Apprentissage de variétés pour la Détection et Reconnaissance de véhicules faiblement résolus en imagerie aérienne

Sebastien Razakarivony

Université de Caen Basse-Normandie

Ecole doctorale

STRUCTURES, INFORMATIONS, MATIERES ET MATERIAUX

Thèse de doctorat

présentée et soutenue le : *03/12/2014*

par

Sébastien RAZAKARIVONY

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

Spécialité : Traitement du signal et des images

Apprentissage de variétés pour la Détection et Reconnaissance de véhicules faiblement résolus en imagerie aérienne

Directeur de thèse : *Frédéric JURIE*

Composition du jury

Laurent HEUTTE	Professeur des Universités, Université de Rouen	Rapporteur
El Mustapha MOUADDIB	Professeur des Universités, Université de Picardie Jules Verne	Rapporteur
Frédéric JURIE	Professeur des Universités, Université de Caen Basse Normandie	Examinateur
Rachid DERICHE	Directeur de Recherche, INRIA	Examinateur
Daniel DUCLOS	Expert Senior, CRT SAFRAN	Examinateur
Farid OUDYI	Expert Traitement d'Images, SAGEM	Examinateur

Table des matières

Table des figures

Liste des tableaux

Notations et Abréviations

Symbole	Signification
M	matrice
d	scalaire
$\mathbf{x}$	vecteur
$*$	convolution
M^T	transposée de M
f	fonction
f^{-1}	inverse de f
$\mathcal{M}$	variété

Abréviation	Signification
ACP	Analyse en Composantes Principales
AP	Average Precision
AE	AutoEncodeur
AED	AutoEncodeur Discriminant
AEDMC	AutoEncodeur Discriminant Multi-Classes
BOW	Bag Of Words / Sac de Mots
CMU	Carnegie Mellon University
CNN	Convolutional Neural Network / Réseau convolutionel
CODA	COnvolutional Discriminant Autoencodeur / Autoencodeur Convolutionel Discriminant
Chap.	Chapitre
DET	Detection Error Tradeoff
DPM	Deformable Parts Model
DRI	Détection Reconnaissance Identification
EM	Expectation Maximisation
ETHZ	Swiss Federal Institute of Zürich
FAST	Features from Accelerated Segment Test
Fig.	Figure
FN	Faux Négatif
FP	Faux Positif
FPPF	Faux Positif Par Fenêtres

FPPI	Faux Positif Par Image
GM	Gaussian Mixture
GRA	GRAdient normalisé
HOG	Histogram of Oriented Gradients / Histogramme de gradients orientés
LBP	Local Binary Pattern
LTP	Local Ternary Pattern
mAP	Average Precision moyenne
MLP	Multi-Layers Perceptron / Perceptron Multi-couches
MR	Miss Rate / Taux d'erreur
NDG	Niveaux De Gris normalisés
NMS	Non-Maximum Suppression / Suppression des Non-Maxima
OIRDS	Overhead Imagery Research DataSet
OIRDS-EGI	OIRDS-Ensemble de Grandes Images
OIRDS-EPI	OIRDS-Ensemble de Petites Images
OP	Operating Point / Point Opérationnel
P	Nombre de Positifs
Prec	Précision
RBF	Radial Basis Function
Rec	Recall / Rappel
ROC	Receiver Operator Characteristic
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Feature
SVM	Support Vector Machine / Séparateur à Vaste Marge
Tab.	Table
TM	Template Matching
TVP	Taux de Vrais Positifs
TFP	Taux de Faux Positifs
VeDAI	Vehicule Detection in Aerial Imagery
VeDAI-GIC	VeDAI, Grandes Images Couleurs
VeDAI-GII	VeDAI, Grandes Images Infrarouges
VeDAI-PIC	VeDAI, Petites Images Couleurs
VeDAI-PII	VeDAI, Petites Images Infrarouges
VN	Vrai Négatif
VP	Vrai Positif
WGSS	Within Group Scatter Sum, distance intra-cluster

Abstract

Abstract

This manuscript addresses the problematics of Detection and Recognition of vehicles of poor resolution in aerial imagery. First, we present these two problems and we give a survey of the different state-of-the-art technics that exist to solve them. We then introduce databases which are used by the Computer Vision community and the databases created and used during our work, that are more suited to our industrial context. Thirdly, we test some of the state-of-the-art algorithms and we present the related results on these databases. Next, we introduce the use of manifolds as generative models in order to decorrelate the modelisation of the vehicles from the modelisation of the background regions. Then, the discriminative autoencoder, a novel algorithm based on metric learning to detect and recognise efficiently vehicles, as well as its extension, the convolutional discriminative autoencoder, are presented with the associated experiments and results. Finaly, we present some experiments on learning the caracteristics of the background of an image. The document is closed by the conclusions and a discussion about the future works.

Résumé

Cette thèse traite les problématiques de Détection et Reconnaissance de véhicules faiblement résolus dans des images aériennes. Nous présentons tout d'abord ces deux problématiques et leurs difficultés, tant générales que spécifiques au contexte industriel. Dans un second temps, nous effectuons un état de l'art des techniques existantes qui permettent de résoudre ces problématiques sur les contextes classiques, puis nous présentons les bases de données utilisées par la communauté de vision par ordinateur et les bases de données utilisées et crées dans cette thèse. Par la suite, nous présentons les expériences et résultats obtenus avec des méthodes de l'état de l'art sur les bases de données présentées, après quoi nous introduisons l'utilisation des variétés de manière générative, afin de découpler la modélisation des véhicules à détecter et la modélisation du fond à traiter. Ensuite, l'autoencodeur discriminant, un nouvel algorithme basé sur de l'apprentissage de métrique à des fins de classification est présenté, ainsi que son extension, l'autoencodeur discriminant convolutionel. Enfin, nous présentons quelques expériences basées sur l'apprentissage des caractéristiques du fond d'une image. Nous exposons finalement les conclusions et perspectives des travaux effectués.

Remerciements

Je tiens en premier lieu à remercier tous les membres de mon jury, à savoir les professeurs des universités Laurent Heutte et El Moustapha Mouaddib, rapporteurs de mon manuscrit, Rachid Deriche, directeur de recherche à l'INRIA, président de mon jury, ainsi que mes encadrants, Frédéric Jurie, professeur des universités, Farid Oudyi et Daniel Duclos, respectivement experts pour Sagem et Safran. Le jury a contribué à une discussion intéressante sur mes travaux, leurs résultats ainsi que leur perspectives.

Mes remerciements s'adressent plus particulièrement à mon directeur de thèse, Frédéric Jurie, qui a toujours fait son maximum pour être disponible, a été à l'écoute de mes remarques et questions et a toujours su porter un regard positif sur mes travaux. Je remercie également Farid Oudyi et Daniel Duclos pour leurs encadrements ainsi que leurs remarques pertinentes qui m'ont permis de conserver en permanence du recul sur ma thèse.

Au cours de ces trois années de thèse, j'ai eu l'occasion de travailler au sein de Sagem DS, je tiens en conséquence à remercier tous mes collègues, que ce soit pour les différentes discussions techniques lors des différentes réunions d'équipes, que pour l'ambiance de travail qui a toujours été très agréable.

Si je ne suis pas souvent allé physiquement sur le site de Caen, je me dois de remercier chaleureusement Pierre Blondeau et Davy Gigan, qui m'ont été d'une aide précieuse pour l'installation et le maintien de la connexion à l'université. Je remercie également tous mes collègues du laboratoire de Caen pour avoir été patients vis-à-vis de mon occupation des clusters de calculs.

Je remercie évidemment ma famille et mes amis, qui ont toujours été là pour moi et m'ont épaulé tout au long de cette thèse.

Enfin, je remercie du plus profond de mon coeur ma compagne et chère amie Héléna, avec qui je me suis lancé dans le périple du doctorat, pour tout son soutien et tout son amour qui m'ont accompagné au cours de ce voyage ! Merci pour tout ce que tu m'as apporté et apporte encore.

Encore un grand merci à tous.

Chapitre 1

Introduction

Chaque jour, toujours plus de données sont fournies par les capteurs qui nous entourent. Ces capteurs sont eux mêmes plus précis et donc plus volubiles. Certains d'entre eux enregistrent les données de l'environnement et les traduisent dans un vocabulaire compréhensible par l'être humain (caméra IR, ultrasons, télescopes...). D'autres, alimentent les entrées d'objets intelligents, par exemple les capteurs de collision pour un robot ou encore un simple détecteur de mouvement pour une lampe d'extérieur. Le domaine de la vision par ordinateur consiste à traiter les données fournies par les nombreux capteurs d'images à notre disposition afin de permettre à un ordinateur d'accomplir des tâches spécifiques sans le concours de l'homme pour pré-traiter les données. Malgré les avancées importantes de ce champ de recherche, le couple capteur-ordinateur est aujourd'hui encore loin de d'égaler les performances de celui formé par l'œil et le cerveau.

Dans le domaine de la vision par ordinateur, la Détection d'objets consiste à trouver automatiquement des objets dans des images, c'est-à-dire donner leurs positions dans ces images. Les applications les plus courantes sont des systèmes de sécurité ou de sûreté (détection de piétons, identification de comportement), ainsi que des systèmes de contrôle (détection de défauts sur des surfaces). La Reconnaissance est un problème lié qui consiste à savoir à quelle catégorie d'objet appartient le contenu d'une image ou d'une portion d'image. La problématique de cette thèse est plus spécifique, car elle se pose dans le cas plus restreint, mais également plus difficile, de la détection de véhicules faiblement résolus présents dans des images aériennes.

Cette thèse s'est inscrite dans le cadre d'un contrat CIFRE entre le laboratoire GREYC [1] Image et la société Sagem – membre du groupe SAFRAN, figurant parmi les leaders mondiaux en optronique et avionique – et plus précisément avec sa division Avionique. Cette division travaille dans les domaines de la navigation inertielle et de l'avionique civile et militaire. Ses systèmes de navigation, de guidage et de pointage équipent les avions, navires de surface et sous-marins, missiles, véhicules blindés et systèmes d'armement terrestre de nombreux pays. Ses systèmes et équipements avioniques sont présents dans les plus grands programmes d'avions et d'hélicoptères. Enfin, ses systèmes d'information et de préparation de mission sont utilisés par de nombreuses forces armées.

Ce premier chapitre est une introduction à ce manuscrit. Il contient la problématique, les difficultés tant générales aux problématiques abordées que spécifiques à cette thèse, suivie du plan. Cette thèse a fait l'objet de différentes publications, dont la liste exhaustive est située en Annexe E.

1. Laboratoire de l'Université de Caen Basse-Normandie et de l'ENSICAEN, UMR 6602 du CNRS

1.1 Problématique

Cette thèse s'intéresse aux problèmes de Détection et Reconnaissance de véhicules faiblement résolus, en imagerie aérienne. Différentes définitions sont possibles quant à ces problématiques et diffèrent selon les travaux dans la communauté de vision par ordinateur. En conséquence, il nous a paru important de définir clairement ces problématiques. Dans cette section, nous définissons ce que couvrent les termes *Détection* et *Reconnaissance* dans ce manuscrit et nous abordons les spécificités contextuelles de cette thèse. Dans le domaine militaire, ces problématiques sont regroupées sous le terme DRI, pour Détection, Reconnaissance et Identification.

Le premier problème, celui de la *Détection d'objets* consiste à prédire la position des objets appartenant à une catégorie donnée dans une image. Une *catégorie d'objets* ou *classe d'objets*, est constituée d'un ensemble d'objets similaires, le plus souvent du point de vue sémantique (par exemple, différents types de voitures forment la catégorie voiture). Dans le cadre de ces travaux, les véhicules sont plus particulièrement étudiés afin de mieux correspondre au cadre d'emploi des imageurs de Sagem. Les véhicules étant des objets rigides et globalement convexes, le cadre est différent des cas étudiés habituellement par la communauté, comme nous le verrons dans le Chap. 3. Cependant, en dehors de ces restrictions, les véhicules à détecter peuvent être sur n'importe quel fond, dans des conditions de prises de vue quelconques. Néanmoins, pour des raisons pratiques, nous nous sommes restreints à des images en vue verticale. La position d'un objet dans une image peut être définie par divers méthodes : les coordonnées de son centre, l'ensemble des pixels de l'objet, sa boîte englobante, son polygone englobant. La figure 1.1 illustre tant le type d'image traitée que le résultat souhaité pour une détection de véhicules en imagerie aérienne.

Par la problématique de *Reconnaissance d'objets*, nous signifions la détermination de la sous-catégorie d'un objet. Par exemple, dans la catégorie des véhicules, nous souhaitons déterminer si l'objet est un van, une voiture ou un pick-up. Nous nous plaçons dans le cadre applicatif pour lequel la problématique de localisation d'un véhicule dans l'image est résolue (par exemple par un détecteur de mouvement ou une désignation manuelle). Nous souhaitons alors connaître le type de véhicule dont il s'agit (véhicule blindé, voiture civile, autre...). Notons que la Reconnaissance peut être vue comme un sous-problème de la Détection dans une image fixe dans des conditions plus restreintes. La figure 1.2 illustre la problématique de Reconnaissance telle qu'abordée ici.

Les travaux de cette thèse s'intéressent à ces problématiques plus particulièrement appliquées aux véhicules *faiblement résolus*. Cette terminologie vient renforcer le contraste entre nos travaux et la majorité des travaux existants dans la communauté de la vision par ordinateur et s'explique par le domaine d'activités de Sagem. En effet, les objets à détecter ou reconnaitre ont des tailles aux alentours de 40x40 pixels, certains pouvant aller jusqu'à 10x20 pixels, tandis que dans l'état de l'art les objets ont des tailles plus conséquentes, comme nous l'expliquerons au Chap. 3. Même si de nouveaux capteurs sont développés, plus performants et plus résolus, travailler sur des faibles résolutions permet de Détecter et Reconnaitre des objets qui sont situés plus loin du capteur (étant plus petits), cette étude reste donc pertinente quelle que soit l'évolution de la technologie. Enfin, les travaux doivent s'appliquer à différents types d'images, que ce soit les images couleur ou infrarouge.

Quant aux des applications, nous traitons l'*imagerie aérienne*, c'est-à-dire que les images traitées sont issues de capteurs situés dans les airs, pointés vers le sol. Cette

FIGURE 1.1 – Illustration de la problématique de Détection. Deux vans, une voiture et un camping car ont été détectés dans cette image, leurs positions respectives sont marquées d'une croix.

imagerie est différente des imageries terrestres, qui sont limitées dans leurs degrés de liberté. Cependant, les algorithmes développés au cours de cette thèse peuvent tout à fait s'appliquer à d'autres types d'imagerie.

1.2 Difficultés

Les problématiques de Reconnaissance et de Détection abordées présentent plusieurs difficultés, certaines intrinsèques, d'autres spécifiques au contexte des imageurs Sagem.

Une des premières difficultés classiques est la disparité des apparences des objets au sein d'une même classe. Cette difficulté est appelée la variabilité intra-classe. Malgré des apparences parfois très différentes dans une classe donnée, il est attendu que les algorithmes soient capables de prédire correctement cette classe. L'exemple classique est la classe 'chaise', qui a une très grande variabilité intra-classe. Pour les véhicules, il est possible de souhaiter classer dans la même catégorie une petite citadine et un 4x4. Cette

FIGURE 1.2 – Illustration de la problématique de Reconnaissance. Nous souhaitons déterminer la classe du véhicule à droite parmi les 6 classes connues présentées à gauche.

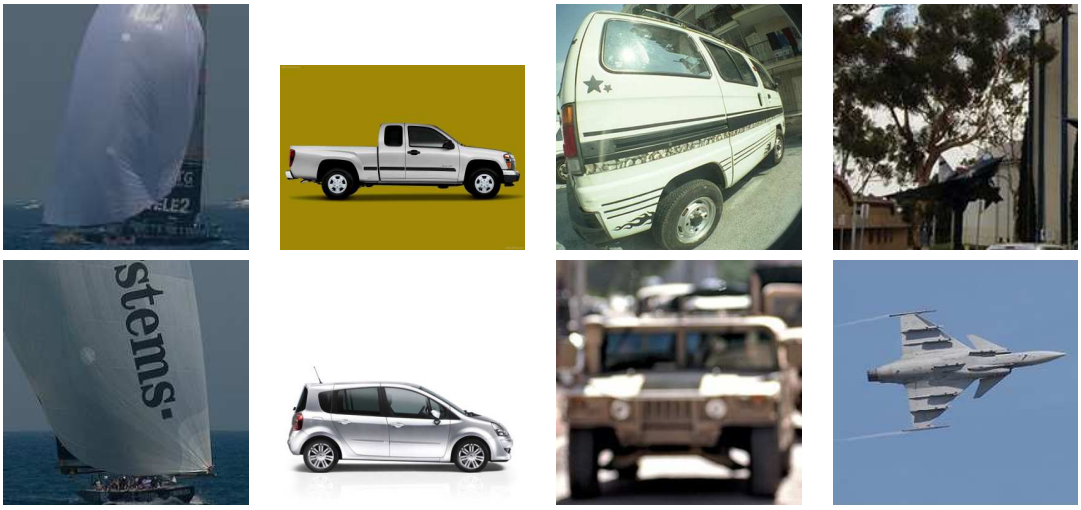


FIGURE 1.3 – Illustrations des différentes difficultés du problème. De gauche à droite : la variabilité intra-classe (deux bateaux à voiles différents), la proximité inter-classes (comparaison d'un pickup et d'une voiture), les conditions d'acquisition (distorsion et flou), les différences de fonds (deux avions, sur fond complexe ou sur fond simple).

variabilité dépend également de la définition des classes, qui caractérise (souvent) une fonction et non pas une apparence. Cette difficulté rend la modélisation d'une classe d'objet plus difficile de par sa disparité interne. À l'inverse les apparences peuvent être très similaires dans deux classes à différencier, comme par exemple, une voiture et un pickup. Il s'agit de la proximité inter-classes. Cette proximité inter-classes pose des problèmes pour séparer deux classes, la frontière entre elles pouvant être mal définie en termes d'apparence. Une autre difficulté classique est la forte variabilité des fonds. Nous définissons comme *fond* toute région de l'image qui ne contient pas de véhicule. Contrairement à la Détection dans des environnements contrôlés, les fonds des images traitées peuvent être divers et il n'est pas possible de les modéliser fidèlement sans restriction. La variabilité des points de vue représente une autre difficulté : un même objet pris sous

deux angles différents peut avoir une apparence complètement différente. De même, les conditions d'acquisition, le type de capteur ou l'environnement extérieur peuvent varier et ainsi changer l'apparence de l'objet dans l'image. Enfin, les objets étudiés peuvent subir des occultations. Quelques unes de ces difficultés sont illustrées Fig. 1.3.

Certaines difficultés sont propres au cadre applicatif de la thèse. Ainsi, nous souhaitons traiter des images infrarouge qui ont des propriétés particulières. En effet, un véhicule change d'apparence dans une image infrarouge en fonction de son historique récent (si son moteur est allumé/éteint, si il a roulé) et de son environnement (les conditions météorologiques sont différentes, ses conditions d'éclairage diffèrent...). Autre difficulté spécifique, la proximité inter-classe est renforcée car nous travaillons sur des véhicules. Par exemple, une voiture civile devra être distinguée d'un véhicule blindé. Les objets sont faiblement résolus, l'information du signal est donc limitée et sensible au bruit. En imagerie aérienne, des degrés de liberté supplémentaires de point de vue sont ajoutés (absence de la contrainte du sol en bas de l'image). De plus, en surveillance, les véhicules à détecter ne sont pas coopératifs avec le système de détection, en conséquence ils n'occupent pas une place privilégiée dans l'image et peuvent disposer de systèmes de camouflage. Enfin, peu de données réelles sont disponibles, en raison des méthodes d'acquisition difficiles à mettre en place (campagne de prises d'images en vol avec un imageur).

1.3 Organisation du document

Cette thèse vise à développer de nouveaux d'algorithmes de Détection et Reconnaissance, appliqués au contexte particulier des véhicules faiblement résolus en imagerie aérienne. La problématique de Détection consiste à prédire à quel endroit se trouvent les véhicules dans une image, tandis que celle de Reconnaissance vise à prédire quel type de véhicule est dans une région donnée. Ces deux problèmes sont intimement liés ; ainsi pour localiser un objet il faut (généralement) le reconnaitre et inversement. Les deux problématiques sont traitées, cependant le manuscrit est plus orienté vers la problématique de Détection. Les défis à relever sont la variabilité forte des fonds, la variabilité entre différentes images d'un même objet (point de vue, illumination), la variabilité intra-classe (deux voitures peuvent être très différentes), la proximité inter-classes (proximité d'un petit camion et d'une grosse voiture), les occlusions partielles.

Dans un premier temps, le Chap. 2 présente différentes méthodes et techniques de l'état de l'art qui tentent de résoudre ces tâches, en détaillant les éléments clefs de celles-ci. Ensuite, les bases de données pré-existantes à la thèse, puis celles créées et utilisées pour la thèse sont présentées dans le Chap. 3. Les expériences et résultats obtenus avec des algorithmes classiques de l'état de l'art sur les bases de données présentées sont détaillés dans le Chap. 4. Le Chap. 5 aborde l'utilisation des variétés génératives, modélisant différemment le fond et les objets. Le Chap. 6 présente une nouvelle méthode de classification, l'autoencodeur discriminant, ainsi que ses applications. Le Chap. 7 présente une extension des autoencodeurs discriminants pour un apprentissage plus global. Le Chap. 8 présente des expériences sur l'apprentissage automatique des caractéristiques propres au fond d'une image. Enfin, les conclusions sont présentées et les perspectives futures sont abordées dans le dernier chapitre.

Chapitre 2

État de l'art

Ainsi que présenté dans l'introduction, cette thèse vise à proposer des méthodes de Détection et de Reconnaissance de véhicules faiblement résolus dans des images. Nous proposons dans ce premier chapitre un état de l'art des différentes approches récentes qui nous ont semblé les plus pertinentes pour aborder ces problématiques. Ce premier chapitre se veut général afin de positionner les travaux par rapport aux techniques employées dans le domaine. Certains points plus spécifiques, tels que les bases de données, les techniques par apprentissage de variétés ou les réseaux convolutionels sont abordés plus en détails dans les chapitres les concernant.

Les algorithmes de Détection présents dans la littérature récente sont fondés sur trois principes clefs : l'apprentissage statistique, la représentation de l'information et la définition de mécanismes de localisation. Nous rappelons que la Reconnaissance peut être vue comme un sous-problème de la Détection. Cette problématique est usuellement abordée en utilisant des techniques similaires, mais dans un cadre différent et plus restreint. Ce chapitre a en conséquence une optique plus orientée vers la problématique de la Détection, tout en notant que la Reconnaissance est implicitement traitée.

Dans ce chapitre, nous aborderons successivement les trois principes évoqués ci-dessus, à savoir l'apprentissage statistique, la représentation de l'information et la localisation. Enfin, nous évoquerons la question de la mesure de performance.

2.1 Apprentissage statistique

Les méthodes de détection d'objets proposées dans la littérature récente reposent sur des mécanismes d'*apprentissage statistique*. L'apprentissage statistique consiste à construire un modèle à partir de données, dans le but d'accomplir une tâche, qui est généralement une tâche de *régression* (c'est-à-dire prédire une grandeur à partir d'observations) ou de *classification* (c'est-à-dire prédire une classe ou label à partir d'observations). Dans notre cas, il s'agira d'apprendre à faire la différence entre les objets à reconnaitre et leur environnement pour la détection (classification binaire) et pour la reconnaissance, d'apprendre à séparer les différentes sous-catégories d'objets (classification multi-classes).

Il est possible de comparer directement les apparences disponibles et les apparences à détecter. Les algorithmes de comparaison directe appartiennent à la large et ancienne famille des algorithmes de K plus proches voisins [49]. De nombreuses difficultés quand à l'échantillonnage, l'exhaustivité, le coût computationel (tant en mémoire qu'en rapidité) et l'absence totale de modélisation de la structure entre les données font que les techniques

basées sur l'utilisation d'ensembles d'apprentissage entiers ne sont pas des techniques couramment utilisées en détection.

En dehors de ces méthodes, l'apprentissage statistique peut être séparé en deux grandes familles d'approches. Les approches dites *génératives*, qui s'attachent à modéliser des objets le plus finement possible, sont à l'opposé des approches dites *discriminantes*, qui s'attachent à modéliser les frontières entre les objets le mieux possible.

Un algorithme d'apprentissage statistique en Détection s'utilise classiquement en deux phases. Lors de la phase dite d'apprentissage, les données disponibles sont utilisées pour effectuer la modélisation souhaitée. Lors de la phase dite de test, la modélisation apprise précédemment est utilisée sur de nouvelles données. Certains algorithmes, dits algorithmes à apprentissage par renforcement, utilisent les données recueillies pendant la phase de test pour apprendre.

Dans cette section, nous présentons différentes méthodes d'apprentissage statistique de modèles, qu'elles soient génératives ou discriminantes, puis nous expliquons les différentes architectures de modèles possibles pour expliquer des données. Enfin, nous donnons quelques détails sur les méthodes de combinaisons d'algorithmes.

2.1.1 Les modèles discriminants

Les modèles discriminants modélisent la frontière entre deux objets. Le modèle ne permettra pas de générer les apparences des classes, mais donne une comparaison entre différentes classes. Il se concentre sur les différences entre les classes plutôt que dans une modélisation précise de chacune d'entre elles.

Séparateurs à Vaste Marge

Les Séparateurs à Vastes Marges (*Support Vectors Machine* en anglais) ont été introduits par Vapnik [20]. Ils consistent à trouver un hyperplan séparateur qui donnera la meilleure généralisation possible, en maximisant la distance de cet hyperplan aux exemples donnés dans la base d'apprentissage (la marge, illustration en Fig. 2.1). De très nombreux travaux en Détection s'appuient sur ce classifieur [40, 109, 26], au vu de sa puissance théorique. L'hyperplan peut être calculé dans un espace dual, permettant ainsi l'obtention de frontière plus complexe qu'un hyperplan [167]. De nombreuses variantes ont été développées, permettant de travailler en multi-classes [177], sur une classe unique [14] ou encore d'incorporer des variables latentes [40]. Plusieurs librairies permettant d'apprendre un SVM sont disponibles librement sur internet, les plus connues et les plus utilisées étant SVMlight [81] et LIBSVM [12].

Réseaux de neurones

Les réseaux de neurones classiques, c'est-à-dire ceux utilisant au moins deux couches de neurones, tels que le perceptron [141], permettent de faire de la classification de données. La théorie des réseaux de neurones montre que ceux-ci peuvent approximer n'importe quelle fonction [25]. Leur principale difficulté d'utilisation est l'apprentissage en lui-même, bien que de nombreuses techniques existent afin d'éviter l'écueil des minima locaux et le sur-apprentissage [97].

Dans le domaine de la Détection d'objets, les réseaux profonds sont utilisés et plus particulièrement les réseaux convolutionels [96] qui ont obtenu de grandes avancées en

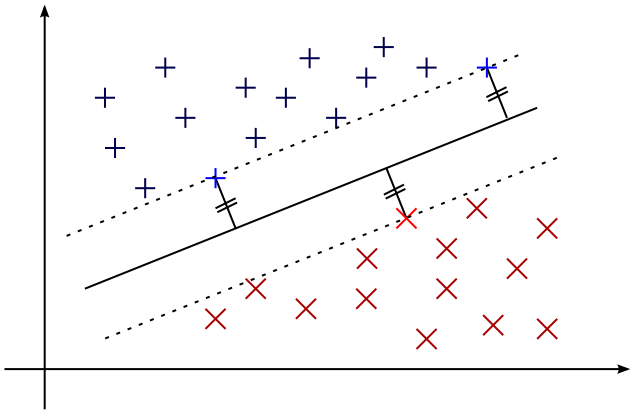


FIGURE 2.1 – Séparateur à vaste marge. L'hyperplan séparateur est choisi de telle sorte qu'il soit le plus éloigné possible des données, par la marge. La marge est ici représentée par les lignes pointillées, la ligne pleine correspondant à l'hyperplan. Les vecteurs supports sont les données les plus proches de l'hyperplan.

termes de résultats [95, 96, 125, 90] encore très récemment [153, 124]. Les architectures profondes prennent en entrée des niveaux de gris normalisés, afin de construire les descripteurs en même temps que l'apprentissage de la frontière entre les classes. Les premiers étages de l'architecture agissent comme les étapes de calculs de descripteurs, puis les étages suivants classent les informations. Un point plus détaillé sur les réseaux convolutionels est effectué dans le Chap. 7.

2.1.2 Modèles génératifs

Les modèles génératifs permettent de modéliser les apparences d'un objet à partir des exemples disponibles. La force de ce genre de modélisation est de permettre de générer des apparences non disponibles pas au départ ou de régénérer des apparences à partir du modèle. Souvent, un modèle génératif permet l'interprétation du modèle. La plupart des techniques d'apprentissage génératif sont des algorithmes dits non supervisés, c'est-à-dire qu'ils ne nécessitent pas de données labellisées. Cependant, dans les tâches de Détection et Reconnaissance, souvent plusieurs modèles génératifs seront mis en correspondance. Dans ce cas, chaque classe doit être apprise séparément.

Partitionnement de données

L'une des techniques les plus simples à mettre en oeuvre est le partitionnement de données (*Data Clustering*). Il s'agit de modéliser les données en regroupant celles qui sont similaires sous forme de représentants clefs, appelés centroïdes. L'information de la base d'exemples est ainsi condensée. L'espace parcouru est découpé en zones labellisées, appelées clusters. De très nombreux autres algorithmes de partitionnement de données ont été développés, le lecteur intéressé pourra se référer à [79]. Il est également possible de faire du partitionnement de données en utilisant des réseaux de neurones. Les cartes auto-organisées (réseaux de Kohonen [87]) cherchent à trouver la structure qui génère les différentes apparences d'un objet. À chaque ensemble d'apparences est associé un

neurone. Inversement, il est possible d'utiliser les activations de ces neurones pour obtenir des apparences. Une autre méthode de partitionnement de données est l'algorithme de K-moyennes [2], qui construit des centroïdes qui sont les barycentres des points proches de la même classe. Ce genre de techniques est envisageable pour des bases de données jugées représentatives de l'espace de travail, comme par exemple dans [178].

Variétés

Une autre manière de modéliser des données de manière non supervisée est d'utiliser des méthodes de réduction de dimension, qui permettent de générer des données à partir d'un petit nombre de variables dites *latentes*, c'est-à-dire inconnues a priori. Des méthodes de réduction de dimension classiques comme l'Analyse en Composante Principale [73] sont couramment utilisées pour résoudre la problématique de Détection [118, 129]. De même, certains travaux utilisent des réseaux de neurones pour apprendre des variétés à des fins de détection [43, 67]. Ce genre de réseaux fait partie de la famille des mémoires associatives, conçues pour conserver le maximum d'information dans un minimum de neurones. Les autoencodeurs [88] et les machines de Boltzmann [1] sont des réseaux capables d'apprendre la variété regroupant toutes les apparences d'un objet. Un point plus détaillé sur les variétés et leur utilisation dans les problématiques traitées est abordé dans le chapitre 5.

Densités mélanges

Enfin, l'utilisation de densités mélanges (*mixture models* en anglais) est une autre méthode de modélisation de données de manière non supervisée utilisée en Détection d'objets ([118, 184, 101]). Elle consiste à approximer une distribution de probabilité quelconque par une somme de distributions plus simples (souvent gaussiennes). Ensuite différentes techniques sont utilisées pour estimer les paramètres de ces distributions, l'une des plus utilisées étant l'algorithme Expectation Minimization (EM [29]).

2.1.3 Structure d'un modèle

Il existe plusieurs structures pour modéliser un objet grâce à l'apprentissage statistique. L'un des modèles les plus classique est le modèle dit *à parties*, introduit dans [47]. L'objet est alors représenté comme des parties interconnectées. Les connexions peuvent être modélisées explicitement, (*ad hoc* ou par apprentissage [126, 172]) ou implicitement [98]. Dérivant de la structure par parties, il existe les modèles dits en constellation [44], en étoile [41], un mélange des deux [22] ou encore en arbre [137].

Si ces approches permettent de gérer les déformations partielles et veulent modéliser la complexité de certains objets, elles restent lourdes à mettre en place. Une structure plus simple consiste à considérer un ensemble de modèle rigide, sans partie et sans lien entre eux. Cette technique est appelée le *Template Matching* [8]. Elle consiste à modéliser un objet par tout ou une partie de ces apparences connues, que ce soit directement ou par le biais de descripteurs comme ceux présentés dans la section suivante. Cette approche est très simple, mais pâtit des difficultés liées aux changements de points de vue ou encore de la variabilité intra-classe. Quelques méthodes plus récentes, appelées *Deformable Template Matching* ou *active shape models* ont été mises en place [21, 18] et consistent à donner un peu de latitude aux modèles en ajoutant de possibles déformations locales.

2.1.4 Combinaison de classifieurs

Il est également possible de combiner plusieurs classifieurs différents afin d'obtenir de meilleurs résultats. Leur apprentissage peut se faire conjointement, comme dans le boosting [50]. Une fusion peut sinon être effectuée a posteriori, comme l'ont fait les organisateurs du challenge PASCAL VOC en créant un super classifieur à partir des résultats de tous les participants [37].

Boosting

Pour l'apprentissage conjoint de classifieurs, la méthode la plus utilisée est le boosting. Le principe est de combiner les classifieurs dits 'faibles' pour les fusionner et obtenir alors un classifieur dit 'fort'. Un classifieur sera dit faible s'il a une probabilité de bonne classification légèrement au dessus du hasard. En utilisant un système de vote entre les différents participants, il peut être montré que le boosting améliore la capacité de décision. Un des algorithmes les plus utilisés en Détection est Adaboost (*Adaptative Boosting*, [168]. Des variantes du boosting tel que gentleboost [51] et realAdaboost [146] sont également utilisées. Une illustration de cette technique est présentée Fig. 2.2.

