ordonnées, nous avons le score de la meilleure fonction de classification de $\mathbf{g}(\mathbf{x})$, et dans les abscisses la soustraction entre le meilleur et le deuxième meilleur score de $\mathbf{g}(\mathbf{x})$. Nous allons définir la frontière dans cet espace à l'aide de la décision bayésienne.

L'équation de la règle de Bayes est donnée par :

$$P(w_i|\mathbf{s}) = \frac{p(\mathbf{s}|w_i) \cdot P(w_i)}{\sum p(\mathbf{s}|w_i) \cdot P(w_i)}$$

où, dans notre cas, w_1, w_2 sont les deux classes qui représentent respectivement : les classifications correctes et les classifications incorrectes. $\mathbf{s}$ est le vecteur obtenu des scores de $\mathbf{g}(\mathbf{x})$ qui a les deux paramètres du nouveau espace.

Pour chaque classe w_i, nous pouvons calculer :

– $P(w_i)$, la probabilité *a priori* de cette classe,

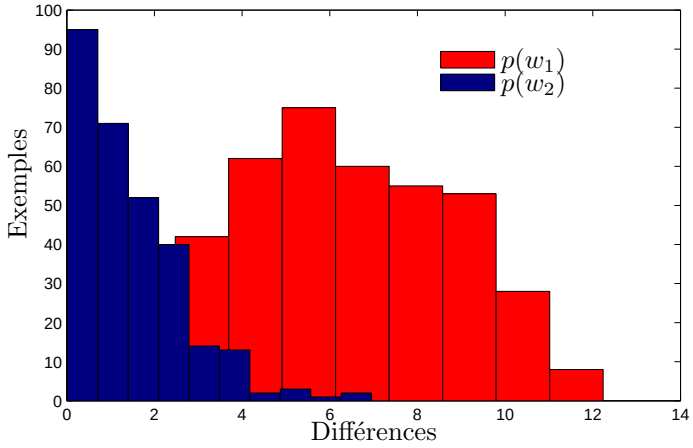


FIGURE 3.3 – Densités de probabilités des sorties du classifieur.
$p(w_1)$ représente la différence pour les classifications correctes et $p(w_2)$ la différence pour les classifications incorrectes.

– $p(\mathbf{s}|w_i)$, la densité de probabilité de $\mathbf{s}$ conditionnée par cette classe. Cette fonction de probabilité normale pour deux variables est donnée par : $N(\mu_i, \mathbf{\Sigma}_i)$. Les matrices de covariance sont différentes pour les deux classes.

La fonction discriminante peut être alors donnée par :

$$f_i(\mathbf{s}) = p(\mathbf{s}|w_i) * P(w_i)$$

ou de façon similaire, en prenant les logarithmes :

$$f_i(\mathbf{s}) = log(p(\mathbf{s}|w_i)) + log(P(w_i)) \tag{3.2}$$

L'équation 3.2 peut être évaluée en remplaçant l'expression de $p(\mathbf{s}|w_i)$ donnée par la forme de probabilité normale bi−variable :

$$f_i(\mathbf{s}) = -\frac{1}{2}(\mathbf{x} - \mu_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{s} - \mu_i) - \frac{d}{2}log(2\pi) - \frac{1}{2}log|\mathbf{\Sigma}_i| + log(\pi_i) \tag{3.3}$$

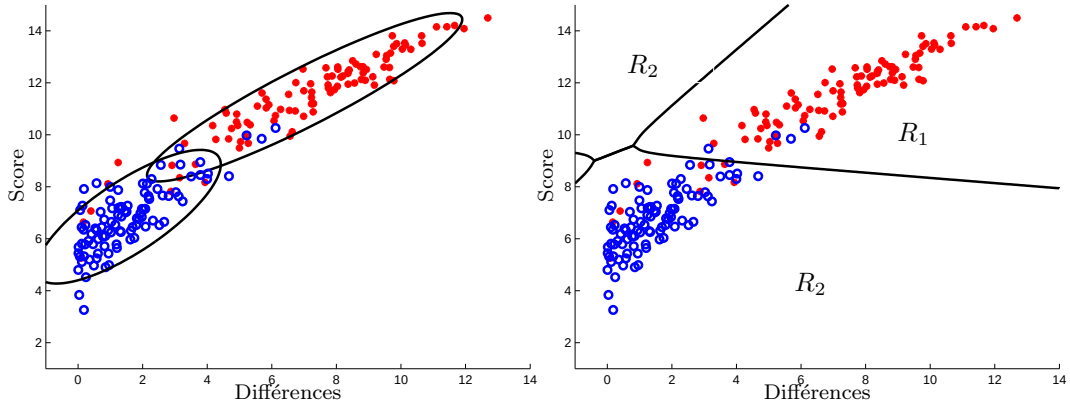
où $\pi_i = P(w_i)$.

Dans la figure 3.4 (a), nous avons dessiné les ellipses que illustrent les distributions gaussiennes des deux classes.

Nous obtenons deux fonctions discriminantes : f_1 qui correspond aux **CC** et f_2 qui correspond aux **MC+FC**. La frontière déterminée par :

$$f_1(\mathbf{x}) \ \leq \ \lambda \ f_2(\mathbf{x}) \tag{3.4}$$

avec $\lambda = 1$, est une hyperbole qui sépare le plan en deux régions R_1 et R_2 (figure 3.4 (b)). La région R_1 correspond à la classe w_1 des classifications correctes et la région R_2 à la classe w_2 des classifications incorrectes. Pour un exemple de test $\mathbf{x}$, la règle 3.4 nous permet d'évaluer s'il se place dans R_1 ou R_2.



(a) (b)

FIGURE 3.4 – Ellipses des distributions et frontière de séparation entre les deux classes. (a) Montre les ellipses correspondantes à la distribution des probabilités normale pour chaque ensemble des points et (b) la frontière de séparation obtenue.

Chapitre 4

Résultats expérimentaux et analyse comparative

Ce chapitre présente l'évaluation de notre système de classification à partir d'une base d'images de véhicules et une comparaison avec les algorithmes décrits par Petrovic [116] et Zafar [153]. Nous avons codé ces deux algorithmes et nous l'avons utilisés avec les mêmes conditions opérationnelles que notre algorithme.

4.1 Bases de véhicules

Dans cette section, nous présentons les bases d'images que nous allons employer pour tester la méthode de classification proposée.

Nous utilisons deux bases d'images de 20 types de véhicules.

Base Parking La première base est constituée de 166 images de haute qualité prises à partir d'appareils numériques dans des parkings. La résolution de ces images varie entre 1280x960 pixels et 1296x976 pixels. Les images vue de face des voiture ont été prises soit en couleur soit en niveaux de gris et à une distance proche (voir figure 4.1 (a)).

Base Galilée La deuxième base est composée de 708 images que nous pries avec deux types de caméscopes numériques, en utilisant un zoom sur une rue d'Ivry-sur-Seine. La résolution des images de cette base varie de 640x480 pixels à 720x576 pixels. La figure 4.1 (b) montrent que ces images ont été prises en différents points de vue.

La constitution des bases est précisée dans le tableau 4.1.

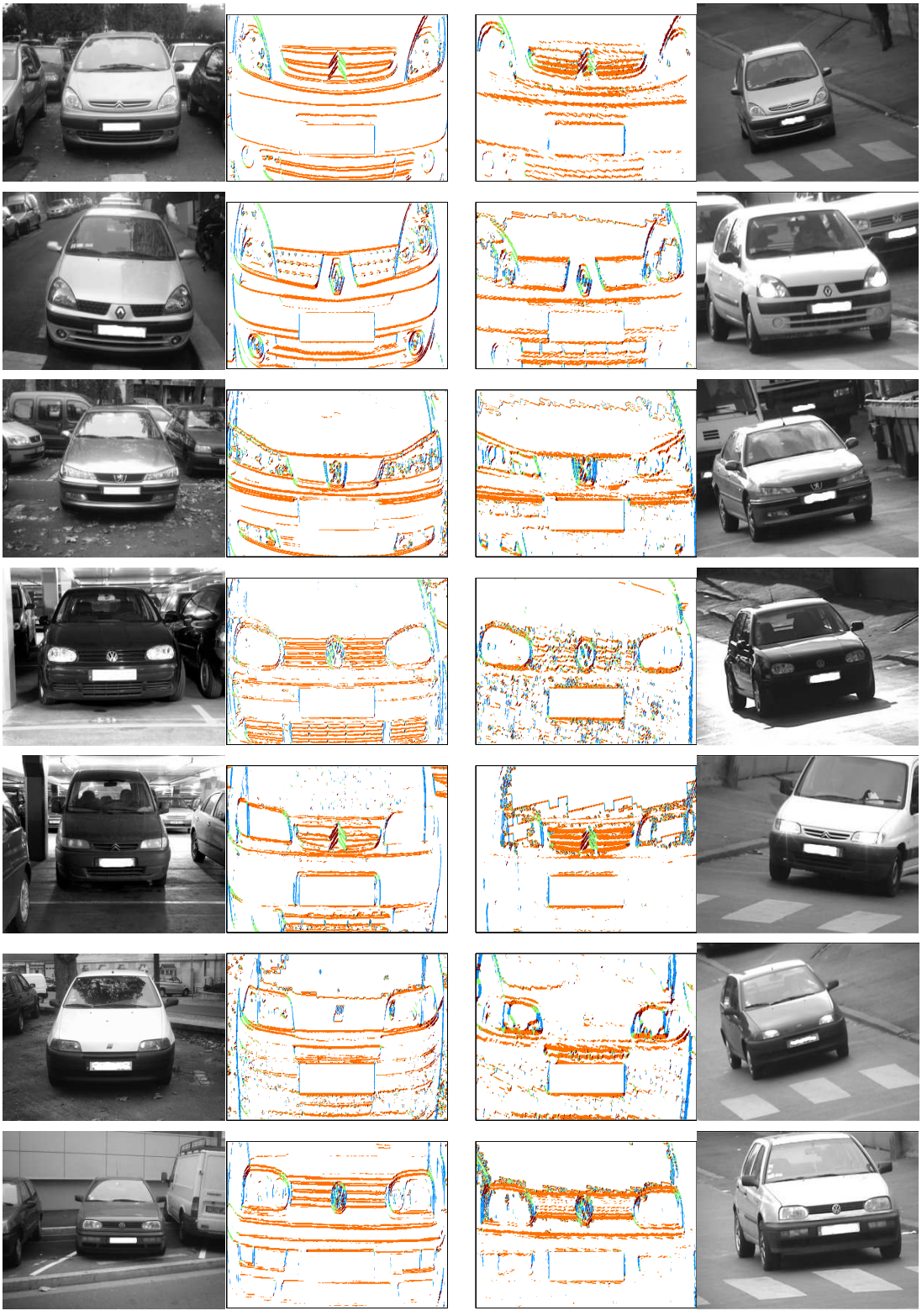
---

[116] V.S. Petrovic and T.F. Cootes. Analysis of features for rigid structure vehicle type recognition. In *British Machine Vision Conference*, volume 2, pages 587–596, Septembre 2004.

[153] I. Zafar, B.S. Acar, and E.A. Edirisinghe. Vehicle make & model identification using scale invariant transforms. In *Visualization, Imaging, and Image Processing*, Août 2007.

Modèle	**Exemples de Parking**	**Exemples Galilée**
Citroen Berlingo A	1	14
Citroen C3	1	41
Citroen Picasso B	4	28
Citroen Saxo B	9	23
Fiat Punt A	3	22
Peugeot 206 B	10	51
Peugeot 307	12	43
Peugeot 405	6	24
Peugeot 406 B	6	30
Renault 19 B	5	36
Renault Clio A	7	29
Renault Clio C	15	47
Renault Clio D	31	80
Renault Laguna B	5	34
Renault Scenic B	4	25
Renault Scenic C	4	27
Renault Twingo A	19	50
Renault Twingo B	15	34
VW Golf C	4	40
VW Golf D	5	30

TABLE 4.1 – Composition des bases de véhicules.