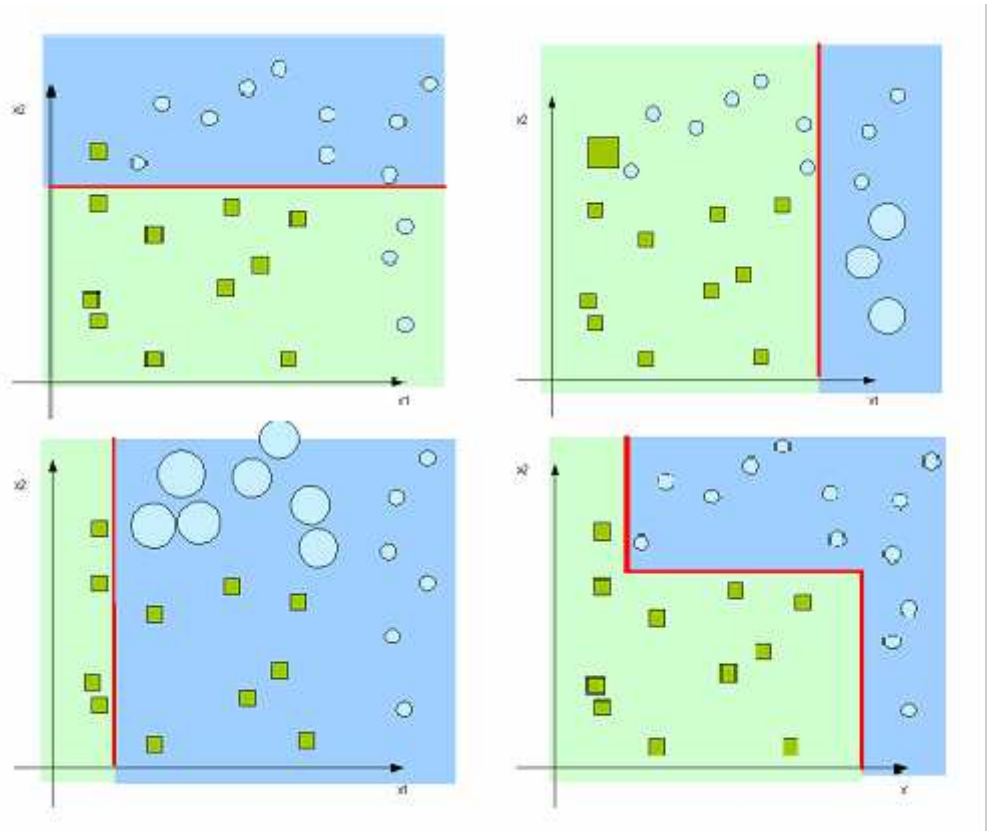


FIGURE 2.2 – Principe du boosting. À chaque nouveau classifieur, un classifieur est appris en priorité sur les exemples mal classés, afin d'apprendre des classifieurs complémentaires. Chaque classifieur seul est mauvais, mais regroupés ils permettent de modéliser la frontière.

Cascades

Une cascade de détecteurs prend plusieurs détecteurs les uns à la suite des autres, chacun éliminant les hypothèses les moins probables [168, 65, 165]. Ainsi, les classifieurs simples sont utilisés pour les premières étapes et enlèvent les négatifs considérés comme les plus évidents, tandis que les derniers classifieurs seront plus précis mais aussi plus lourds à utiliser sur une fenêtre. Cela permet de gagner en temps de calcul, mais il est possible de perdre de la performance. Si le dernier classifieur est très performant, il est possible que ses résultats soient dégradés du fait que certaines bonnes détections ont été rejetées par les étages précédents de la cascade. La Fig. 2.3 illustre le principe.

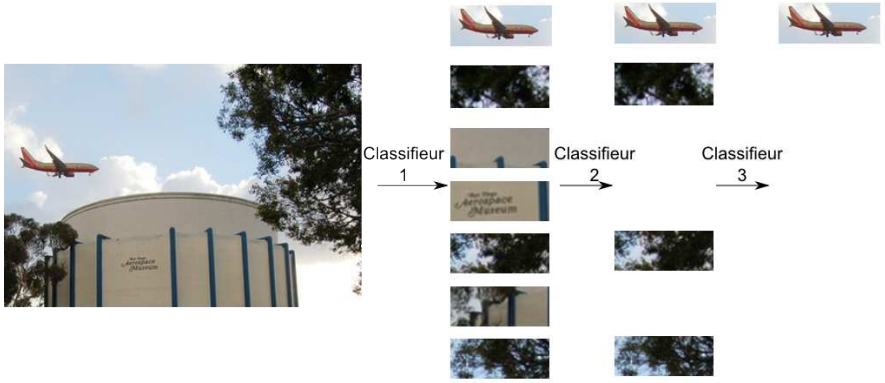


FIGURE 2.3 – Principe d'une cascade de classifieur. Chaque étage de la cascade retire des exemples négatifs.

Fusion

Enfin, il est également possible de fusionner les classifieurs en utilisant différentes théories de fusion, comme de simples combinaisons linéaires, moyennes [84], la logique floue [35], les probabilités [164] ou encore la théorie de Dempster Shafer [152]. Certaines fusions, plus spécifiques, tendent à maximiser la performance plutôt que de travailler sur les probabilités d'occurences des objets, comme par exemple [85]. Certains travaux apprennent un classifieur sur les classifieurs.

2.2 Localisation

L'apprentissage statistique permet d'apprendre des modèles à partir de données, mais ne permet pas en lui même de résoudre le problème de la localisation de l'objet d'intérêt dans une image. À cette fin, deux approches s'opposent. La première consiste à effectuer un parcours exhaustif de l'espace des poses, c'est-à-dire l'ensemble des positions de l'image, tandis que la seconde consiste à ne regarder qu'un sous ensemble de l'espace des poses, et à en déduire la position de l'objet.

2.2.1 Localisation par parcours exhaustif

Ainsi qu'annoncé, la première approche se base sur l'apparence de l'objet dans son ensemble. La méthode la plus simple consiste à scanner densément à différentes échelles et

différentes positions l'image à traiter. De très nombreux travaux utilisent cette technique [168, 126, 120, 40, 26, 183]. Ce parcours de l'image, version évoluée des premiers temps du *Template Matching* [8], est appelé *parcours par fenêtre glissante.*

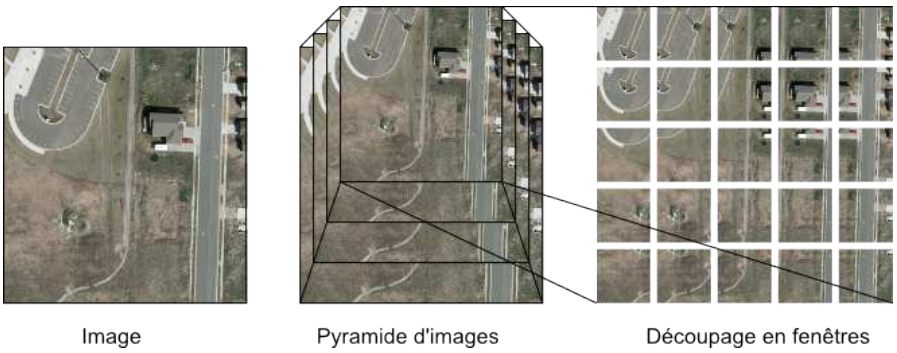


FIGURE 2.4 – Illustration du concept de parcours par fenêtre glissante. Le parcours s'effectue en deux étapes : la constitution d'une pyramide d'image, puis le découpage de cette pyramide d'image de manière exhaustive par des fenêtres qui se recouvrent.

Fig. 2.4 permet de voir les 3 étapes principales d'un parcours par fenêtre glissante. Tout d'abord, une pyramide multi-échelles de l'image est construite. Nous avons $h_{n-1} = r.h_n$ et $w_{n-1} = r.w_n$, avec w_n et h_n la largeur et la hauteur du niveau n de la pyramide, r le ratio entre deux niveaux, le niveau 0 étant l'image de départ sur échantillonnée. Dans un deuxième temps, chacun de ces niveaux est découpé en un certain nombre de fenêtres rectangulaires (celles-ci peuvent se chevaucher). Les méthodes pour choisir les tailles des fenêtres et leur chevauchement dépendent des chaînes algorithmiques. Enfin, pour chacune de ces fenêtres, les étapes de description et classification présentées dans les sections précédentes déterminent la présence ou l'absence du véhicule. Si cette technique permet d'être exhaustif sur les positions, elle peut être mise en difficulté dans des situations d'occlusion, où seule une partie de l'objet est visible. Les objets non convexes ou déformables présentent également des difficultés supplémentaires de par la présence de fond à l'intérieur des fenêtres. De même, les objets fortement déformables ont tendance à fausser ces méthodes car la taille des fenêtres est fixée à l'avance.

Des approches utilisant des parties déformables [42, 183] ont été conçues pour contourner ces problèmes. Des techniques ont été développées pour accélérer la recherche exhaustive, comme utiliser des images intégrales ou histogrammes intégraux [134, 168], ne traiter que certaines sous-fenêtres [16] ou encore effectuer une stratégie de séparation et évaluation (*branch and bound* en anglais [93]).

2.2.2 Localisation par hypothèse/validation

La deuxième technique la plus utilisée consiste à décrire l'objet comme une réunion d'éléments caractéristiques. Il s'agit de l'approche par hypothèse/validation. Chaque élément caractéristique localisé permet ensuite d'en déduire les positions probables de l'objet recherché. La Fig. 2.5 illustre le concept. Cette technique peut se décliner en de nombreuses approches différentes. Certaines d'entre elles utilisent des parties définies de manière *ad hoc*, (membres, tête, torse pour un détecteur de piéton par exemple [115, 119]). D'autres apprennent des dictionnaires de caractéristiques [99, 149, 108]. Le modèle liant ces caractéristiques/parties doit ensuite être appris [46, 136, 44, 147]. Ces travaux se rap-

prochent de l'idée de la transformée de Hough [75] et sont parfois appelés transformées de Hough généralisées [54, 53].

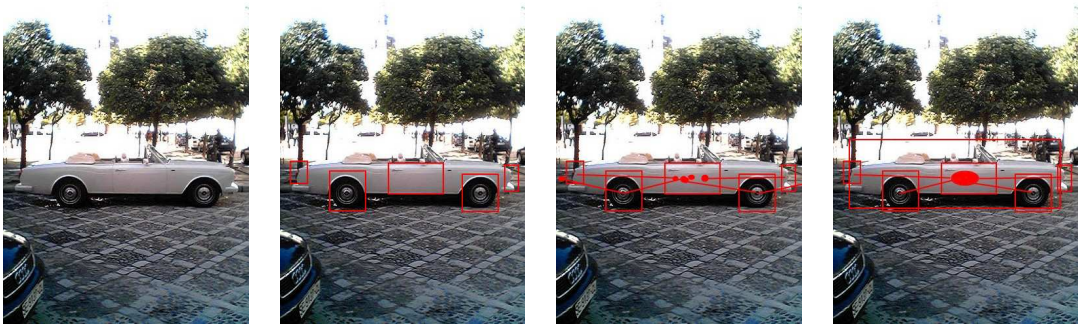


FIGURE 2.5 – Transformée de Hough généralisée. De gauche à droite : l'image de départ, les détections d'éléments caractéristiques, le vote des différentes parties, le résultat.

Ces approches permettent de gérer les occlusions et les déformations, mais sont beaucoup plus complexes à mettre en œuvre. L'hypothèse de parties bien définies et détectables doit être applicable, ce qui peut être limitant. Le problème a simplement été repoussé d'un cran, avec d'autres objets plus faiblement résolus à détecter, mais en plus grand nombre pour un seul objet. La difficulté de détection des parties est augmentée, mais les restrictions imposées par les modèles des relations entre les parties et le nombre de ces parties contrebalancent cet effet.

2.2.3 Suppression des Non Maxima

Lorsqu'un ensemble de localisations de l'objet a été déterminé, il est fort possible d'avoir deux détections pour deux emplacements proches dans l'image correspondant au même objet. Pour fusionner les résultats, l'algorithme glouton de Suppression des Non Maxima (NMS pour *Non Maximum Suppression* en anglais), est souvent utilisé [40, 168]. Il consiste à prendre toutes les détections, à les trier par score, puis la détection de meilleur score est retenue comme candidat. Les détections qui ont un critère de recouvrement trop élevé (aire en commun ou critère de Jaccard par exemple) par rapport à celle-ci sont enlevées de l'ensemble des détections, puis le processus est réitéré avec la détection de plus fort score restante jusqu'à ce qu'aucune détection n'en recouvre une autre (au sens du critère choisi).

2.2.4 Utilisation du contexte

Il est également possible d'ajouter des informations de contexte pour mieux détecter l'objet. Dans un cadre applicatif où il est possible que des images ne contenant pas l'objet à détecter soient présentes, l'image en elle même permet d'extraire de l'information [65]. De même, des a priori sur la position peuvent être retirés d'informations globales, comme dans [121], qui propose de localiser d'après le *gist* (essence) de l'image, les positions les plus probables pour un objet. Les travaux de [182] proposent de détecter les routes pour ensuite détecter les voitures. Il est à noter que dans notre contexte applicatif, nous ne souhaitons pas avoir de biais dû au contexte. En effet, il est important de détecter pareillement un véhicule sur une route que sur un terrain quelconque.

2.3 Primitives visuelles

Si l'utilisation de l'apprentissage statistique permet d'obtenir de bons résultats, il est nécessaire de l'appliquer sur des données ayant de bonnes propriétés intrinsèques. Ainsi, les données brutes des capteurs sont rarement utilisées telles quelles, mais sont plutôt transformées en primitives. Ces dernières sont conçues ou apprises afin de maximiser l'efficacité de l'apprentissage statistique, que ce soit en réduisant le bruit, en ayant des propriétés topologiques intéressantes ou en mettant en valeurs des caractéristiques particulières. Dans cette section, nous présentons différents types de primitives utilisées pour traiter les tâches de Détection et de Reconnaissance.

2.3.1 Primitives 2D

Un objet peut être décrit au moyen d'une collection d'images. Il s'agit alors d'extraire des primitives de ces images. Cette technique est la plus répandue dans l'état de l'art, d'une part du fait qu'il est plus facile de disposer d'images d'un objet plutôt que de son apparence 3D complète, et d'autre part que cette approche permet d'avoir des données équivalentes entre la phase de construction du modèle et la phase de test. Les primitives 2D peuvent avoir une approche globale, c'est-à-dire prendre l'objet dans sa globalité ou locale, c'est-à-dire voir l'objet comme la somme de parties.

Information globale

La méthode par extraction d'information globale se fonde sur des indices de caractéristiques de l'objet modélisé comme un tout. Certains descripteurs s'appuient sur des transformations d'espace comme les coefficients de Fourier, les coefficients de Fourier-Clifford [5], les ondelettes de Haar [110] ou de Daubechies [27]. Dans la même idée, les filtres de Gabor [52] donnent une décomposition en orientation et en échelle d'une image (quelques illustrations Fig. 2.6). Des descripteurs de gradients, de contours ou de silhouettes [55, 19, 7] permettent également de décrire l'objet dans son ensemble. De par leur nature, ces descripteurs sont peu robustes aux occlusions, changements de points de vues et déformations. Ils peuvent cependant donner des résultats remarquables en détection [168, 169, 36].

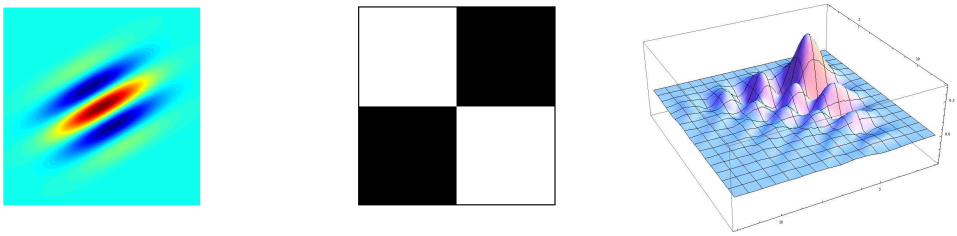


FIGURE 2.6 – À gauche : filtre de Gabor. Au centre : ondelette de Haar. À droite : ondelette de Daubechies

Information locale

L'une des méthodes de description utilisée est la méthode par extraction de point d'intérêt. Elle consiste à considérer des parties de l'objet dans l'image qui seront invariantes

à de légers changements de l'image, à savoir des invariances par rotation, translation et intensité. L'objet est ainsi représenté comme une collection de détails. Par conséquent, s'il est vu selon un angle légèrement différent et sous une intensité différente, les points d'intérêt resteront en grande partie les mêmes. Les points d'intérêt les plus utilisés sont les points d'intérêt SIFT (Scale Invariant Feature Transform) [105], ceux de Harris [64], les points SURF (Speeded Up Robust Feature [6]) ou FAST (Features from Accelerated Segment Test [138]). Ces différentes techniques permettent de trouver des points qui sont remarquables dans une image (illustration Fig. 2.7).

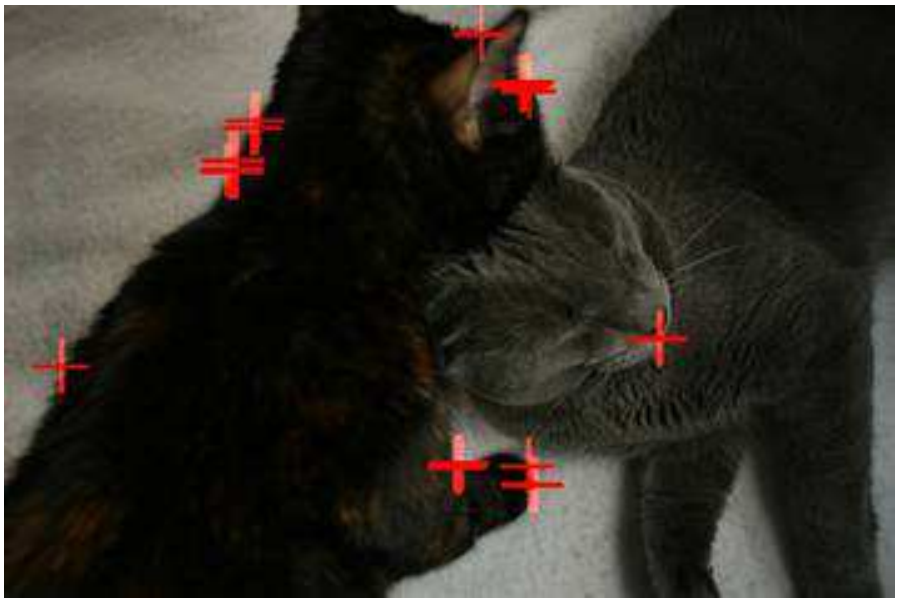


FIGURE 2.7 – Résultat d'une détection de points de Harris dans une image.

Une fois ces points détectés, des descripteurs locaux leurs sont associés. Les points SIFT et ceux de Harris possèdent des descripteurs associés, qui sont basés sur les gradients. Le descripteur HOG [26] peut être vu comme une version plus simple de SIFT (illustration Fig. 2.8). D'autres, tels les LBP (Local Binary Pattern [171], LTP (Local Trinary Pattern [159], BRIEF [10] sont basés sur des différences de pixels par paires, sur des voisinages plus ou moins grands (illustration Fig. 2.8). Enfin, d'autres descripteurs se basent sur la couleur, comme les histogrammes de couleurs [158] ou les corrélogrammes de couleurs [77].

Agrégation de descripteurs

Il est également possible de mélanger les deux approches. Dans ce cas, il s'agit de donner une structure aux informations locales. La méthode par sac de mots [23] est une des techniques les plus courantes (son nom provient de l'analyse et la classification de texte). Elle consiste à prendre des descripteurs locaux de points d'intérêts et à considérer chacun d'entre eux comme un mot visuel. Un histogramme de ces mots visuels est constitué et les techniques de classification sont ensuite utilisées sur ces histogrammes [44]. L'histogramme peut être brut [24] (dans ce cas là il s'agit presque d'une méthode purement globale) ou spatialement organisé (en pyramides de descripteurs par exemple [94]). La décomposition d'un objet en mots visuels est illustrée Fig. 2.9. Il est possible de

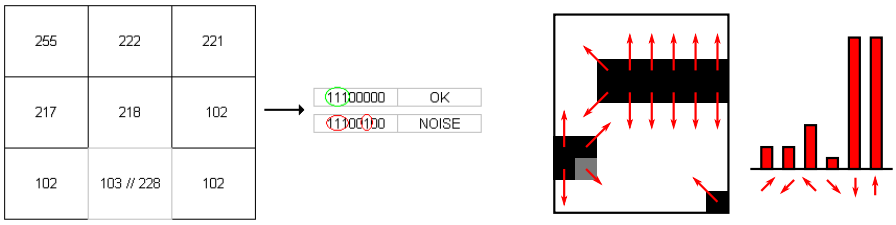


FIGURE 2.8 – Illustration des descripteurs LBP et HOG. Pour le LBP (à gauche) le pixel central est comparé à ses voisins. Cette comparaison donne lieu à des codes binaires, qui sont ensuite gardés ou rejetés en fonction de critères simples (pour plus de détails se référer à [171]). Pour le HOG (à droite), les gradients sont calculés sur le voisinage, puis, pour un nombre prédéterminé d'orientations. Chaque gradient vote selon sa magnitude sur son orientation. Le résultat est un histogramme des orientations du gradient.

calculer densément les descripteurs en chaque pixel, plutôt que de prendre uniquement les descripteurs sur des points d'intérêts, comme dans [26, 171, 159]. Ces histogrammes peuvent de même être spatialement organisés en cellules, le plus souvent cartésiennes, mais aussi en des configurations polaires [26]. Souvent, les extractions denses se basent sur des histogrammes intégraux de descripteurs [134], qui permettent de diminuer le temps de calcul.

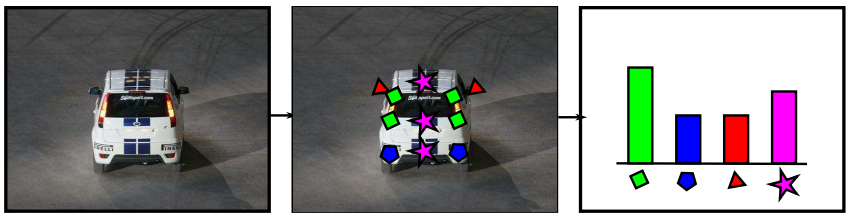


FIGURE 2.9 – Principe du sac de mots. Des points d'intérêt et leurs descripteurs locaux sont extraits, puis, après avoir généré un dictionnaire de ces mots visuels, leur histogramme est construit. Cet histogramme sert de descripteur de l'image.

Une autre manière d'organiser les descripteurs est d'utiliser un découpage en régions pour éviter le côté arbitraire de la découpe en cellules [61] et ainsi ajouter des descripteurs propres à la région, comme son contour, sa taille, ses couleurs.

Enfin, une des techniques les plus étudiées [40, 183] consiste à modéliser l'objet à la fois par de manière globale par ce qui est appelé *racine*, chacune d'entre elle étant en relation avec des descriptions locales, nommés *parties*. Un point plus complet sur les racines est abordé Chap. 4. Cette représentation permet de gagner en flexibilité sur des objets déformables ou articulés, tout en gardant la force des modèles globaux (toutes les structures de modèle présentées précédemment sont possible).

2.3.2 Apprentissage de descripteurs

Il est également possible d'apprendre des descripteurs de manière automatique, au lieu d'utiliser les descripteurs mentionnés. Les Fisher Vector [78] par exemple utilisent

un modèle génératif pour caractériser une image, en étendant l'idée des BOW, tout en apprenant la structure de l'espace des mots visuels. De même, les travaux basés sur les réseaux convolutionels, tels que [151, 57] ont pour but d'apprendre les descripteurs les plus adaptés possibles. Le principe consiste à apprendre un réseau convolutionel, puis à utiliser la sortie de la dernière couche de convolution comme un descripteur. Les travaux récents de [153] montrent que l'utilisation de tels descripteurs permet de gagner significativement en performance, et ce dans de nombreuses problématiques proche de la Détection. Comme la plupart des travaux dans le domaine de la détection, ces techniques sont appliquées à des images web plutôt bien résolues. En effet, les images à détecter sont ré-échantillonnées afin d'avoir une taille fixe de 221x221 pixels, très éloignée des tailles d'objets que nous souhaitons détecter. Plus de détails sur les réseaux convolutionels sont donnés dans le Chap. 7.

2.3.3 Primitives 3D

Pour cette approche un modèle 3D est supposé disponible. Étant donné qu'il doit *in fine* le comparer à une image 2D, la difficulté supplémentaire de la comparaison 2D/3D doit être surmontée. Ceci est compensé par l'apport d'informations plus riches. Il est toujours possible d'utiliser ce modèle 3D pour revenir au cas précédent en générant des images de vues caractéristiques de l'objet dans un panel de conditions représentatives. La comparaison 3D/3D se fait également, en particulier dans la reconnaissance de visages [34], avec une acquisition de modèle 3D et une base de modèles 3D. Dans cette thèse, nous ne nous intéresserons cependant qu'à la reconnaissance dans des images 2D.

Il est possible d'approximer un modèle 3D en tant que suite de patchs de vues 2D, positionnés en 3D [139], puis de chercher un appariement entre les patchs du modèle et les patchs 2D extraits de l'image. Cet appariement s'effectue en utilisant les contraintes géométriques entre les patchs (méthode RANSAC améliorée). La limitation principale de cette approche est le besoin d'un grand nombre de patchs positionnés en 3D et donc la nécessité d'avoir une bonne résolution à la fois pour construire les modèles 3D et pour reconnaître l'objet. Cette technique ne permet également que de reconnaître un objet particulier et n'est pas robuste au changement intra-classes.

L'utilisation de la 3D est plus efficace pour des objets bien connus, souvent pour des objets industriels dont un modèle CAO est disponible [101, 154].

2.4 Évaluation des performances

Une fois l'information pertinente extraite des images et une fois celle-ci modélisée par rapport aux applications souhaitées, il est nécessaire d'évaluer la performance d'une chaîne de traitement algorithmique. C'est pourquoi il est très important d'utiliser des protocoles clairs et reproductibles, correspondant à l'application visée.

2.4.1 Décision en Détection

Une fois les modèles appris, qu'ils soient discriminants ou génératifs, il est nécessaire de transformer les modèles en une décision, qui peut être continue (scalaire) ou binaire (présence/absence de l'objet).

Pour les modèles génératifs, il est possible soit d'utiliser directement la probabilité d'apparence de l'objet sachant le modèle et la donnée, soit d'utiliser des règles telles que la probabilité a posteriori ou le maximum de vraisemblance, comme par exemple dans

[99]. Pour les modèles discriminants, l'utilisation d'une distance à la frontière modélisée est utilisée pour prendre une décision.

Pour les deux types de modélisations, les décisions scalaires sont transformées en décisions binaires par un seuillage. Le plus souvent, le seuil est choisi en fonction des performances à atteindre. Par la suite, nous appellerons une *détection* un point de l'espace des poses qui a été étiqueté comme contenant l'objet recherché.

2.4.2 Positifs/Négatifs : quelques définitions pour la Détection

Les algorithmes précédemment présentés ont comme sortie un ensemble de *détections* sur l'espace des poses. Celles-ci doivent être reliées à la réalité terrain, c'est-à-dire à des annotations définies comme le meilleur résultat possible (dans le cas d'annotations parfaites), afin d'évaluer les performances d'un algorithme de Détection.

Un Vrai Positif (VP) sera défini comme une détection donnée par un algorithme qui est effectivement l'objet recherché. Un Faux Positif (FP), en revanche, consistera en une détection sur une position pour laquelle l'objet n'est pas présent. Enfin, naturellement, les Vrais Négatifs sont les positions pour lesquelles aucune détection n'est fournie et il n'y a effectivement pas d'objet et les Faux Négatifs sont les positions pour lesquelles un objet est présent, mais aucune détection n'a cependant été retournée.

Un protocole d'évaluation se doit de définir si une détection donnée est un Vrai Positif. Une fois l'objet détecté, il peut être nécessaire de savoir le format sous lequel le résultat est souhaité. La plupart du temps, ce résultat est assez direct et découle du modèle qui a été appris. Il suffit en effet de donner le rectangle constitué des bords de la fenêtre utilisée lors de la détection [26]. Cependant, pour les détecteurs possédant plusieurs parties mobiles de positions inconnues, il s'agit de calculer les coordonnées maximales de la boîte englobante. Ces coordonnées peuvent être calculées par une méthode de régression linéaire, comme dans [40].

La plupart des algorithmes prennent en compte une délimitation rectangulaire autour des objets, appelée boîte englobante. Une prédiction est considérée comme correcte si la position et l'échelle de cette boîte englobante sont correctes par rapport à une boîte englobante *vraie*. Un des moyens de vérifier les deux à la fois est d'utiliser l'indice de Jaccard (par exemple [38]), défini comme la fraction de l'aire de l'intersection entre les boîtes sur l'aire de l'union de ces boîtes. D'autres travaux utilisent simplement le recouvrement entre les deux boîtes [83].

Quand plusieurs objets sont sur une image, plusieurs détections pourraient être assignées à une même vérité terrain, par exemple lorsque différents objets sont alignés et que chacun masque partiellement un objet derrière lui. Une méthode naïve peut se contenter d'assigner chaque détection à son plus proche voisin par un algorithme glouton, comme fait dans la plupart des challenges. Ce problème peut être géré de différentes manières, par exemple par la construction d'un graphe bipartite entre les détections et les objets [103, 80] ou bien en utilisant un critère de relaxation lors de l'optimisation de l'attribution des détections [72, 112]. Lors de multiples détections sur le même objet, soit la détection est considérée comme un faux positif supplémentaire, soit la détection est ignorée et retirée du décompte des faux positifs.

Il est à savoir que l'utilisation des boîtes englobantes n'est pas sans poser des problèmes, ainsi qu'indiqué dans [71]. La Fig. 2.10 présente deux illustrations de problèmes courants sur l'évaluation de performances. Les problèmes évoqués peuvent être évités en fonction de l'application visée.

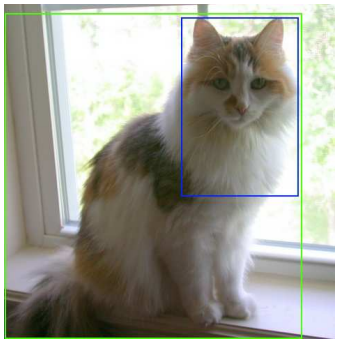



FIGURE 2.10 – À gauche : la prédiction (rectangle bleu) est considérée comme mauvaise de par son indice de Jaccard, car la vérité terrain (rectangle vert) ne couvre pas que l'objet. À droite : bien que la prédiction couvre l'intégralité du chat et seulement lui, (rectangle bleu), il s'agit d'un faux positif car l'indice de Jaccard est trop faible. Images extraites de Pascal VOC.

Certains challenges [142] proposent d'annoter l'intégralité de l'image, pixel par pixel et utilisent donc en métrique le taux de pixels bien classés. Ce taux permet de s'affranchir des problèmes posés par l'annotation rectangulaire mais reste problématique quant à la performance d'un algorithme vis-à-vis de la détection d'un seul objet contre tous les autres.

2.4.3 Le calcul de la performance en Détection

Une fois les détections associées à des objets ou considérés comme des faux positifs, il reste à évaluer la performance totale. Tout d'abord, ainsi qu'expliqué dans [31], il est possible d'opposer la mesure de performance *par fenêtre*, utilisée dans plusieurs travaux de détection [26], à la mesure de performance *par image*, autre norme pour la détection [133]. Le cas *par fenêtre* suppose que la tâche de détection est évaluée en classant des fenêtres centrées sur des objets d'intérêt, contenant ou ne contenant pas l'objet à détecter. L'hypothèse étant que meilleur est le score *par fenêtre*, meilleur sera le score *par image*. En pratique, les travaux présentés dans [31] ont montré que ce n'était pas toujours le cas. Mesurer la performance par fenêtre permet de s'affranchir d'une taille d'image standardisée.

Ainsi qu'expliqué par [28], les courbes Receiver Operator Characteristic (ROC) [106] sont couramment utilisées (par exemple [168, 140]) pour présenter des résultats sur des problèmes de classification. Ces courbes font l'hypothèse que la chaîne algorithmique renvoie une liste de détections chacunes associées à des scores. La courbe ROC est dessinée en affichant la fraction de Vrais Positifs sur le nombre de Positifs (TVP = Taux de Vrai Positif $TVP = \frac{VP}{P}$) contre la fraction de Faux Positifs sur le nombre de Négatifs (TFP = Taux de Faux Positifs $TFP = \frac{FP}{N}$) et ce pour différentes valeurs de seuil. La courbe ROC est parfois substituée par la courbe Detection Error Tradeoff (DET) (par exemple dans [26, 165]), qui représente le Taux de Faux Rejet contre le Taux de Fausses Acceptation. Elle fournie la même information, mais utilise 1-TVP au lieu de TVP.

Cependant, les courbes ROC peuvent présenter une vue très optimiste d'un algorithme de performance s'il existe un biais dans la distribution des classes. C'est typiquement le cas des algorithmes utilisant des *parcours par fenêtre glissante*, car la majorité

des fenêtres à tester sont des négatifs. En conséquence, une grande variation dans le nombre de faux positifs peut amener à de petites variations sur la courbe ROC. En effet, que l'ensemble des fenêtres négatives soit évalué une seule fois ou des milliers de fois, la courbe ne change pas. Cependant, le nombre de faux positifs est alors multiplié par mille.

En conséquence, pour ce genre d'algorithmes travaillant sur des répartitions de données pour lesquelles le ratio positif/négatif est très déséquilibré, la courbe Précision Rappel [135] donne une image plus représentative de la performance d'un algorithme. Le Rappel est défini comme la fraction des Vrais Positifs sur le nombre de Positifs (P) c'est-à-dire $Rappel = \frac{VP}{P}$, tandis que la Précision est la fraction du nombre de Vrais Positifs (VP) parmi toutes les détections considérées comme positives (Faux Positifs FP plus Vrais Positifs VP), c'est-à-dire $Precision = \frac{VP}{VP+FP}$. Le Rappel est parfois appelé taux de détection, car il représente le pourcentage d'objets effectivement détectés parmi tous les objets à détecter. Cette courbe est très souvent utilisée pour représenter des résultats de détection [60, 166, 128, 42, 38]. Évidemment, elle présente le défaut inverse de la courbe ROC, qui est qu'un ajout arbitraire de fenêtres à tester comprenant des négatifs cause une chute de performance.

Enfin, il est possible d'utiliser une troisième courbe, en traçant le Rappel en fonction du taux de Faux Positifs Par Image (FPPI) ou du taux de Faux Positifs Par Fenêtre (FPPF), définis tels que $FPPI = \frac{FP}{nbImages}$ et $FPPF = \frac{FP}{nbFenTest}$.

Au lieu de représenter la performance par une courbe, il est souvent plus approprié d'utiliser des points opérationnels. Par exemple, dans [26], le point choisi est Rappel à 10^{-4} FPPI, tandis que dans [31], le Rappel pour 1 FPPI a été utilisé. Dans un contexte industriel, les traitements algorithmiques rencontrent des spécifications, c'est pourquoi ces métriques sont plus souvent utilisées par les entreprises. D'autres points sont parfois utilisés, comme le point d'erreur égale, qui correspond au point où le Taux de Faux Positif est égal au rappel. C'est un choix arbitraire mais facile à calculer.

Enfin, une autre manière de représenter les performances des algorithmes consiste à utiliser des intégrales des courbes précédemment vues (ROC,PR). Ces intégrales sont appelées, respectivement, Area Unders ROC (AUROC) et Average Precision (AP). Ces deux scalaires ont l'avantage d'être compris entre 0 et 1, 1 étant un classifieur parfait.

2.4.4 Le calcul de la performance en Reconnaissance

Lors de la reconnaissance, chaque donnée se voit attribuer un label. Il est alors possible de définir la matrice de confusion telle que :

$$M(i,j) = \frac{\sum O_{i,j}}{\sum O_i} \tag{2.1}$$

où O_i est l'objet appartenant à la classe i et $O_{i,j}$ est l'objet appartenant à la classe i attribué à la classe j. Ainsi, la diagonale de la matrice de confusion donne les taux de bonne reconnaissance de chaque classe. De plus, elle permet de voir à quelles classes correspondent les faux positifs et en déduire si des classes sont proches ou éloignées, donnant ainsi une idée de la proximité inter-classe.

En plus de la matrice de confusion, il est possible de définir deux indicateurs qui permettent de résumer les performances d'un classifieur multi-classes. Le macro averaging est la moyenne de la diagonale de la matrice de confusion. Cette mesure permet de connaitre la performance moyenne sur n'importe quelle classe et ce quel que soit le nombre d'éléments dans chaque classe. Le micro averaging consiste à prendre le nombre de Vrais

Positifs et à diviser par le nombre d'éléments à classifier. Ainsi une performance globale sur la probabilité de reconnaissance attendue est donnée, toutes classes confondues.

2.4.5 Robustesse statistique

Les indicateurs donnés précédemment permettent d'avoir des performances sur des ensembles de données. Une problématique importante est de savoir dans quelles mesures les performances obtenues sont représentatives des applications visées. Afin de mesurer la robustesse des algorithmes lors du changement de données, plusieurs méthodes, appelées méthodes par validation croisée existent. Une méthode simple et qui permet d'utiliser un maximum de données disponibles est d'utiliser toutes les données à l'exclusion d'une pour créer le modèle, puis de tester le modèle sur cette donnée mise à l'écart et enfin de recommencer pour chaque donnée disponible. Cette estimation, appelée *Leave-One-Out cross-validation* (LOOCV, littéralement validation croisée Laisser-un-dehors) est la meilleure disponible, cependant elle est en pratique difficile à mettre en place au vu du nombre de simulations à effectuer (une par donnée disponible). La technique de validation croisée simple est plus souvent utilisée. Les données sont découpées en trois ensembles, un pour apprendre (l'ensemble *d'apprentissage*), un pour tester afin de régler les paramètres (appelé ensemble de *validation*) et un dernier pour tester les performances effectives de l'algorithme une fois terminé (ensemble de *test* à proprement parler). Cette technique est utilisée par exemple dans [38]. Enfin, une technique à mi-chemin appelée méthode par "folds" consiste à découper les données disponibles en k ensembles, les *folds*. Il s'agit ensuite d'apprendre sur $k-1$ folds, tandis que le test se fait sur le fold restant et ce pour les k folds. Ainsi, il est possible de calculer une performance moyenne et une variance de cette performance, permettant d'estimer la généralisation de l'algorithme développé.

2.5 Conclusions

Dans ce chapitre, nous avons vu les principales techniques utilisées afin de résoudre les problématiques de Détection et de Reconnaissance. Ces techniques sont basées sur l'apprentissage statistique, couplé à des représentations efficaces et des techniques de localisation de l'objet pour ce qui est de la Détection. Enfin, nous avons donné les bases pour effectuer une évaluation des performances rigoureuse.

Chapitre 3

Bases de données

Comme expliqué dans le Chap. 2, la plupart des approches modernes de Détection et Reconnaissance se fondent sur des techniques d'apprentissage statistique, qui nécessitent des données d'entrainement. Il est de plus crucial de pouvoir comparer les algorithmes les uns avec les autres, ce qui suppose là encore de pouvoir disposer de données annotées (images pour lesquels les véhicules sont localisés et identifiés). En conséquence, de nombreuses bases de données d'images ont été construites et rendues publiques afin que la communauté de vision par ordinateur puisse les utiliser pour entraîner ses algorithmes et comparer ses résultats. Si la plupart d'entre elles se concentrent sur de la reconnaissance de scène ou de la classification [120, 96, 163, 130, 144], seule une petite partie couvre la détection d'objets à proprement parler.

Dans ce chapitre, nous ferons la différence entre un *ensemble d'images*, que nous définissons comme un regroupemet d'images éparses, une *collection d'images*, que nous définissons comme un ensemble d'images annotées (disposant donc d'une vérité terrain) et une *base de données d'images*, que nous définissons comme une collection d'images enrichie d'un protocole d'évaluation de performance.

Dans ce chapitre, nous allons d'abord présenter les bases de données et collections qui sont disponibles pour entraîner et évaluer des algorithmes de Détection et Reconnaissance dans la section 3.1. Dans la section 3.2, nous exposerons la collection OIRDS, ainsi que les modifications que nous avons dû y apporter pour la transformer en base de données. La base VeDAI, nouvelle base de données publique pour la détection d'objets en imagerie aérienne, développée dans le cadre de cette thèse, sera présentée dans la section 3.3. Enfin, les bases de données Sagem seront abordées section 3.4.

3.1 Bases de Données de référence

Il est possible de catégoriser les sujets de Détection et Reconnaissance que couvrent ces bases de données comme suit : (i) les piétons, (ii) les visages, (iii) les objets de la vie de tous les jours, (iv) les véhicules. Un résumé des différentes bases est donné au Tab. 3.4.

3.1.1 Personnes et piétons

La détection de personnes/piétons est l'une des tâches les plus populaires, grâce aux très nombreuses applications qui peuvent en découler (surveillance, sécurité routière, sauvetage, etc). L'une des premières bases de données de référence de détection de piétons est la base INRIA person dataset [26]. Elle contient des centaines d'images découpées

à différentes résolutions (64x128, 70x134, 96x160). Des images de fonds sont également fournies, ainsi que la séparation en un ensemble d'apprentissage et un ensemble de test. Les images originales sont des photographies personnelles des membres du laboratoire de l'époque, ainsi que quelques images provenant d'internet. Les personnes apparaissent selon diverses poses et orientations, et ce sur des fonds très variés. Cette base de données a été créée pour aller au delà de la première base de données de détection de piétons – *MIT person dataset* [126] – qui était devenue trop simple.

De même, la *CalTech pedestrian dataset* [31] a été introduite pour remplacer la base de données de l'INRIA. Elle pose le problème de la détection de piétons pour une application de sécurité routière : les images sont prises depuis un véhicule roulant dans un environnement urbain. Elle contient 350,000 piétons labellisés dans 250,000 frames. Les occlusions sont annotées avec un système de deux boîtes englobantes, et les annotations sont reliées d'une frame à l'autre, permettant d'évaluer également des algorithmes de poursuite.

L'augmentation du nombre de données dans ces bases reflète une tendance actuelle d'augmentation de la production de données.

Pour ce qui est de la Reconnaissance, la base de Gavrila [120] regroupe des fenêtres déjà pré-découpées de piétons et autres négatifs difficiles.

3.1.2 Détection de visages

La détection de visages est une autre tâche de détection bien connue. Contrairement à la détection de piétons et bien que les applications soient tout aussi importantes, peu de bases de données libres existent. La plupart des bases de données existantes sont orientées vers de l'identification de visage [130]. La *CMU-MIT dataset* [140], qui inclut la *MIT dataset* [157], est une collection d'images de référence, qui a beaucoup été utilisée par le passé. Elle ne contient que 130 d'images différentes pour un total de 507 visages, tous de face et elle ne propose pas de protocole d'évaluation bien défini. Les résultats présentés dans les différents articles ne peuvent donc pas être comparés de manière fiable, comme expliqué dans [70]. Plus récemment, Kodak a rassemblé de nouvelles images afin d'évaluer les algorithmes de détection et reconnaissance de visages [104]. Cette base de données propose 300 images de tailles différentes, les visages présents variant d'une résolution de 13×13 pixels à 300×300 pixels. Pour les visages, il est à noter que beaucoup de bases de données sont tournées vers l'identification, qui est un problème différent de la Détection et la Reconnaissance, où on ne souhaite pas reconnaître un visage mais savoir à qui appartient ce visage (par exemple [76]).

3.1.3 Les objets du quotidien

D'autres bases de données poursuivent un but plus général, souvent avec des applications pour internet. Ainsi qu'expliqué en introduction de ce chapitre, la plupart d'entre elles sont tournées vers la Reconnaissance, comme ([96, 122, 59, 39]). D'autres sont plus orientées vers la Détection. Par exemple, l'*ETHZ dataset* [45] contient 5 classes différentes ('logo', 'bottle', 'giraffe', 'mug', et 'swan'), qui sont caractérisées par leurs formes. Le découpage en ensemble d'entrainement et ensemble de test a été effectué. Ses images proviennent de Flickr ou Google Search Images. Cette base contient des photographies, mais aussi des dessins, esquisses, peintures et images de synthèse. 40 images sont disponibles pour l'apprentissage, 40 pour le test pour chaque classe.

FIGURE 3.1 – Quelques exemples de véhicules dans le challenge Pascal VOC 2007 [38]. Les véhicules sont clairement centrés et bien résolus.

Les challenges récents traitent souvent les deux problématiques à la fois, en proposant de classer les images en catégories mais également de détecter les objets inclus dans ces images. Une des bases les plus connues contenant des objets du quotidien est celle utilisée pour le challenge Pascal VOC [38] – l'un des challenges les plus connus et reconnus dans la communauté de la vision par ordinateur, qui a eu lieu entre 2007 et 2012. Les dernières éditions de Pascal VOC présentent 20 classes réparties sur des milliers d'images. Le découpage en ensembles d'apprentissage, de validation et de test ainsi que le protocole d'évaluation sont fournis. Enfin, la collection *LabelMe dataset* [142] est celle présentant le plus de données. Elle est différente par le fait que, ainsi que son nom l'indique (Annote Moi), LabelMe peut être annotée par n'importe qui. Le nombre précis de classes change en permanence, ainsi que le nombre d'images annotées. En décembre 2006, date à laquelle un décompte précis a été effectué, elle contenait 111 490 polygones sur 11 845 images et 18 524 séquences vidéo.

3.1.4 Les véhicules

Plus proche de notre problématique, la détection de véhicules a également reçu l'attention de la communauté de vision par ordinateur. Cependant, la plupart des bases de véhicules se concentrent sur des images vues du sol, dans lesquelles le véhicule est le sujet principal (par exemple *INRIA Car dataset* [11]). Certains travaux utilisent des bases de données en vue aérienne, mais aucune d'entre elles n'est publique ([58, 156]). La seule collection existant avant la création de la base VeDAI (présentée section 3.3) est OIRDS (Overhead Imagery Reearch DataSet) [160], qui est présentée en détail section suivante.

En se restreignant aux bases de données de reconnaissance de véhicules, la plupart d'entre elles sont construites dans une optique de surveillance du traffic, tels que [107, 15] et ne sont souvent pas, elles non plus, disponibles. Il existe également des bases de données portées sur la classification très fine de véhicules, telles que [155] et [89]. Le challenge de ces bases de données est de donner pour chaque voiture sa marque, son modèle et son année de sortie. Les images de voitures analysées sont cette fois ci issues d'images web et ont une très bonne résolution (une résolution trop faible rendrait la tâche impossible).

3.1.5 Adéquation avec la problématique

Les bases de données précédemment présentées ne conviennent pas pour développer les applications de détection et reconnaissance de véhicules faiblement résolus, et ce, pour trois raisons principales.

Tout d'abord, la question du contexte applicatif reste en suspens. La plupart des données présentes dans ces bases sont des images extraites de Flickr, c'est-à-dire des photographies prises à l'aide d'appareils photo grand public. Si ces images sont bien

adaptées aux algorithmes de recherches d'images sur le web, le contexte est beaucoup trop éloigné des applications de détection en imagerie aérienne. Les problèmes de points de vue, de résolution des objets et de conditions météorologiques ne sont pas présents sur la plupart des images.

Ensuite, nous traitons des véhicules. En dehors des bases dédiées à cette catégorie d'objet, les piétons, les visages et une grand partie des objets issus d'images web sont fortement déformables. De plus, la variabilité intra-classe est souvent très forte (sofa, chaise). Ces variabilités ne correspondent pas à celles à traiter dans nos problématiques.

De plus, les objets d'intérêt sont souvent le sujet principal de la photographie, étant donné que la personne ayant pris cette dernière souhaitait justement capturer un sujet précis. Nous traitons une problématique différente, pour laquelle les objets n'occupent aucune place privilégiée dans l'image. Quelques photographies présentées Fig. 3.1 extraites du challenge Pascal VOC 2007 permettent de visualiser ce biais.

3.2 Overhead Imagery Research DataSet

3.2.1 Présentation

La base OIRDS (Overhead Imagery Research DataSet) est une collection de 900 images issues de différentes sources (USGS et VIVID), contenant 1800 vehicules répartis sur quatre classes ('truck', 'pickup', 'cars' et 'unknown'). Les annotations sont très riches. En effet la couleur, la présence de reflets spéculaires, la distance au sol lors de la prise de la photo, la taille du véhicule en pixel (aux alentours de 20x20 pixels) sont disponibles. Quelques images de cette base sont présentées Fig. 3.2 et quelques fenêtres typiques sont présentées Fig. 3.3. Il est important de noter que cette collection présente plusieurs problèmes, tant par sa difficulté d'utilisation que par des problèmes intrinsèques aux données.

En ce qui concerne l'utilisation de cette collection, l'ensemble des images est découpé en 20 sous-ensembles de données hétérogènes, répartis dans 20 dossiers différents. Les images provenant de sources différentes, les noms ne sont pas uniformisés et nécessitent une indexation à part, qui n'est pas fournie. Les annotations sont fournies en format Excel et la définition des différentes colonnes se trouvent sur la page web de présentation de la collection. Ces défauts nécessitent d'être corrigés avant toute utilisation rigoureuse de la collection.

En plus de ces défauts mineurs mais qui rendent la collection difficile à utiliser et à interfacer, d'autres problèmes viennent s'y ajouter. Tout d'abord, ainsi qu'annoncé, OIRDS est une collection et non une base de données. Aucun protocole d'évaluation et aucune séparation des ensembles entre test et validation ne sont fournis. Ensuite, deuxième faiblesse, les images proviennent de 20 sources différentes (ce qui implique des capteurs différents, des résolutions différentes, des contextes différents). Cette multiplication des sources rend difficile la reproductibilité des résultats en fonction du découpage de la base, car en moyenne chaque source ne fournit que 45 images. Troisièmement, les annotations n'ont pas été définies correctement lors de leur génération. En conséquence, les orientations des véhicules, en particulier, sont moyennées sur les 3 annotateurs et donnent donc des résultats erronés.

Les articles utilisant cette base de données (par exemple [83] qui sépare OIRDS en trois ensembles : un facile, un moyen et un difficile) utilisent leurs propres partages mais sans le définir précisément. De plus, les images de OIRDS sont parfois mélangées avec

FIGURE 3.2 – OIRDS : images représentatives.

des images de Google Earth, qui sont protégées et ne peuvent pas être distribuées. Les résultats ne sont donc pas reproductibles.

3.2.2 De la collection à la base de données

Les premières expériences concernant le codage des méthodes de l'état de l'art ont été effectuées sur la base OIRDS. Étant donné la disparité des données, nous avons décidé de découper la base en 2 ensembles, un contenant des images de tailles similaires, que nous appellerons *Ensemble des Petites Images* (EPI), et un contenant des images de tailles plus importantes (EGI, *Ensemble des Grandes Images*). Nous avons utilisé un protocole de validation croisée en 10 folds (voir 2.4.5 pour plus de détails). Comme la problématique

FIGURE 3.3 – OIRDS : fenêtres représentatives. Haut : fenêtres centrées sur des véhicules. Bas : fenêtres centrées sur des fonds

nous intéressant est la détection de véhicules faiblement résolus, les images ont été sous-échantillonnées afin que les véhicules ne soient pas plus grands que 20×20 pixels. Les Vrais Positifs sont définis comme les détections associées à une vérité terrain avec une distance de 10 pixels ou moins autour du centre. Si plusieurs détections sont attribuées à la même vérité terrain, celle de meilleur score est conservée et les autres sont éliminées. L'AP est utilisée pour mesurer les performances, en utilisant une interpolation à 11 points de la courbe précision rappel, comme c'est habituellement effectué ([111]). Nous avons utilisé la classe 'voiture' uniquement, les autres classes étant parfois ambigües dans leurs apparences. Enfin, il a été choisi comme indicateur l'Average Precision moyenne (mAP), qui est la moyenne sur les 10 folds de l'AP.

3.3 Vehicle Detection in Aerial Imagery

La base de données VeDAI (Vehicule Detection in Aerial Imagery) présentée dans cette section a été constituée au vu de l'absence de base de données se rapprochant suffisamment du cadre applicatif, la base OIRDS ayant montré de nombreuses limites. VeDAI se veut être le premier benchmark de détection de véhicules faiblement résolus.

3.3.1 Contraintes

Après avoir utilisé les bases INRIA, PASCAL VOC, et OIRDS, il a été possible d'effectuer les spécifications qu'une base de données qui suit les critères souhaités doit satisfaire :

- Les images doivent être libres de droit. Ce critère est assez limitant au vu du coût de production d'une image aérienne, en effet, la plupart des organismes affrétant un avion souhaitent conserver leurs droits sur les images prises en vol.
- Le nombre d'objets à détecter et leur type doit être assez varié. En pratique, 5 classes est un minimum pour des analyses pertinentes.
- Les fonds doivent être les plus diversifiés possible.
- Les objets doivent être faiblement résolus.
- Les annotations et les images doivent être faciles à parcourir et à utiliser.
- Un protocole de test clair doit être mis en place.

3.3.2 Images

Après avoir recherché différentes sources d'images aériennes disponibles, nous avons décidé d'utiliser les images proposées par le site Utah AGRC [173], qui donne accès à de nombreuses images aériennes orthonormées. Il est toutefois à noter que la métropole de Rennes propose une couverture satellite disponible gratuitement `http://www.data.rennes-metropole.fr/`, de même que le site `http://gis.ny.gov`. Un projet de regroupement d'images aériennes recensait divers sites d'images aériennes gratuites. Après quelques temps d'arrêt, il semblerait que le projet est récemment repris `http://wiki.openstreetmap.org/wiki/Vertical_Aerial_Photographs`. Sur le site de l'AGRC, nous avons sélectionné les données *HRO 2012 6inch photography*, qui ont une résolutions de 4.92in $\times$ 4.92in par pixel (12.5cm $\times$ 12.5cm par pixel). Toutes les images sont orthonormées et sont prises à la même distance du sol. Les images ont quatre canaux de couleur (trois canaux visibles et un en proche infrarouge).

Les images originelles étant beaucoup trop grandes pour un traitement efficace ou pour correspondre à une quelconque application pour des algorithmes de détection utili-

FIGURE 3.4 – VeDAI : images représentatives. Haut : images en couleurs. Bas : images proche infrarouge.

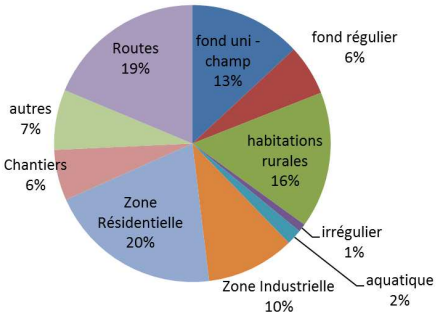


FIGURE 3.5 – Répartition des images en fonction de leur principal fond.

sant une seule caméra à un instant donné, nous les avons découpées en images plus petites. Cela nous a permis également de concentrer les images sur les régions intéressantes. En effet, la large majorité des images contiennent des régions de textures similaires, vides de véhicules (par exemple des lacs, des montagnes, des forêts). Ces nombreuses images ne feraient que baisser arbitrairement la performance lors des évaluations, ainsi qu'expliqué section 2.4.3.

Etant donné que l'annotation exhaustive n'était pas possible, un choix a été fait en

nom de l'ensemble	sigle	taille d'image	résolution(cmpp)	#canaux	type de canaux
Grandes Images Couleur	GIC	1024x1024	12.5x12.5	3	couleurs
Petites Images Couleur	PIC	512x512	25.0x25.0	3	couleurs
Grandes Images Infrarouge	GII	1024x1024	12.5x12.5	1	proche infrarouge
Petites Images Infrarouge	PII	512x512	25.0x25.0	1	proche infrarouge

TABLE 3.1 – Les différents ensembles de données présent dans la base VeDAI

faveur de la diversité. La Fig. 3.5 indique la répartition de différents types d'images, voir Annexe B pour quelques exemples. En effet, en sélectionnant les régions dites intéressantes, nous avons essayé autant que possible d'avoir la plus grande diversité dans la base de données, tant en termes de véhicules qu'en termes de fonds et de potentiels faux positifs. Tandis que certains travaux de l'état de l'art se concentrent sur un seul type d'environnement ([156]), en particulier des véhicules sur des routes dans des environnements urbains, VeDAI propose un grand nombre d'environnements différents (des champs, de l'herbe, des forêts, des montagnes, des bordures de lacs et de rivière, mais aussi des villes, des usines, des carrières...)

Les images qui contenaient trop de véhicules, par exemple des images de parking comme celle présentée dans [83] ont été exclues. En effet, un algorithme qui renverrait des positions aléatoires donnerait de bons scores sur de telles images, et évaluer la performance serait difficile. La Fig. 3.4 montre quelques images de VeDAI.

Au total, 1210 images 1024×1024 ont été annotées et inclues dans 4 sous-bases de données différentes : (i) les images réduites à leurs canaux couleurs, à leur résolution originelle, que nous avons nommée Grandes Images Couleur (GIC), (ii) ces mêmes images, en une résolution deux fois moindre, Petites Images Couleur (PIC), (iii) le canal infrarouge à la résolution originelle Grandes Images Infrarouge, (GII) et (iv), ce même canal infrarouge mais à une résolution moindre, appelé Petites Images Infrarouge (PII). Un résumé des différentes sous-bases est disponible Tab. 3.1.

La position exacte des images 1024×1024 dans les images de l'AGRC est également disponible. Les bases contenant les images "petites" sont les mêmes images que les grandes, sous échantillonnées jusqu'à une résolution de (512×512), dans le but de rendre les véhicules plus faiblement résolus. Les images couleurs sont constituées de trois canaux 8-bits (R,G, B), tandis que les images infrarouges sont constituées d'un seul canal 8-bits, proche infrarouge. Le découpage de la base en fonction de ces deux critères (taille et résolution), permet de couvrir deux besoins dans les applications de détection. Tout d'abord, nous pouvons étudier l'influence de la résolution sur les performances. Plus les objets détectés de manière fiable peuvent être petits dans l'image et être détecté, plus grande sera la portée de détection. Ainsi, ces performances permettront de dimensionner un capteur pour une application donnée. Ensuite, la différence de bande de fréquence permet de voir la robustesse vis-à-vis de l'utilisation des couleurs ou de la transposition des fréquences. De même que pour la portée, il est important de connaître l'apport d'un capteur 3 bandes par rapport à un capteur 1 bande.

3.3.3 Véhicules

La base de données contient 9 classes de véhicules, à savoir 'Airplane', 'Boat', 'Camping car', 'Car', 'Pick-up', 'Tractor', 'Truck', 'Van', et une classe 'Other' (respectivement avion, bateau, camping car, voiture, pick-up, tracteur, van, et autre). Des images typiques

de ces classes sont présentées Fig. 3.6. De plus, deux méta-classes ont été définies : la classe des 'small land vehicle', qui contient les classes 'car', 'pick-up', 'tractor' et 'van', et la classe 'large land vehicle', qui contient les classes 'Truck' et 'Camping Car'.

FIGURE 3.6 – VeDAI : Images illustrant les différentes catégories de la base de données. De gauche à droite : 'Car', 'Truck', 'Camping car', 'Tractor', 'Plane', 'Boat', 'Other', 'Pick-up' et 'Van'.

3.3.4 Protocole d'évaluation

L'ensemble des images a été découpé en 10 'folds' (2.4.5) afin de pouvoir effectuer une validation croisée. Les 'folds' ont été constitués de manière à ce qu'ils contiennent chacun le même nombre de véhicules de chaque classe, à l'exception de la classe 'Planes', pour laquelle trop peu d'occurrences étaient disponibles pour les diviser de manière équitable. Des statistiques représentatives de la base sont présentées Tab. 3.2.

Les indicateurs choisis pour étudier les performances sont l'AP moyenne (mAP, calculée sur 11 points), ainsi que différents points opérationnels, qui sont le taux de Rappel pour 0.01 FPPI, 0.1 FPPI et 1 FPPI. Ces indicateurs sont tels que décrits dans le Chap. 2.

Nous avons défini les Vrais Positifs de la manière suivante. En principe, il est nécessaire de trouver le meilleur appariement entre les prédictions et la vérité terrain, ainsi qu'expliqué 2.4.2. Cependant, dans notre cas, tout comme pour OIRDS, les véhicules ne se recouvrent pas et sont à une échelle similaire. Ainsi, chaque détection est assignée à la vérité terrain la plus proche.

En notant $p = (x, y)$ les coordonnées (en pixels) de la détection et $P = (X, Y)$ les coordonnées du véhicule le plus proche dans la vérité terrain, la détection est considérée comme correcte (c'est-à-dire comme un VP) si elle se situe dans une ellipse centrée sur la vérité terrain, suivant ce critère :

$$(p-P)^t \begin{pmatrix} \cos a & -\sin a \\ \sin a & \cos a \end{pmatrix}^t \begin{pmatrix} \frac{1}{W^2} & 0 \\ 0 & \frac{1}{L^2} \end{pmatrix} \begin{pmatrix} \cos a & -\sin a \\ \sin a & \cos a \end{pmatrix} (p-P) \leq 1 \qquad (3.1)$$

où W et H sont la moitié de la hauteur et la moitié de la largeur du véhicule en pixels, et a est l'orientation (véridique) du véhicule. Par construction, l'ellipse touche les bords du véhicule (en effet, les véhicules traités sont tous approximativement convexes). En

conséquence, il ne peut pas y avoir de recouvrement entre deux ellipses de deux véhicules différents et en conséquence il n'y a aucun besoin d'envisager une procédure d'assignation plus complexe (voir Fig. 3.7). Si plusieurs détections sont associées au même véhicule, celle ayant le meilleur score sera comptée comme VP tandis que les autres seront écartées de l'ensemble des détections.

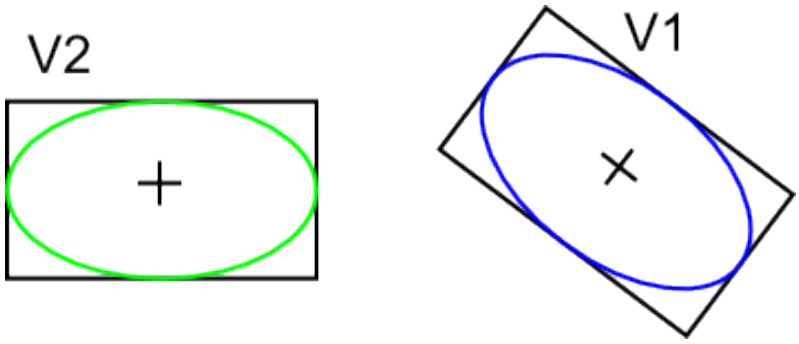


FIGURE 3.7 – Deux véhicules et leurs ellipses de détection. Il ne peut y avoir de détection ambigüe, du fait que deux véhicules ne peuvent se recouvrir.

Nom de la classe	abréviation	nombre par fold	total	orientation
Boat	Boa	17	170	$[-\pi\ \pi]$
Camping Car	Cam	39	390	$[0\ \pi]$
Car	Car	134	1340	$[-\pi\ \pi]$
Other	Oth	20	200	$[0\ \pi]$
Pick-Up	Pic	95	950	$[-\pi\ \pi]$
Plane	Pla	–	47	$[-\pi\ \pi]$
Tractor	Tra	19	190	$[-\pi\ \pi]$
Truck	Tru	30	300	$[-\pi\ \pi]$
Van	Van	10	100	$[-\pi\ \pi]$
small land vehicle	slv	295	2950	$[-\pi\ \pi]$
large land vehicle	llv	69	690	$[0\ \pi]$

TABLE 3.2 – VeDAI : Statistiques sur les données

3.3.5 Annotations

Une fois les images rassemblées, il a fallu constituer des annotations. Nous conseillons de suivre la même procédure à toute personne souhaitant constituer une base de données en détection, cette procédure permettant d'avoir toutes les informations nécessaires pour l'évaluation d'un algorithme de détection de manière simple à utiliser .

Chaque véhicule de la base de données est annoté de la manière suivante : les coordonnées de son centre (en pixels), son orientation, les coordonnées de ses quatre coins (en pixels également), le label de la classe. Enfin, deux booléens viennent compléter les annotations, en ajoutant si le véhicule est partiellement masqué et s'il est complètement dans l'image (troncature). L'orientation correspond à l'angle que fait le véhicule par rapport à l'horizontale de l'image, et ce, modulo π pour les véhicules où il est difficile de

déterminer l'avant ou 2π pour les autres. En conséquence, les orientations des 'Camping cars' et de la classe 'Other' ne sont données que dans l'intervalle $[0 - \pi]$ – la différence entre l'avant et l'arrière du véhicule n'est pas évidente pour le premier cas et n'a pas de sens dans le deuxième. Un exemple de fichier d'annotation est donné Tab. (3.3.5).

Les annotations sont mises à disposition à la fois dans des fichiers indépendants, un par image et dans un seul fichier qui regroupera toutes les annotations. Les annotations ont été données à la fois pour les images 512×512 et les images 1024×1024.

00000000	580.69	1009.22	3.012318	554	[...]	1021	002	1	0
00000001	344.82	812.36	-0.013888	326	[...]	819	001	1	0
00000001	413.21	811.24	-0.011363	392	[...]	819	009	1	0

TABLE 3.3 – VeDAI : Extrait d'un fichier d'annotation. Ce fichier contient, pour chaque véhicule, de gauche à droite : l'ID de l'image, les coordonnées du centre, l'orientation du véhicule, les coordonnées des quatre coins, l'ID de la classe, le booléen de troncature, le booléen d'occlusion.

3.3.6 Reconnaissance sur VeDAI

Afin de traiter la problématique de Reconnaissance, nous avons également créé une base de données de classification à partir de la base VeDAI-PII. Des fenêtres de taille 64x64 pixels autour des 8 classes de véhicules [1] de la base VeDAI-PII ont été extraites et transformées en base de classification. Les fenêtres sont à la fois disponibles en niveau de gris, et transformées en descripteur HOG. Pour la mesure de la performance, nous donnerons les taux de reconnaissance par classe, le microaveraging et le macroaveraging (voir 2.4.4 pour plus de détails).

3.4 Base Sagem

Si OIRDS et VeDAI permettent de tester les algorithmes de détection dans des conditions proches de la problématique des applications Sagem, il était important pour Sagem de tester ces derniers sur des données opérationnelles. Ces données étant confidentielles industrie, il n'est pas possible de présenter plus en détail les bases de données.

Ces bases de données ont été constituées soit à partir d'images de synthèse, générées à partir de modèles 3D, de scène et de véhicules, ainsi que d'une modélisation de capteur propre aux applications Sagem, soit à partir d'images prises en vol par des capteurs Sagem. Voir Annexe Confidentielle C pour plus de détails.

3.5 Conclusions

Il est connu que les bases de données présentées ont un biais intrinsèque, comme l'ont montré les travaux de [162] et de [132]. En effet, chaque base a ses propres limitations, que ce soit au niveau des images en elles-même (pas d'occlusion, pas de problème météorologique...), au niveau des annotations (quelles informations ont été conservées, ont été annotées – un polygone, la boîte englobante, le centre) ou au niveau du protocole

1. nous rappelons que la classe 'aeroplane' a été annotée, mais n'est pas utilisée lors des évaluations du fait de son faible nombre de représentants)

(comment sont définis les Vrais Positifs, les Faux Positifs, quelle est la métrique utilisée – mAP, ROC...).

En conséquence, une base de données doit être représentative du contexte applicatif à laquelle elle est destinée. Plus la base sera représentative, meilleures seront les évaluations des performances.

Dans ce chapitre, nous avons vu un état de l'art des différentes bases de données existantes touchant aux problématiques de Détection et de Reconnaissance. Nous avons vu que la plupart d'entre elles se concentrent sur des images issus du web, ne correspondant pas à la Détection et Reconnaissance de véhicules faiblement résolus. Nous avons ensuite présenté la collection OIRDS et les changements ajoutés afin d'en faire une base de données. Enfin, nous avons présenté la base VeDAI, premier benchmark pour la Détection de véhicules faiblement résolus.

Database	Classes	Ensembles	# Images	Eval.
Inria Pedestrian	1	train/test	2,000	DET+ OP
CalTech Pedestrian	1	train/test	250,000	MR/FPPI+ OP
CMU	1	no protocol	130	no protocol
ETHZ	6	train/test	500	rap/FPPI
PASCAL	20	train/val/test	$\geq$10,000	AP+prec/rec
LabelMe	$\geq$400	no cut	$\geq$40,000	no protocol
OIRDS	4	no cut	900	no protocol
VeDAI	9	train/test	1200	AP+OP

TABLE 3.4 – Résumé de bases de données existantes pour la Détection. Les abréviations correspondantes peuvent être trouvées dans la table des notations et abréviations

Chapitre 4

Étude algorithmique en Détection et Reconnaissance

L'objectif visé par la thèse est de produire des algorithmes capables de localiser dans des images des véhicules appartenant à différentes catégories (tâche de détection), ainsi que de prédire à quelle classe de véhicules appartient un objet dont la position est connue (tâche de reconnaissance).

De tels algorithmes reposent sur des chaînes de traitement complexes, comportant de nombreuses étapes pour lesquelles des choix (type de modèles, etc.) doivent être faits. Dans un premier temps, nous avons réalisé plusieurs chaînes complètes reposant sur les choix qui nous ont paru les meilleurs après analyse de l'état de l'art. Dans un second temps, nous avons caractérisé et évalué ces chaînes afin de déterminer sur quels éléments il était le plus opportun de concentrer nos efforts.

Dans ce chapitre, nous décrirons les chaînes algorithmiques choisies dans la section 4.1, puis nous donnerons quelques intuitions quant à la sensibilité de différents paramètres intervenant dans chacune des étapes dans la section 4.2. Enfin nous présenterons les résultats généraux obtenus dans les sections 4.3 et 4.4.

4.1 Chaînes algorithmiques

Ainsi que présenté Chap. 2, toutes les chaînes algorithmiques se décomposent en deux étapes : l'apprentissage et le test. Lors de la phase d'apprentissage, un ou plusieurs modèles sont construits par apprentissage statistique. Puis, dans la phase de test, ces modèles sont utilisés pour effectuer les tâches de détection et/ou reconnaissance. Les différentes chaînes proposées sont illustrées dans la Fig. 4.2. Afin de simplifier la lecture des différentes chaînes algorithmiques, un système de symboles a été mis en place pour chaque étape. La correspondance entre ces symboles et leurs significations respectives est décrite en détail dans l'Annexe A. Dans cette section, nous présentons la modélisation, le parcours de l'image, les descripteurs et les classifieurs utilisés, que ce soit pour la Détection ou la Reconnaissance.