(a) (b)

FIGURE 4.1 – Exemples des images du même modèle des bases des images. (a) base Parking, (b) base Galilée.

4.2 Choix des paramètres

Notre système utilise quelques paramètres pour lesquels il faut choisir une valeur optimale, du point de vue de la charge de calculs et aussi du point de vue du matériel, par exemple la mémoire disponible. Ces paramètres sont :

- La quantité de points pour la classification : c'est le nombre T des pixels orientés d'un exemple de test choisis pour créer la matrice $\mathbf{P}$.
- La quantité des orientations du gradient : la valeur de la variable N quantifie les valeurs des orientations du gradient.
- Le rayon du voisinage : la valeur du rayon r qui définit les cercles de voisinage.
- La taille du modèle : cette valeur est définie à partir de la largeur de la plaque minéralogique. Nous avons fixé cette valeur à 200 pixels (voir section 2.1.2).

Dans la suite, nous allons étudier la sensibilité des résultats à chacun de ces paramètres.

4.2.1 Variation de la quantité de points de travail

Nous allons faire la classification de la base de test en faisant varier la quantité de points de travail. Nous utilisons d'habitude 2000 points, mais il pourrait être avantageux, pour réduire la charge de calculs, de travailler avec une quantité plus faible.

Les modèles sont créés avec 4 directions du gradient et un rayon de voisinage de 5 pixels. Le résultat est illustré par la figure 4.2. Un choix des points convenable serait de 2000 points où nous avons une stabilisation du pourcentage des classifications correctes : ajouter des points incrémente la charge de calculs pour le même taux de classifications correctes.

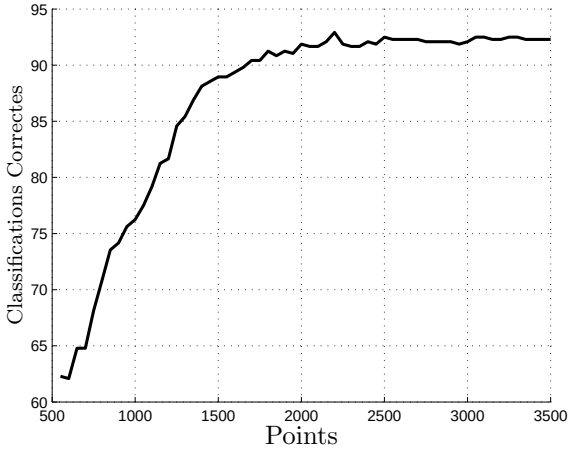


FIGURE 4.2 – Influence de la quantité de points de travail sur le taux de classifications correctes.

4.2.2 Variation de la quantité des orientations

Nous allons faire la classification de la base de test en faisant varier la quantité d'orientations N.

Nous utilisons 2000 points et un rayon de voisinage de 5 pixels. La figure 4.3 illustre le résultat obtenu. Comme nous pouvons le voir, il faut éviter de prendre un nombre d'orientations impair. En effet, un nombre d'orientation impair place certaines limites entre les orientations à 0, 90, 180 ou 270 degrés. Ainsi du fait du bruit, les pixels des contours horizontaux et verticaux auront tendance à osciller entre 2 classes d'orientation possibles. Or ce sont les points les plus nombreux sur un véhicule. Par ailleurs, nous pouvons aussi remarquer que si nous prenons un nombre d'orientation égal à 6 les résultats ne sont que légèrement meilleurs qu'avec un nombre d'orientation égal à 4. Le choix du nombre d'orientation se fera donc plus en fonction de la ressource mémoire disponible. La quantité totale de mémoire est une fonction de $WxHxNxK$, où (W, H) est la taille des vignettes, N la quantité d'orientations et K la quantité des classes dans la base.

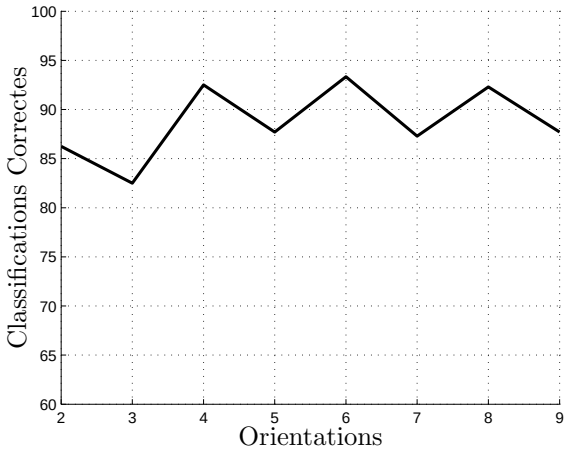


FIGURE 4.3 – Influence de la quantité des orientations sur le taux de classifications correctes.

4.2.3 Variation du rayon de voisinage

Dans ce test, nous allons faire varier le seuil appliqué à la distance de Chamfer pour obtenir la matrice $\mathbf{M}^i_{dch}$ du modèle i. Nous allons étudier une variation du rayon de voisinage entre 2 et 15 pixels.

Nous utilisons deux valeurs de T : soit égal à 550 points, le minimum de la courbe 4.2, soit égal à 2300 points, où le taux de corrections correctes se stabilise. Nous avons toujours 4 orientations du gradient. Le résultat est présenté dans la figure 4.4.

Nous constatons que si nous prenons un rayon de voisinage trop important le nombre de classifications correctes diminue. De plus, plus le nombre de points est important, plus

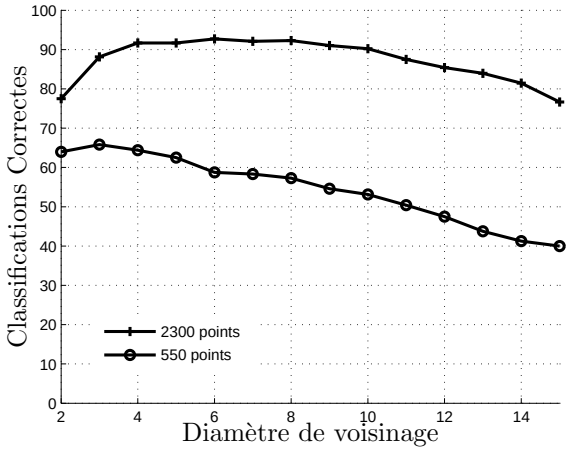


FIGURE 4.4 – Influence du rayon de voisinage sur le taux de classifications correctes.

large est la plage de valeurs possibles [1] pour le rayon de voisinage.

4.2.4 Bilan

Le choix de paramètres est fortement lié à la taille de la vignette. Travailler avec des tailles plus petites que 200 pixels de longueur de la plaque minéralogique nous oblige à réduire le nombre T de points de travail. Principalement en raison d'une réduction des points de contour trouvés dans la vignette.

L'autre paramètre influencé est le rayon de voisinage. Si la taille est réduite, il doit aussi se réduire.

Nous avons fait une étude sur la base d'images récupérée par Petrovic et utilisée plus tard par Zafar, qui nous l'a fournit déjà recadrée et sous-échantillonnée à une taille 140x70 pixels. Les paramètres $T = 750$ et $r = 3$, nous ont donné un taux de classifications correcte de 93.1 % qui améliore le taux trouvé par Zafar dans [153] sur cette même base.

Bien que nous ayons trouvé de bons résultats pour cette taille de vignette, le sous-échantillonage signifie aussi une perte d'information originale de l'image. Notamment dans le cas de la base Galilée où l'inclinaison des voitures a une forte incidence sur la position des primitives locales. Notre algorithme remplace le sous-échantillonage par le voisinage qui le rend plus robuste aux erreurs d'encadrement.

Ces paramètres sont utilisés pour l'apprentissage et pour les tests. La section suivante évalue la performance de notre système de classification et fait une comparaison avec l'existant.

1. ce sont des valeurs que l'on peut choisir sans conséquence notable sur le résultat final.

[153] I. Zafar, B.S. Acar, and E.A. Edirisinghe. Vehicle make & model identification using scale invariant transforms. In *Visualization, Imaging, and Image Processing*, Août 2007.

4.3 Résultats de Classification

Bases mélangées

La base d'apprentissage **TrB** est composée des exemples de la base Galilée et de la base Parking. Chacune des 20 classes contient le même nombre n d'images. Nous allons varier cette valeur pour analyser son impact sur les résultats. La base d'apprentissage sera ainsi composée d'une, trois, six et neuf images de véhicules par classe. Nous effectuons trois tirages au hasard pour choisir les exemples dans les bases pour chacun de ces cas. Pour le cas à une seule image l'exemple est choisi entre les images de la base Parking, étant donnée sa meilleure qualité. Le résultat final pour chaque cas, est obtenu à partir de la moyenne de ces trois tirages. Les images des bases Galilée et Parking qui ne sont pas utilisées pour l'apprentissage, sont placées dans la base de test.

Notre algorithme de classification est appliqué en utilisant deux RoI différentes (voir 4.5) : ROI1, définie par Petrovic [116] et ROI2, la région d'intérêt définie par Negri [105].

Les deux RoI prennent en compte la taille de la plaque d'immatriculation (w, h) pour calculer la vignette. La figure 4.5 nous montre la différence de tailles des deux vignettes. ROI1 fait 520 pixels (2.6 fois la largeur w de la plaque d'immatriculation) et ROI2 a 600 pixels (3 fois w). La vignette ROI2 a une vue plus importante du capot du véhicule que la ROI2.

La ROI1 se concentre plus précisément sur la calandre du véhicule. La ROI2, en ayant une taille plus importante, introduit des sources de bruit, telles que la vue hors la voitures dans ces cotés et les reflets sur le capot.

ROI 1 ROI 2

FIGURE 4.5 – Les deux RoI utilisés dans l'algorithme.

Dans tous les cas, notre algorithme obtient des meilleurs résultats que les deux autres. La ROI1 a une performance légèrement supérieure que la ROI2.

Nous constatons aussi que le fait d'augmenter le nombre des exemples d'apprentissage accroît aussi le taux de classifications correctes. Dans le cas de Petrovic et Zafar, ils comptent avec plus des échantillons dans la base de référence pour comparer les exemples

[116] V.S. Petrovic and T.F. Cootes. Analysis of features for rigid structure vehicle type recognition. In *British Machine Vision Conference*, volume 2, pages 587–596, Septembre 2004.

[105] P. Negri, X. Clady, M. Milgram, and R. Poulenard. An oriented-contour point based voting algorithm for vehicle type classification. In *IAPR International Conference on Pattern Recognition*, volume 1, pages 574–577, Août 2006.

Exemples	ROI1	ROI2	Petrovic	Zafar
1	**91.56**	81.85	86.76	69.08
3	**93.16**	91.76	91.27	90.9
6	**96.9**	96.1	95.75	93.37
9	**98.36**	97.45	96.54	95.11

TABLE 4.2 – Comparaison des taux de classification de notre algorithme et des algorithmes de la littérature.

de test. Dans notre algorithme, un nombre important d'images dans la base d'apprentissage permet d'améliorer le modèle de la classe et d'éliminer du bruit.

Cette observation est confirmée par la courbe Cumulative Match Characteristic (CMC) de la figure 4.6. La courbe CMC est utilisée pour mesurer la performance d'un système d'identification qui utilise une liste triée de candidats. Cette courbe donne la probabilité que la classe correcte de l'exemple de test soit présente dans un rang de la liste. Nous disons alors que le système reconnait au rang 1 lorsqu'il choisit la classe au meilleur score comme résultat. Ensuite, le système reconnait au rang 2, lorsqu'il considère comme bons les deux meilleurs scores de la liste.