4.1.1 Modélisation utilisée

Ainsi que décrit dans le Chap. 2, nous avons vu que différents types de modèles existent. Une classe d'objets sera pour nous décrite par des primitives 2D, extraites de

différentes images. Le but sera soit de localiser de telles primitives dans une image, soit de les différencier entre différentes classes d'objets.

Pour la modélisation utilisée dans la détection, nous avons utilisé un modèle séparé en différents groupes de primitives, chaque groupe s'appelant une *racine*. Une racine regroupe des apparences similaires. Par exemple, la classe voiture peut être représentée dans un cas classique par 3 racines, une regroupant les primitives des vues de côtés, une regroupant les primitives des vues de face et enfin, une regroupant celles des vues du dessus. Une racine peut être ensuite modélisée par un modèle génératif ou classée par un modèle discriminant. Lors de la phase de test, chaque primitive est testée en rapport avec le modèle appris pour la racine correspondante. Les scores des différentes racines sont fusionnées par un algorithme de NMS.

De plus, chaque classe d'objet est évaluée séparément, ainsi chaque classe est détectée par une chaîne algorithmique. Dans les travaux présentés les résultats des différents classifieurs ne sont pas fusionnés. En ce qui concerne la problématique de Reconnaissance, nous sommes restés sur une modélisation par une seule racine, afin de disposer de la même primitive pour tous les véhicules.

4.1.2 Parcours de l'image

Tous les algorithmes présentés se basent sur un parcours par *fenêtres glissantes*, ainsi que présenté à la section 2.2.1 et ce pour deux raisons. Tout d'abord, cette méthode donne le plus souvent de meilleurs résultats que les parcours par hypothèse validation (voir 2.2.2) dans l'état de l'art. Ensuite nous rappelons que dans le cadre de notre problématique, les objets sont faiblement résolus. En conséquence, ils sont difficilement séparables en différentes parties qui permettraient d'en déduire une position. Que le classifieur soit génératif ou discriminant, il est appris sur les fenêtres extraites pendant la phase d'apprentissage et est utilisé sur les fenêtres extraites lors de la phase de test.

Taille et formes des fenêtres, configuration des racines

La fenêtre traitée par un classifieur doit avoir une taille déterminée. Dans le cas simple où l'objet à détecter est modélisé par une seule racine, la taille de cette dernière est la taille moyenne des fenêtres représentant des positifs. Elles sont déduites des annotations et correspondent au plus petit rectangle englobant en entier un positif.

Dans le cas où plusieurs racines constituent le modèle, nous avons retenu une technique usuelle, qui consiste à prendre l'histogramme des ratios (hauteur/largeur) des fenêtres positives et à en sélectionner les pics ou, dans un cas uniforme, à échantillonner cet histogramme de manière régulière. Ces ratios sont ensuite mis en rapport avec l'aire (en pixel) des fenêtres, afin d'en déduire la hauteur et la largeur de différentes fenêtres typiques. Ensuite, chaque racine est dédoublée, afin d'avoir pour deux objets d'orientation contraire deux racines différentes. La Fig. 4.1 illustre un modèle à 8 racines, découlant de quatre ratios hauteur/largeur, chacun découpé en deux racines selon l'orientation.

Pour les expériences effectuées sur OIRDS, nous avons utilisé un modèle à une seule racine, du fait des différents problèmes d'annotations. Pour VeDAI, nous avons utilisé un modèle à 12 racines, c'est-à-dire 6 ratios hauteur/largeur découplés en deux orientations.

FIGURE 4.1 – Illustration d'un modèle à 8 racines. Les différentes apparences possibles de voiture sont séparées en 4 ensemble de ratios hauteur/largeur fixe, puis chacun de ces ensemble est séparé en deux orientations distinctes.

Extraction des fenêtres

Afin d'extraire les exemples des images d'apprentissage, chacune d'entre elles est transformée en pyramide d'images, tel que décrit dans le Chap. 2. Pour chaque niveau d'une pyramide, toutes les fenêtres de hauteur et largeur précédemment calculées sont extraites. Dans la pratique, les niveaux de la pyramide sont espacés d'un ratio 1,07. Les fenêtres ne sont généralement pas toutes extraites, en prendre une tous les huit pixels est ce qui est habituellement effectué. Dans le cadre des études sur l'imagerie aérienne, lors de la phase d'apprentissage, nous avons décliné chaque image en 8 exemplaires différents, créés à partir d'opérations de miroir et de transposition, le repère de l'image étant en effet arbitraire. De plus, seuls les 4 premiers étages de la pyramide sont utilisés, étant donné que l'altitude à laquelle la photo a été prise est plus ou moins connue. Enfin, pour ne pas avoir de frontière trop floue entre les positifs et les négatifs, les fenêtres négatives trop proches d'un positif ne sont pas extraites.

4.1.3 Descripteurs

Le choix de bons descripteurs est crucial quant à la réussite d'un algorithme de classification. Au vu du nombre de primitives existantes (voir section 2.3) et au vu du cadre d'application spécifique, un choix restreint de descripteurs a été effectué.

Les méthodes à bases de sacs de mots ou de descripteurs de régions, semblaient peu adaptées à la détection d'objets faiblement résolus. En effet, il semble difficile d'extraire des points d'intérêts ou de décrire des régions quand la taille de l'objet est de l'ordre de grandeur de celle d'une région ou d'un point d'intérêt. À l'inverse, les descripteurs à base de gradients, qui capturent les caractéristiques géométriques des objets, nous ont parus les mieux à même de tirer l'information pertinente des images à traiter. Ainsi, le choix s'est porté sur les Histogrammes de Gradients Orientés (HOG [26]) et les Local Binary/Ternary Pattern (LBP [171] et LTP [159]). Afin de voir l'apport de tels descripteurs, l'utilisation des niveaux de gris normalisés (NDG) et des gradients normalisés (GRA) a également été effectuée. La normalisation consiste en un centrage (moyenne nulle), suivi d'un redimensionnement pour étaler les valeurs entre -1 et 1.

4.1.4 Classifieurs

Afin d'avoir une base de comparaison solide, nous avons implémenté et/ou testé différents classifieurs. En ce qui concerne la détection, nous avons ainsi utilisé le Séparateur

à Vaste Marge (SVM [20]), une approche bayésienne à base de mixtures de gaussienne (GMM apprises par Expectation Maximisation [29]), une méthode de Template Matching (TM [117]) ainsi que le code du Deformable Part Model (DPM [40]). Pour la Reconnaissance, nous avons utilisé le SVM et un perceptron multi-couches (MLP [141]).

Template Matching (TM)

La méthode de Template Matching [8] utilisée est basée sur les étapes suivantes. Les descripteurs positifs sont regroupés dans des clusters, par un algorithme de partitionnement de données. En pratique, nous avons testé l'algorithme des K-moyennes [2] et des K-Médoïdes [82], ce dernier étant retenu pour l'ensemble des expériences. L'algorithme des K-Médoïdes retient comme élément représentatif du cluster la donnée la plus au centre de ce cluster. Cette représentation s'appelle un médoïde. Le nombre de clusters est choisi par validation croisée. Lors du parcours par fenêtre glissante, une corrélation entre le descripteur de la fenêtre et les médoïdes est effectuée. Ainsi, pour une racine, plusieurs scores sont disponibles et le maximum est conservé en tant que score final.

Approche bayésienne

Pour la comparaison avec des modèles génératifs, une méthode de modélisation par mixture de gaussiennes (GMM) a été implémentée. Les différents descripteurs positifs sont modélisés par une mixture de gaussiennes, apprise par l'algorithme EM (Expectation Maximisation [29], de même pour les négatifs. Ainsi, lors de la phase de test, les deux modèles donnent les probabilités qu'un descripteur appartienne à la mixture des positifs ou celle des négatifs. Le nombre de composants des différentes mixtures est choisi par une validation croisée préalable.

Séparateurs à Vaste Marge (SVM)

En ce qui concerne le SVM, deux librairies ont été utilisées, SVMlight [81] et libSVM [12]. Ces librairies donnent des résultats similaires pour des réglages similaires, en conséquence nous ne ferons pas la distinction de la librairie utilisée dans ce manuscrit. Le SVM apprend sur les descripteurs précédemment cités ou sur des combinaisons concaténés de ceux-ci.

Pour la détection, nous avons utilisé un noyau linéaire, en raison de sa rapidité d'exécution, tant pour l'apprentissage que pour le test. Afin de ne pas avoir à entraîner le SVM sur l'intégralité des exemples d'apprentissage en une seule fois (principalement pour des problèmes de place mémoire), la technique classique de réentraînement sur des *négatifs difficiles* [40] a été appliquée. Ainsi, seule une partie des négatifs est conservée en mémoire. Sur un SVM, typiquement tous ceux dont le score dépasse -1.0 sont conservés. Ces négatifs sont ceux qui sont près de la frontière modélisée. Par défaut, nous n'avons pas normalisé les scores, voir la section 4.2.3 pour plus de détails.

Pour la Reconnaissance, des tests ont été effectués avec le noyau linéaire, mais aussi avec un noyau Gaussien. Plusieurs approches sont possibles avec le SVM pour des problèmes classification. Nous avons utilisé la stratégie un contre tous, en apprenant ainsi 8 SVM (nous rappelons que nous traitons 8 classes différentes), chacun opposant une classe à toutes les autres. Nous avons également utilisé le SVM multi-classes de [86] qui effectue une optimisation de la classification sur toutes les classes. Dans le cas de classification multi-classes, il est important de normaliser les sorties des classifieurs afin de pouvoir

les comparer correctement. De même que pour la détection, nous avons testé l'apport de l'ajout d'un modèle probabiliste. La normalisation effectuée est issue de [102], qui est une version améliorée de la version de Platt [131]. D'autres hypothèses et normalisations sont toutefois possibles. Il serait par exemple possible de rééquilibrer les classes par un jeu de coefficients qui maximise le micro ou le macro averaging. Toutefois, nous sommes restés dans l'hypothèse selon laquelle la base de données d'apprentissage a le même biais (déséquilibre entre les classes) que la base de test.

Deformable Part Model

Le code du DPM ([40]) est disponible en ligne. Nous avons utilisé la version 5 [56]. Le code effectue un parcours à fenêtre glissante, identique au notre et utilise un HOG en guise de primitive. La différence provient du classifieur. Le classifieur utilise des racines, mais également des *parties déformables*. Chaque racine est reliée à un certain nombre de parties et leurs relations, ainsi que la forme des parties, sont apprises lors de l'optimisation d'un SVM à variables latentes. L'optimisation recherche les meilleures formes et relations, afin de maximiser la performance.

S'il nous a été possible d'interfacer le code aux bases de données, ce dernier a été implémenté à l'origine pour couvrir les bases Pascal VOC et INRIA (voir Chap. 3 pour plus de détails). En conséquence, il n'a pas été toujours possible d'utiliser toutes les options de ce code correctement et certains résultats sont partiels ou n'ont pas répondu à nos attentes.

Perceptron Multi-couches (MLP)

Le perceptron multi-couches ([141]) est un réseau de neurones classique en classification. Il est composé de deux couches de neurones entièrement connectées, chacune utilisant comme fonction d'activation une fonction non linéaire. Le réseau est appris en associant chaque descripteur à un label, par descente de gradient. Pour cette expérience, nous avons utilisé la toolbox Neural Network de Matlab. Le nombre de neurones de la couche centrale est choisi par validation croisée.

4.1.5 Post-traitements

En post traitement, une étape de NMS a été mise en place. Elle consiste à éliminer les détections qui se recouvrent avec un indice de Jaccard supérieur à 0,5. Ainsi, chaque chaîne algorithmique complète retourne une liste de positions avec un score. Par défaut, les positions non retournées sont considérées comme étant du fond.

4.1.6 Analyse de la complexité

Si l'analyse de la complexité d'un algorithme est cruciale dans les tâches de détection et reconnaissance automatique, en pratique, nous ne sommes intéressés que par la complexité de la phase de test, étant donné que l'apprentissage peut être effectué offline et une fois pour toutes. Tous les algorithmes testés ont une complexité linéaire vis-à-vis du nombre de fenêtres testées lors du processus de parcours de fenêtres. Tous les algorithmes ont également une complexité linéaire au regard de la taille du descripteur mis en entrée. Enfin, tous les algorithmes à base de variétés, qui seront présentés dans les chapitres suivants, ont une complexité linéaire vis-à-vis de l'espace latent. En ce qui concerne le temps de calcul, tous les algorithmes présentés peuvent être implémentés à

base de multiplications et de sommes, ainsi qu'à l'aide de tables références pour ce qui est des fonctions. Ils peuvent ainsi être utilisés pour des applications temps réel.

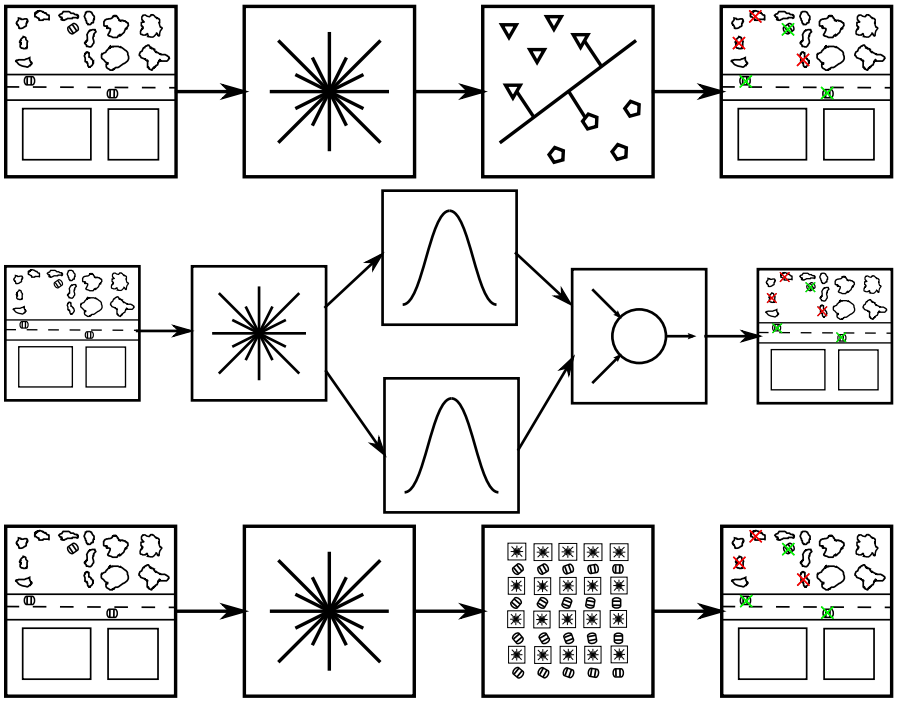


FIGURE 4.2 – Trois chaînes algorithmiques classiques. En haut, descripteur + SVM. Au centre, descripteur + Mixtures de Gaussiennes. En bas, descripteur + Template Matching

4.2 Étude paramétrique en Détection

Dans cette section, nous présentons quelques résultats relatifs à la sensibilité de différents paramètres des chaînes algorithmiques présentées. En effet, que ce soit lors du parcours, de l'utilisation des données disponibles, du choix de structure du modèle, des post-traitements ou encore de la mesure des performances, un certain nombre d'opérations doivent être réglées avec soin, tandis que d'autres peuvent être plus lâches. Les performances sont données sur différentes bases de données, en majorité la base VeDAI, mais également sur la base Pascal VOC 2007. La configuration par défaut est une expérience sur la classe 'slv', sur la base VeDAI-PII, utilisant un SVM à 12 racines sur des descripteurs HOG. Dans les autres cas, nous précisons pour chaque expérience la base et la chaîne algorithmique utilisées. Dans un premier temps, nous présentons l'apport de l'ajout d'exemples virtuels. Ensuite, le problème de la structure du modèle est abordé, avec les racines et les parties, puis l'influence du réglage de l'étape de NMS est analysée. Enfin, le parcours de l'image et la tolérance lors de l'évaluation sont étudiés.

4.2.1 Exemples virtuels

Il est important, afin d'obtenir le meilleur détecteur possible par rapport à l'ensemble des données disponibles, de multiplier au maximum ses positifs, en utilisant des transformations simples [13]. En effet, le simple fait d'ajouter ces exemples permet de gagner

significativement en performance, les meilleurs transformations à effectuer sont celles qui ne se basent pas sur des interpolations/déformation de l'information contenu dans les pixels, à savoir les techniques de découpage, les miroirs et les inversions, à opposer aux rotations, dégradations par ajout de bruit artificiels, warping [127].

Une manière simple de faire est de les multiplier par des miroirs et des flips, ce qui permet de gagner en performance ainsi que montré en Tab. 4.1. L'étirement n'a pas été retenu dans le cas des algorithmes de Détection, en raison de l'homogénéité globale des objets d'une même classe. En effet, les différences de longueur entre deux véhicules de la même classe sont négligeables dans notre problématique.

Classe	aer	bik	bird	boa	bot	bus	car	cat	chair	cow	–
Référence	9.1	17.1	0.1	1.4	12.9	12.3	24.4	0.3	9.6	3.7	–
Étirement	11.5	22	0.3	1	12.4	14.7	24.5	0.3	10.0	11.3	–
Miroir	8.2	19.9	0.7	0.7	13.6	14.2	26.8	0.3	9.9	7.2	–
Étirement+Miroir	11.5	22.7	0.5	3.2	12.3	14.9	23.7	0.4	6.1	11.5	–
Classe	din	dog	hor	mot	per	pot	she	sof	tra	tv	gain
Référence	1.1	0.4	10.1	13.3	3.7	3.3	4.8	5.0	7.5	21	–
Étirement	1.9	1.1	13.2	17	12.4	5	7.2	1.7	11.2	18.1	1.79
Miroir	10.4	2.2	14.8	14.6	12.6	9.5	11.8	6.3	13.1	23	2.94
Étirement+Miroir	1.8	3.5	13.9	17.7	12.2	9.4	12.6	5.2	13.1	18.6	2.69

TABLE 4.1 – AP pour différentes classes. Expérience effectuée sur PASCAL VOC 2007, avec un SVM mono-racine appliqué à un descripteur HOG, sur l'influence de l'étirement et la réflexion des exemples.

Il est possible d'augmenter encore la performance en entrant dans l'entrainement toutes les fenêtres qui peuvent être considérées comme positives lors du test. Ainsi, chaque fenêtre dont le centre est dans l'ellipse inscrite d'une vérité terrain (comme défini dans la section 3.3.4) peut être ajoutée à l'ensemble des positifs au lieu de retenir seulement la fenêtre la mieux placée. Il est possible également de restreindre ces petites translations en limitant l'ellipse inscrite, ainsi qu'illustré Fig. 4.3.

Ainsi que montré table 4.2, une trop grande latitude sur les données positives ne permet pas de gagner en performances. Cela provient du fait que les exemples deviennent trop peu spécifiques et contiennent alors trop de fond. La fenêtre peut même tronquer l'image du véhicule à détecter. Cependant, en restreignant les translations et en ajoutant des exemples qui recouvrent tout de même suffisamment les exemples initiaux, il est possible de conserver une certaine cohérence et ainsi améliorer les résultats.

	mAP	0.01 FPPI	0.1 FPPI	1 FPPI	10 FPPI
Référence	58.9 ± 3.5	13.2 ± 5.1	30.5 ± 3.7	72.6 ± 4.3	90.3 ± 2.6
$1.0r$	58.0 ± 5.3	12.8 ± 5.1	32.6 ± 5.8	70.5 ± 4.8	90.3 ± 2.9
$0.25r$	62.8 ± 4.0	16.5 ± 8.6	39.4 ± 5.8	72.7 ± 3.5	90.2 ± 3.0

TABLE 4.2 – Impact de l'ajout d'exemples virtuels en fonction de l'éloignement au centre réel de l'objet. r correspond au rayon de l'ellipse inscrite. Expérience réalisée sur la classe 'car', avec la chaîne HOG+SVM, sur la base VeDAI-PII.

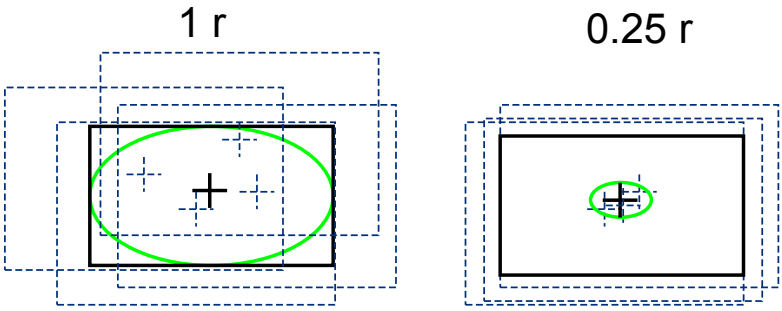


FIGURE 4.3 – Un véhicule et les fenêtres déclarées positives, en utilisant toutes les fenêtres dans un rayon $1r$ (à gauche) ou rayon $0.25r$ (à droite), r étant le rayon de l'ellipse dans un axe donné.

4.2.2 Étude sur les racines et les parties

Le nombre de racines d'un modèle à partie est un paramètre critique afin d'obtenir des bonnes performances, il s'agit même en pratique du paramètre le plus sensible. En effet, l'utilisation de racines différentes permet de découper le problème en sous problèmes mieux définis. De plus, le découpage est en conséquence plus précis autour des véhicules, les fenêtres contiennent mécaniquement moins de pixels n'appartenant pas au véhicule. Une expérience simple de variation du nombre de racines montre qu'un minimum de 4 racines est nécessaire pour obtenir de bonnes performances (voir Fig. 4.4). Les fenêtres d'un même véhicule ont des ratio hauteur-largeur très différents, du fait de toutes les possibilités d'orientation du véhicule par rapport au capteur.

Le test du DPM sur VeDAI nous a montré que l'utilisation des parties déformables permet de gagner légèrement en performance, cependant, le code utilisé pouvait parfois être instable en raison de la petite taille des objets (des erreurs d'exécution pouvaient arrêter le programme avant la fin de l'apprentissage). Un gain de 3% de mAP a été observé lorsque le code fonctionnait (3 folds sur les 10 disponibles, les résultats n'ont donc pas été inclus dans les tableaux). Ces résultats sont en concordance avec les travaux de Divvala [30], qui montrent que l'utilisation de nombreuses racines améliore bien plus significativement les performances que l'ajout de parties, au point que l'apprentissage de parties est inutile dès lors que les racines sont suffisamment nombreuses.

4.2.3 Paramétrage du NMS

Nous avons également testé l'influence du seuil sur l'indice de Jaccard utilisé pour l'algorithme de NMS. La Fig. 4.5 montre que choisir un très petit indice de Jaccard diminue les performances, mais de peu (3% de mAP par exemple). En revanche, choisir un indice de Jaccard trop élevé ne permet pas d'enlever les détections multiples sur des faux positifs, la performance chute alors de l'ordre de 10%. Les résultats présentés sont donnés avec la configuration par défaut, mais des résultats similaires ont été observés sur les autres classes et avec les autres détecteurs.

Les classifieurs ne sont pas normalisés lors de cette étape. En effet, nous avons pu observer qu'une étape de normalisation ne change pas les performances, comme montré en Tab. 4.3.

Il est à noter que dans le cas d'une cascade, il est essentiel de placer l'algorithme

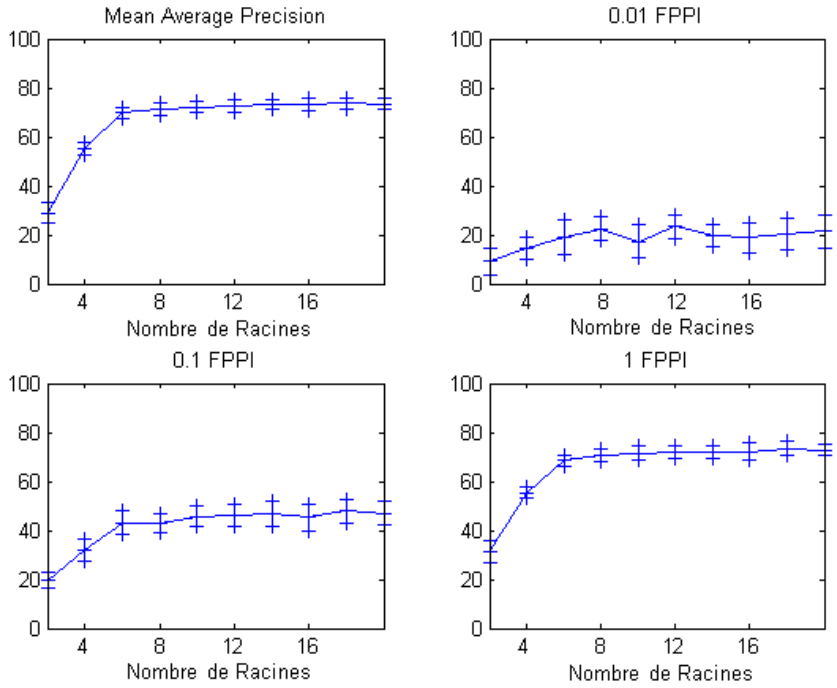


FIGURE 4.4 – Performances en fonction du nombre de racines utilisées dans l'apprentissage. Expérience réalisée avec la configuration par défaut (excepté pour les racines). Les barres d'erreur représentent l'écart type.

Méthode	mAP	Rappel à 0.01 FPPI	Rappel à 0.1 FPPI	Rappel à 1 FPPI
Non normalisée	71.5±2.5	17.5±6.5	41.9±5.7	71.4±3.0
Normalisée	71.4±1.6	18.4±7.2	41.6±5.9	71.3±2.7

TABLE 4.3 – Étude de la normalisation des scores des différents classifieurs avant une étape de NMS. Expérience réalisée avec la configuration par défaut.

de NMS en toute fin, pour être sûr de conserver toutes les fenêtres potentielles, et en conséquence celle qui est la mieux alignée avec l'objet à détecter.

4.2.4 Paramétrage du parcours de l'image lors de la phase de test

Le pas de parcours dans un algorithme à fenêtre glissante peut impacter les performances. La Fig. 4.6 montre que si le pas est trop grand, (plus grand que deux fois le pas utilisé lors de l'apprentissage), la performance chute significativement. Inversement, prendre un pas de 2 ou 4 pixels permet de gagner légèrement en performance, pour un coût calculatoire important. Dans les expériences nous avons conservé un pas de 8 pixels, afin d'avoir un temps d'exécution raisonnable.

Le nombre d'échelles utilisées affecte également les performances, mais son impact est faible tant que l'échelle réelle est approximativement connue, ainsi que montré Fig. 4.7. L'impact peut être plus important si dans une même classe sont réunis des véhicules de taille différentes, mais dans notre cas il reste réduit, grâce à l'utilisation des racines de tailles différentes. Le nombre d'échelles utilisées affecte également les performances, mais

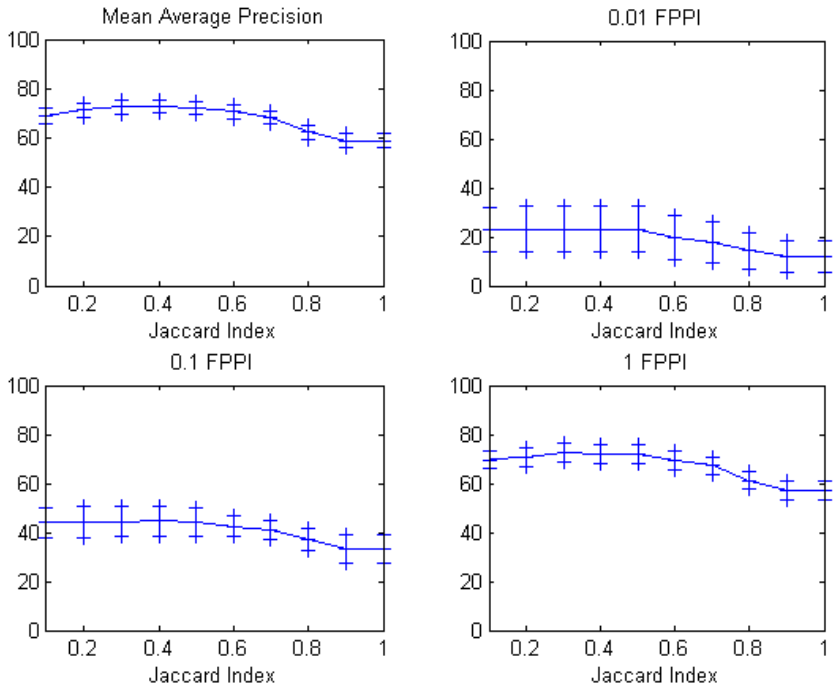


FIGURE 4.5 – Influence du seuil de l'étape de Suppression des Non-Maxima sur les performances. Les barres d'erreur représentent l'écart type. Cette expérience a été réalisée avec la configuration par défaut.

son impact est faible tant que l'échelle réelle est approximativement connue, ainsi que montré Fig. 4.7. L'impact peut être plus important si dans une même classe sont réunis des véhicules de taille différentes, mais dans notre cas il reste réduit, grâce à l'utilisation des racines de tailles différentes.

4.2.5 Influence de l'ellipse de détection

Les expériences suivantes ont été faites afin de voir l'influence de la taille de l'ellipse vis-à-vis de la mesure des performances. Nous rappelons ici qu'une détection est considérée comme bonne si elle se trouve dans une ellipse inscrite dans le rectangle formé par le véhicule, ainsi que présenté en 3.3.4. Il est possible de faire varier la tolérance de la détection en agrandissant ou diminuant arbitrairement les axes de cette ellipse. Un seuil de 1.0 correspond à l'ellipse inscrite, tandis qu'un seuil de 0 correspond uniquement au point central du véhicule. La Fig. 4.8 montre que la performance est relativement stable lorsque ce seuil varie, ce qui signifie que : (i) les détections sont plutôt précises, (ii), la valeur choisie pour ce seuil est adéquat pour la mesure de performance puisqu'elle n'est en rien critique.

4.2.6 Invariance à de petites translations

Afin de mesurer l'invariance des performances à de petites translations de la base de données, nous avons créé 7 nouveaux ensembles d'images, chacun étant une transposition, un miroir ou une combinaison des deux des images de départ. L'idée est de montrer que

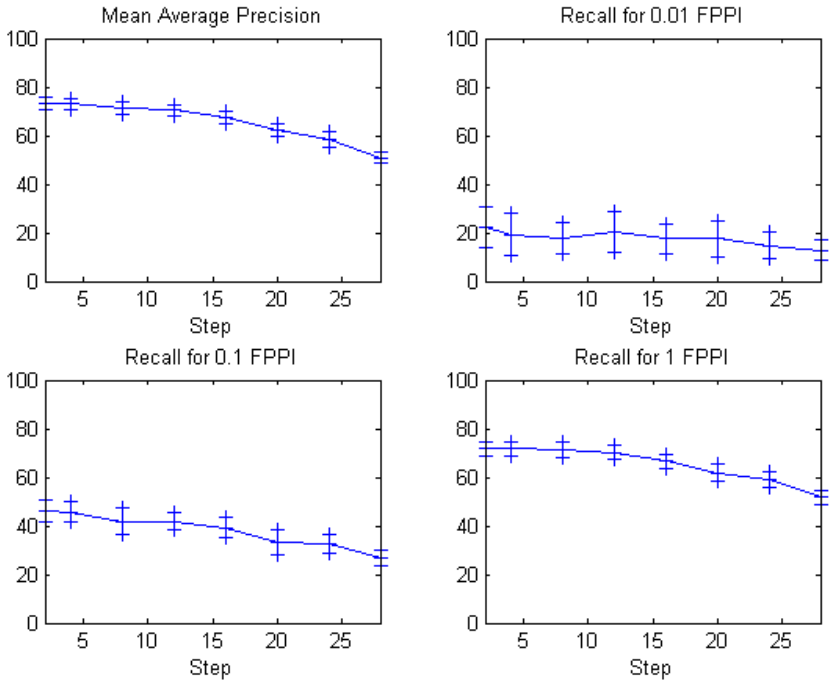


FIGURE 4.6 – Performance en fonction du pas de l'algorithme de parcours par fenêtre glissante. Expérience réalisée avec la configuration par défaut. Les barres d'erreur représentent l'écart type.

même en déplaçant l'ensemble des images de la base de données de quelques pixels, les résultats restent stables et qu'ainsi le choix d'une orientation arbitraire pour parcourir les images ne fausse pas l'évaluation. Nous avons testé SVM+HOG sur la classe 'slv' et mesuré l'écart type de performance sur ces différentes bases. Ainsi que montré table 4.4, l'écart type est plus petit que celui entre les folds, ce qui montre la relative invariance du détecteur à de petites translations.

slv	mAP	0.01 FPPI	0.1 FPPI	1 FPPI	10 FPPI
Mean	72.9	20.8	45.0	72.5	90.5
Standard Deviation	0.25	1.6	0.80	0.44	0.35

TABLE 4.4 – Moyenne et écart type sur les 8 types de transpositions et miroirs possibles. Il est à noter que l'écart type est très faible. L'expérience est réalisée avec la configuration par défaut.

4.3 Détection : résultats généraux

Nous présentons dans cette section les résultats de Détection et Reconnaissance obtenus avec les méthodes décrites dans la première section de ce chapitre. Les résultats sur OIRDS sont présentés en Tab. 4.5, ceux sur VeDAI en Tab. 4.6 et en Tab. 4.7.

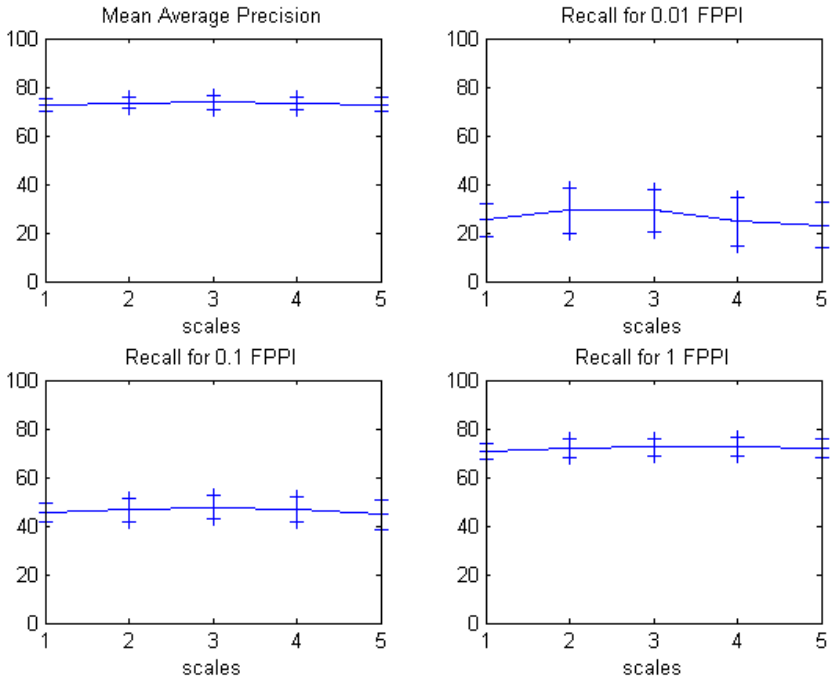


FIGURE 4.7 – Influence du nombre d'échelles sur les performances. Expérience réalisée avec la configuration par défaut. Les barres d'erreur représentent l'écart type.

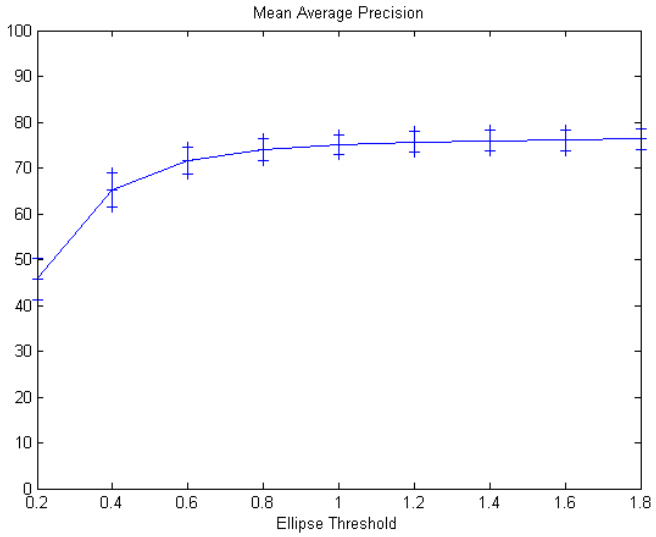


FIGURE 4.8 – mAP sur la classe 'slv' en fonction de la largeur de l'ellipse utilisée lors de la détermination des bonnes détections. Expérience réalisée avec la configuration par défaut. Un ratio de 1 correspond à l'ellipse inscrite, un seuil inférieur à 1 correspond à une contrainte plus forte. Les barres d'erreurs correspondent à l'écart-type.

4.3.1 Comparaison des classifieurs

De la comparaison des classifieurs, le SVM est apparu comme le plus simple à mettre en place. En effet, ce classifieur n'a pas nécessité de validation croisée poussée, l'utilisation

Base	Méthode	NDG	GRA	HOG
OIRDS-EPI	TM	1.79	0.97	0.77
	GMM-GMM	8.3	21,3	17.7
	SVM [26]	**10.5**	**35.2**	**46.8**
	DPM [40]	–	–	6.55
OIRDS-EGI	SVM [26]	1.5	12.1	12.6

TABLE 4.5 – Average Precision sur les différentes chaînes algorithmiques expérimentées. En haut, sur la base OIRDS-EPI, en bas, sur le set OIRDS-EGI. Le SVM et le DPM sont mono-racine.

des paramètres recommandés dans l'état de l'art a suffi à donner de bonnes performances. De plus son adaptation est facile à implémenter et à maîtriser. À l'inverse, les mixtures de gaussiennes et le Template Matching nécessitent tout deux des méta paramètres qui dépendent de la classe utilisée et du problème traité.

Quel que soit le descripteur, et quelle qu'ait été la base, les résultats obtenus avec le Template Matching sont très médiocres, la chaîne de détection ne fonctionne absolument pas. L'absence de modèle de fond ne permet pas d'éviter de trop nombreux faux positifs, mettant en défaut une approche basée seulement sur un modèle génératif. Les images de VeDAI ou de OIRDS présentent de nombreuses textures qui corrèlent fortement avec les médoïdes utilisés de par leur aspect pseudo aléatoire. Le fait que les descripteurs soient de grandes dimensions et que la corrélation prenne en compte chacune de ces dimensions sans distinction augmente les problèmes dus au bruit sur les données.

En ce qui concerne l'utilisation de mixtures de gaussiennes, si la présence d'un modèle de fond permet d'obtenir des performances moins mauvaises que le Template Matching, les performances restent très inférieures à celles données par le SVM. la grande dimension des descripteurs rend l'apprentissage des gaussiennes difficile. L'algorithme EM a du mal à converger et a tendance à créer des gaussiennes centrées en un descripteur.

Sur OIRDS, le DPM n'a pas donné de bons résultats. La principale cause est la spécialisation du code sur des images de grandes dimensions. Ce dernier n'est pas fait pour traiter d'aussi petites images de positifs. De plus, les négatifs difficiles sont récoltés uniquement dans des images sans véhicule, il a donc fallu ajouter des images supplémentaires de fond. Malgré les modifications et l'ajout d'images de fond, il n'a pas été possible d'obtenir de meilleurs résultats. Pour ce qui est de VeDAI, la Tab. 4.6 montre que le DPM est légèrement meilleur qu'un HOG+SVM, cependant, l'ajout de meilleurs descripteurs permet de mieux gagner en performance. Cette expérience valide le fait que les améliorations proposées par le DPM ne permettent pas de gagner significativement en performances. En conséquence, il n'a pas été décidé d'implémenter un modèle à base de parties déformables.

4.3.2 Comparaison des descripteurs

Nous observons une grande différence entre les descripteurs basiques que sont NDG et GRA, comparé à un descripteur tel que HOG (Tab. 4.5). Malgré un très bon classifieur, comme le SVM, il n'est pas possible d'obtenir de bonnes performances avec des primitives aussi simples.

D'autres expériences, résumées en Tab. 4.6, indiquent que l'utilisation de HOG ou

mAP										
Détecteur	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
DPM	72.4±2.6	40.7±4.6	26.1±19	41.9±11.7	60.5±4.2	6.0±4.5	52.3±5.3	**33.8±16**	34.3±5.9	36.3±10
SVM +HOG	71.5±2.5	36.0±4.7	32.2±15	33.4±8.9	55.4±2.6	6.9±4.2	48.6±4.9	07.4±3.6	32.5±8.0	40.6±15
SVM +LBP	64.3±3.2	22.9±5.4	07.6±5.4	24.1±9.5	51.7±5.2	1.0±0.9	48.2±4.5	06.3±2.8	25.7±6.3	38.1±14
SVM + LTP	73.1±3.1	40.9±5.5	17.9±8.0	45.6±7.9	60.4±4.0	**6.5±3.1**	56.9±5.5	21.2±8.6	35.7±9.5	**51.6±17.1**
SVM +HOGLBP	**75.0±2.2**	**42.2±5.4**	**34.1±16**	**47.5±11**	**61.3±3.9**	2.0±1.3	**57.5±4.5**	17.5±8.2	**37.4±8.4**	44.7±14
TM	2.2±1.7	1.0±1.2	0.03±0.03	3.2±2.4	5.3±1.5	0.1±0.02	0.6±1.0	0.2±0.2	0.4±0.6	1.7±3.5
rappel à 1 FPPI										
Détecteur	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
DPM	71.8±2.6	**62.9±5.5**	48.0±32	**72.9±8.6**	74.5±4.5	**22.5±12**	70.6±5.0	**61.4±19**	60.1±4.7	85.9±7.5
HOG + SVM	71.4±3.0	56.8±5.8	64.8±15	64.9±8.8	70.5±3.9	32.9±3.5	69.5±4.7	34.9±14	51.1±9.2	80.5±11
LBP + SVM	62.9±2.9	38.0±4.8	29.4±14	42.0±8.5	65.9±4.7	8.6±5.1	65.2±6.0	29.5±8.3	48.8±9.4	76.0±16
LTP + SVM	72.0±3.5	59.2±4.8	51.6±14	67.6±6.6	77.2±4.8	26.5±11	72.8±5.0	60.2±14	59.1±11	85.2±10
HOGLBP + SVM	**74.7±2.9**	59.7±5.8	**56.9±16**	72.4±7.6	**77.6±3.8**	16.2±8.9	**75.1±4.4**	53.1±14	**60.6±10**	**86.8±9.6**
TM	3.8±2.4	3.5±3.6	0.6±1.9	12.3±3.7	10.5±2	0.0±0.0	1.2±1.7	1.9±4.2	1.8±3.9	9.5±12
rappel à 0.1 FPPI										
Détecteur	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
DPM	46.1±4.4	28.5±4.3	23.4±19	37.4±11	**31.4±5.8**	**10.7±8.2**	30.9±6	**39.3±18**	33.7±3.7	**54.2±15**
HOG + SVM	41.9±5.7	24.0±4.1	39.0±14	30.1±11	24.4±5.7	9.3±5.9	27.7±5.4	13.3±3.5	33.5±9.8	48.9±17
LBP + SVM	37.7±5.8	17.9±6	8.9±7.1	24.6±9.5	26.3±7.9	1.9±3.1	29.3±5.5	10.8±5.7	26.9±6.7	47.1±18
LTP + SVM	43.7±6.5	29.0±5.4	22.4±7.2	42.6±7.4	29.1±6.2	8.3±6.0	37.1±5.1	25.4±12	35.3±9.4	64.3±16
HOGLBP + SVM	**49.1±5.3**	**30.1±6.9**	**40.2±16**	**45.1±12**	30.7±7.0	3.0±4.3	**36.0±5.9**	20.1±6.5	**36.4±7.7**	53.3±14
rappel à 0.01 FPPI										
Détecteur	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
DPM	**23.4±4.9**	6.8±3.0	15.3±13	10.4±12	**13.4±6.8**	1.9±3.2	12.5±4.2	**18.9±9.9**	18.5±7.8	17.0±16
HOG + SVM	17.5±6.5	9.5±4.3	19.7±15	7.1±5.7	7.8±5.5	**2.8±3.9**	6.9±4.4	1.9±3.1	17.4±7.0	29.0±16
LBP + SVM	16.2±5.6	6.1±13.5	3.6±5	11.8±6.6	5.5±2.2	0.0±0.0	10.9	1.2±2.4	12.5±8.0	29.5±12
LTP + SVM	16.5±6.9	13.9±4.9	8.9±7.3	20.0±7.6	9.3±3.7	2.6±3.6	11.8±7.3	7.8±7.3	17.6±10	**35.5±18**
HOGLBP + SVM	19.1±9.6	**13.5±4.2**	**22.8±11**	**20.6±13**	8.3±5.2	0.0±0.0	**15.1±8.4**	9.6±7.0	**20.2±9.2**	32.6±15

TABLE 4.6 – Performances de différents détecteurs sur l'ensemble PII de VeDAI. Sont présentés l'average precision et le rappel à 1, 0.1 et 0.01 FPPI.

du LTP est plus efficace que le LBP, tout en restant dans la même gamme de performance. Enfin, la fusion de deux descripteurs (HOGLBP) permet encore de gagner en performance, sans pour autant permettre un saut important.

4.3.3 Influence des bandes spectrales

Grâce à la base VeDAI, il a été possible de quantifier l'apport de différentes bandes spectrales du capteur. Les résultats obtenus sont présentés Tab. 4.7. Les descripteurs ont été implémentés en étant très peu (HOG) voire pas du tout (LBP/LTP) basés sur la couleur, nous ne notons pratiquement pas d'amélioration. En effet, le HOG calcule le gradient sur chaque canal de couleur, puis prend le maximum. Les LBP et LTP travaillent sur une image en niveaux de gris (attention, l'image en niveau de gris est cependant différente de l'image en proche infrarouge). Il est possible qu'en changeant les descripteurs, afin de profiter au maximum des différents canaux, les résultats s'améliorent d'avantage. Cependant, étant donné que le cadre d'application visé est l'imagerie infra-rouge, cette étude n'a pas été effectuée.

4.3.4 Influence de la résolution

En ce qui concerne de la résolution, la multiplication par deux de la résolution permet de gagner en performance sur toutes les classes et tous les indicateurs. Cependant, même si les performances augmentent, elles restent néanmoins dans le même ordre de grandeur. En effet, pour un taux de faux positif donné, il est possible de gagner entre 5 et 10 pour cent de taux de détection. Ce gain est significatif mais ne change pas la classe de performance. Pour des résolutions plus fortes, d'autres approches devraient être envisagées.

mAP										
Ensemble	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
SII	75.0±2.2	42.2±4.4	34.0±16	47.5±11	61.3±3.9	2.0±1.3	57.5±4.5	17.6±8.2	37.4±8.4	44.7 ±14.4
PIC	74.9±2.5	44.1±6.7	38.1±16	58.8±9.7	63.6±5.3	2.9±2.3	55.2±5.3	15.6±9.2	36.1±7.2	50.3±17.5
GII	77.0±1.6	45.7±3.9	36.9 ±16	52.9±9.3	62.6±4.5	4.4±1.8	54.4±5.2	10.9±6.2	34.3±6.8	48.1±19.9
GIC	76.8±1.5	45.6±4.2	40.4±16	55.4±8.1	63.8±4.1	5.6±3.7	51.5±4.8	13.9±9.5	38.1±9.6	49.4±17.9
rappel à 1 FPPI										
Ensemble	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
PII	74.7±2.9	59.7±5.8	56.9±16	67.6±6.6	77.6±3.8	16.2±8.9	75.1±4.4	53.1±14	60.6±10	86.8 ±9.6
PIC	74.6±2.8	62.8±6.4	64.4±15	80.2±7.1	79.8±4.3	15.9±7.9	73.8±4.5	48.2±15	63.8±8.8	84.6±15
GII	76.8±2.3	65.1±4.6	61.0±12	76.1±6.0	78.3±4.1	24.3±7.8	75.7±5.5	43.8±15	58.7±9.4	83.4±10
GIC	76.5±2.3	66.2±4.5	68.5±13	80.6±6.5	78.9±3.5	29.6±8.3	73.1±3.5	42.4±14	61.0±12	83.3±14
rappel à 0.1 FPPI										
Ensemble	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
PII	49.1±5.3	30.1±6.9	40.2±16	45.1±12	30.7±7.0	3.0±4.3	36.0±5.9	20.1±6.2	36.3±7.7	53.3±14
PIC	47.7±7.2	32.8±7.0	42.9±13	55.4±9.8	32.7±10	3.9±4.4	33.7±6.4	16.6±13	36.1±8.0	64.2±18
GII	51.6±5.9	33.8±5.3	39.9±17	47.4±12	31.5±6.6	7.1±4.2	31.0±5.3	14.5±8.2	35.9±5.8	56.3±20
GIC	51.9±3.3	32.6±5.6	42.7±16	48.7±7.7	35.5±7.9	5.5±5.2	29.3±6.2	19.9±9.9	37.5±10	64.1±20
rappel à 0.01 FPPI										
Ensemble	slv	llv	Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
PII	19.1±9.6	13.5±4.2	22.8±11	20.7±13	8.3±5.2	0.0±0.0	15.1±8.4	7.8±7.3	20.2±9.2	32.7 ±15
PIC	19.7±8.5	13.7±8.9	26.1±13	29.2±14	8.9±7.0	1.0±2.2	11.8±8.3	6.6±5.6	16.5±7.6	36.6±18.4
GII	19.3±8.1	15.0±7.7	30.2±16	20.6±13	11.1±7.0	1.1±2.3	7.7±4.5	2.8±4.7	17.0±5.7	39.7±21
GIC	24.1±8.6	12.6±5.7	28.5±15	24.6±9.5	10.0±6.4	2.1 ±3.6	9.2±4.1	6.1±7.5	20.0±9.1	34.3±16.6

TABLE 4.7 – Performance de la chaîne HOGLBP+SVM. La Tab. présente la moyenne sur les dix folds ainsi que l'écart type sur ces dix folds.

4.3.5 Présence/absence d'autres véhicules

Nous avons souhaité étudier le scénario dans lequel seul le véhicule à détecter est présent dans l'image, à l'exclusion de tout autre véhicule. Cette expérience permet de mesurer l'impact de la proximité inter-classes en terme de faux positifs.

La Tab. 4.8 montre que dans ce cas, la performance n'augmente que peu. Seule la classe 'car', et la classe 'pickup', deux classes qui se ressemblent beaucoup, gagnent en performance. Cela montre que la très large majorité des faux positifs sont des éléments du fond.

	slv	llv	boa	cam	car
Avec les autres véhicules	49.1±5.3	30.1±6.9	40.2±16	45.1±12	30.7±7
Sans les autres véhicules	54.2±4.8	33.1±6.5	45.5±15.7	48.6±11.8	65.7±6.5
gain	+5.1	+3.0	+5.3	+3.5	+30.5
	Oth	Pic	Tra	Tru	Van
Avec les autres véhicules	3.0±4.3	36.0±5.9	20.1±6.5	36.4±7.7	53.3±14
Sans les autres véhicules	4.0±4.8	61.0±5.0	24.5±9.1	42.4±9.1	62.4±13.7
gain	+1.0	+25.0	+4.4	+6.0	+9.1

TABLE 4.8 – Dans cette expérience, les faux positifs dus aux autres véhicules ne sont pas comptés. Les résultats sont données sur le VeDAI-PII, avec un détecteur HOGLBP + SVM.

4.4 Reconnaissance : résultats généraux

Les résultats sur le problème de Reconnaissance sont présentés en Tab. 4.9. Un exemple de matrice de confusion par HOG+SVM linéaire est donné en Tab. 4.10.

Pour ce qui est de la comparaison entre les classifieurs, le MLP est bien en dessous des performances des SVMs. En plus d'avoir des performances plutôt mauvaises, les résultats ne sont pas reproductibles, ainsi qu'en témoigne le fort écart type des performances.

Le SVM multi-classes n'apporte pratiquement rien comparé à une approche de 8 SVM en parallèle. Le multi-classes permet de lisser les variations de performances classe par classe, mais au final, le taux de reconnaissance est légèrement moins élevé. L'ajout d'un modèle probabiliste est crucial pour l'obtention de bonnes performances, sinon le biais envers les catégories sur-représentées dans la base (comme les voitures) prend trop d'importance. L'utilisation d'un SVM RBF permet de gagner significativement en performance.

méthode	Boa	Cam	Car	Oth	Pic
8 SVM linéaires	5.7±6.75	89.5±4.4	91.8±3.4	14.0±10.0	15.2±5.0
8 SVM linéaires probabilistes	21.4±18.3	83.9±4.6	82.8±5.6	24.2±7.6	53.0±8.4
8 SVM RBF probabilistes	61.9±15.4	86.4±6.9	86.7±3.9	39.8±6.2	77.4±3.5
1 SVM MC linéaire	16.2±9.3	76.2±5.2	70.7±6.8	25.4±7.5	59.2±7.4
1 SVM MC RBF	56.1±13.4	83.7±7.7	84.6±3.9	47.8±6.5	74.5±1.9
MLP	1.4.±3.9	36.4±47.1	63.1±34.0	32.9±29.8	69.9±25.7
méthode	Tra	Tru	Van	Macro	Micro
8 SVM linéaires	17.0±9.4	5.2±4.7	1.5±2.1	29.98±1.8	50.8±2.1
8 SVM linéaires probabilistes	48.6±8.2	11.9±7.2	27.1±8.0	44.1±3.4	60.3±3.2
8 SVM RBF probabilistes	71.6±10.8	56.4±9.5	57.3±13.5	**67.2±2.2**	**76.4±2.0**
1 SVM MC linéaire	61.6±14.0	22.4±7.4	49.1±11.3	47.6±3.3	58.6±2.7
1 SVM MC RBF	68.4±11.6	57.7±7.8	60.9±12.0	66.8±3.3	75.1±3.2
MLP	30.7±39.4	18.7±26.0	0.3±0.8	31.7±8.3	50.7±11.3

TABLE 4.9 – Résultats de classification donnés, sur des descripteurs HOG, par différents classifieurs, à savoir de haut en bas : SVM linéaire, SVM linéaire + modèle probabiliste, SVM à noyau RBF probabiliste, SVM multi-classes RBF probabiliste, SVM multi-classes linéaire, et enfin Perceptron Multi-Couches.

		Classe prédite							
		Boa	Cam	Car	Oth	Pic	Tra	Tru	Van
Classe évaluée	Boa	59.1	10.2	21.6	0	1.1	0	8.0	0
	Cam	0.6	90.3	4.9	0	2.8	0	0	1.4
	Car	0	0	87.6	0	10.7	0.6	1.1	0
	Oth	0	0	45.6	30.0	10.6	12.5	1.3	0
	Pic	0	0.3	36.8	1.6	57.0	1.1	3.3	0
	Tra	0	0	30.3	4.6	10.5	54.6	0	0
	Tru	0	19.6	21.8	2,7	40.2	0	15.2	0.5
	Van	0	20.0	27.5	0	10.0	0	2.5	40.0

TABLE 4.10 – Matrice de confusion en utilisant 8 SVM Linéaires sur des descripteurs HOG.

4.5 Conclusions

Une étude paramétrique des hypothèses de test et d'apprentissage a mis en évidence que le protocole de test était suffisamment robuste pour permettre une comparaison claire des algorithmes. Après l'étude des différentes chaînes algorithmiques classiques, il ressort que la chaîne plusieurs descripteurs+SVM donne les meilleurs résultats. L'étude des différents paramètres montre que l'utilisation d'exemples virtuels et de nombreuses racines permet de gagner en performance avec cet algorithme. Cependant, l'ajout d'apprentissage latent ou de parties ne permet pas d'avoir un gain significatif, de même que l'utilisation de concaténation de descripteurs classiques.

La conclusion majeure reste que les performances globales ne sont pas assez élevées. En effet, pour avoir un taux de détection de l'ordre de 90 pour cent, qui serait un taux de détection minimum nécessaire pour toute application, le taux de faux positifs est très élevé (10 FPPI). Afin de développer des algorithmes plus performants, il est nécessaire d'envisager d'autres approches.

Pour ce qui est de la Reconnaissance, là encore, il parait difficile d'exploiter les résultats tels quels, le macro averaging ne dépassant pas les 70 pour cent, tandis que le micro averaging dépasse à peine les 75 pour cent.

Il nous a donc paru nécessaire d'explorer de nouvelles voies, ce que nous nous proposons de faire dans les chapitres suivants, en commençant par chercher à modéliser plus finement les apparences possibles des véhicules à détecter/reconnaitre.

Chapitre 5

Variétés génératives

Dans le chapitre précédent, nous avons présenté et étudié plusieurs chaînes algorithmiques de Détection et Reconnaissance reposant sur des composants issus de l'état de l'art et nous les avons appliquées au cadre de la problématique des véhicules faiblement résolus. Il nous est apparu que, si certaines techniques (par exemple la modélisation par plusieurs SVM utilisant des descripteurs efficaces) donnent de bons résultats, les performances ne sont pas encore suffisantes pour les applications visées. Cela nous a conduit à développer de nouvelles approches. Dans ce chapitre, nous proposons une approche qui prend en compte de manière plus spécifique le fait que les objets sont de faible résolution et que les fonds sont complexes.

Étant donnée la faible résolution des objets, nous souhaitons modéliser le plus finement possible l'ensemble de leurs apparences possibles dans les images, tout en notant que ces apparences peuvent être très variées. À l'inverse, l'ensemble des fonds [1], très vaste, ne semble pas être modélisable dans sa globalité ou du moins, s'il l'est, il ne semble pas qu'utiliser un modèle semblable pour les véhicules et le fond soit pertinent. Nous avons donc développé une approche basée sur des modèles (génératifs) distincts, afin de modéliser le fond et les véhicules différemment.

Dans ce chapitre, nous introduisons le cadre théorique des variétés, par lequel nous avons choisi de modéliser nos véhicules et nos fonds. Dans un second temps, nous présentons comment utiliser les variétés pour de la classification et plus particulièrement comment utiliser les autoencodeurs et l'Analyse en Composantes Principales. Enfin, nous présentons les validations expérimentales des méthodes proposées, les résultats obtenus et leurs interprétations.

5.1 Cadre théorique et adéquation à la problématique

Afin de travailler sur une modélisation fine des véhicules, nous nous sommes intéressés à la théorie des variétés. Dans cette section, nous définissons ce qu'est une variété et quels sont les concepts mathématiques associés, puis nous présentons en quoi cette théorie est pertinente vis-à-vis de notre problématique. Ensuite, nous introduisons les grandes familles d'algorithmes existants pour l'apprentissage de variété et enfin, après quelques remarques générales, nous présentons comment utiliser ces variétés comme classifieur.

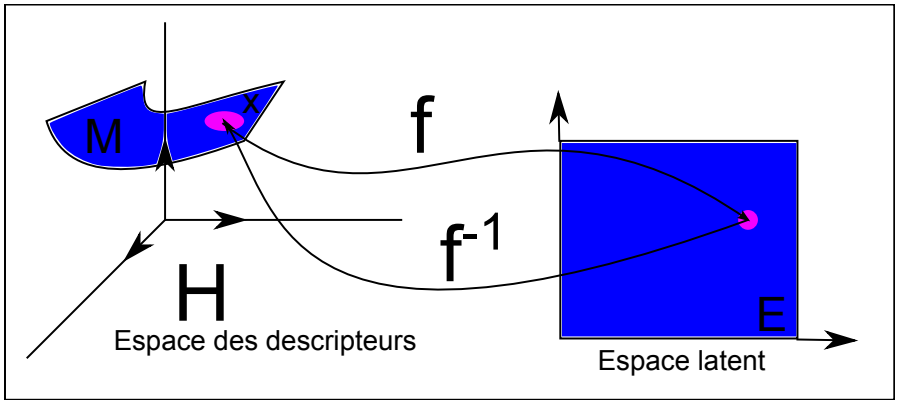


FIGURE 5.1 – Illustration d'une variété $\mathcal{M}$ dans un espace H

5.1.1 Définitions

Définition. Variété : Une *variété* (*manifold* en anglais) est un espace topologique abstrait, qui se comporte localement comme un espace euclidien. Les variétés seront notées $\mathcal{M}$. L'illustration d'une variété est présenté en Fig. 5.1.

Définition. Espace tangent : Un espace euclidien local de la variété $\mathcal{M}$ est appelé *espace tangent*. Cet espace sera noté E et sa dimension sera notée d. d est appelée par extension la dimension de la variété $\mathcal{M}$.

Définition. Espace d'origine : L'espace vectoriel euclidien contenant $\mathcal{M}$, de dimension n sera noté H. Dans le cadre de cette thèse, n est supposé grand. En pratique, nous avons $n > 100$. De plus, toujours dans le cadre de ces travaux, $d < n$. Cet espace correspond à l'espace des descripteurs.

Définition. Variété riemannienne : Les *variétés riemanniennes* sont des variétés qui ont la propriété supplémentaire que leurs espaces tangents varient de manière continue. Elle peuvent donc être dotées d'une métrique. Ici, les variétés que nous utilisons sont toujours considérées riemanniennes.

Définition. Plongement : La fonction f définie telle que :

$$\forall \mathbf{x} \in \mathcal{M}, \exists! \bar{\mathbf{x}} \in E, \bar{\mathbf{x}} = f(\mathbf{x}) \tag{5.1}$$

est appelée le plongement de la variété $\mathcal{M}$.

Théorème. Plongement de Nash : Toute variété riemannienne peut être plongée de manière isométrique dans un espace euclidien (preuve de l'existence d'une fonction de plongement f).

Propriété. Isométrie : La fonction de plongement f conserve les distances entre l'espace de départ et l'espace tangent.

1. Nous rappelons qu'un *fond* désigne toute région de l'image qui ne contient pas de véhicule

Définition. Distance géodésique : La *distance géodésique* correspond à la longueur du plus court chemin pour aller d'un point à un autre d'une variété, en restant dans cette variété.

5.1.2 Adéquation à la problématique

La théorie des variétés s'adapte bien à notre problématique. En effet, les problèmes de Détection et Reconnaissance dans des images de faible résolution se heurtent à des espaces de grandes dimensions. Par exemple, l'espace des images 20x20 pixels (ce qui est une taille plutôt petite pour une image) est déjà un espace de dimension 400. Dans le cadre de la Détection et de la Reconnaissance, un objet a une apparence qui décrit un ensemble de points possibles dans cet espace de dimension 400. Cependant, cet ensemble de points n'a qu'un nombre restreint de degrés de liberté dans cet espace. Ces degrés de libertés correspondent aux variations d'un certain nombre de paramètres, tels que la position ou l'illumination. L'hypothèse selon laquelle l'ensemble des points représentant un objet dans l'espace vectoriel des images peut être généré par un nombre restreint de paramètres (plus petit que la dimension de l'espace) est équivalent à dire que ces points se trouvent sur une *variété* de dimension inférieure à la dimension de l'espace de départ.

De plus, si la faible résolution nous fait perdre les détails de l'objet à détecter, elle a l'avantage d'estomper la variance intra-classe. Ainsi, contrairement à des apparences fortement résolues, la variété ne présente pas de discontinuités fortes entre les différences instances d'une même classe. La figure 5.2 illustre une variété d'images générée par deux paramètres, la pose et le contraste.

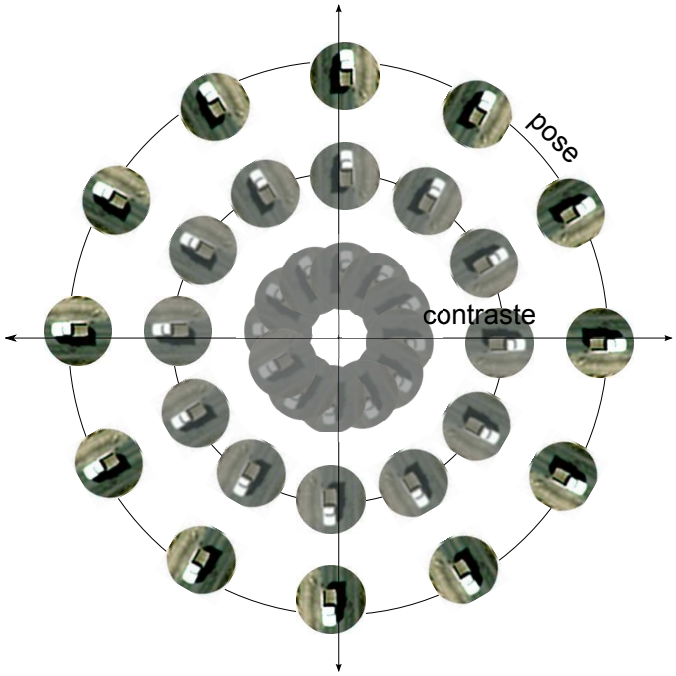


FIGURE 5.2 – Illustration d'une variété d'images générée par deux paramètres, la pose et le contraste.

5.1.3 Principaux algorithmes d'apprentissage de variété

Les algorithmes de modélisation de données peuvent être classés en deux grandes catégories : les méthodes linéaires et les méthodes non linéaires. Les méthodes linéaires considèrent que la variété est un hyperplan. Cette restriction est très forte mais permet dans certains cas d'obtenir de bonnes performances. La technique la plus simple est l'Analyse en Composante Principale (ACP, [73]). Le principe fondateur est de changer de repère dans l'espace de départ, afin de décorréler les variables. L'Analyse Discriminante Linéaire (LDA, [48]) reprend le principe de l'ACP mais dans un problème pour lequel les données sont labellisées. Une ACP projette les données sur leurs axes de plus grandes variances, tandis que la LDA les projette sur les axes de plus grandes variances inter-classes et de plus petites variances intra-classe. Enfin, l'Analyse en Composantes Indépendantes (ICA [17]), projette les données selon des axes principaux qui permettent de les séparer au mieux, en considérant que les différentes classes sont issues de sources différentes.

Les méthodes non linéaires se basent sur différentes propriétés d'une variété. Trois types peuvent être distingués : les techniques fondées sur la conservation des distances c'est-à-dire sur la propriété d'isométrie du plongement, celles fondées sur l'approximation locale par des espaces euclidiens et enfin, celles fondées sur la conservation globale des propriétés topologiques.

Pour les techniques basées sur la propriété d'isométrie du plongement, le Multidimensional Scaling (MDS, [91]), Isomap [161] et les cartes de diffusion [92] sont les techniques les plus utilisées. MDS force la distance euclidienne entre les points de H à être aussi près que possible de leur distance dans E. Isomap est une extension de MDS et approxime les distances dans la variété par la longueur d'un chemin dans un graphe de distance, puis utilise l'algorithme MDS vu précédemment pour effectuer la transformation de H dans E, ce qui va permettre de conserver les distances géodésiques et donc la topologie de la variété. Du fait de son approximation des distances géodésiques, cet algorithme ne peut pas (ou difficilement) modéliser des variétés dont l'espace des paramètres est non convexe [33]. De plus, il est très sensible au bruit et à la taille du voisinage choisi. Les cartes de diffusion [92] sont quant à elles basées sur les marches aléatoires markoviennes. En effectuant une marche aléatoire entre 2 points, une pseudo-distance peut être déduite. À partir de cette pseudo-distance, une cartographie de la variété est inférée. Cette cartographie est censée être plus robuste que celle estimée par l'approximation des distances géodésiques d'Isomap puisque l'ensemble de tous les chemins dans le graphe est considéré.

Des techniques pour la modélisation de variétés en utilisant la propriété d'approximation par sous-espaces euclidiens, les plus connues sont les Local Linear Embedding (LLE, [145]) et ses extensions [32], Local Tangente Space analysis (LTSA [181]) ou encore Local Preserving Projection (LPP [123]). LLE représente la variété comme un ensemble de patchs localement linéaires. LLE utilise l'algorithme des plus proches voisins et construit le meilleur hyperplan générant ces points. Comme pour HLLE, LTSA décrit localement la variété par des espaces tangents. Chaque point décrit avec ses voisins un espace tangent par ACP, puis ces espaces tangents sont alignés. Enfin, LLP est l'approximation linéaire optimale de LLE. Cette approximation permet en particulier de résoudre le problème de la représentation de nouveaux points dans la variété.

Enfin, certaines méthodes non linéaires se basent sur la conservation globale de l'information. Les autoencodeurs [88] permettent d'effectuer une régression pour trouver une expression du plongement. La régression s'effectue avec un réseau de neurones non linéaire (l'utilisation d'un réseau linéaire revient théoriquement à effectuer une ACP). Les

sorties dans E n'étant pas connues, un second réseau de neurones symétrique au premier est ajouté. Il permet d'aller de E dans H pour ensuite retrouver les données d'entrée. Ainsi, nous souhaitons trouver la fonction Identité qui va de H dans H et qui associe un vecteur de données x à lui-même. Ainsi, la fonction f réduit la dimension de la donnée x tout en conservant un maximum d'information. La Fig. 5.3 illustre des architectures classiques d'autoencodeurs. Plus de détails seront donnés section 5.2. L'approche par rétro propagation du gradient converge assez difficilement, mais une approche par les Restricted Boltzmann Machines [1] permet de bien initialiser les poids pour que la convergence soit plus aisée, ainsi que démontré dans [68]. Nous rappelons que les travaux de [25] ont montré qu'un réseau de neurones à deux couches dont la fonction d'activation n'est pas polynomiale est un approximateur universel. Ainsi, toute fonction peut être approximée par un Réseau de Neurones d'aussi proche que souhaité, tant qu'assez de neurones sont utilisés. L'algorithme Maximum Variance Unfolding (MVU [175, 174]) déplie la variété, en éloignant les points qui sont éloignés dans la variété (grande distance géodésique) mais qui sont proches dans l'espace euclidien de départ (faible distance euclidienne), tout en conservant la proximité des points qui sont proches dans la variété. Enfin, la méthode d'ACP par noyau [148] est une extension non linéaire de l'analyse en composante principale. Il s'agit d'utiliser le principe de l'astuce du noyau, utilisée dans les Séparateurs à Vastes Marges . Il a été démontré dans [62] que ISOMAP, LLE et LPP peuvent être considérés comme une ACP à noyaux utilisant des noyaux particuliers.