Dans la figure 4.6 nous observons que le classifieur appris avec un et trois exemples a un comportement similaire : ils approchent du 100 % vers le septième rang. Les classifieurs appris avec six et neuf exemples arrivent au 100 % vers le cinquième rang.

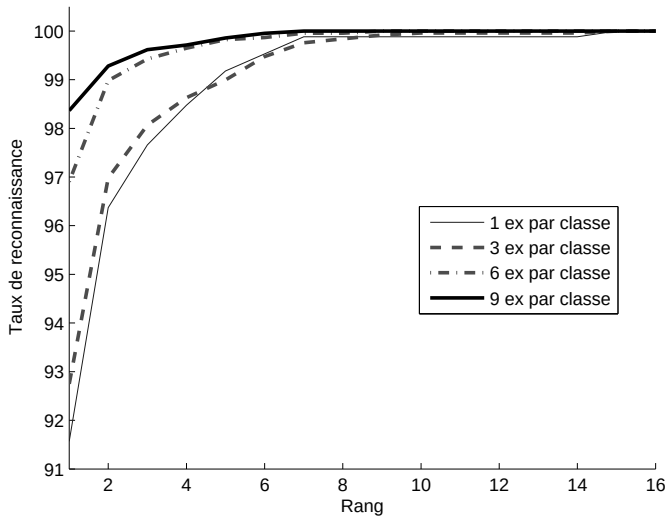


FIGURE 4.6 – Courbe CMC pour les classifieurs de notre algorithme basés sur la ROI1.

Scénario opérationnel

Dans une situation plus réaliste d'installation d'un système de reconnaissance de véhicules, l'apprentissage *off-line* est effectué à partir d'images différentes de celles qui seront acquises dans le fonctionnement final (différents points de vue, qualité de l'image, etc). Même le nombre des images d'apprentissage par classe peut être très différent. Nous pouvons simuler ce cas en utilisant la base Parking pour l'apprentissage et la base Galilée pour les tests. Nous allons donc, employer les images de haute résolution pour obtenir les modèles de classes et les images de plus mauvaise qualité, avec des rotations et prises sous différentes conditions climatiques pour les tests.

ROI1	ROI2	Petrovic	Zafar
94.2	90.11	88.84	83.75

TABLE 4.3 – Comparaison des résultats de notre algorithme et les algorithmes de la littérature pour un scénario opérationnel.

Le tableau 4.3 montre les résultats obtenus dans ce cas. Notre algorithme obtient une performance meilleure que les deux autres. Cette fois, la ROI1 obtient une différence plus importante par rapport à la ROI2 parce que cette dernière intègre dans la région d'intérêt une part importante du capot, qui est une source importante de bruit (reflets, etc.). Étant donné qu'il existe des classes comportant un seul exemple d'apprentissage, ce bruit ne peut pas être filtre. Sa présence fait, effectivement, chuter la performance des classifieurs.

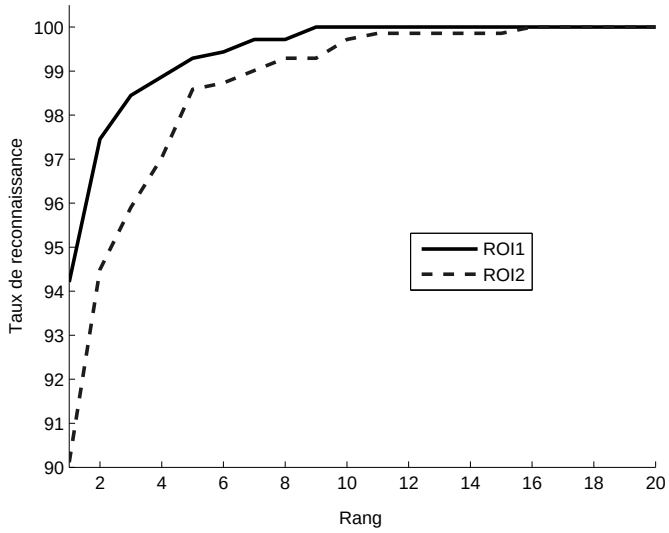


FIGURE 4.7 – Courbe CMC pour les classifieurs appris sur la base Parking.

La figure 4.7 présente les courbes CMC de notre classifieur avec les deux ROIs différentes. Nous constatons une supériorité dans la performance de la ROI1 sur la ROI2.

Le tableau 4.4 montre la matrice de confusion issue de la classification de notre algorithme avec ROI1.

	Berlingo	C3	Picasso	Saxo	Punto	206B	307	405	406B	19B	ClioA	ClioC	ClioD	Laguna	ScenicB	ScenicC	TwingoA	TwingoB	GolfC	GolfD	**Total**
Berlingo	13	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	14
C3	.	37	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	2	41
Picasso	.	.	28	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	28
Saxo	1	.	.	22	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	23
Punto	.	.	.	.	18	1	.	.	.	.	2	.	.	.	.	.	.	1	.	.	22
206B	.	.	.	.	.	51	.	.	.	.	.	.	.	.	.	.	.	.	.	.	51
307	.	.	.	.	.	.	43	.	.	.	.	.	.	.	.	.	.	.	.	.	43
405	.	.	.	.	.	.	.	23	.	1	.	.	.	.	.	.	.	.	.	.	24
406B	.	.	.	.	.	.	.	.	27	.	3	.	.	.	.	.	.	.	.	.	30
19B	.	.	.	.	.	.	.	.	.	35	.	.	.	.	.	.	.	.	1	.	36
ClioA	.	.	.	.	.	.	.	.	.	.	29	.	.	.	.	.	.	.	.	.	29
ClioC	.	.	.	.	.	.	.	.	.	.	1	45	1	.	.	.	.	.	.	.	47
ClioD	.	.	.	.	.	.	.	1	.	.	.	1	78	.	.	.	.	.	.	.	80
Laguna	.	.	.	.	.	3	.	2	2	.	.	.	.	27	.	.	.	.	.	.	34
ScenicB	.	.	.	.	.	.	.	.	.	.	.	1	.	.	24	.	.	.	.	.	25
ScenicC	.	.	.	.	.	.	.	.	1	.	.	.	2	.	.	23	.	.	.	1	27
TwingoA	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	49	1	.	.	50
TwingoB	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	34	.	.	34
GolfC	.	.	.	.	.	.	.	1	.	2	.	.	.	.	.	.	.	.	37	.	40
GolfD	.	.	.	.	.	.	.	6	.	.	.	.	.	.	.	.	.	.	.	24	30

TABLE 4.4 – Matrice de confusion sur la ROI1.

Résultat : **94.2 %** des classifications correctes.

4.4 Résultats avec le critère de rejet

Bases d'images

Nous effectuons d'abord un apprentissage de notre système de classification par votes à partir d'une base composée de 9 exemples par classe.

Les autres 694 exemples appartenant aux nos 20 classes, vont composer la base de test. Nous complétons cette base avec 397 exemples de véhicules des classes inconnues pour notre système.

Les résultats de la classification du système de votes dans la base peuvent être :

- classifications correctes (**CC**),
- mauvaises classifications (**MC**), les exemples qui sont dans la base de test, mais mal classifiés par le système,
- fausses classifications (**FC**), les véhicules dont le type sera considéré comme appartenant à la base de classes, tandis qu'il n'en fait pas partie.

Résultats

Nous avons divisée en deux la base de test, ainsi que lŠensemble des exemples des classes inconnues. La première moitié est utilisé pour apprendre les deux fonctions de classification f_1 et f_2 de la règle 3.4, que nous transcrivons ici :

$$f_1 < \lambda f_2$$

où f_1 correspond aux classifications correctes et f_2 aux classifications incorrectes.

Nous définissons ainsi les frontières qui séparent les régions R_1 et R_2 dans l'espace de classification pour λ égal à 1.

Ensuite, nous avons projeté la deuxième moitié des exemples de test dans cet espace. 93.7 % des exemples appartenant à la base des véhicules connus ont été bien classés (région R_1). Seuls 5 % des exemples de la base de véhicules inconnus nŠont pas été rejetés. De la même manière, 18.2 % des exemples appartenant à la base des véhicules connus ont été rejetés. Nous avons donc finalement 93.6 % de bonnes classifications (entre classifications correctes et rejets).

Nous pouvons construire un courbe ROC en faisant varier la valeur de λ (voir figure 4.8). La valeur finale de λ dépend de l'application. Si le système final est combiné avec d'autres méthodes d'identification (lecture de plaque d'immatriculation, cartes électroniques, etc), nous pouvons nous placer dans un point de travail avec un taux de **CC** important avec un taux de **MC** aussi élevé, étant donné que l'erreur totale sera compensé dans la fusion.

En conclusion, ce critère maximise la quantité des exemples **CC** et minimise le nombre des exemples **MC** et **FC** qui sont dans R_1. Nous pouvons considérer comme le meilleur résultat obtenu, étant donné que les deux classes définies dans ce rapport ne sont pas séparables.

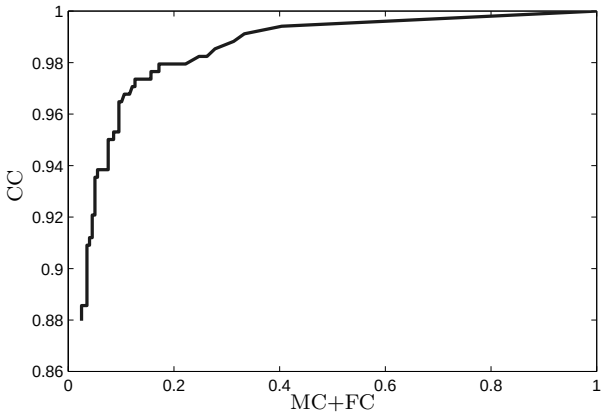


FIGURE 4.8 – Sensibilité au seuil sur le critère de confiance.

En conclusion, ce critère maximise la quantité des exemples **CC** et minimise le nombre des exemples **MC** et **FC** qui sont dans R_1. Nous pouvons considérer comme le meilleur résultat obtenu, étant donné que les deux classes définies dans ce rapport ne sont pas séparables.

4.5 Présence de la barrière

Dans la littérature, parmi les systèmes conçus pour être installés dans des parkings, aucun auteur n'a évalué leur performance face aux occultations de la barrière.

Dans cette section nous allons simuler la présence de la barrière du parking comme dans les exemples réels de la figure 1.4.

Étant donné que les images dans les bases ne sont pas prises dans un contexte d'accès au parking, elles n'ont pas une barrière que cache une partie de la calandre. Nous simulons sa présence qui cache la calandre à trois endroits différents, comme nous pouvons observer dans la figure 4.9.

Position 1 Position 2 Position 3

FIGURE 4.9 – Les trois positions d'une barrière virtuelle.

Nous présentons dans le tableau 4.5 les résultats obtenus pour le scénario opérationnel.

Nous remarquons une chute de la performance des classifieurs par rapport aux résultats sans occultations. Notre algorithme continue à avoir un meilleur comportement que celui de Petrovic.

Une analyse des résultats par position de la barrière nous permet de constater que la première position n'a pas une incidence dans les résultats. La deuxième position, par

Barriere	ROI1	ROI2	Petrovic
1	**93.22**	88.84	89.4
2	**92.09**	87.14	87.82
3	**93.22**	89.40	90.11

TABLE 4.5 – Résultats de reconnaissance en présence d'occultations dans le scénario opérationnel.

contre, peut cacher des logos qui représentent des caractéristiques discriminantes de la calandre. La dernière position de la barrière a un meilleur taux de reconnaissance que les deux autres.

Dans le chapitre suivant, nous allons conclure cette partie et proposer quelques perspectives.

Chapitre 5

Conclusions

Nous avons présenté dans cette deuxième partie, un système de votes pour réaliser la classification du type de véhicule (marque et modèle) basé sur les pixels de contour orientés. Une image moyenne de contours est utilisée pour construire le modèle de chaque classe. Nous proposons une représentation en régions pondérées voisines des pixels du modèle pour calculer les votes de manière rapide. La fonction discriminante combine les scores obtenus de trois types de classifieurs basés sur les votes et une sur l'erreur de recalage. Notre méthode obtient de meilleurs résultats que d'autres systèmes présentés dans la littérature. Les occultations d'une partie de la calandre des véhicules n'ont pas une incidence importante sur les résultats. Nous avons présenté finalement une méthode pour évaluer la réponse du classifieur. Notre système est aussi capable de réaliser une vérification que le type de véhicule soit le même pour deux images différentes (par exemple à l'entrée et sortie d'un parking). Enfin, une méthode pour obtenir une mesure de confiance dans les résultats est présentée, basée sur un critère de rejet. Elle s'avère utile dans le cas où le système est fourni avec des images de nouvelles classes de véhicules ou s'il existe de doutes à la réponse du classifieur.

Depuis la fin des travaux de thèse présentés dans ce mémoire, plusieurs améliorations de la méthode développée ont été proposées [106, 30], notamment pour accroître la robustesse face à un nombre plus important de modèles. En effet, le système a une perte importante de performance quand sont ajoutées plus de classes à la base. Pour 50 modèles différents, nous obtenons des performances de l'ordre de 80 % en micro précision (pour la ROI2).

Dans [106] nous proposons une cascade dynamique qui permet de réduire l'ambiguïté entre différents modèles. Nous avons constaté un gain global de l'ordre de 3-4 %. Pour les modèles le plus ambigus, le gain de taux de reconnaissance individuel peut atteindre le

[106] P. Negri, X. Clady, M. Milgram, and R. Poulenard. Système de reconnaissance multi-classes du type de véhicules. In *Congrès francophone des jeunes chercheurs en vision par ordinateur, ORASIS*, Juin 2007.

[30] X. Clady, P Negri, M Milgram, and R Poulenard. Multi-class vehicle type recognition system. In *IAPR International Workshop on Artificial Neural Networks in Pattern Recognition*, pages 228–239, 2008.

20 %. Dans [30], nous avons testé d'autres stratégies de décision que le winner-take-all. Le principe appliquée a consisté à concaténer dans un vecteur les sorties des fonction de classification (votes et distance) entre l'image à tester et les modèles. Le vecteur obtenu est une nouvelle représentation de l'image dans l'espace de classification. Une procédure de plus proche voisins permet alors de classifier le véhicule à tester par rapport à des véhicules de référence dans cet espace. Cette approche améliore sensiblement les résultats en taux de reconnaissance : pour 50 modèles de véhicules, la meilleure stratégie atteints 93 %.

[30] X. Clady, P Negri, M Milgram, and R Poulenard. Multi-class vehicle type recognition system. In *IAPR International Workshop on Artificial Neural Networks in Pattern Recognition*, pages 228–239, 2008.

CONCLUSION GENERALE

Cette thèse est vouée à la recherche de méthodes de reconnaissance de formes avec application dans le domaine du transport intelligent.

Les problématiques étudiées concernent deux types différents d'algorithmes d'apprentissage supervisé. Le premier décrit une approche de détection et classification en temps réel des objets. Ce type de problème peut compter habituellement avec des bases des données de grande taille permettant de généraliser l'aspect ou l'apparence des classes d'objets, tels que les véhicules de tourisme, camions, etc. Le deuxième classifie les objets à partir de leurs caractéristiques ou formes, telle que la calandre des voitures ; toutefois les bases des données disponibles sont limitées au point de se retrouver parfois avec un seul exemple par classe.

Nous avons abordé la détection de véhicules via une architecture en cascade obtenue à la manière de Viola et Jones [142] ; l'algorithme Adaboost étant utilisé pour l'apprentissage de l'ensemble.

La performance de cette méthode a été évaluée pour deux familles de descripteurs : les filtres rectangulaires, associés à des fonctions de classification de type discriminant et les histogrammes de gradients orientés, liés à des fonctions de type génératif. De plus, nous avons proposé une approche mixte qui combine les deux espaces. L'architecture de ce détecteur Mixte, obtenue automatiquement par Adaboost, est composée de classifieurs génératifs dans les premiers étages de la cascade. Ces classifieurs permettent d'éliminer les exemples négatifs qui sont " loin " du modèle de véhicule. Puis Adaboost sélectionne majoritairement des classifieurs discriminants dans les derniers étages de la cascade afin de filtrer les exemples négatifs les plus difficiles. Enfin, cette combinaison permet d'optimiser le comportement de la cascade sur trois points fondamentaux : le taux de détections correctes, l'élimination de fausses alarmes et la réduction du temps de traitement.

Le second problème abordé dans la deuxième partie de ce mémoire, la reconnaissance de type (modèle et constructeur) de véhicules, requiert une approche différente de la

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

précédente. La méthode générative qui a été développée à cet effet emploie les pixels de contour orientés comme espace de paramètres. Ces descripteurs permettent de modéliser les classes de véhicules à travers une méthode qui détermine une image moyenne des contours orientés pour chaque modèle de sorte à conserver les points les plus représentatifs de la classe. Sa robustesse face aux changements des images de test (différents éclairages, qualité, etc.) est un avantage non négligeable du point de vue opérationnel. Pour l'étape de classification, la fonction de discrimination que nous avons proposée utilise les votes comme mesure de similarité. Chaque vote est pondéré par une matrice qui gratifie les pixels les plus discriminants d'un modèle de classe par rapport aux autres classes de la base. Enfin, l'analyse de nos résultats de classification permet d'estimer une mesure de confiance sur les réponses pour en valider certaines ou pour requérir éventuellement un post-traitement. Grâce à ces deux apports (utilisation des pixels de contour orientés comme descripteurs et classification à travers le vote des pixels les plus discriminants), cette méthode s'avère la plus performante, la plus robuste aux occultations partielles et la plus fiable parmi les méthodes décrites dans la littérature récente.

Les travaux ici présentés peuvent être le point de départ du développement de systèmes sur la reconnaissance et détection de véhicules. A cette fin, plusieurs autres aspects doivent être mis en oeuvre.

Par rapport à la première méthode de détection de véhicules, nous pourrions l'enrichir en fusionnant l'information fournie par d'autres capteurs, tels que le radar, le laser, etc., ainsi que par d'autres algorithmes, comme la détection de la route [4]. Leur incorporation permettrait d'apporter une information précieuse pour éliminer les fausses alarmes et focaliser la recherche des véhicules sur des régions d'intérêt mieux délimitées. De plus, cela réduirait considérablement le temps de traitement.

D'autres méthodes de dopage (Adaboost Réel) et architectures en cascade (de type *Nested*) sont envisageables pour améliorer les performances de notre approche. Quelques premiers essais dans ce sens ont été réalisés. Concernant Adaboost Réel, les résultats que nous avons obtenu pour la détection de véhicule ont été similaires à ceux du Adaboost Discret. Nous pensons cependant l'utiliser dans un avenir proche afin d'améliorer la phase de classification. En effet, les sorties réelles des classifieurs faibles permettraient de réduire l'espace de descripteurs à l'aide d'un ACP, particulièrement dans le cas du classifieur multi-modèle. Concernant la cascade de type *Nested*, les tests réalisés ont montré une meilleure discrimination des exemples positifs et négatifs, notamment pour les détecteurs entraînés avec les trois classes différentes.

En ce qui concerne notre méthode de reconnaissance du type de véhicule, nous avons

[4] R. Aufrère, R. Chapuis, and F. Chausse. A dynamic vision algorithm to locate a vehicle on a non-structured road. *International Journal of Robotics Research*, 19(5) :411–423, Mai 2000.

étudié ses limites à partir de l'inclusion de nouvelles classes dans la base. Ce dernier travail nous a permis de développer des nouvelles méthodes pour contourner cette contrainte. Il reste d'autres aspects à parfaire. Notamment notre méthode nécessite une détection de la plaque d'immatriculation relativement précise. Nous avons commencé à travailler sur un algorithme de détection de la plaque ; les résultats développés par une stagiaire, Mlle ABDALLAH, sont encourageants. Une autre approche consisterait à développer une représentation robuste aux changement de perspective. L'espace de paramètres, tels que les "sacs des mots" [108], serait par exemple envisageable.

Ces travaux ont donné lieu à un projet CEDRE de collaboration entre l'Université du Liban et l'ISIR. Celui-ci a pour but la conception d'un système complet de détection de véhicule, de lecture de la plaque de immatriculation et de reconnaissance du modèle de véhicule (LAPI+VMMR).

---

[108] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

Annexe A

Cascade Attentionnelle : analyse de sensibilité

Dans cet annexe, nous allons étudier la sensibilité de l'apprentissage en cascade à deux paramètres :DC_{min} et FA_{max}.

Nous avons entraîné des détecteurs Mixtes en cascade, sans contrôle du nombre de fonctions faibles par étage (Cascade Contrôlé) en faisant varier DC_{min} et FA_{max}. La base d'apprentissage utilisée contient 1490 véhicules de la classe 1 (voitures de tourisme). Dans le tableau A.1 nous présentons les résultats obtenus

DC_{min} (%)	FA_{max} (%)	# Etages	# Descripteurs	DC (%)	FA
95	40	11	53	82.7	0.00035
95	50	12	44	81.2	0.00035
99	40	12	153	90.4	0.0010
99	50	14	143	90.47	0.0008
99.5	40	13	234	95.57	0.00128
99.5	50	16	292	95.7	0.0014

TABLE A.1 – Tableau de résultats pour différentes cascades obtenues de faire varier les paramètres d'apprentissage.

Le tableau nous confirme que choisir une valeur de DC_{min} de plus en plus important augmente le taux de détections correctes (DC). En même temps, le nombre de fausses alarmes augmente aussi, car le classifieur doit être moins discriminant pour valider un plus grand nombre des exemples positifs.

La figure A.1 montre que les meilleurs résultats (DC et FA) sont obtenus en choisissant $DC_{min} = 99.5$. De plus, le choix de FA_{max} égal à 40 % ou égal à 50 % donne de résultats semblables. La différence principale se trouve dans la constitution du détecteur dans le nombre d'étages. Les détecteurs avec $FA_{max} = 40$ minimisent le nombre d'étages, étant donnée qu'ils arrivent au taux de fausses alarmes totale FA_{max} plus vite que les détecteurs avec $FA_{max} = 50$. Cependant, ils sont construits avec des classifieurs forts plus complexes (une quantité plus importante de fonctions faibles), qui arrivent à éliminer le taux de

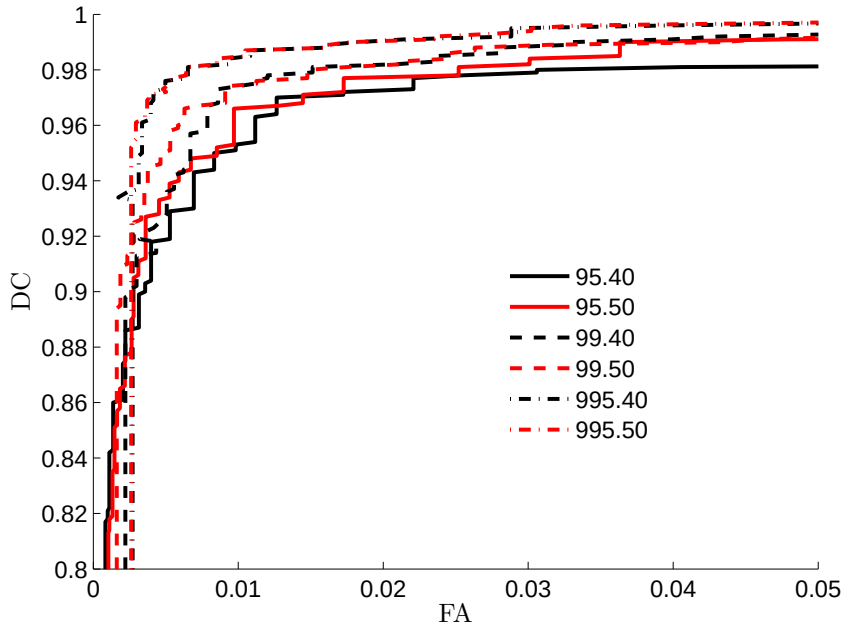


FIGURE A.1 – Courbes ROC des détecteurs entraînés avec différents paramètres d'apprentissage.

fausses alarmes souhaité.