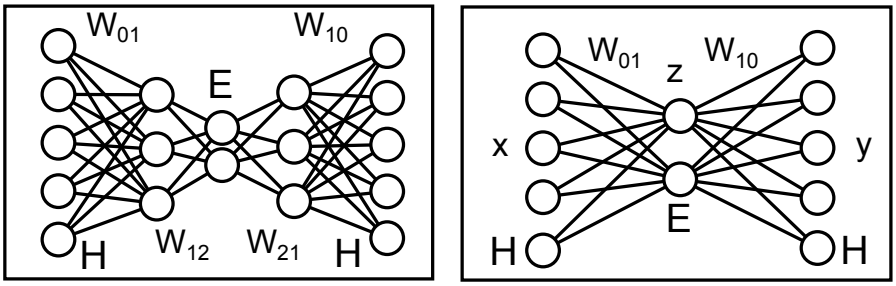


FIGURE 5.3 – Exemples d'architectures d'autoencodeurs.

5.1.4 Remarques générales

Un des problèmes clefs de l'utilisation des variétés est la dimension intrinsèque du problème. Elle n'est généralement pas connue. Ainsi, lors de l'utilisation de telles techniques, il sera intéressant d'optimiser le paramètre d. Il est toutefois possible de donner une borne minimum à d. Cette borne correspond au nombre de paramètres connus qui a un effet indépendant sur les données.

Un second problème dans l'utilisation des variétés, ainsi que présenté dans [179], qui compare ACP, MDS, LLE, HLLE, Isomap, LTSA, Kernel ACP, Diffusion Maps et autoencodeurs, est la robustesse au bruit. Si les données sont trop bruitées, la topologie de l'espace peut être difficile à modéliser.

5.1.5 Utilisation dans un classifieur

Ce qui est présenté ici se rapproche des travaux de Pentland [118]. Il est possible d'utiliser les variétés comme modèle génératif, en effet une fois l'espace E appris et le

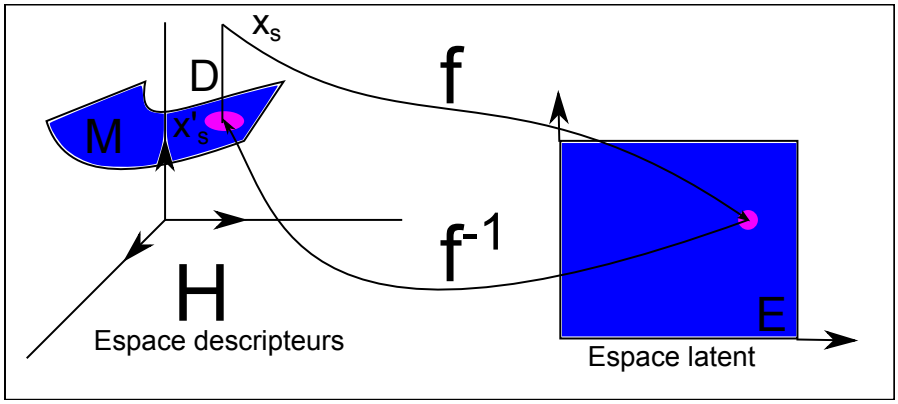


FIGURE 5.4 – Illustration du concept de *distance à la variété*. Soit X_s un vecteur et $X'_s = f^{-1} \circ f(X_s)$ sa projection sur la variété. $||X'_s - X_s||$ est la distance à la variété.

plongement f connu, il est possible de générer l'ensemble de la variété grâce à la fonction f^{-1}. Il est toutefois important de noter que beaucoup d'algorithmes dans leur forme classique ne donnent f^{-1} que pour les points du set d'apprentissage. Seules les variantes d'ACP et d'autoencodeurs le permettent.

Pour créer un classifieur simple à partir d'une variété apprise, il suffit de remarquer qu'il est possible de créer une distance à la variété. Soit $\mathcal{H}$ l'espace des descripteurs et $\mathbf{x} \in \mathcal{H}$ un descripteur visuel extrait d'une image (par exemple la signature d'une région de l'image). Nous avons :

$$\forall \mathbf{x} \in \mathcal{M}, f^{-1} \circ f(\mathbf{x}) = \mathbf{x} \tag{5.2}$$

Nous définissons la projection sur la variété comme suit :

$$\mathbf{x} \in \mathcal{H}, P_{\mathcal{M}}(\mathbf{x}) = f^{-1} \circ f(\mathbf{x}) \tag{5.3}$$

puis la distance à la variété :

$$D_{\mathcal{M}}(\mathbf{x}) = ||\mathbf{x} - P_{\mathcal{M}}(\mathbf{x})|| \tag{5.4}$$

Le principe de cette projection est illustré en Fig. 5.4. Il est ensuite possible de modifier cette distance en probabilité comme suit :

$$p(\mathbf{x} \in \mathcal{M}|\mathbf{x}) = \mathcal{C} \exp - \frac{\mathcal{D}_{\mathcal{M}}(\mathbf{x})}{\sigma^{\in}} \tag{5.5}$$

avec σ et C deux scalaires constants, le premier modélisant le bruit sur la variété, le second permettant la normalisation en probabilité. Une fois une ou plusieurs variétés apprises et transformées en probabilité, il est possible d'utiliser un cadre bayésien afin de créer un classifieur.

Les modèles génératifs ne décrivent que la probabilité que l'observation soit effectivement l'objet. Elle ne permet cependant pas d'écarter les faux positifs. En conséquence, les travaux utilisant des modèles génératifs mettent en opposition un modèle génératif d'objet à détecter et un modèle génératif de fond. La décision est prise grâce à un ratio de log-vraisemblance :

$$Score = log(\frac{p_{objet}}{p_{fond}}) \tag{5.6}$$

Remarque : Étant donné que le modèle de fond et le modèle d'objet sont indépendants, nous n'avons pas p_{objet}=1 − p_{fond}.

Avec l'expression des probabilités données en équation 5.5, nous obtenons :

$$log(\frac{p_{objet}}{p_{fond}}) = log(\frac{C_o e^{(-\frac{D_{\mathcal{M}_l}(\mathbf{x})}{\sigma_o^2})}}{C_f e^{(-\frac{D_{\mathcal{M}\{}(\mathbf{x})}{\sigma_f^2})}}}) \tag{5.7}$$

$$= log(C_o) - log(C_f) - \frac{D_{\mathcal{M}_l}(\mathbf{x})}{\sigma_o^2} + \frac{D_{\mathcal{M}\{}(\mathbf{x})}{\sigma_f^2} \tag{5.8}$$

$$\sim C - \alpha D_{\mathcal{M}_l}(\mathbf{x}) + D_{\mathcal{M}\{}(\mathbf{x}) \tag{5.9}$$

avec C un scalaire et α le ratio des deux variances. Ainsi, dans un contexte de classification où seules les valeurs relatives des scores nous intéressent, il est possible de négliger C. Le paramètre α peut se régler par validation croisée.

5.2 Utilisation d'autoencodeurs et d'ACP pour la Détection et la Reconnaissance

Dans cette section, nous expliquons comment calculer une distance à une variété avec une ACP, puis avec un autoencodeur. Ensuite, nous décrivons les chaînes algorithmiques utilisées dans le cadre de la détection et l'utilisation des variétés à des fins de Reconnaissance.

5.2.1 Analyse en Composantes Principales

Il est possible d'utiliser l'ACP afin de projeter des données de dimension n sur un espace de plus faible dimension d, en ne conservant que les d premières composantes principales lors de la projection. Pour rappel, nous avons :

$$\mathbf{x}_{ACP} = M\mathbf{x} \tag{5.10}$$

avec M la matrice résultant de l'ACP. La variable réduite s'obtient alors :

$$for\ j = 1..d\ \mathbf{x}_{reduit}(j) = M\mathbf{x}_{ACP}(j) \tag{5.11}$$

Il est possible de directement tronquer la matrice M afin d'obtenir le résultat :

$$\mathbf{x}_{reduit} = M_d\mathbf{x} \tag{5.12}$$

Le projeté sur $\mathcal{M}$ s'obtient donc :

$$\tilde{\mathbf{x}} = M_d^T\mathbf{x}_{reduit} \tag{5.13}$$

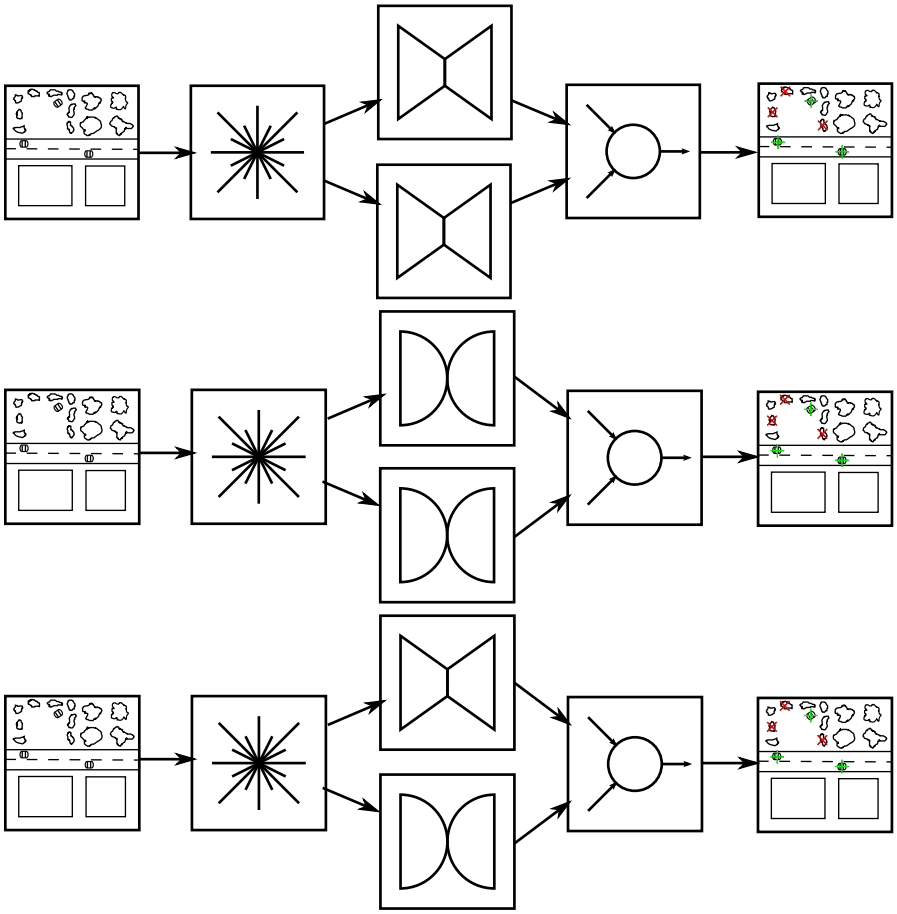


FIGURE 5.5 – Trois chaînes algorithmiques basées sur des variétés. En haut et au centre, une modélisation identique pour les véhicules et pour les fonds par une ACP ou un autoencodeur. En bas, une modélisation différente, avec une ACP pour les fonds, un autoencodeur pour les objets.

5.2.2 Autoencodeur

Nous présentons ici plus en détail l'autoencodeur et en particulier comment l'utiliser afin d'apprendre une variété. Dans cette section, nous nous référerons à l'autoencodeur comme un autoencodeur *standard*, afin de le différencier des autoencodeurs des sections suivantes. Les autoencodeurs standards sont des réseaux de neurones symétriques, qui apprennent la fonction identité sous contrainte. Deux architectures possibles de l'autoencodeur sont montrés Fig. 5.3 mais des architectures plus complexes peuvent être utilisées.

Un neurone de la couche i est connecté à la couche $i+1$ et seulement à ces neurones. Soit W_{ij} la matrice des poids entre la couche i et j. Les couches sont numérotées de 0 (entrée) à N (couche centrale) puis de nouveau de N (couche centrale) à 0 (sortie), tel que montré Fig. 5.3. Comme le réseau est symétrique, on a :

$$dimension(W_{ji}) = dimension(W_{ij}^T) \tag{5.14}$$

Chaque couche j a sa sortie $\mathbf{r}(\mathbf{x})$ complètement définie par la couche précédente $\mathbf{x}$ et la matrice des poids W_{ij} par :

$$\mathbf{r}(\mathbf{x}) = h(W_{ij}\mathbf{x}) \tag{5.15}$$

h est appelée la *fonction d'activation* – typiquement la fonction sigmoïde ou la fonction arc tangente sont utilisées, comme souvent dans les réseaux de neurones. Quand h est linéaire pour toutes les couches, l'autoencodeur effectue une ACP [88], il est donc important d'utiliser des fonctions non linéaires. Nous rappellons qu'utiliser une ou plusieurs couches avec des fonctions d'activation non linéaires parmi les couches du réseau permet à celui-ci d'approximer n'importe quelle fonction [25].

Soit χ l'ensemble des vecteurs d'entrainements $\mathbf{x}$. L'autoencodeur standard minimise la fonction de coût [88] :

$$L(\chi) = \sum_{\mathbf{x} \in \chi} ||\mathbf{x} - \tilde{\mathbf{x}}||^2, \tag{5.16}$$

minimisant ainsi l'erreur de reconstruction des exemples positifs, $\tilde{\mathbf{x}}$ étant la reconstruction de $\mathbf{x}$ donnée par l'autoencodeur. Cette erreur est généralement minimisée en utilisant une descente de gradient stochastique, dans le cadre d'une rétro propagation de l'erreur [97]. f et son inverse sont ainsi apprises simultanément. L'espace latent E est disponible à la sortie de la couche centrale. Des techniques pour avoir une meilleure convergence existent, telles que l'utilisation de Restricted Boltzmann Machine [1] et de la *Contrastive Divergence* [66] ou bien l'utilisation du *dropout*. Le lecteur intéressé peut se reporter à [69] et [68] pour plus de détails.

Dans le contexte de l'apprentissage de variétés, le réseau est utilisé pour apprendre f, donnant ainsi un plongement des données [68]. Ici à l'inverse, le réseau est entièrement appris, ce qui donne la projection $P_{\mathcal{M}}(x)$ dont nous avons besoin. Nous rappelons que la distance s'obtient par l'équation 5.4, puis la probabilité par l'équation 5.5.

5.2.3 Chaînes algorithmiques

Ainsi qu'expliqué en introduction de ce chapitre, les fonds n'ont aucune raison d'avoir une structure ou des caractéristiques identiques aux objets à détecter. En conséquence, nous proposons d'utiliser un autoencodeur afin de modéliser les non-linéarités qui lient les différentes apparences d'un d'objet (rotation par exemple) et une Analyse en Composantes Principales afin de modéliser les fonds. Afin de valider l'hypothèse selon laquelle utiliser un autoencodeur pour modéliser les véhicules et une ACP pour modéliser les fonds est pertinente, nous avons testé une méthode modélisant les deux domaines par des autoencodeurs et une autre modélisant les deux par une ACP. Les trois architectures sont présentées Fig. 5.5, de même que précédemment la table des symboles est située en Annexe A.

Pour OIRDS, nous disposons en point de comparaison des chaînes de traitement de l'état de l'art (HOG+SVM, DPM, TM et GM) présentées chapitre 4, qui servent de comparaison aux trois architectures décrites ci-dessus. Afin de valider l'approche qui doit fonctionner quel que soit l'espace des descripteurs, nous avons testé les trois architectures sur trois descripteurs, à savoir (i) les niveaux de gris normalisés (NDG), (ii) les gradients normalisés (GRA) et (iii) les HOG, calculés tels qu'expliqué Chap. 4.

Pour VeDAI, nous avons effectué la comparaison entre un SVM mono-racine, un SVM 12 racines et l'algorithme AE contre ACP.

Enfin, pour la problématique de Reconnaissance, nous avons utilisé 8 autoencodeurs, chacun modélisant une classe. Nous avons également appris un modèle statistique sur les scores retournés par les autoencodeurs, afin de normaliser les décisions.

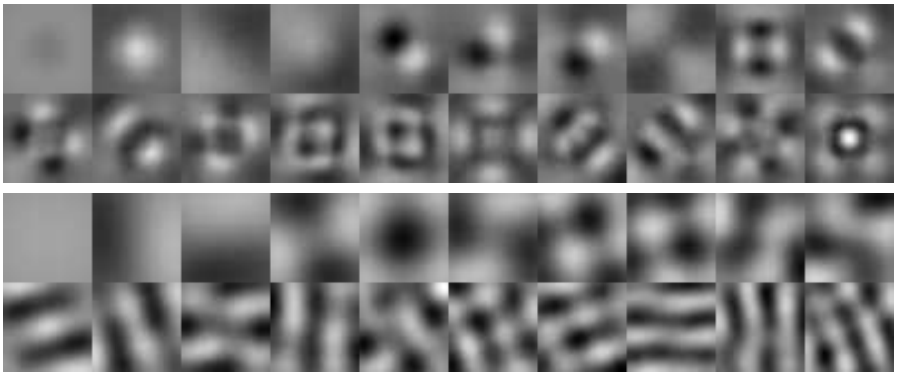


FIGURE 5.6 – Les 20 premières composantes principales d'une ACP, sur les voitures en haut, sur les fonds, en bas.

5.3 Résultats et analyse

5.3.1 Visualisation des variétés et des reconstructions

Fig. 5.6 montre les premières composantes principales d'une ACP (i) sur les fonds et (ii) sur les voitures, en utilisant les niveaux de gris normalisés. Dès les premières composantes, le modèle de fond contient beaucoup de basses fréquences et de textures, tandis que les composantes de voitures permettent de reconstruire des objets centrés sur un fond uni ou bicolore. L'allure des composantes principales des fonds sont typiques des images naturelles (comme montré par exemple dans [63]).

La Fig. 5.7 montre quelques apparences que l'autoencodeur peut générer une fois entrainé à partir de niveaux de gris normalisés (à gauche sur la figure). La rotation apprise par l'autoencodeur est assez apparente. Cependant les apparences générées ressemblent à des voitures mais n'en sont pas tout à fait. Ce résultat illustre le défaut de l'autoencodeur de généraliser abusivement. A droite de la figure est présenté ce qui peut être généré par un autoencodeur appris sur des fenêtres de fonds. Dans ce cas, l'autoencodeur se concentre sur les différences d'intensité et n'apprend aucune rotation, ce qui est tout à fait attendu.

La Fig. 5.8 montre des fenêtres test (1ère ligne), leurs projections sur une variété de fonds obtenue par ACP (2ème ligne) et enfin leur projection sur une variété de voitures, apprise par un autoencodeur (dernière ligne). Les fenêtres de voitures sont mieux reconstruites par la variété de voitures et inversement pour les fenêtres de fonds. En revanche, bien que la différence entre les reconstructions soit flagrante, la reconstruction des véhicules reste approximative.

5.3.2 Cartes d'erreur

La Fig. 5.9 montre des *cartes d'erreur* (c'est-à-dire l'image dans laquelle un pixel représente la distance entre une région centrée en ce pixel et une variété, de fond ou de véhicule. L'autoencodeur seul donne beaucoup de faux positifs et doit être contrebalancé par le modèle de fond, en concordance avec les expériences précédentes (Template Matching du Chap. 4). La variété apprise par l'autoencodeur peut générer n'importe quel motif uni, qui correspond aux véhicules qui se confondent avec le fond. Ces faux positifs peuvent être enlevés facilement grâce à l'ACP, étant donné qu'un fond uniforme est reconstruit par les premières composantes principales.

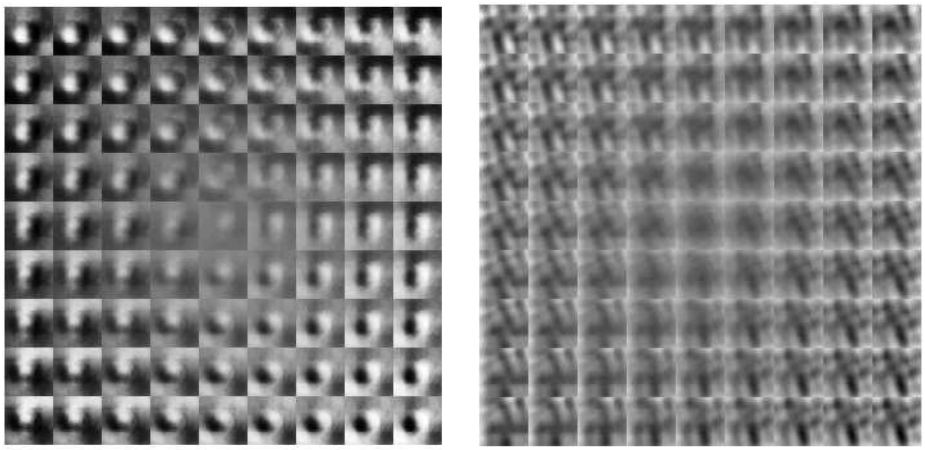


FIGURE 5.7 – Visualisation de deux parties de variétés apprises par un autoencodeur, à gauche pour des véhicules, à droite pour des fonds.

5.3.3 Analyse des faux positifs et faux négatifs

La Fig. 5.10 montre les faux positifs de meilleurs scores obtenus avec un SVM, en comparaison avec le détecteur AE contre ACP. La figure montre également les véhicules les plus difficiles, c'est à dire les Vrais Positifs à bas score. Chaque fenêtre est légèrement agrandie afin de voir le contexte qui l'entoure.

Il est notable que pour le SVM ou l'AE contre ACP, certains négatifs difficiles pourraient être comptabilisés par un être humain comme positifs. En réalité, la localisation est légèrement excentrée, donnant un label négatif. Comme montré Chap. 4, nous rappelons que ce genre de comportement n'est pas significatif en terme de dégradation des performances et découle d'un choix arbitraire de tolérance autour du centre de l'objet.

Il est également possible de remarquer que si les faux positifs sont très différents entre le SVM et l'AE contre ACP, les positifs difficiles sont le plus souvent identiques. Cela tend à montrer qu'en ce qui concerne ces fenêtres particulières, le descripteur pose problème. Sur certains négatifs difficiles, des points spéculaires et des ombres sont visibles. Ces deux éléments perturbent la détection.

5.3.4 Étude paramétrique

Différents paramètres doivent être estimés, en particulier la dimension de l'espace latent, correspondant au nombre de composantes conservées lors de l'ACP ou au nombre de neurones cachés de l'AE. Nous les avons fixés lors d'expériences préalables. La Fig. 5.11 montre la performance du détecteur AE contre ACP en fonction des dimensions des espaces latents, en utilisant les niveaux de gris normalisés. Le paramètre α (5.1.5) de combinaison des scores a été fixé à 1. Quelques expériences ont montré qu'une légère optimisation de ce paramètre est possible, cependant elle est négligeable vis-à-vis des autres paramètres comme les dimensions des variétés. Nous observons qu'il y a une certaine corrélation entre les deux modèles, avec un optimum plat des paramètres. Les dimensions des variétés trouvées par ces quelques expériences préliminaires, sont pour les fonds de 40, 10 et 16 pour NDG, GRA et HOG. Les autoencodeurs utilisés ont une seule couche cachée, ayant respectivement 35, 8 et 10 neurones pour les différents descripteurs

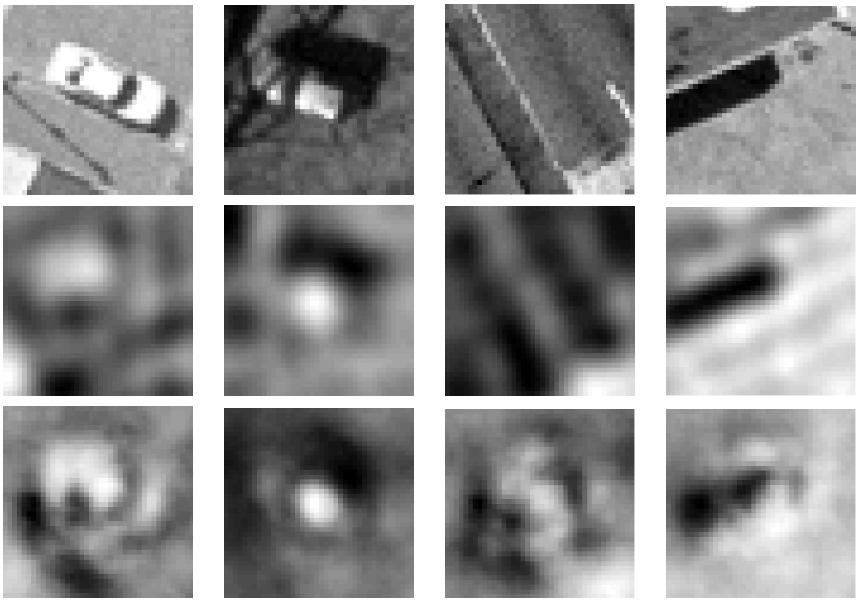


FIGURE 5.8 – Quatre fenêtres test (1ère ligne), suivies par leur reconstruction grâce à une ACP (2ème ligne), puis leur reconstruction par une variété apprise par un autoencodeur (dernière ligne).

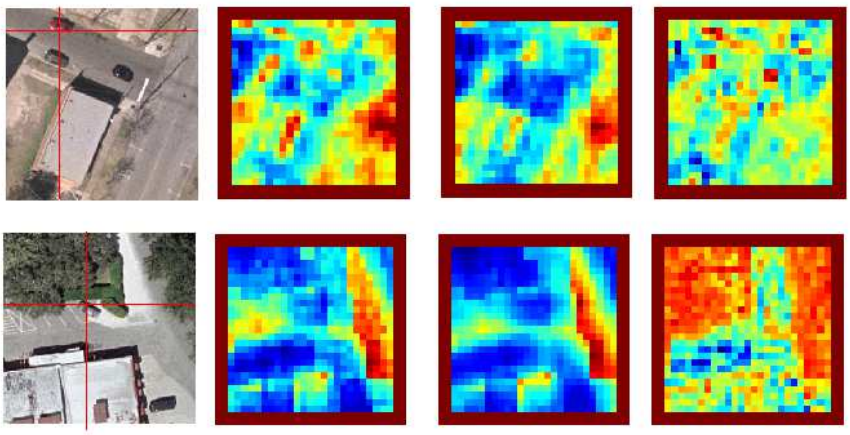


FIGURE 5.9 – De gauche à droite : (i) l'image originale, avec le maximum de vraisemblance marqué d'une croix rouge, (ii) la carte du logarithme de l'erreur de reconstruction donné par l'autoencodeur, (iii) la négative de la carte du logarithme de l'erreur de reconstruction donné par le modèle ACP, (iv), le résultat de la log vraisemblance.

NDG, GRA et HOG.

L'étape de raffinement est nécessaire pour obtenir de bons résultats, comme montré Fig. 5.11. La mAP chute au delà de 45 neurones. Ce résultat est attendu, du fait même que les autoencodeurs apprennent l'identité sous contrainte, ainsi, ajouter des neurones revient à enlever des contraintes et donc à apprendre l'identité. Ainsi qu'illustré Fig. 5.12, l'erreur de reconstruction n'est pas totalement corrélée avec la performance. Si une baisse

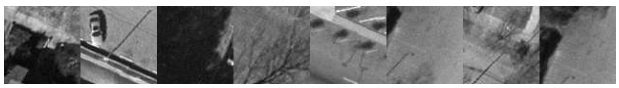
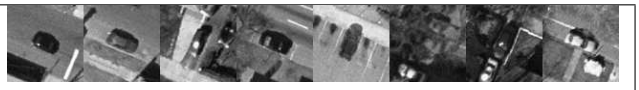
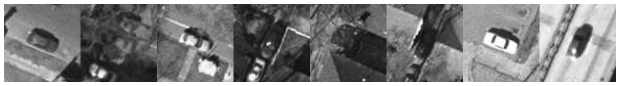
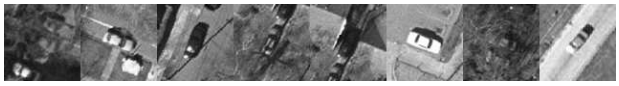
Classifieur	Descripteur	Faux positifs à haut score
SVM	NDG	
AE contre ACP	NDG	
SVM	HOG	
AE contre ACP	HOG	
Classifieur	Descripteur	Faux Négatifs
SVM	NDG	
AE contre ACP	NDG	
SVM	HOG	
AE contre ACP	HOG	

FIGURE 5.10 – Visualisation de quelques Faux Positifs forts (régions négatives mais avec un haut score) sur les 4 premières lignes, puis les positifs à plus bas score sur les 4 dernières lignes et ce, en fonction du classifieur ou du descripteur. Expérience réalisée sur OIRDS.

de l'erreur de reconstruction d'une valeur de 22 à 14 permet de gagner en mAP, le passage de 14 à 10 n'améliore rien.

5.3.5 Résultats quantitatifs

Pour chacun des détecteurs testés sur la base OIRDS, la Tab. 5.1 donne l'AP moyenne sur les dix folds. Dans cette table sont rappelées les performances obtenues par les algorithmes de référence du Chap. 4.

La principale conclusion que nous pouvons tirer de ces résultats est que l'approche AE contre ACP dépasse tous les algorithmes de référence sur la base OIRDS et ce pour n'importe quel type de descripteur. Les meilleurs résultats sont, comme attendu, obtenus avec le descripteur HOG. Le détecteur ACP-ACP bat le SVM sur des descripteurs simples tels que NDG et GRA mais le HOG-SVM reste meilleur si des HOG sont utilisés.

Utiliser deux autoencodeurs, un pour les véhicules, un pour les fonds, ne donne pas de meilleurs résultats. Le modèle par autoencodeur, bien que plus complexe qu'une ACP, ne permet pas de générer la diversité de tous les fonds, certainement à cause d'un fort sur-apprentissage. En effet, l'utilisation d'un nombre de neurones réduit ne permet pas d'éviter ce sur-apprentissage, étant donné que l'utilisation de fonctions d'activation telle que la sigmoïde amène une forte non linéarité.

Nous avons également testé l'algorithme AE contre ACP sur OIRDS-EGI. Les résul-

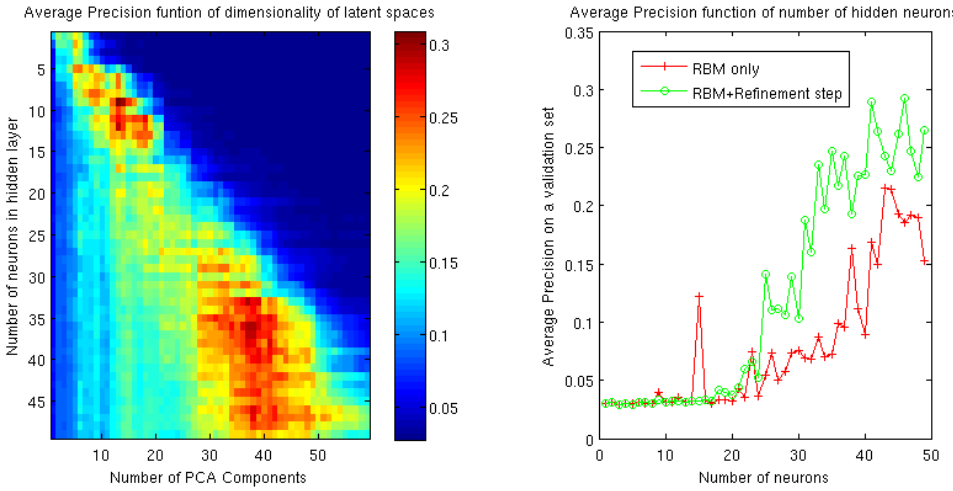


FIGURE 5.11 – (Gauche : précision moyenne sur un set de validation, pour différentes paires (nombre de composantes de l'ACP, nombre de neurones dans la couche cachée). Droite : Average Précision sur ce même set de validation, en fonction du nombre de neurones, avec et sans l'étape de raffinement par rétro propagation du gradient.

tats sont présentés Tab. 5.1. De même que le SVM, AE contre ACP donne de moins bons résultats que sur OIRDS-EPI, ce qui est attendu. Cependant, là encore, AE contre ACP gagne en performance par rapport à SVM.

Base	Classifieur	NDG	GRA	HOG
OIRDS-EPI	TM	1.79%	0.97%	0.77%
	GMM-GMM	8.3%	21,3%	17.7%
	SVM	10.5%	35.2%	46.8%
	DPM [40]	–	–	6.55%
	ACP contre ACP	35.0%	37.9%	42.5%
	AE contre AE	35.3%	33.5%	47.5%
	AE contre ACP	**35.5%**	**39.9%**	**48.9%**
Base	Classifieur	NDG	GRA	HOG
OIRDS-EGI	SVM	1.5%	12.1%	12.6%
	AE contre ACP	**3.3%**	**16.4%**	**17.1%**

TABLE 5.1 – Average Precision moyenne sur OIRDS-EPI (en haut) pour les 7 détecteurs expérimentés et OIRDS-EGI (en bas) pour les 2 meilleurs (meilleur classique et le classifieur proposé).

Nous avons également fait des expériences sur la base VeDAI en utilisant le classifieur AE contre ACP. De même que les expériences sur OIRDS, nous avons comparé notre approche à un SVM mono-racine. Les résultats sont présentés en Tab. 5.2.

Il apparait clairement que le modèle AE contre ACP permet de gagner en performance si les exemples fournis sont suffisants. En effet, AE contre ACP gagne sur un SVM pour les classes 'car', 'pickup' et 'bateau'. À l'inverse, la classe 'tra' (tracteurs) est très mal apprise. Pour les autres classes, les performances sont similaires, sans plus. Cependant, ce

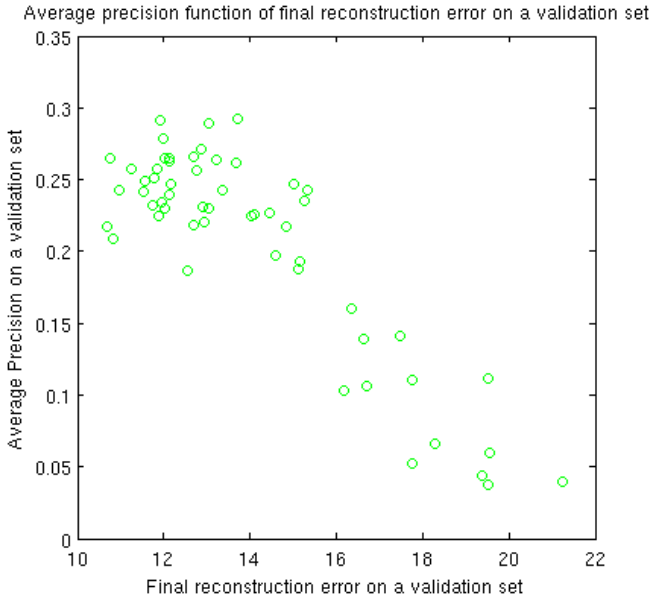


FIGURE 5.12 – Couples (AP, Erreur de reconstruction). Chaque point correspond à une architecture différente. Diminuer l'erreur de reconstruction permet de faire augmenter la performance jusqu'à un certain point. Ensuite, une erreur trop faible fait diminuer les performances.

gain de performance ne permet pas d'atteindre les performances obtenues avec un SVM disposant de 12 racines. Ainsi, la non linéarité de rotation de l'angle de vue est mieux modélisée par une multiplication des racines que par l'utilisation d'un modèle plus fin.

Détecteur	car	tru	tra	cam	van
HOG+SVM 1 racine	19.0±3.0	9.2 ± 5.7	7.0±5.3	20.6±5.8	18.8±11.1
HOG+SVM 12 racines	55.4±2.6	32.5±8.0	7.4±3.6	33.4±8.9	40.6±14.8
HOG+AE contre ACP	35.2±4.4	8.1 ± 5.1	6.1±1.2	18.0±7.9	19.1±5.3
Détecteur	oth	pic	boa	slv	llv
HOG+SVM	0.35 ± 0.1	14.7 ± 3.2	8.4 ± 6.4	21.1±2.0	19.7 ± 4.3
HOG+SVM 12 racines	6.9±4.2	48.6±4.9	32.2±14.9	71.5±2.5	36.0±4.7
HOG+AE contre ACP	0.6 ± 0.3	21.5 ± 5.2	14.5 ± 6.2	36.9±2.8	19.7 ± 5.6

TABLE 5.2 – Comparaison du modèle AE contre ACP et SVM 1 racine et SVM 12 racines. Expériences réalisées sur VeDAI-PII.

5.3.6 Reconnaissance

Les résultats de l'utilisation d'autoencodeurs pour la Reconnaissance sont présentés en Tab. 5.3. Tout d'abord, l'apprentissage d'un modèle probabiliste dégrade les performances en faveur des classes les plus représentées. Ensuite, l'utilisation d'autoencodeurs standards pour la classification permet d'obtenir des résultats meilleurs que le SVM linéaire, comme pour la détection. Cependant, les autoencodeurs sont largement moins

bons que l'utilisation d'un SVM à noyau RBF. Quelques expériences de fusion entre les autoencodeurs et le SVM RBF ont été tentées (une fusion multiplicative et une fusion par somme pondérée avec optimisation de la pondération par validation croisée), mais il n'a pas été possible d'améliorer significativement les performances.

méthode	Boa	Cam	Car	Oth	Pic
8 SVM linéaires	21.4±18.3	83.9±4.6	82.8±5.6	24.2±7.6	53.0±8.4
8 AE+modèle probabiliste	1.5±1.9	50.0±12.0	98.5±2.0	0.4±1.2	14.2±4.1
8 AE	31.9±11.6	74.8±5.7	77.8±3.4	37.1±12.6	71.0±2.6
1 SVM MC RBF	56.1±13.4	83.7±7.7	84.6±3.9	47.8±6.5	74.5±1.9
méthode	Tra	Tru	Van	Macro	Micro
8 SVM linéaires	48.6±8.2	11.9±7.2	27.1±8.0	44.1±3.4	60.3±3.2
8 AE+modèle probabiliste	0.0±0.0	0.0±0.0	27.2±17.2	24.0±2.7	46.9±1.8
8 AE	50.1±6.4	35.0±6.7	14.0±13.4	49.0±4.3	64.7±5.7
1 SVM MC RBF	68.4±11.6	57.7±7.8	60.9±12.0	66.8±3.3	75.1±3.2

TABLE 5.3 – Comparaison des résultats de Reconnaissance entre SVM et autoencodeurs.

5.4 Application à la base de données Sagem

Les architectures développées dans ce chapitre ont également été testées sur la base de données Sagem. De même, nous avons pu comparer les résultats entre les classifieurs SVM, ACP contre ACP et AE contre ACP. Les résultats sont détaillés en Annexe. D.

5.5 Conclusions

Dans ce chapitre, nous avons montré que l'utilisation de variétés génératives permet d'obtenir des performances intéressantes en détection. L'utilisation d'un modèle différent entre les fonds et l'objet à détecter est effectivement pertinente. La non linéarité de l'autoencodeur permet d'apprendre efficacement les rotations possibles et les variations d'apparences du véhicule et ainsi de gagner en performance. Cependant, l'algorithme nécessite un grand nombre d'exemples de descripteurs pour être efficace. De plus, résoudre le problème de la rotation en multipliant les racines ayant des ratios hauteur/largeur différents est plus efficace que d'apprendre par un modèle non linéaire. Pour ce qui est de la Reconnaissance, le SVM non-linéaire surpasse toujours le modèle AE.

Nous avons également pu voir que l'autoencodeur permet de générer une variété bien plus vaste que nécessaire, en englobant des apparences qui ne correspondent pas à la variété effective des véhicules. Nous présentons dans le chapitre suivant une solution pour que l'autoencodeur ne génère pas des apparences qui ne correspondent pas à des véhicules.

Chapitre 6

Variétés Discriminantes

Il a été montré dans le chapitre précédent qu'utiliser des autoencodeurs afin de modéliser les objets à reconnaître permettait d'apprendre des non-linéarités de leurs apparences. Cependant, ces autoencodeurs standards généralisent abusivement les apparences des objets ; ils nécessitent donc d'être contrebalancés par un modèle de fond. Pour résoudre ce problème, nous avons conçu et développé les *autoencodeurs discriminants*.

Dans ce chapitre, nous présentons d'abord le concept d'*autoencodeurs discriminants*, puis nous présentons les chaînes algorithmiques les incluant. Enfin, nous présentons les résultats obtenus en Détection qu'en Reconnaissance.

6.1 Autoencodeurs Discriminants

Dans cette section, nous introduisons le concept d'*autoencodeur discriminant*, puis nous développons les équations nécessaires à son apprentissage. Enfin, nous présentons une extension dans le cas multi-classes.

6.1.1 Principe

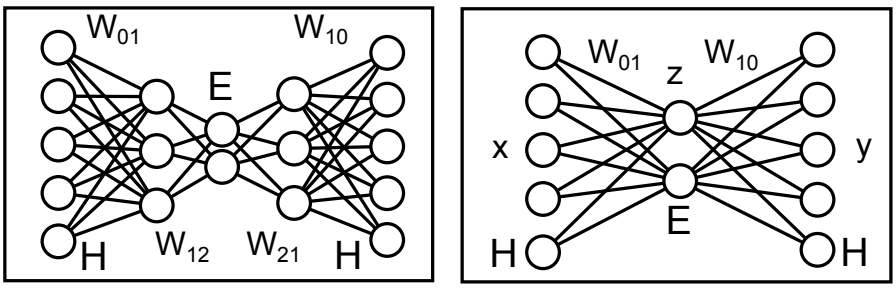


FIGURE 6.1 – Architectures possibles et notations associées pour un autoencodeur.

Un *autoencodeur discriminant* (AED) permet d'utiliser les données de deux classes (notées par la suite χ^+ pour les positifs et χ^- pour les négatifs) afin d'apprendre un modèle génératif qui sera adapté à la reconstruction des positifs tout en assurant une mauvaise reconstruction des négatifs. Ainsi, nous essayons de tirer également parti de l'information des exemples négatifs, afin d'éviter la généralisation abusive propre à l'autoencodeur classique.

L'architecture d'un autoencodeur discriminant est identique à celle d'un autoencodeur classique. Nous rappelons en Fig. 6.1 les notations prises. Soit $t(\mathbf{x})$ le label de l'exemple $\mathbf{x}$, avec $t(\mathbf{x}) \in \{-1, 1\}$ et $e(\mathbf{x})$ la distance de cet exemple à la variété, avec $e(x) = ||\mathbf{x} - \tilde{\mathbf{x}}||$. La fonction optimisée par un autoencodeur classique est donnée dans le chapitre précédent, équation 5.16. Un autoencodeur discriminant optimise la fonction suivante :

$$L_d(\chi^+ \cup \chi^-) = \sum_{\mathbf{x} \in \chi^+ \cup \chi^-} \max\left(0, t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1)\right) \tag{6.1}$$

qui n'est rien d'autre que la fonction de coût Hinge Loss. En pratique, nous utilisons une version légèrement différente de la Hinge Loss classique, comme proposée par [114] – la fonction standard étant $L_d = \sum_{\mathbf{x} \in \chi^+ \cup \chi^-} \max(0, 1 - t(\mathbf{x}) \cdot e(\mathbf{x}))$ – plus adaptée à notre problème, étant donné que les erreurs de reconstruction sont toutes positives. Ces fonctions sont illustrées en Fig. 6.2. En un sens, notre problème se rapproche plus des algorithmes d'apprentissage de métrique que de ceux de classification. Quand le minimum est atteint, les exemples positifs (resp. négatifs) ont une erreur de reconstruction plus petite (resp. plus grande) que 1.

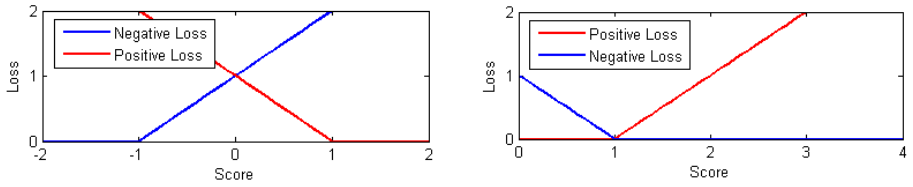


FIGURE 6.2 – Hinge Loss. Haut : Hinge loss standard. Bas : Hinge loss pour l'apprentissage de métrique.

6.1.2 Apprentissage

Pour optimiser la fonction de coût, nous utilisons une rétro-propagation de l'erreur. Les équations sont données pour une seule couche cachée, cependant, elles sont valables pour N couches. Repartant des équations de l'autoencodeur, nous avons $\tilde{\mathbf{x}} = \mathbf{y} = h(W_{10}\mathbf{z})$ et $\mathbf{z} = k(W_{01}\mathbf{x})$. Comme proposé par [68], nous prenons comme fonction k la fonction identité et une sigmoïde pour h. k étant l'identité, la transformation apprise de l'espace latent vers l'espace des descripteurs est linéaire. k est introduit pour conserver la généralité des équations. Pour simplifier les notations, notons $\mathbf{u}$ et $\mathbf{v}$ tels que : $\mathbf{u} = W_{10}\mathbf{z}$ et $\mathbf{v} = W_{01}\mathbf{x}$. L'objectif alors d'estimer les coefficients w_{ki} de W_{01} et W_{10} en minimisant $L(\chi^+ \cup \chi)$.

La valeur optimale des w_{ki} vérifie : $\frac{\partial L}{\partial w_{ki}} = 0$, qui peut être résolue par une descente de gradient. Les dérivées partielles peuvent s'écrire :

$$\frac{\partial L}{\partial w_{ki}} = \frac{\partial L}{\partial \mathbf{e}^i} \cdot \frac{\partial \mathbf{e}^i}{\partial \mathbf{y}^i} \cdot \frac{\partial \mathbf{y}^i}{\partial \mathbf{u}^i} \cdot \frac{\partial \mathbf{u}^i}{\partial w_{ki}} \tag{6.2}$$

avec :

$$\frac{\partial \mathbf{e}^i}{\partial \mathbf{y}^i} = -1; \quad \frac{\partial \mathbf{y}^i}{\partial \mathbf{u}^i} = \frac{\partial h(\mathbf{u}^i)}{\partial \mathbf{u}^i}; \quad \frac{\partial \mathbf{u}^i}{\partial w_{ki}} = z^k \tag{6.3}$$

De plus, dans le cas d'une fonction sigmoïde, $\frac{\partial h(\mathbf{u}^i)}{\partial \mathbf{u}^i} = \mathbf{y}^i \cdot (1 - \mathbf{y}^i)$. Jusqu'ici, les équations sont identiques à la rétro-propagation classique d'erreur. Ensuite, nous introduisons la

fonction hinge loss avec :

$$\frac{\partial L}{\partial \mathbf{e}^i} = \begin{cases} \mathbf{e}^i & \text{si } t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1) > 0 \\ 0 & \text{sinon} \end{cases} \tag{6.4}$$

Nous obtenons la descente de gradient suivante : $\Delta w_{ki} = -\eta \delta_i \mathbf{z}^k$, avec $\delta_i = \mathbf{e}^i \frac{\partial h(\mathbf{u}^i)}{\partial \mathbf{u}^i}$ si $t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1) > 0$ et 0 sinon.

Pour la couche cachée :

$$\frac{\partial L}{\partial w_{lk}} = \frac{\partial L}{\partial \mathbf{z}^k} \cdot \frac{\partial \mathbf{z}^k}{\partial \mathbf{v}^l} \cdot \frac{\partial \mathbf{v}^l}{\partial w_{lk}} \tag{6.5}$$

Les deux derniers termes ne changent pas, cependant le troisième devient :

$$\frac{\partial L}{\partial \mathbf{z}^k} = \sum_n e^n \frac{\partial e^n}{\partial \mathbf{z}^k} \text{ si } t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1) > 0$$

ce qui nous donne :

$$\frac{\partial L}{\partial \mathbf{z}^k} = \sum_n e^n \frac{\partial (\mathbf{x}^k - h(\mathbf{u}^n))}{\partial \mathbf{u}^n} \cdot \frac{\partial \mathbf{u}^n}{\partial \mathbf{z}^k}$$

$$= -\sum_n e^n \frac{\partial (h(\mathbf{u}^n))}{\partial \mathbf{u}^n} w_{kn} = -\sum_n \delta(n) w_{kn} \tag{6.6}$$

L'incrément devient $\Delta w_{lk} = -\eta \delta_k \mathbf{x}^l$ avec $\delta_k = \frac{\partial h(\mathbf{v}^k)}{\partial \mathbf{v}^k} \sum_n \delta(n) w_{kn}$ si $t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1) > 0$ et 0 sinon. Pour les deux incréments, η est le pas d'apprentissage. Différentes manières existent pour l'optimiser. Le lecteur intéressé pourra se référer à [97] pour plus d'informations.

Ces équations peuvent être rendues plus robustes par l'ajout d'une marge w à l'équation $t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1) > 0$, qui devient $t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1) + w > 0$. w est déterminée par validation croisée. Cette marge évite que l'algorithme d'optimisation s'arrête dès que les données d'entraînement sont parfaitement séparées et permet d'obtenir de la robustesse.

L'échelle des données est également à considérer. Le choix de la constante arbitraire 1 dans l'équation 6.1 ne porte pas à conséquence théoriquement. Cependant, il peut être judicieux d'utiliser une constante autre, plus adaptée à la dynamique des réseaux de neurones. En effet leurs sorties sont bornées entre -1 et 1, ainsi avoir une erreur de reconstruction supérieure à 1 pour les négatifs et inférieure à 1 pour les positifs ne semble pas indiqué. En pratique, nous avons utilisé une valeur de 0.2.

Enfin, l'utilisation de la Hinge Loss fait apparaitre une discontinuité dans la fonction de coût. En pratique, il peut être plus efficace d'utiliser un lissage de cette fonction, en utilisant la fonction logistique généralisée [180] plutôt que la Hinge Loss :

$$L_d(\chi^+ \cup \chi^-) = \sum_{\mathbf{x} \in \chi^+ \cup \chi^-} l_\beta(t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1)) \tag{6.7}$$

avec $l_\beta = \frac{1}{\beta} log(1 + e^{\beta \mathbf{x}})$. Différentes stratégies d'optimisation existent, consistant à faire varier β. Pour nos applications pratiques, nous avons fixé $\beta = 1000$.

Finalement, $e(x_{new})$ l'erreur de reconstruction pour un nouveau vecteur x_{new} peut être utilisée directement comme score de classification.

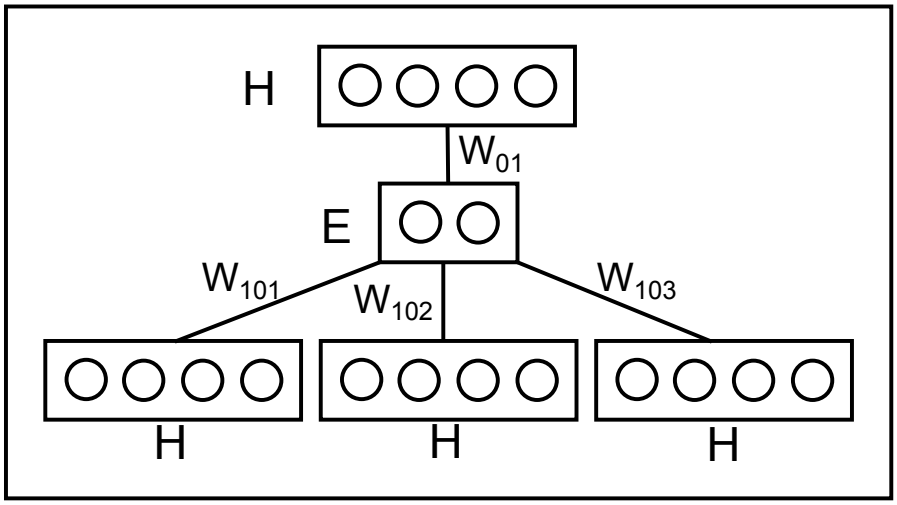


FIGURE 6.3 – Autoencodeur discriminant multi-classes, illustration pour 3 classes.

6.1.3 Extension 3 classes et plus

Afin d'être utilisé dans des conditions de classification plus avancées, nous avons également mis en place une architecture multi-classes, que nous désignerons par *autoencodeur discriminant multi-classes* (AEDMC). Celle-ci est illustrée Fig. 6.3. Le principe est le même que l'autoencodeur discriminant proposé précédemment, cependant, au lieu d'avoir un espace de sortie égal à l'espace d'entrée, l'espace de sortie a une dimension de $C \times n$, où C est le nombre de classe et n la dimension de l'espace d'entrée.

Soit $\chi = \bigcup \chi_C$ l'ensemble regroupant toutes les instances de toutes les classes et $t(\mathbf{x}) \in \{-1, 1\}^C$ le label de $\mathbf{x}$. Nous avons $t^j(\mathbf{x}) = 1$ si la classe de $\mathbf{x}$ est j, -1 sinon. En reprenant les notations précédentes, nous introduisons la fonction f qui correspond à la première couche d'entrée, telle que

$$f(\mathbf{x}) = k(W_{01}\mathbf{x}) \tag{6.8}$$

puis les fonctions g^j telles que :

$$g^j(\mathbf{z}) = h(W_{10j}\mathbf{z}) \tag{6.9}$$

avec W_{10j} comme sur la Fig. 6.3. Nous définissons alors $\tilde{\mathbf{x}}^{\mathbf{j}}$ tel que :

$$\tilde{\mathbf{x}}^{\mathbf{j}} = g^j \circ f(\mathbf{x}) \tag{6.10}$$

Enfin, nous définissons $e^j(\mathbf{x})$ tel que :

$$e^j(\mathbf{x}) = ||\mathbf{x} - \tilde{\mathbf{x}}^{\mathbf{j}}|| \tag{6.11}$$

La fonction à minimiser est alors la suivante :

$$L(\chi) = \sum_{\chi} \sum_{j} max(0, t^j\mathbf{x}(e^j(\mathbf{x} - 1)) \tag{6.12}$$

De même que précédemment, cette fonction est optimisée par descente de gradient. Il est possible de l'adapter avec la fonction logistique généralisée, d'ajouter une marge et d'utiliser un paramètre d'échelle de 0.2.

6.2 Chaînes algorithmiques

Dans cette section, nous présentons les différentes chaînes algorithmiques qui ont été comparées en ce qui concerne la tâche de détection, puis les expériences réalisées dans le cadre de la Reconnaissance.

6.2.1 Détection

La chaîne de détection effectuée est une cascade à deux étages. Ainsi, le premier étage est constitué de 12 SVM linéaires (2 orientations $\times$ 6 ratios largeur/longueur), tandis que le second étage affine les scores de détection en utilisant 12 autoencodeurs, chacun apparié avec le SVM correspondant.

Lors de la phase d'apprentissage, les détecteurs du premier étage utilisent toutes les données disponibles (l'ensemble des fenêtres des pyramides d'images), tandis que ceux du second étage n'utilisent que les négatifs difficiles obtenus après ce premier apprentissage. Afin de voir l'apport d'un autoencodeur *discriminant*, nous avons testé ce second étage avec un autoencodeur classique et avec un autoencodeur discriminant. En pratique, les négatifs difficiles sont conservés si leur score est supérieur à -1.0.

Durant la phase de test, les 12 SVM parcourent l'image. Seules les fenêtres obtenant un score supérieur à -1.0 (le même seuil que celui pour l'apprentissage) sont gardées. Ensuite, les fenêtres sélectionnées sont reclassées avec l'autoencodeur (standard ou discriminant). Enfin, comme habituellement, une étape de filtrage des non-maxima est effectuée, en utilisant le même algorithme glouton que dans les différentes expériences présentées dans les chapitres précédents.

Notre algorithme peut prendre en entrée n'importe quel descripteur. Afin d'illustrer ce fait, nous avons effectué les expériences avec les descripteurs HOG et les descripteurs LBP. Ces descripteurs sont assez rapides à calculer mais permettent d'obtenir des performances très bonnes, comme expliqué Chap. 4.

Deux variantes de notations des fenêtres sont possibles : conserver uniquement le score du deuxième étage de la cascade ou bien combiner les scores successifs par une fusion. Nous avons combiné les deux scores issus des deux étapes de la cascade. Dans ce cas, le résultat final est obtenu par combinaison linéaire avec $\alpha S_{AE}(\mathbf{x}) + (1-\alpha) S_{SVM}(\mathbf{x})$, où α est fixé par validation croisée. De même et comme précédemment, le nombre de neurones est fixé par des expériences préliminaires, par validation croisée.

Les deux architectures utilisées sont schématisées Fig. 6.4.

6.2.2 Reconnaissance

Pour les expériences de classification, nous avons testé deux approches. La première consiste à apprendre un autoencodeur discriminant pour chaque classe, en prenant comme ensemble négatif l'union des autres classes. La seconde approche utilise l'extension de l'autoencodeur discriminant, en effectuant une optimisation multi-classes. Ces expériences sont ensuite comparées aux différents résultats obtenus dans les chapitres précédents.

6.3 Résultats

Dans cette section, nous présentons quelques résultats préliminaires sur les autoencodeurs discriminants, puis les résultats obtenus en détection. Enfin, nous présentons les

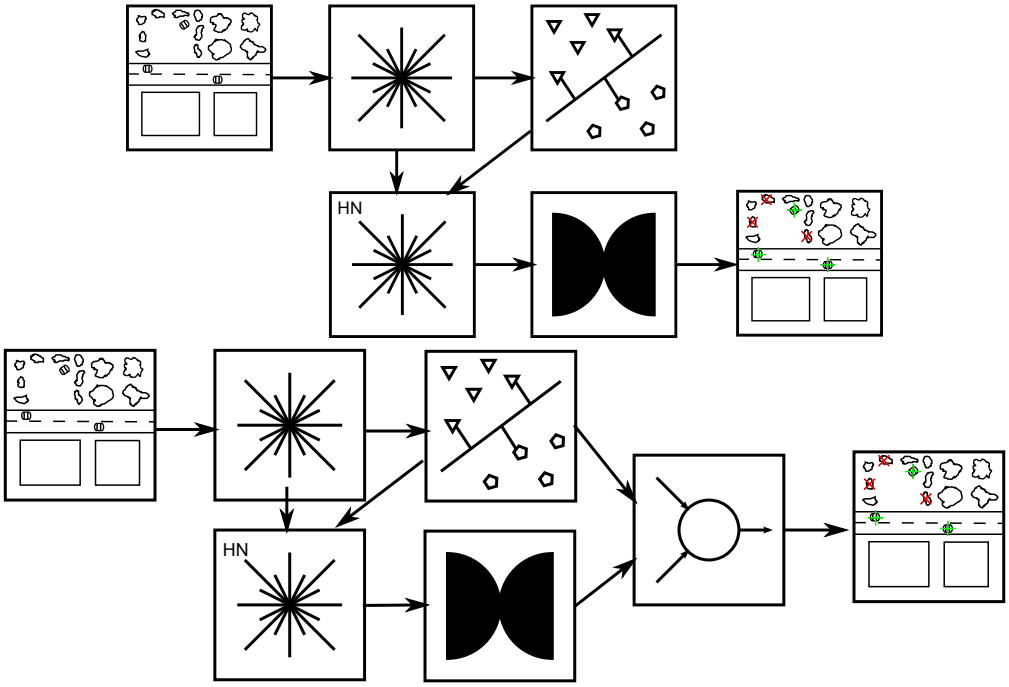


FIGURE 6.4 – Deux chaînes algorithmiques basées sur des variétés. En haut : sans fusion, en bas : avec fusion.

résultats obtenus en Reconnaissance. L'ensemble des expériences a été effecuté sur la base VeDAI-PII. Pour la détection, la classe voiture a été étudiée.

6.3.1 Étude paramétrique

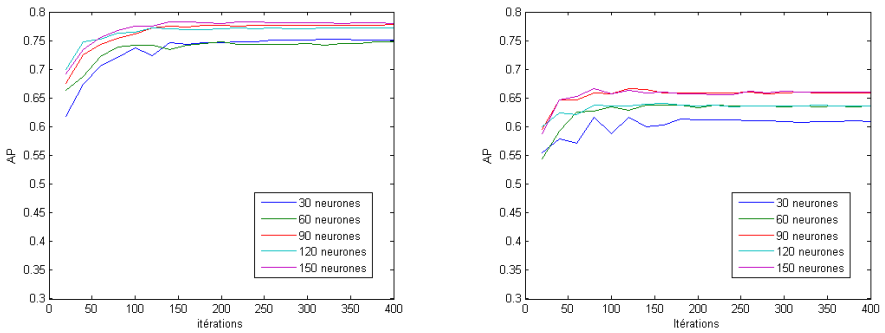


FIGURE 6.5 – AP en fonction du nombre de neurones et du nombre d'itérations, à gauche, sur un sous ensemble ayant fait partie de l'apprentissage, à droite, sur un sous ensemble disjoint.

Le paramètre le plus important, pour tout algorithme basé sur les variétés, est le choix de la dimension de la variété. Nous avons choisi de fixer cette dimension par validation croisée.

En ce qui concerne la détection, il est visible sur la Fig. 6.5 que la performance change en fonction du nombre de neurones. Il est à voir que le nombre de neurones optimal sur

la validation est en accord avec le nombre de neurones optimal sur l'apprentissage. Les performances varient en fonction de l'optimum local dans lequel aura convergé le réseau de neurones, mais, du moment que suffisamment de neurones sont utilisés, de bonnes performances sont atteintes. À l'inverse, un nombre de neurones insuffisant (30 sur la figure) dégrade fortement les performances.

Pour la Reconnaissance, nous observons de même une grande stabilité des performances en fonction du nombre de neurones du moment qu'il y en a suffisamment, comme illustré en Fig. 6.6. Cette stabilité est valable tant pour l'autoencodeur discriminant à deux classes que pour son extension multi-classes.

Pour ce qui est du nombre d'itérations et de la convergence en général, que ce soit pour la Détection (Fig. 6.5) ou la Reconnaissance (Fig. 6.6), nous observons une courbe de croissance classique lors de l'apprentissage de réseaux de neurones, avec un fort gain au départ, qui se ralentit petit à petit. Les équations d'optimisation utilisées (moment, pas dégressif...), permettent de maintenir un plateau de performance suffisamment large pour que la sélection du nombre maximal d'itérations n'ait pas d'influence sur les performances.

La Fig. 6.8 montre que le micro et le macro averaging augmentent avec le nombre d'itérations. De plus, pour certaines classes, les performances ont tendance à décroître avant de se stabiliser, tandis que pour d'autres elles augmentent. Ce comportement est tout à fait prévisible. En effet, au début de l'optimisation, certaines classes ont des scores plus élevés que d'autres, de par l'initialisation aléatoire des poids du réseau. Ces classes agissent comme classe par défaut et ont donc un score de bonne reconnaissance très élevé (Dans un cas extrême, si l'ensemble une classe répond toujours plus fort que les autres, son taux de reconnaissance est de 100%). Cependant, cette représentation erronée amène les autres classes à porter beaucoup de leurs fausses reconnaissaces sur cette classe.

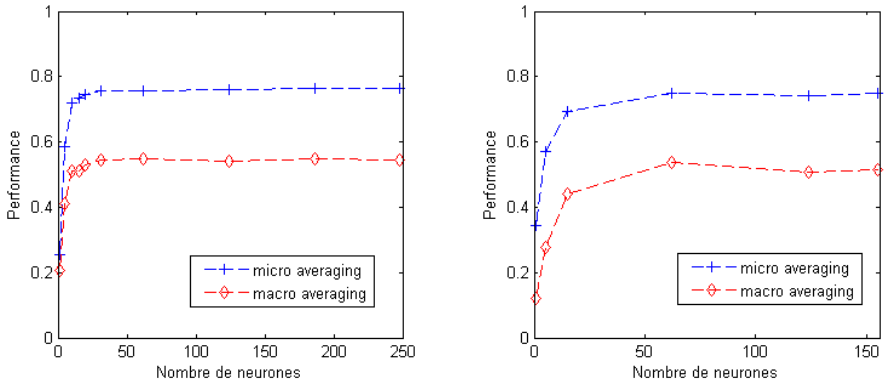


FIGURE 6.6 – Robustesse du micro et macro averaging en fonction du nombre de neurones. À gauche, pour 8 autoencodeurs discriminants, à droite, pour 1 autoencodeur multi-classes.

Lors de la fusion, un paramètre d'équilibrage, α, est utilisé pour balancer les différents algorithmes. Ce paramètre a été choisi par validation croisée. Ainsi, la Fig. 6.7 montre l'influence de ce paramètre sur les performances de test. Au mieux, il est possible de gagner 2 points de mAP. La fusion n'apporte donc que peu de performance.

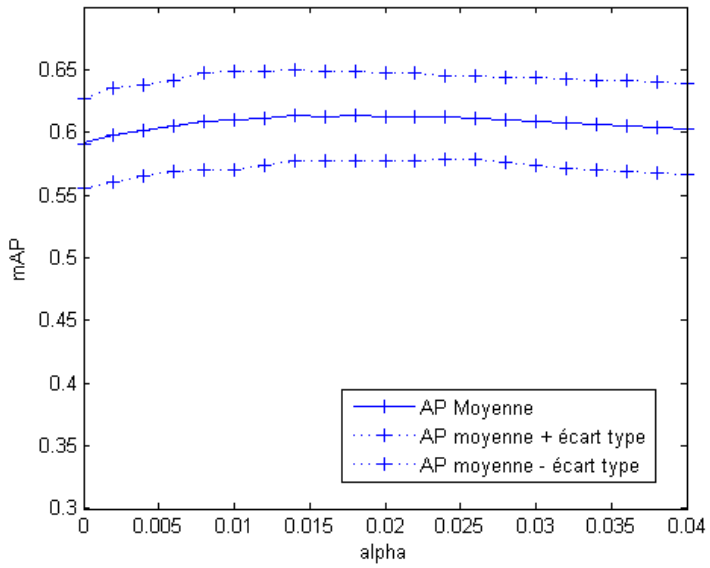


FIGURE 6.7 – Variation des performances en fonction du paramètre alpha lors de la fusion.

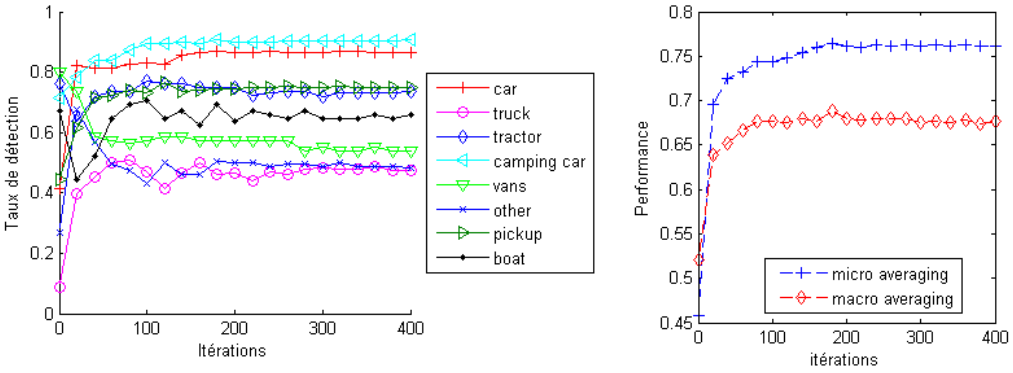


FIGURE 6.8 – À gauche : taux de reconnaissance en fonction de la classe et du nombre d'itérations. À droite : micro et macro averaging sur l'ensemble des classes.

6.3.2 Résultats quantitatifs en Détection

Les résultats obtenus sur VeDAI sont donnés en Tab. 6.1. Premièrement, nous avons observé que l'autoencodeur standard (n'utilisant que des exemples positifs) ne donne pas de bons résultats. Même combiné au premier étage de la cascade, il n'améliore pas significativement les performances du SVM. Ceci confirme que l'autoencodeur seul n'est pas capable de discriminer correctement et il ne permet pas d'apporter une information complémentaire au SVM.