Nous choisissons pour nos détecteurs $DC_{min} = 99.5$ et $FA_{max} = 40$, pour le bon taux de DC total et la quantité minimale des étages (cascade plus rapide).

Annexe B

Performances des classifieurs

Nous présentons les performances en détection et classification obtenues pour les quatre architectures de classification de véhicule du chapitre 5. Les performances montrent les résultats de chaque architecture en fonction du critère de recouvrement.

Dans la première ligne des figures, nous avons placé le taux de détections correctes (DC) et le nombre total de fausses alarmes (FA) obtenues dans la base de test. Les courbes de Det/VT indiquent le nombre des détections par véhicule (ou vérité terrain) qui remplissent le critère de recouvrement. Les courbes Clf/VT calculent le taux de classifications correctes par véhicule entre toutes ces détections. Dans la dernière ligne nous avons placé le taux de micro précision et de macro précision.

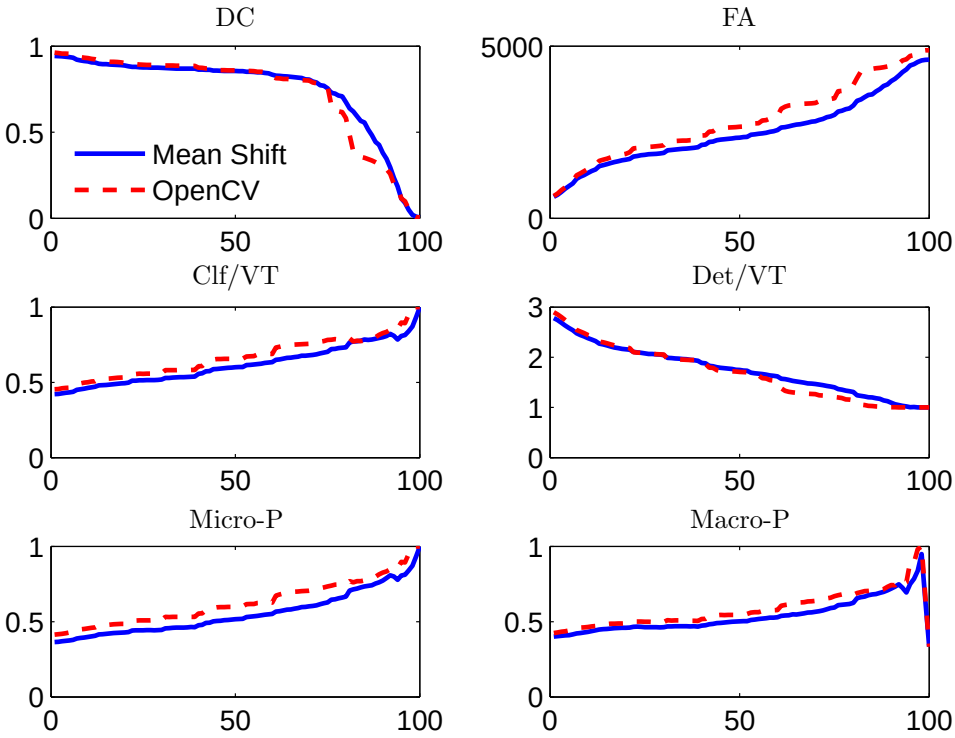


FIGURE B.1 – Performances pour les détecteurs individuels.

La figure B.1 montre que les résultats obtenus pour les deux méthodes de regroupement

sont similaires. Cela est dû au nombre faible des hypothèses générés par les détecteurs.

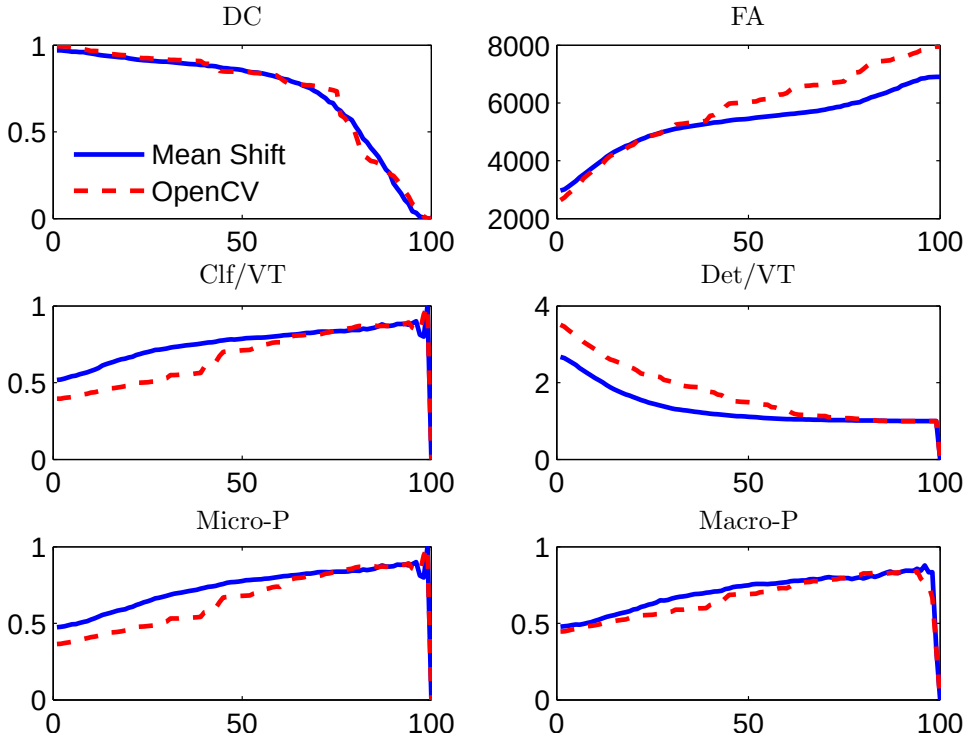


FIGURE B.2 – Performances pour le classifieur pyramidal.

Les statistiques du classifieur pyramidal montrent une meilleur performance de la méthode *Mean Shift* en minimisant les fausses alarmes et la quantité de détections par véhicule. Le taux de classification par détection est maximale par rapport aux autres architectures.

Le classifieur en parallèle montre une meilleure performance en classification pour la méthode de filtrage *OpenCV* pour des taux de recouvrement supérieures à 50 %.

La figure B.4 qui présente les statistiques pour le classifieur multi-modèle montre que la méthode de filtrage de *Mean Shift* obtient les plus mauvais résultats, soit en détection, soit en classification.

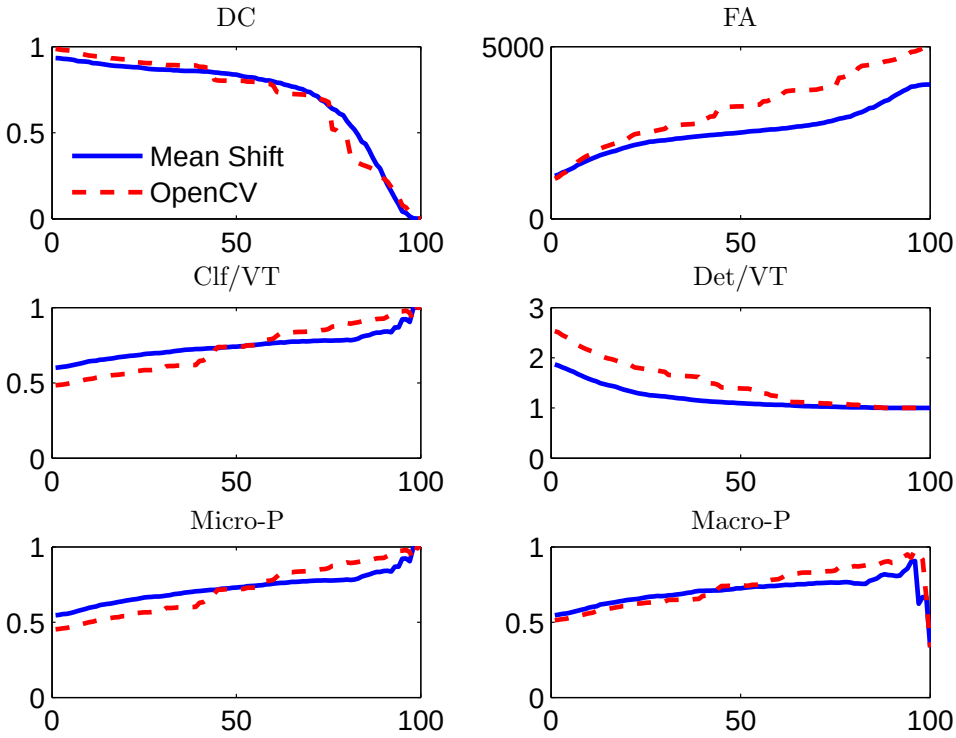


FIGURE B.3 – Performances pour le Classifieur en Parallèle.

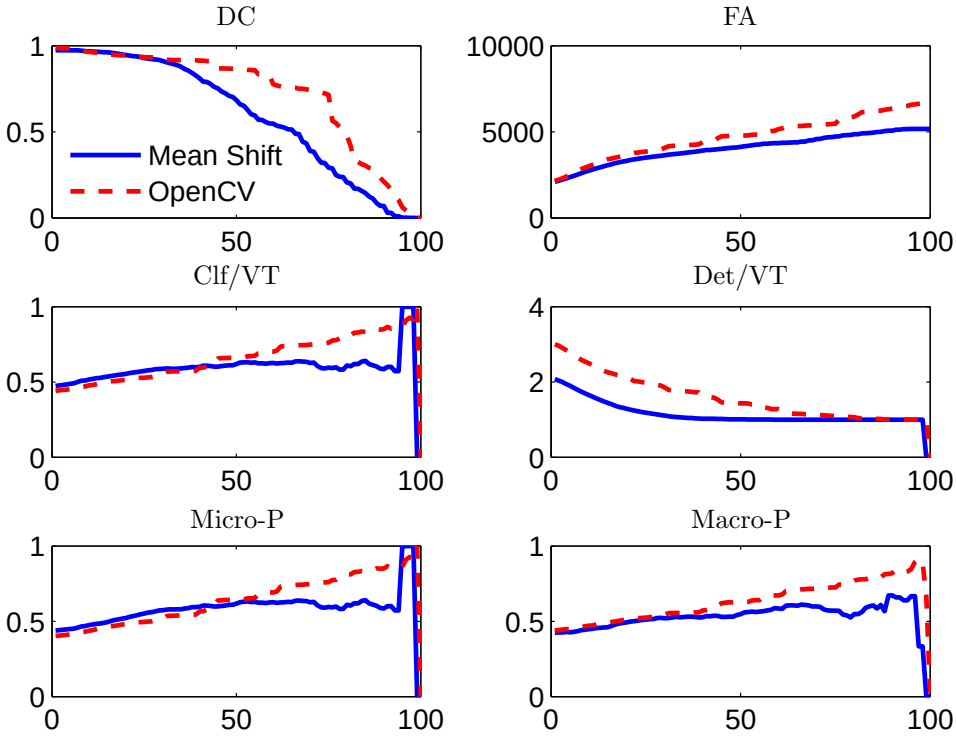


FIGURE B.4 – Perforamances pour le Classifieur Multi-Modèle.

Annexe C

Alignement des imagettes par couples

L'objectif est l'alignement des contours d'un couple d'exemples de la même classe. Cette tâche est réalisée pour résoudre des erreurs commises au moment de réaliser le redressement (section 2.1.1). Par exemple, il peut avoir des erreurs dans l'algorithme qui cherche la position des coins de la plaque minéralogique et les vignettes obtenues à partir de ces points peuvent être donc décalées. Cet alignement nous permettra de mieux trouver une image des contours moyenne entre les exemples d'apprentissage, pour obtenir le modèle de la classe.

Nous prenons, par exemple, une classe qui compte avec 4 exemples dans la base d'apprentissage où chaque exemple i est représenté par la matrice $\mathbf{E}_i$. Cela nous fait 6 couples différents à travailler (cf. figure C.1). Dans la section 2.2.2, nous avons montré la méthode pour obtenir le modèle de classe à partir de ces couples. D'abord, nous obtenons l'image moyenne qui corresponde A partir d'une couple $(\mathbf{E}_i, \mathbf{E}_j)$ nous obtenons une matrice d'accumulation des votes $\mathbf{A}_{ij}$.

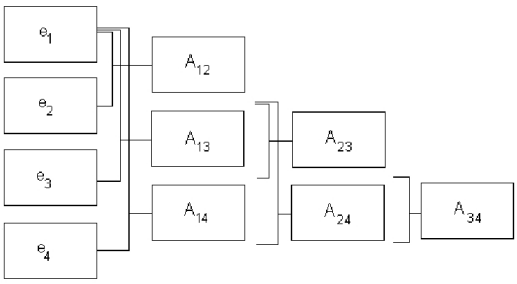


FIGURE C.1 – Creation des matrices d'accumulation des votes $\mathbf{A}_{ij}$ à partir des images $\mathbf{E}_i$ et $\mathbf{E}_j$.

Les matrices $\mathbf{E}_1$ et $\mathbf{E}_2$ représentent les pixels de contour orientés de l'exemple 1 et 2 respectivement. Nous allons trouver une matrice de Transformation Affine (section 2.1.1) pour aligner l'exemple 2 vers 1. Deux matrices des points de contour $\mathbf{v}_1$ et $\mathbf{v}_2$ créés à partir des $\mathbf{E}_1$ et $\mathbf{E}_2$ respectivement, vont être utilisés pour trouver la matrice $\mathbf{M}_{af}$:

$$\mathbf{v}_1 = \mathbf{M}_{af} \cdot \mathbf{v}_2$$

La matrice $\mathbf{v}_1$ est composé de Q points $\mathbf{p}_i$ de $\mathbf{E}_1$ choisis au hasard.

Ensuite, nous construisons la matrice $\mathbf{v}_2$ en cherchant, pour chaque point de $\mathbf{v}_1$, le point de $\mathbf{E}_2$ le plus proche (distance euclidienne) avec la même direction du gradient.

Pour le calcul de la matrice de Transformation Affine, nous changeons par 1 la dernière file des matrices $\mathbf{v}_1$ et $\mathbf{v}_2$ et gardons seulement la position (x, y) de chaque pixel $\mathbf{p}$:

$$\mathbf{v}_1 = \begin{pmatrix} x'_0 & x'_1 & ... \\ y'_0 & y'_1 & ... \\ o'_0 & o'_1 & ... \end{pmatrix} \quad \longrightarrow \quad \mathbf{v}_1 = \begin{pmatrix} x'_0 & x'_1 & ... \\ y'_0 & y'_1 & ... \\ 1 & 1 & ... \end{pmatrix}$$

et

$$\mathbf{v}_2 = \begin{pmatrix} x_0 & x_1 & ... \\ y_0 & y_1 & ... \\ o_0 & o_1 & ... \end{pmatrix} \quad \longrightarrow \quad \mathbf{v}_2 = \begin{pmatrix} x_0 & x_1 & ... \\ y_0 & y_1 & ... \\ 1 & 1 & ... \end{pmatrix}$$

Conformément à 2.2, nous trouvons la matrice de Transformation Affine :

$$\mathbf{M}^k_{af} = \mathbf{v}_1 \cdot \mathbf{v}_2)^{-1}$$

où $\mathbf{v}_2)^{-1}$ corresponde à la pseudoinverse de $\mathbf{v}_2$).

Afin de rendre robuste ce calcul, nous appliquons une procédure dit "médiane". Nous allons calculer un nombre K des matrices de Transformation Affine, différente à chaque fois due au choix au hasard des points de $\mathbf{v}_1$. Après K itérations, nous obtenons la matrice $\mathbf{M}_{af}$ finale, de la médiane des valeurs de tous les $\mathbf{M}^k_{af}$. Ainsi donc, nous trouvons le meilleur alignement de l'exemple 2 vers le 1. Puis, la position (x, y) des éléments du vecteur de pixels orientés de $\mathbf{E}_2$ est recalculée avec la matrice $\mathbf{M}_{af}$. Nous conservons la direction du gradient original :

$$\begin{pmatrix} x'_0 & x'_1 & ... \\ y'_0 & y'_1 & ... \\ 1 & 1 & ... \end{pmatrix} = \mathbf{M}_{af} \cdot \begin{pmatrix} x_0 & x_1 & ... \\ y_0 & y_1 & ... \\ 1 & 1 & ... \end{pmatrix}$$

$$\mathbf{e}'_2 = \begin{pmatrix} x'_0 & x'_1 & ... \\ y'_0 & y'_1 & ... \\ o_0 & o_1 & ... \end{pmatrix}$$

Annexe D

Bibliographie Personnelle

Nous présentons ici les travaux de recherche effectuées par Pablo Augusto Negri au sein du laboratoire ISIR.

Revue Internationale

- *Pablo Negri, Xavier Clady, Shehzad Muhammad Hanif, Lionel Prevost* "**A cascade of boosted generative and discriminative classifiers for vehicle detection**", EURASIP Journal on Advances in Signal Processing, vol. 2008, Article ID 782432, 12 pages, 2008. doi :10.1155/2008/782432.

Congrès Internationaux

- *Shehzad Muhammad Hanif, Lionel Prevost, Pablo Negri* "**A Cascade Detector for Text Detection in Natural Scene Images**" , International Conference on Pattern Recognition, 8-11 Decembre 2008, USA.
- *Shehzad Muhammad Hanif, Lionel Prevost, Pablo Negri* "**Text Detection in Natural Scene Images Using a Cascade of Boosted Ensembles** " , Colloque International Francophone sur l'Ecrit et le Document, 28-31 Octobre 2008, France.
- *Xavier Clady, Pablo Negri, Maurice Milgram* "**Multi-class Vehicle Type Recognition System**" , The Third International Workshop on Artificial Neural Networks in Pattern Recognition, 2-4 Juillet 2008, France.
- *Pablo Negri, Xavier Clady, Lionel Prevost* "**Benchmarking Haar and Histograms of Oriented Gradients features applied to vehicle detection**", Fourth International Conference on Informatics in Control, Automation and Robotics, 9-12 Mai, 2007, Angers, France.
- *Pablo Negri, Xavier Clady, Lionel Prevost, Maurice Milgram, Raphael Poulenard* "**Multiclass Vehicle Type Recognition System using an Oriented Points Set based Model**", Fourth IEEE International Multi-Conference on Systems, Si-

gnals & Devices SSD'07, 19-22 Mars, 2007, Tunisie.

- *Pablo Negri, Xavier Clady, Lionel Prevost, Maurice Milgram, Raphael Poulenard* "**An Oriented-Contour Point Based Voting Algorithm for Vehicle Type Classification**", International Conference of Pattern Recognition, Hong Kong, 20-24 Aout 2006, Pages 574 - 577.
- *Pablo Negri, Xavier Clady, Maurice Milgram* "**A New Voting Algorithm for Human Grasping Gestures**", Advanced Concepts for Intelligent Vision Systems, Anvers, Belgique, septembre 2005, Volume 3708, Pages 130 - 137.

Congrès Francophones

- *Pablo Negri, Lionel Prevost, Xavier Clady*, "**Cascade de classifieurs génératifs et discriminants pour la détection de véhicules**" , 16e Congrès francophone AFRIF-AFIA, Reconnaissance des Formes et Intelligence Artificielle, 22-25 Janvier 2008, Amiens, France.
- *Xavier Clady, Pablo Negri, Maurice Milgram, Raphael Poulenard*, "**Reconnaissance multiclasses de type de véhiclues à l'aide d'algorithme de votes sur des contours orientés**" , 21 Colloque GRETSI, 11-14 Septembre 2007, Troyes, France.
- *Pablo Negri, Xavier Clady, Maurice Milgram, Raphaël Poulenard* "**Système de reconnaissance multi-classes de type de véhicules**", Congrès Francophone des Jeunes Chercheurs en Vision par Ordinateur, 4-8 juin, 2007, Obernai, France.
- *Pablo Negri, Xavier Clady, Lionel Prevost, Maurice Milgram, Raphael Poulenard* "**Reconnaissance par vision du type d'un véhicule automobile**", MAnifestation des Jeunes Chercheurs STIC, 22-24 Novembre 2006, Lorient, France.
- *Pablo Negri, Xavier Clady, Maurice Milgram*, "**Perception visuelle du geste de préhension**", 20e Colloque GRETSI Louvaine-la-Neuve, septembre 2005, Belgique.
- *Pablo Negri, Xavier Clady, Maurice Milgram* "**Perception visuelle du geste de préhension pour des applications robotiques**", 18e Journées des Jeunes Chercheurs en Robotique, Ecole des Mines de Douai, Septembre 2004.

Bibligraphie

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11) :1475–1490, 2004.

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, pages 113–130, Londres, Royaume Uni, 2002. Springer-Verlag.

[3] B. Alefs. Embedded vehicle detection by boosting. In *Intelligent Transportation Systems Conference*, pages 536–541, Vienna, 2006.

[4] R. Aufrère, R. Chapuis, and F. Chausse. A dynamic vision algorithm to locate a vehicle on a non-structured road. *International Journal of Robotics Research*, 19(5) :411–423, Mai 2000.

[5] R. Aufrère, F. Marmoiton, R. Chapuis, F. Collange, and J. P. Dérutin. Détection de route et suivi de véhicules par vision pour l'acc. *Traitement du Signal*, 17 :233–247, 2000.

[6] L. Bai, W. Tompkinson, and Y. Wang. Computer vision techniques for traffic flow computation. *Pattern Analysis and Applications*, 7(4) :365–372, 2004.

[7] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Readings in computer vision : issues, problems, principles, and paradigms*, pages 714–725, 1987.

[8] H. Bay, T. Tuytelaars, and L.J. Van Gool. Surf : Speeded up robust features. In *European Conference on Computer Vision*, volume 3951, pages 404–417, 2006.

[9] M. Beauvais and S Lakshmanan. Clark : a heterogeneous sensor fusion method for finding lanes and obstacles. *Image and Vision Computing*, 18(5) :397–413, 2000.

[10] J. Begard, N. Allezard, and P. Sayd. Real-time humans detection in urban scenes. In *British Machine Vision Conference*, University of Warwick, UK, 2007.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4) :509–522, 2002.

[12] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet. A cooperative approach to vision-based vehicle detection. In *Proceedings on Intelligent Transportation Systems*, pages 207–212, Août 2001.

[13] M. Bertozzi and A. Broggi. Vision-based vehicle guidance. *Computer*, 30(7) :49–55, 1997.

[14] M. Bertozzi, A. Broggi, and S. Castelluccio. A real-time oriented system for vehicle detection. *Journal of Systems Architecture*, 43(1-5) :317–325, 1997.

[15] M. Betke, E. Haritaoglu, and L. S. Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine Vision and Applications*, 12(2) :69–83, Août 2000.

[16] M. Betke, E. Haritaoglu, and L.S. Davis. Multiple vehicle detection and tracking in hard real time. In *Intelligent Vehicles Symposium*, pages 351–356, 1996.

[17] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino Lo Bianco, and A. Piazzi. Visual perception of obstacles and vehicles for platooning. *IEEE Transactions on Intelligent Transportation Systems*, 1(3) :164–176, 2000.

[18] S. Charles Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1-3) :65–86, 2008.

[19] T. Bucher, C. Curio, J. Edelbrunner, C. Igel, D. Kastrup, I. Leefken, G. Lorenz, A. Steinhage, and W von Seelen. Image processing and behavior planning for intelligent vehicles. *IEEE Transactions on Industrial Electronics*, 50(1) :62–75, Febrary 2003.

[20] S. D. Buluswar and B. A. Draper. Color machine vision for autonomous vehicles. *International Journal for Engineering Applications of Artificial Intelligence*, 1(2) :245–256, 1998.

[21] M.J.J. Burden and M.G.H. Bell. Vehicle classification using stereo vision. In *Image Processing and Its Applications, Sixth International Conference on*, volume 2, pages 881–885, Dublin, Juilliet 1997.

[22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligent*, 8(6) :679–698, 1986.

[23] J. Cano and J. C. Pérez-Cortés. Vehicle license plate segmentation in natural images. In *Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science, pages 142–149, Juin 2003.

[24] S.H. Cha and S. N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6) :1355–1370, 2002.

[25] R. Chapuis, R. Aufrere, and F. Chausse. Accurate road following and reconstruction by computer vision. *IEEE Transactions on Intelligent Transportation Systems*, 3(4) :261–270, December 2002.

[26] T. Chateau, V. Gay Belille, F. Chausse, and J.T. Lapreste. Real-time tracking with classifiers. In *Workshop on Dynamical Vision*, pages 218–231, 2006.

[27] H. Cheng, N. Zheng, and C. Sun. Boosted gabor features applied to vehicle detection. In *IEEE International Conference on Pattern Recognition*, pages 662–666, 2006.

[28] Sung Yug Choi and Jang Myung Lee. Applications of moving windows technique to autonomous vehicle navigation. *Image and Vision Computing*, 24(2) :120–130, 2006.

[29] X. Clady. *Contribution à la navigation autonome d'un véhicule automobile par vision.* PhD thesis, Université Blaise Pascal, France, 2003.

[30] X. Clady, P Negri, M Milgram, and R Poulenard. Multi-class vehicle type recognition system. In *IAPR International Workshop on Artificial Neural Networks in Pattern Recognition*, pages 228–239, 2008.

[31] J.M. Collado, C. Hilario, A. de la Escalera, and J.M. Armingol. Model based vehicle detection for intelligent vehicles. In *International Symposium on Intelligent Vehicles*, pages 572–577, 2004.

[32] D. Comaniciu and P. Meer. Mean shift : A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5) :603–619, 2002.

[33] J. Crisman and C. Thorpe. Color vision for road following. In *SPIE Conference on Mobile Robots*, volume 1007, page 175, 1990.

[34] G. Csurka, C. R. Dance, F. Perronnin, and J. Willamowski. Generic visual categorization using weak geometry. In *Toward Category-Level Object Recognition*, pages 207–224, 2006.

[35] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 2, pages 886–893, Juin 2005.

[36] E. De Micheli, R. Prevete, G. Piccioli, and M. Campani. Color cues for traffic scene analysis. In *Intelligent Vehicles Symposium*, pages 466–471, 1995.

[37] F. Dellaert. Canss : A candidate selection and search algorithm to initialize car tracking. Technical report, Robotics Institute, Carnegie Mellon University, 1997.

[38] F. Dellaert and C. Thorpe. Robust car tracking using kalman filtering and bayesian templates. In *Conference on Intelligent Transportation Systems*, volume 3207, Octobre 1997.

[39] C. Demirkir and B. Sankur. Face detection using look-up table based gentle adaboost. In *International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 339–345, 2005.

[40] C. Demonceaux, A. Potelle, and D. Kachi-Akkouche. A multi-cameras 3d volumetric method for outdoor scenes : a road traffic monitoring application. *IEEE Transactions on Vehicular Technology*, 53(6) :1649–1656, November 2004.

[41] L. Dlagnekov. *Video-based car surveillance : License plate make and model recognition.* PhD thesis, University of California, San Diego, 2005.

[42] J. Douret and R. Benosman. A multi-cameras 3d volumetric method for outdoor scenes : a road traffic monitoring application. In *IAPR International Conference on Pattern Recognition*, pages III : 334–337, 2004.

[43] M. Dubuisson, S. Lakshmanan, and A. Jain. Vehicle segmentation and classication using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3) :293–308, Mars 1996.

[44] M.-P. Dubuisson and A.K. Jain. A modified hausdorff distance for object matching. In *IAPR International Conference on Computer Vision and Image Processing*, volume 1, pages 566–568, Octobre 1994.

[45] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2001.

[46] F. Estrada, A. Jepson, and D. Fleet. Local features tutorial. Technical report, Université de Toronto, 2004. Disponible dans http ://www.cs.toronto.edu/ jepson/csc2503/tutSIFT04.pdf.

[47] J. M. Ferryman, A. D. Worrall, G. D. Sullivan, and K. D. Baker. A generic deformable model for vehicle recognition. In *British Machine Vision Conference*, pages 127–136, 1995.

[48] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *Proceedings IEEE Intelligent Vehicles Symposium 2000*, pages 273–278, 2000.

[49] U. Franke and I. Kutzbach. Fast stereo based object detection for stop&go traffic. In *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, pages 339–344, Septembre 1996.

[50] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[51] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[52] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : a statistical view of boosting. *The Annals of Statistics*, 28(2) :337–374, April 2000.

[53] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminative models for object category detection. In *IEEE International Conference on Computer Vision*, volume 2, pages 1363–1370, 2005.

[54] P.F. Fung, W.S. Lee, and I. King. Randomized generalized hough transform for 2-d grayscale object detection. In *IAPR International Conference on Pattern Recognition*, volume 2, pages 511–515, 1996.

[55] T. Gandhi and M. M. Trivedi. Video based surround vehicle detection, classification and logging from moving platforms : Issues and approaches. In *IEEE Intelligent Vehicles Symposium*, pages 1067–1071, Istambul, Juin 2007.

[56] Y. Gao and M. K.H. Leung. Face recognition using line edge map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6) :764–779, 2002.

[57] A. Gepperth, J. Edelbrunner, and T Bocher. Real-time detection and classification of cars in video sequences. In *IEEE Intelligent Vehicles Symposium*, pages 625–631, Juin 2005.

[58] D. Geronimo, A. Lopez, D. Ponsa, and A.D. Sappa. Haar wavelets and edge orientation histograms for on-board pedestrian detection. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 418–425, 2007.

[59] H. Ghorayeb, B. Steux, and C. Laurgeau. Boosted algorithms for visual object detection on graphics processing units. In *Asian Conference on Computer Vision*, pages 254–263, Janvier 2006.

[60] C. Goerick, D. Noll, and M. Werner. Artificial neural networks in real-time car detection and tracking applications. *Pattern Recognition Letters*, 17(4) :335–343, Avril 1996.

[61] V. Graefe and W. Efenberger. A novel approach for the detection of vehicles on freeways by real-time vision. In *IEEE Intelligent Vehicles Symposium*, Tokyo, Japon, Septembre 1996.

[62] D. Guo, T. Fraichard, M. Xie, and C. Laugier. Color modeling by spherical influence field in sensing drivingenvironment. In *IEEE Intelligent Vehicles Symposium*, pages 249–254, 2000.

[63] S. Gupte, O. Masoud, R.F.K. Martin, and N.P. Papanikolopoulos. Detection and classification of vehicles. *IEEE Intelligent Transportation Systems*, 3(1) :37–47, Mars 2002.

[64] D. Han, M. J. Leotta, D. B. Cooper, and J. L. Mundy. Vehicle class recognition from video-based on 3d curve probes. In *IEEE International Conference on Computer Communications and Networks*, pages 285–292, 2005.

[65] S. Han, E. Ahn, and N. Kwak. Detection of multiple vehicles in image sequences for driving assistance system. In *International Conference on Computational Science and Its Applications*, volume 3480, pages 1122–1128, 2005.

[66] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. Computer vision for driver assistance systems. In *SPIE Congress on Signal processing, sensor fusion, and target recognition*, volume 3364, pages 136–147, 1998.

[67] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen. An image processing system for driver assistance. In *IEEE International Conference on Intelligent Vehicles*, pages 481–486, 1998.

[68] W. Heisele, B. end Ritter. Obstacle detection based on color blob flow. In *IEEE Intelligent Vehicles Symposium*, pages 282–286, Septembre 1995.

[69] C. Huang, H. Ai, B. Wu, and S. Lao. Boosting nested cascade detector for multi-view face detection. In *IEEE International Conference on Pattern Recognition*, volume 2, pages 415–418, 2004.

[70] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(9) :850–864, Septembre 1993.

[71] K. Ichikawa, T. Mita, and O. Hori. Component-based robust face detection using adaboost and decision tree. In *International Conference on Automatic Face and Gesture Recognition*, pages 413–420, 10-12 April 2006.

[72] R. Isukapalli and A.M. Elgammal. Learning policies for efficiently identifying objects of many classes. In *IAPR International Conference on Pattern Recognition*, pages III : 356–361, 2006.

[73] B. Jahne, H. Haubecker, and P. Geibler. *HandBook of Computer Vision and Applications*, volume 2. Academic Press, 1999.

[74] M. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-96, MERL, 2003.

[75] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *IEEE International Conference on Computer Vision*, volume 1, pages 604–610, 2005.

[76] A. Torralba K. Murphy and W. T. Freeman. Using the forest to see the trees : a graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems*, 2003.

[77] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1) :52–60, 1967.

[78] T. Kalinke, C. Tzomakas, and W. v. Seelen. A texture-based object detection and an adaptive model-based classification. In *IEEE International Conference on Intelligent Vehicles 1998*, volume 1, pages 143–148, 1998.

[79] T. Kalinke and W. von Seelen. A neural network for symmetry-based object detection and tracking. In *DAGM-Symposium*, pages 37–44, 1996.

[80] T. Kato, Y. Ninomiya, and I. Masaki. An obstacle detection method by fusion of radar and motion stereo. *Intelligent Transportation Systems*, 3(3) :182–188, Septembre 2002.

[81] T. Kato, Y. Ninomiya, and I. Masaki. Preceding vehicle recognition based on learning from sample images. *IEEE Transactions on Intelligent Transportations Systems*, 3(4) :252–260, Decembre 2002.

[82] F. M. Kazemi, S. Samadi, H. Pourreza, and M. R. Akbarzadeh. Vehicle recognition based on fourier, wavelet and curvelet transforms - a comparative study. *International Journal of Computer Sciences and Network Security*, 7(2) :130–135, 2007.

[83] A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau. Vehicle detection combining gradient analysis and adaboost classification. In *IEEE International Conference on Intelligent Transportation Systems*, pages 66–71, Vienna, Austria, Septembre 2005.

[84] K. I. Kim, K. Jung, and J. H. Kim. Color texture-based object detection : An application to license plate localization. In *Pattern Recognition with Support Vector Machines*, volume 2388/2002, pages 293–309, Août 2002.

[85] A. Kimura and T. Watanabe. An extension of the generalized hough transform to realize affine-invariant two-dimensional (2d) shape detection. In *IAPR International Conference on Pattern Recognition*, page 10065, Washington, DC, États Unis, 2002.

[86] H. Kwasnicka and B. Wawrzyniak. License plate localization and recognition in camera pictures. In *Artificial Intelligence Methods*, Gliwice, Pologne, Novembre 2002.

[87] D. Larlus and F. Jurie. Latent mixture vocabularies for object categorization. In *British Machine Vision Conference*, 2006.

[88] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1) :29–44, 2001.

[89] M. Li and B.Z. Yuan. 2d-lda : A statistical linear discriminant analysis for image matrix. *Pattern Recognition Letters*, 26(5) :527–532, Avril 2005.

[90] Q. Li, N. Zheng, and H. Cheng. Springrobot : A prototype autonomous vehicle and its algorithms for lane detection. *Intelligent Transportation Systems*, 5(4) :300–308, Decembre 2004.

[91] S. Z. Li and Z. Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9) :1112–1123, 2004.

[92] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *European Conference on Computer Vision*, pages 67–81, London, 2002.

[93] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM Symposium for Pattern Recognition*, pages 297–304, 2003.

[94] Yen-Yu Lin and Tyng-Luh Liu. Robust face detection with multi-class boosting. In *IEEE Computer Vision and Pattern Recognition*, volume 1, pages 680–687, Washington, DC, États Unis, 2005.

[95] D. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[96] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, pages 91–110, 2004.

[97] R. Mandelbaum, L. McDowell, L. Bogoni, B. Reich, and M. Hansen. Real-time stereo processing, obstacle detection, and terrain estimation from vehicle-mounted stereo cameras. In *IEEE Workshop on Applications of Computer Vision*, pages 288–289, Octobre 1998.

[98] F. Marmoiton. *Detection et suivi par vision monoculaire d'obstacles mobiles cooperatifs a partir d'un vehicule experimental automobile*. PhD thesis, Ecole Doctorale Sciences Pour l'Ingénieur de Clermont-Ferrand, Université Blaise Pascal, Clermont-Ferrand, France, Janvier 2000.

[99] N. D. Matthews, P. E. An, D. Charnley, and C. J. Harris. Vehicle detection and recognition in greyscale imagery. *Control Engineering Practice*, 4(4) :473–479, Avril 1996.

[100] Neil D. Matthews. *Visual Collision Avoidance*. PhD thesis, University of Southampton, Advanced Systems Research Group, Department of Electronics and Computer Science, Octobre 1994.

[101] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, pages 69–82, 2004.

[102] S. Mohottala, M. Kagesawa, and K. Ikeuchi. Vehicle class recognition using 3d cg models. In *Intelligent Transportation Systems*, 2003.

[103] H. Mori and N. Charkai. Shadow and rythm as sign patterns of obstacle detection. In *International Symposium on Industrial Electronics*, pages 271–277, 1993.

[104] D. T. Munroe and M. G. Madden. Multi-class and single-class classification approaches to vehicle model recognition from images. In *Conference on Artificial Intelligence and Cognitive Science*, Septembre 2005.

[105] P. Negri, X. Clady, M. Milgram, and R. Poulenard. An oriented-contour point based voting algorithm for vehicle type classification. In *IAPR International Conference on Pattern Recognition*, volume 1, pages 574–577, Août 2006.

[106] P. Negri, X. Clady, M. Milgram, and R. Poulenard. Système de reconnaissance multi-classes du type de véhicules. In *Congrès francophone des jeunes chercheurs en vision par ordinateur, ORASIS*, Juin 2007.

[107] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. In MIT Press, editor, *Conference on Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[108] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[109] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*, volume 4, pages 490–503. Springer, 2006.

[110] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4) :353–363, 1993.

[111] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *IEEE Computer Vision and Pattern Recognition*, page 193, 1997.

[112] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *IEEE International Conference on Computer Vision*, pages 555–562, Janvier 1998.

[113] C.P. Papageorgious and T. Poggio. A trainable object detection system : Car detection in static images. Technical report, MIT AI Memo 1673 (CBCL Memo 180), 1999.

[114] P. Parodi and G. Piccioli. A feature based recognition scheme for traffic scenes. In *IEEE Intelligent Vehicles Symposium,*, pages 229–234, Tokyo, Japon, 1995.

[115] V. S. Petrovic and T. F. Cootes. Vehicle type recognition with match refinement. In *IAPR International Conference on Pattern Recognition*, volume 3, pages 95–98, 2004.

[116] V.S. Petrovic and T.F. Cootes. Analysis of features for rigid structure vehicle type recognition. In *British Machine Vision Conference*, volume 2, pages 587–596, Septembre 2004.

[117] D. Ponsa and A. Lopez. Cascade of classifiers for vehicle detection. In *Advanced Concepts for Intelligent Vision Systems*, volume 4678, pages 980–989. Springer, 2007.

[118] F. Porikli. Integral histogram : A fast way to extract histograms in cartesian spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.

[119] A. Prati, I. Mikic, R. Cucchiara, and M. M. Trivedi. Analysis and detection of shadows in video streams : A comparative evaluation. In *IEEE Computer Vision and Pattern Recognition Workshop on Empirical Evaluation Methods in Computer Vision*, volume 2, Decembre 2001.

[120] L. Prevost, L. Oudot, A. Moises, C. Michel-Sendis, and M. Milgram. Hybrid generative/discriminative classifier for unconstrained character recognition. *Pattern Recognition Letters*, 26(12) :1840–1848, 2005.

[121] U. Regensburger and V. Graefe. Visual recognition of obstacles on roads. In *International Conference on Intelligent Robots and Systems*, volume 2, pages 980–987, Septembre 1994.

[122] K. Reumann and A.P. Witkam. Optimizing curve segmentation in computer graphics. In *International Computing Symposium*, pages 467–472, 1974.

[123] J.C. Rojas and J.D. Crisman. Vehicle detection in color images. In *IEEE Conference on Intelligent Transportation System*, pages 403–408, Novembre 1997.

[124] A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *Association for Computing Machinery*, 13(4) :471–494, 1966.

[125] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Pattern Analysis and Machine Intelligence*, 20(1) :23–38, Janvier 1998.

[126] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3) :297–336, 1999.

[127] M. Schmid. An approach to model-based 3-d recognition of vehicles in real time by machine vision. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, volume 3, pages 2064–2071, Septembre 1994.

[128] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 746–751, 2000.

[129] M.A. Sotelo, M.A. Garcia, and R. Flores. Vision based intelligent system for autonomous and assisted downtown driving. In *International Workshop on Computer Aided Systems Theory*, volume 2809, pages 326–336. Springer, 2003.

[130] N. Srinivasa. Vision-based vehicle detection and tracking method for forward collision warning in automobiles. In *IEEE Intelligent Vehicle Symposium*, volume 2, pages 626–631, Juin 2002.

[131] B. Steux. *Maps, un environnement logiciel dédié à la conception d'applications embarqués temps-réel. Utilisation pour la détection automatique de véhicules par fusion radar/vision.* PhD thesis, Ecole des Mines de Paris, Paris, France, Decembre 2001.

[132] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using evolutionary gabor filter optimization. *IEEE Transactions on Intelligent Transportation Systems*, 6(2) :125–137, Juin 2005.

[133] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection : A review. *IEEE Transanctions on Pattern Analalysis and Machine Intelligence*, 28(5) :694–711, 2006.

[134] T. N. Tan and K. D. Baker. Efficient image gradient based vehicle localization. *IEEE Transactions on Image Processing*, 9 :1343–1356, 2000.

[135] M.A.P. Taylor and W. Young. *Traffic Analysis : New Technology and New Solutions.* Hodder Arnold, 1988.

[136] F. Thomanek, E.D. Dickmanns, and D Dickmanns. Multiple object recognition and scene interpretation for autonomous road vehicle guidance. In *IEEE Intelligent Vehicles Symposium*, pages 231–236, Octobre 1994.

[137] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features : Efficient boosting procedures for multiclass object detection. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 762–769, 2004.

[138] D. A. Torres. More local structure information for make-model recognition. University of California, San Diego, 2005.

[139] N. Trujillo, R. Chapuis, F. Chausse, and C. Blanc. On road simultaneous vehicle recognition and localization by model based focused vision. In *IAPR Conference on Machine Vision Applications*, pages 388–391, Mai 2005.

[140] L. G. Valiant. A theory of the learnable. *ACM Symposium on Theory of Computing*, 27(11) :1134–1142, 1984.

[141] M.B. van Leeuwen and F.C.A. Groen. Vehicle detection with a mobile camera. Technical report, Computer Science Institute, University of amsterdam, The Netherlands, Octobre 2001.

[142] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Decembre 2001.

[143] P. A. Viola and M. J. Jones. Robust real-time face detection. In *International Conference on Computer Vision*, volume 2, page 747, Juillet 2001.

[144] C.-C. Wang and J.-J. Lien. Automatic vehicle detection using statistical approach. In *Asian Conference on Computer Vision*, volume 3852, pages 171–182. Springer, 2006.

[145] H. Wang, P. Li, and T. Zhang. Boosted gaussian classifier with integral histogram for face detection. *International Journal of Pattern recognition and Artificial Intelligence*, 21(7) :1127–1139, 2007.

[146] X. Wen, H. Yuan, C. Yang, C. Song, B. Duan, and H. Zhao. Improved haar wavelet feature extraction approaches for vehicle detection. In *IEEE Intelligent Transportation Systems Conference*, pages 1050–1053, Septembre 2007.

[147] D. Withopf and B. Jahne. Improved training algorithm for tree-like classifiers and its application to vehicle detection. In *IEEE Intelligent Transportation Systems Conference*, pages 642–647, Septembre 2007.

[148] J. Wu and X. Zhang. A pca classifier and its application in vehicle detection. In *IEEE International Joint Conference on Neural Networks*, volume 1, pages 600–604, 2001.

[149] W. Wu, Z. QiSen, and W. Mingjun. A method of vehicle classification using models and neural networks. In *Vehicular Technology Conference*, volume 4, pages 3022–3026, 2001.

[150] T. Xiong and C. Debrunner. Stochastic car tracking with line- and color-based features. *Advances and Trends in Research and Development of Intelligent Transportation Systems : An Introduction to the Special Issue*, 5(4) :324–328, Decembre 2004.

[151] H. Yang, J. Lou, H. Sun, W. Hu, and T. Tan. Efficient and robust vehicle localization. In *International Conference on Image Processing*, volume 2, pages 355–358, Octobre 2001.

[152] A.H.S. Yung, N.H.C.and Lai. Detection of vehicle occlusion using a generalized deformable model. In *IEEE International Symposium on Circuits and Systems*, volume 4, pages 154–157, Mai 1998.

[153] I. Zafar, B.S. Acar, and E.A. Edirisinghe. Vehicle make & model identification using scale invariant transforms. In *Visualization, Imaging, and Image Processing*, Août 2007.

[154] I. Zafar, E. A. Edirisinghe, S. Acar, and H. E. Bez. Two-dimensional statistical linear discriminant analysis for real-time robust vehicle-type recognition. In *SPIE Real-Time Image Processing*, volume 6496, Février 2007.

[155] M. Zayed and J. Boonaert. Detection et suivi d'une plaque d'immatriculation pour l'attelage virtuel. In *18 Journées des Jeunes Chercheurs en Robotique*, Douai, France, Septembre 2004.

[156] G. Zhao and S. Yuta. Obstacle detection by vision system for an autonomous vehicle. In *Intelligent Vehicles Symposium*, pages 31–36, 1993.

[157] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE International Converence on Computer Vision and Pattern Recognition*, pages 1491–1498, Washington, DC, États Unis, 2006.

[158] T. Zielke and J. GroSS. Advances in computer vision-based obstacle detection for cars. In *In Proceedings 27th International Symposium on Automotive Technology and Automation*, pages 431–436, 1994.