Á l'inverse, pour l'autoencodeur discriminant, le gain est significatif pour les deux descripteurs, montrant que cette technique apporte en performance quel que puisse être l'espace des représentations. En effet, l'autoencodeur discriminant donne de bien meilleurs

Détecteur	mAP		Rappel @ 0.01 FPPI		Rappel @ 0.1 FPPI		Rappel @ 1 FPPI	
DPM [40]	60.5±4.2		13.4±6.8		31.4±5.8		74.5±4.5	
Descripteurs	HOG	LBP	HOG	LBP	HOG	LBP	HOG	LBP
SVM (1er étage)	58.9±3.5	52.9±3.5	13.2±5.1	10.5±6.3	30.4±3.9	30.8±3.9	72.1±4.1	63.2±5.0
SVM puis AE	30.0±3.9	34.2±4.4	1.5 ±1.6	3.6±1.6	6.8 ±1.8	13.3±3.9	39.5±4.1	43.2±7.1
SVM fusion AE	58.8±3.8	55.5±4.0	12.9±3.5	13.4±4.7	34.0±4.5	32.1±3.1	71.8±5.4	66.7±5.0
SVM puis AED	68.0±4.2	59.2±3.6	**21.2±6.9**	14.5±4.5	46.7±6.8	39.2±4.0	78.7±3.4	70.8±4.3
SVM fusion AED	**69.6±3.4**	**60.0±3.7**	20.4 ±6.2	**17.3±6.5**	**49.0±3.6**	**40.0±3.5**	**80.3±3.1**	**72.0±5.3**

TABLE 6.1 – Comparaison de différentes chaînes algorithmiques, résultats sur la base VeDAI-PII, pour la classe voiture. (Deformable Part Model, Autoencodeur, Autoencodeur Discriminant)

résultats que l'autoencodeur classique (+38.0 de mAP avec HOG, +25.0 avec LBP) et est significativement meilleur que le SVM (+9.1 de mAP avec HOG et +6.3 avec LBP), alors qu'il utilise les mêmes exemples d'entrainement. Pour les différents points de fonctionnement, l'autoencodeur discriminant gagne en performance. Le grand écart-type pour 0.01 FPPI rend la conclusion non fiable pour ce point, mais le gain est important pour les deux descripteurs.

La combinaison du score de l'autoencodeur discriminant et du score du SVM donne des résultats légèrement meilleurs que l'autoencodeur discriminant seul, montrant ainsi que l'autoencodeur a conservé la majorité de l'information du premier étage de la cascade.

Enfin, lorsque nous comparons notre approche avec le Deformable Part Model de [40] (en utilisant la version 5 et en désactivant l'apprentissage des parties du fait de la petite taille des objets), le gain est d'environ 10% de mAP également. Ces résultats montrent clairement que non seulement les autoencodeurs discriminants donnent de meilleurs résultats que les autoencodeurs classiques, mais qu'ils permettent d'améliorer significativement les performances d'un SVM ou du DPM dans une tâche de détection de petits objets.

6.3.3 Résultats quantitatifs en Reconnaissance

Les résultats sont présentés Tab. 6.2. De même qu'en détection, l'autoencodeur discriminant fait significativement mieux qu'un autoencodeur classique. Nous retrouvons ainsi ce qui avait été montré en détection.

De plus, les résultats sont similaires à un SVM RBF en terme de classification. Il est à noter que, si les résultats sont similaires, un SVM RBF a un nombre d'opérations non-linéaires en $O(N)$, avec N le nombre de vecteurs support (environ 17 000 dans notre cas), tandis que l'autoencodeur discriminant a une complexité en $O(n)$, n étant la dimension de l'espace des descripteurs, aux alentours de 400. De plus, un SVM RBF a une complexité d'apprentissage en $O(n_e^2)$, avec n_e le nombre d'exemples d'apprentissage disponibles, tandis que l'autoencodeur discriminant a une complexité d'apprentissage en $O(n_e)$. L'autoencodeur discriminant permet donc d'obtenir des résultats similaires à un SVM RBF, mais avec un temps d'apprentissage et de test plus petit. Il peut également traiter plus de données d'apprentissage.

Ensuite, l'utilisation d'un autoencodeur discriminant multi-classes fonctionne aussi bien que 8 autoencodeurs classiques. Les performances sont à peine dégradées, cependant, le nombre d'opérations est quasiment divisé par deux, permettant de gagner à la fois sur le temps d'apprentissage et sur le temps de test.

Le Tab. 6.3 présente les temps d'exécution en Matlab des trois algorithmes utilisés. Même si ces temps ne sont pas représentatifs dans l'absolu, nous retrouvons le fait que

Méthode	Boa	Cam	Car	Oth	Pic
8 SVM linéaires	21.4±18.3	83.9±4.6	82.8±5.6	24.2±7.6	53.0±8.4
8 AE	31.9±11.6	74.8±5.7	77.8±3.4	37.1±12.6	71.0±2.6
1 SVM MC RBF	56.1±13.4	83.7±7.7	84.6±3.9	47.8±6.5	74.5±1.9
8 SVM RBF	61.9±15.4	86.4±6.9	86.7±3.9	39.8±6.2	77.4±3.5
8 AED	58.3±12.4	87.9±6.4	86.0±3.6	41.6±5.9	76.7±2.0
1 AEDMC	50.4±10.1	87.0±5.5	86.7±4.7	38.9±6.5	74.1±2.8
8 AED + 1 SVM RBF	57.7±13.1	86.8±6.5	86.5±3.8	47.1±7.4	76.6±2.6
Méthode	Tra	Tru	Van	Macro	Micro
8 SVM linéaires	48.6±8.2	11.9±7.2	27.1±8.0	44.1±3.4	60.3±3.2
8 AE	50.1±6.4	35.0±6.7	14.0±13.4	49.0±4.3	64.7±5.7
1 SVM MC RBF	68.4±11.6	57.7±7.8	60.9±12.0	66.8±3.3	75.1±3.2
8 SVM RBF	71.6±10.8	56.4±9.5	57.3±13.5	67.2±2.2	76.4±2.0
8 AED	69.2±8.6	53.5±9.6	57.4±11.6	66.3±5.8	75.7±6.9
1 AEDMC	67.9±8.2	52.3±9.0	48.5±10.0	63.2±2.3	74.4±2.4
8 AED + 1 SVM RBF	71.9±10.7	57.9±9.1	60.7±12.1	**68.1±3.0**	**76.7±2.7**

TABLE 6.2 – Résultats de Reconnaissance comparés entre AE, AED, AEDMC et différents SVM.

Méthode	SVM RBF	8 AED	1 AEDMC
Temps d'exécution (ms)	450.0	0.45	0.30

TABLE 6.3 – Comparaison des temps d'exécution en Matlab, sur la même machine, pour 3 algorithmes différents. Le SVM RBF est l'implémentation proposée par libsvm.

le SVM RBF est lent, tandis que l'autoencodeur multi-classes est à peu près 1.5 fois plus rapide que les 8 autoencodeurs discriminants.

Afin de voir si l'information encodée par un AED et par un SVM RBF était la même, nous avons effectué une expérience de fusion entre les deux classifieurs. Il apparait alors que les performances sont augmentées en ce qui concerne le macro et le micro averaging, de 1 ou 2% en comparaison avec la meilleure performance précédente. Nous pouvons donc en déduire que, si les performances sont similaires, la frontière générée est toutefois différente et apporte une information supplémentaire utile.

6.4 Conclusions

Dans ce chapitre, nous avons présenté un nouveau classifieur, l'autoencodeur discriminant. Ce classifieur se base sur une architecture de réseau de neurones et plus particulièrement sur celle d'un autoencodeur classique. Cependant, sa fonction d'optimisation est différente et se fonde sur une approche d'apprentissage de métriques. Nous avons développé les équations de la version de classifieur bi-classes ainsi que son extension multi-classes. Finalement, nous avons montré par différentes expériences que les autoencodeurs discriminants permettent d'améliorer les résultats d'algorithmes de Détection, d'égaler un SVM non linéaire en Reconnaissance, tout en étant plus rapide et en permettant un apprentissage sur un grand nombre de données.

Ainsi, nous avons pu développer un classifieur qui s'adapte bien au contexte de la

Détection et Reconnaissance de véhicules faiblement résolus. Cependant, ce classifieur reste tributaire de l'espace de représentation donné en entrée de part les descripteurs choisis. Ces descripteurs, bien qu'obtenant de bonnes performances, ont été conçus empiriquement sans aucun critère d'optimalité quant aux performances de détection. Dans le chapitre suivant, nous présentons des algorithmes permettant d'apprendre des descripteurs en utilisant le formalisme des autoencodeurs discriminants.

Chapitre 7

Réseaux convolutionels discriminants

Dans les chapitres précédents, nous avons majoritairement travaillé sur les classifieurs. Cependant, ainsi qu'expliqué dans le Chap. 4, l'utilisation de bons descripteurs est une des clefs de la réussite d'algorithmes de Détection et Reconnaissance. Dans ce chapitre, nous présentons quelques bases sur les réseaux convolutionels et leur théorie. Dans un second temps, nous introduisons le concept d'autoencodeur convolutionel discriminant. Enfin, nous présentons différentes utilisations possibles de tels autoencodeurs en tant que nouveau classifieur ou en tant que générateur de descripteurs, ainsi que les résultats et interprétations en découlant dans le cadre de la détection.

7.1 Réseaux convolutionels

Dans cette section, nous présentons les bases des réseaux convolutionels, suivies de quelques définitions et équations nécessaires à leur utilisation.

7.1.1 Concept

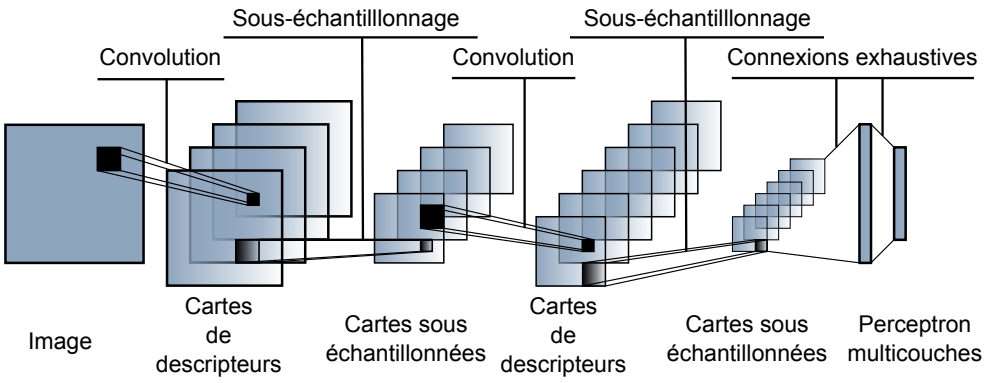


FIGURE 7.1 – Réseau convolutionel. L'architecture présentée ici est Lenet-5, l'une des premières architectures de réseaux convolutionels [96].

Les réseaux convolutionels (abrégés CNN pour Convolutional Neural Network) [96] sont une technique connue pour être efficace dans des tâches de classification [150] et,

plus récemment, de Détection [90, 124]. Comme évoqué dans le Chap. 2, les méthodes par réseaux convolutionels profonds donnent dans des travaux récents les meilleures performances dans de nombreux domaines. Nous rappelons que les travaux effectués sont effectués sur des images dans lesquelles les objets sont de taille importante (221x221 pixels pour la fenêtre englobante) et qu'en conséquence, l'utilisation d'une architecture usuelle de réseau convolutionel telle quelle ne peut s'appliquer de manière réaliste à nos images. En effet, un véhicule étant approximativement contenu dans une fenêtre 40x40 pixels, un sur-échantillonnage d'un facteur 5 devrait être effectué. Cependant, vu les gains de performances, il nous a paru pertinent d'utiliser le formalisme des CNN afin d'apprendre de nouveaux descripteurs.

Les CNN sont une catégorie de réseau de neurones différente des autoencodeurs. Ils apprennent directement depuis les niveaux de gris de l'image, en faisant l'hypothèse que les pixels ont des dépendances locales. L'architecture classique d'un CNN est une suite de couches de convolution intercalées par des couches de sous-échantillonnage (*pooling* en anglais). Les résultats d'une couche de pooling sont passés par une fonction d'activation classique (voir Chap. 5), puis servent d'entrées à la couche de convolution suivante. Enfin, pour les deux dernières couches, les neurones sont interconnectés simplement de la même manière qu'un perceptron multi-couches. La Fig. 7.1 illustre une telle architecture. Les premières couches servent à fabriquer des descripteurs efficaces, tandis que les couches finales sont utilisées pour la classification. Les convolutions permettent de conserver les dépendances locales, tandis que les étapes de sous-échantillonnage condensent l'information, tout en permettant l'invariance à des petites transformations. Finalement, un perceptron classique utilise ces descripteurs intermédiaires efficaces pour attribuer un label à l'entrée.

De tels réseaux sont conçus afin d'apprendre des descripteurs efficaces en même temps que les classifieurs, cependant, les approches récentes tendent à montrer que l'utilisation des descripteurs combinés avec des classifieurs donne de bons résultats en soi [90]. Les paramètres à apprendre sont les noyaux de convolution ainsi que les poids entre les couches. Le nombre de noyaux, le nombre de couches, le nombre de neurones sont autant de paramètres à ajuster. La règle d'apprentissage est la rétro-propagation de l'erreur.

7.1.2 Théorie et définitions

Nous donnons ici quelques définitions et équations de base nécessaires à la compréhension d'un CNN et introduisons brièvement les autoencodeurs convolutionels.

Cartes de descripteurs

La sortie d'une couche de convolution est appelée *carte de descripteurs*. Les filtres appris lors de cette étape permettent de capturer les dépendances locales des pixels. La k-ième carte $M_k(I)$ calculée par le noyau de convolution W_k sur l'entrée I est donnée par :

$$M_k(I) = h(I * W_k + b_k) \tag{7.1}$$

avec h une fonction d'activation classique comme la sigmoïde ou la tangente hyperbolique. b_k est un biais. W_k et b_k sont appris par rétro-propagation de l'erreur. $*$ est l'opérateur convolution. La propagation de l'erreur s'effectue ainsi :

$$E_k = \hat{W} * E_{k+1} \tag{7.2}$$

avec E_{k+1} l'erreur propagée précédemment, E_k l'erreur à propager, $\hat{W}$ étant la double transposée du noyau de convolution W, tandis que pour la mise à jour des poids, nous avons :

$$dW = M_{k+1} * E_{k+1} \tag{7.3}$$

avec dW la mise à jour du noyau W, E_{k+1} l'erreur propagée et M_{k+1} la carte de descripteurs issue de W.

Ré-échantillonnage

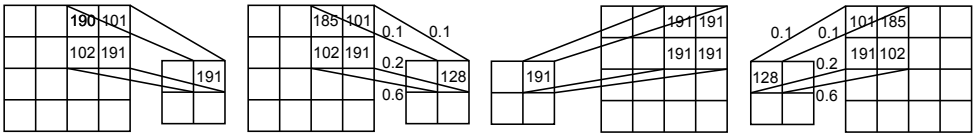


FIGURE 7.2 – Étapes de sous-échantillonnage (gauche) et sur-échantillonnage (droite), correspondant respectivement aux techniques de max-pooling (1er et 3ème) et de somme pondérée (2ème et 4ème).

Les étapes de ré-échantillonnage sont illustrées Fig. 7.2.

Ainsi que décrit précédemment, une étape de sous-échantillonnage est effectuée sur ces cartes de descripteurs. Habituellement deux méthodes sont utilisées : le max-pooling ou la somme pondérée. Le max-pooling consiste à transférer en entrée de la couche suivante les valeurs des maxima locaux, pour un voisinage donné. Aucun paramètre ne doit être appris. La somme pondérée consiste à transférer la moyenne pondérée dans un voisinage donnée. Dans ce cas, les coefficients de la somme pondérée doivent être appris. Par la suite, nous noterons les opérations de sous-échantillonnage ds, ce qui nous donne :

$$\hat{M}k(I) = ds(M_k(I)) \tag{7.4}$$

avec $\hat{M}k(I)$ la carte de descripteur $M_k(I)$ sous-échantillonnée. Cette étape permet d'augmenter la robustesse vis-à-vis de légères translations de l'image à tester. La propagation de l'erreur devient pour le max-pooling, pour un point (i, j) de E_k correspondant au voisinage V :

$$E_k(i_{max}, j_{max}) = E_{k+1}(i, j) \; sinon \; E_k = 0 \tag{7.5}$$

avec i_{max} et j_{max} les indices du maximum sur V lors de l'étape de calcul de la sortie du réseau.

Sur-échantillonnage

Parallèlement au sous-échantillonnage, le sur-échantillonnage s'obtient ainsi :

$$\hat{M}k(I) = us(M_k(I)) \tag{7.6}$$

La propagation de l'erreur devient :

$$E_k = \sum_V E_{k+1}(V) \tag{7.7}$$

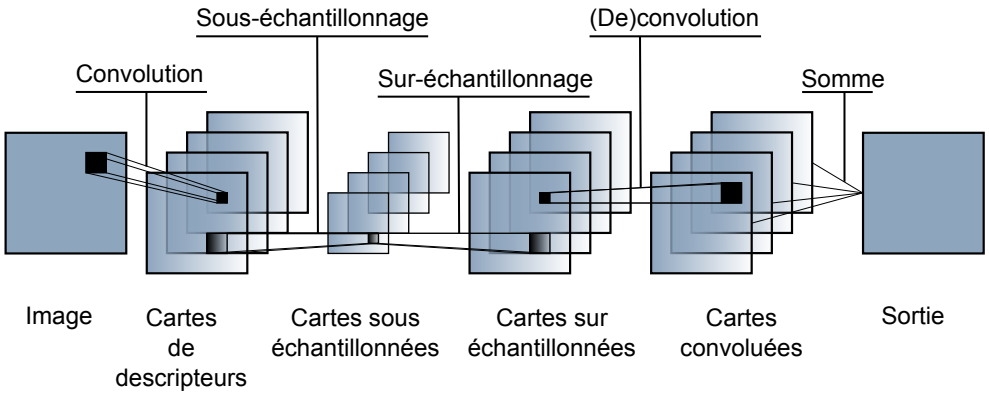


FIGURE 7.3 – Autoencodeur convolutionel, architecture générale.

Autoencodeurs convolutionels

Les *autoencodeurs convolutionels* [113] ont été conçus afin d'initialiser les réseaux convolutionels. Le principe est le même que les autoencodeurs : apprendre l'identité sous des contraintes de parcimonie. Ensuite, une fois le réseau convergé sur un ensemble de noyaux, les noyaux appris sont utilisés pour initialiser un CNN classique. L'équation pour la carte de descripteur est la même que l'équation 7.1. La reconstruction est donc donnée par :

$$R(I) = h(\sum_k \hat{M}k(I) * \tilde{W}_k + c) \tag{7.8}$$

avec $R(I)$ la reconstruction, h la fonction d'activation utilisée pour calculer M_k. $\tilde{W}_k$ est W_k retournée gauche-droite et haut-bas, c est un biais. Dans ce cas, Wk, bk et c sont appris.

7.2 L'autoencodeur convolutionel discriminant

7.2.1 Architecture

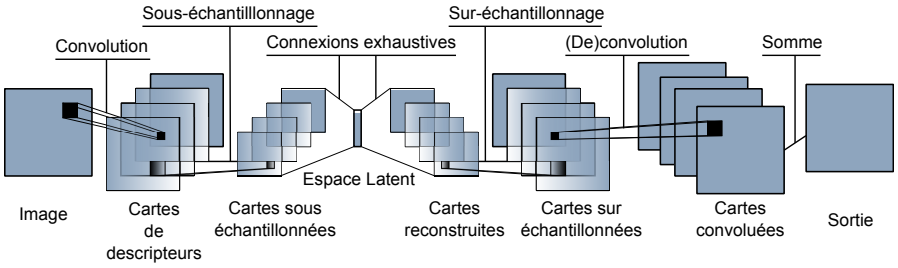


FIGURE 7.4 – Autoencodeur convolutionel discriminant : architecture globale. Un autoencodeur convolutionel discriminant est constitué de couches de convolution et de sous-échantillonnage, suivies d'une couche de projection dans un espace latent. Ensuite, après reconstruction, les cartes sont sur-échantillonnées et convoluées de nouveau. Enfin, les différentes cartes sont fusionnées.

L'architecture présentée ici peut être vue comme une extension des autoencodeurs discriminants présentés au chapitre précédent. En relation avec les CNN, les AED et les AE,

nous introduisons le concept d'*Autoencodeurs Convolutionels Discriminants*, (CODA, pour COnvolutional Discriminant Autoencoder), qui utilise les données de deux classes pour apprendre des descripteurs efficaces.

L'architecture d'un tel réseau est composée d'une couche classique de CNN (convolution + sous-échantillonnage + fonction d'activation), qui construit des descripteurs à partir des niveaux de gris, suivie d'une couche de neurones entièrement connectée. Ensuite, de la même manière qu'un autoencodeur, les deux couches sont reflétées, ainsi nous avons une couche de sur-échantillonnage suivie par une convolution. Les deux blocs sont reliés entre eux par une connexion dense. Le réseau est présenté en Fig. 7.4. Le réseau peut être vu comme un autoencodeur amplifié de deux couches classiques de réseaux convolutionels, une à l'entrée et une à la sortie.

7.2.2 Apprentissage

De même que pour les *autoencodeurs discriminants*, les *autoencodeurs discriminants convolutionels* minimisent la fonction de coût suivante :

$$L_d(\chi^+ \cup \chi^-) = \sum_{\mathbf{x} \in \chi^+ \cup \chi^-} \max\left(0, t(\mathbf{x}) \cdot (e(\mathbf{x}) - 1)\right) \tag{7.9}$$

avec $e(\mathbf{x})$ l'erreur de reconstruction sur les niveaux de gris. Les règles de propagation de l'erreur sont celles présentées dans les sections précédentes. De même que dans les chapitres précédents, il est possible d'utiliser une marge w, ainsi que de lisser cette fonction par la fonction logistique généralisée pour obtenir une convergence plus simple.

7.3 Utilisation des réseaux convolutionels discriminants

7.3.1 Architectures de détection : utilisation comme classifieur

Afin d'effectuer une tâche de détection, nous avons implémenté deux architectures, semblables à celles utilisées dans le chapitre précédent et illustrées en Fig. 7.5, de même que précédemment, les symboles sont individuellement expliqués en Annexe A. Celles-ci sont basées sur une cascade à deux étages. Le premier étage est constitué de 12 SVM, le second étage est constitué de 12 CODA, chacun apparié à un SVM. Dans la phase d'apprentissage, chaque CODA apprend sur les fenêtres en niveau de gris extraites de chaque position des Négatifs Difficiles retenus par le SVM correspondant. Lors de la phase de test, les SVM retournent un ensemble de positions. Sur chacune d'entre elles, les niveaux de gris sont extraits. Enfin, chaque CODA retourne un score pour ces fenêtres. De même que précédemment, il est possible soit d'utiliser le score seul, soit d'utiliser une fusion entre le SVM et le CODA. Les paramètres suivants ont été utilisés pour configurer CODA : les filtres de convolution sont des filtres 5x5, les cartes sont sous-échantillonnées par un max-pooling dans un voisinage 4x4. Une étude sur le nombre de filtres et le nombre de neurones a été effectuée. Par défaut, les expériences effectuées utilisent 16 noyaux et 400 neurones sur la couche cachée, sur des niveaux de gris normalisés. Les SVM utilisent des descripteurs HOG.

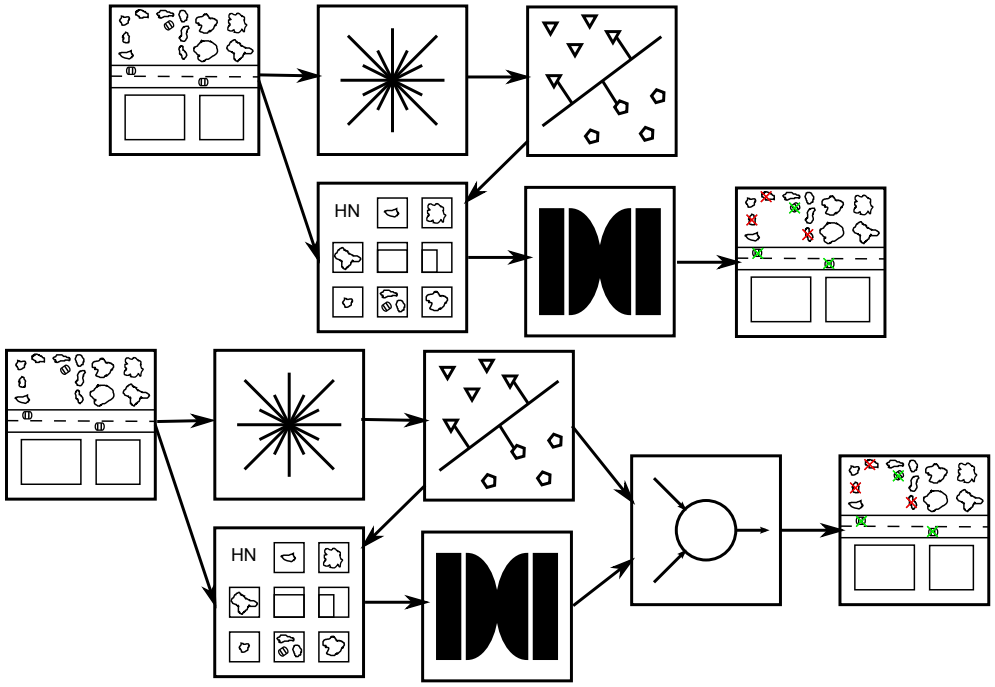


FIGURE 7.5 – Les deux chaînes algorithmiques utilisées contenant des autoencodeurs convolutionels discriminants. Le SVM sélection des fenêtres, qui servent ensuite d'entrée à l'algorithme CODA.

7.3.2 Architecture de détection : utilisation des couches intermédiaires comme descripteurs

Il est également possible d'utiliser les CODA afin d'apprendre de nouveaux descripteurs. En effet, les couches de convolution et de sous-échantillonnage sont une information de plus haut niveau que les niveaux de gris et peuvent par là même servir de primitives pour un algorithme de classification. Ainsi, une fois l'apprentissage terminé, il est possible de ne conserver que les cartes de descripteurs ou leurs projetés dans l'espace latent. Ces expériences permettent également de décorréler l'apprentissage de primitive (plutôt effectuée par la couche de convolution), de l'apprentissage de classifieur (plutôt présent dans l'autoencodeur discriminant central). Nous notons Desc1 le descripteur issu des cartes de descripteurs et Desc2 celui issu de leurs projetés dans l'espace latent. Ces deux descripteurs sont normalisés à une norme unitaire, afin d'être correctement conditionnés pour le SVM. Les deux possibilités sont illustrées en Fig. 7.6.

7.4 Résultats

7.4.1 Résultats en détection : architecture complète

L'évolution de l'AP selon le nombre de neurones est représentée Fig. 7.7, droite. Le nombre de neurones de la couche centrale doit être suffisamment élevé pour ne pas trop compresser l'information. Lorsque trop de neurones sont ajoutés, le temps de calcul devient prohibitif. Cependant, la courbe suggère qu'un nombre de neurones plus grand permettrait de continuer à gagner en performances.

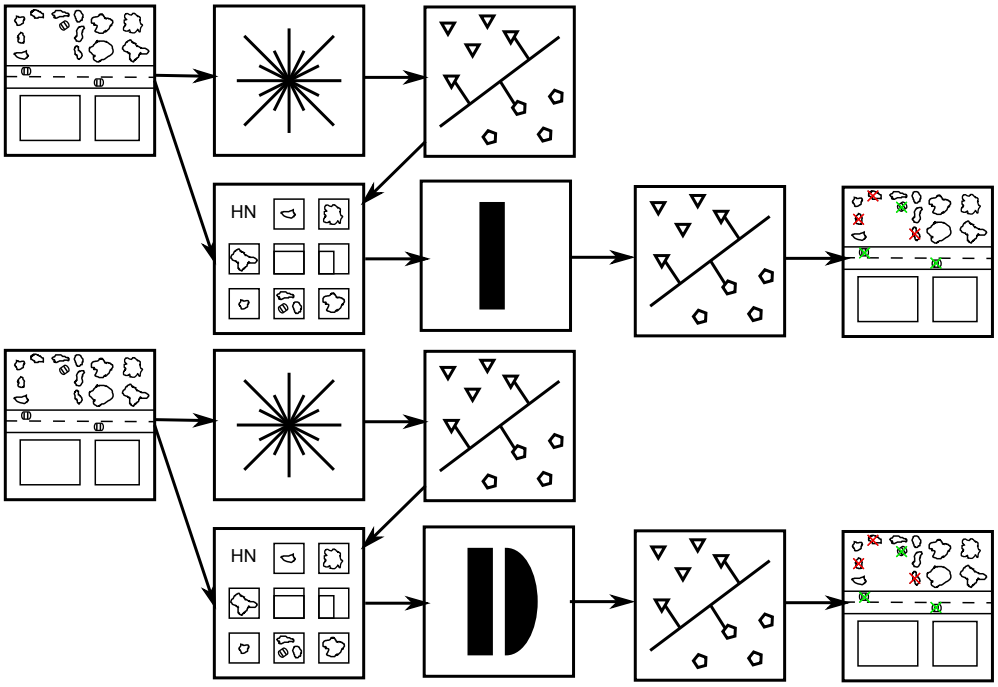


FIGURE 7.6 – Deux architectures utilisant les couches intermédiaires comme descripteurs. En haut : utilisation de la carte de descripteurs (Desc1). En bas : utilisation de la projection de cette carte sur l'espace latent (Desc2).

L'évolution de l'AP selon le nombre de noyaux est représenté en Fig. 7.7, à gauche. De même que pour les noyaux, un nombre de filtres trop faible ne permet pas d'obtenir des performances intéressantes. Il semblerait cependant qu'augmenter le nombre de noyaux permettrait d'augmenter légèrement les performances, mais le plafond semble presque atteint.

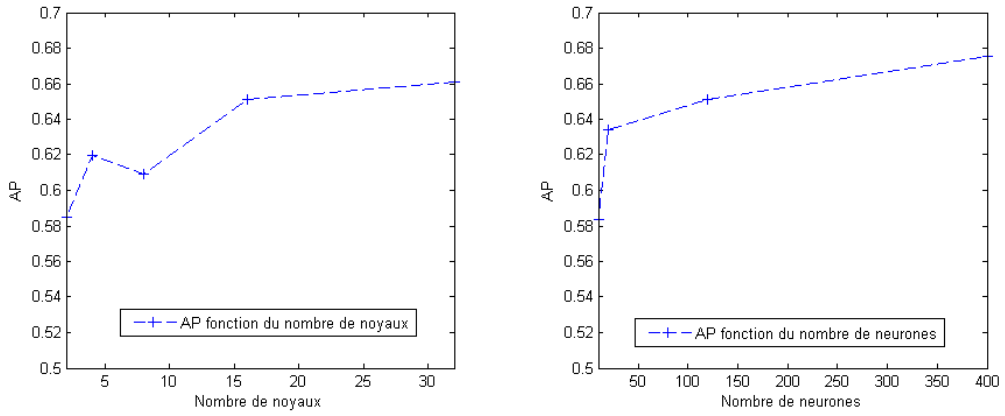


FIGURE 7.7 – Gauche : augmentation de l'AP en fonction du nombre de noyaux. Étude menée sur un seul fold, avec comme configuration par défaut 120 neurones. Droite : Étude menée sur un seul fold, avec comme configuration par défaut 16 noyaux.

Quelques filtres appris sont visibles figure 7.8. Les filtres restent bruités après leur

Détecteur	mAP	Rappel @ 0.01 FPPI	Rappel @ 0.1 FPPI	Rappel @ 1 FPPI
SVM (1er étage)	58.9±3.5	13.2±5.1	30.4±3.9	72.1±4.1
SVM fusion AE Discriminant	**69.6±3.4**	**20.4±6.2**	**49.0±3.6**	**80.3±3.1**
SVM suivi de CODA	58.2±4.4	10.0±4.0	34.0±6.2	71.1±5.0
SVM fusion CODA	65.8±3.7	20.0±5.2	41.6±3.8	77.5±5.2

TABLE 7.1 – Résultats sur la base VeDAI de l'algorithme SVM-CODA

optimisation. La différence entre leur initialisation et leur état final indique que ces filtres convergent entre autres vers des filtres de gradients ou des structures présentant un ou plusieurs pics, une ou plusieurs bandes. Étonnamment, cette convergence n'a pas besoin d'être terminée pour atteindre le plafond de performances.

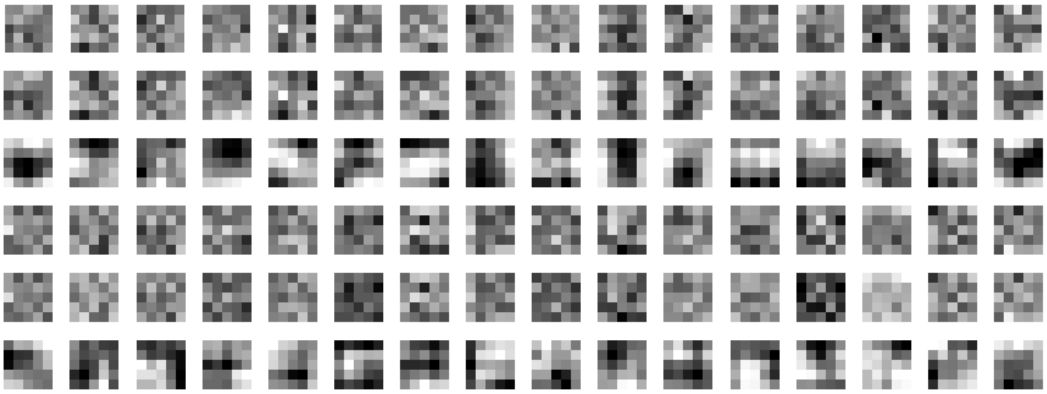


FIGURE 7.8 – Filtres et delta d'apprentissage pour une orientation. De haut en bas : filtres de départ pour la couche de convolution, filtres appris pour cette même couche, différence entre ces filtres, initialisation des filtres pour la couche de déconvolution, filtres appris, différence entre ces filtres.

Les résultats de détection sont présentés en Tab. 7.1. L'algorithme CODA seul ne permet pas d'apporter un gain en performances, en effet ses performances sont similaires à celle du HOG+SVM. Ainsi, aucune information n'est perdue alors que seuls les niveaux de gris sont utilisés, mais aucun gain n'en est retiré directement. La fusion entre CODA et HOG+SVM permet en revanche de gagner en performances et ce sur tous les indicateurs (+6.9% de mAP par exemple). Cependant, il faut noter que cette amélioration n'est pas aussi bonne que celle apportée par les autoencodeurs discriminants, comme rappelé dans le tableau de résultats.

7.4.2 Résultats en détection : utilisation des descripteurs

La Fig. 7.9 (gauche) montre la convergence de l'architecture CODA, pour 400 neurones et 16 noyaux et la compare à la convergence de Desc1+SVM et Desc2+SVM. Si la performance de CODA et de Desc1+SVM sature, un effet de sur-apprentissage est visible sur Desc2. En effet, la performance asociée à Desc2 diminue au delà de 100 itérations.

Il est possible d'analyser la dimensionalité intrinsèque de la carte de descripteur obtenue après convergence d'un CODA. La Fig. 7.9 (droite) montre que la carte de descripteurs contient une information relativement redondante. En effet, l'utilisation d'une ACP sur cette carte permet d'obtenir des résultats similaires en ayant une dimension

plus réduite. Il est donc possible de diviser par 3 (environ) la taille du descripteur pour une performance équivalente.

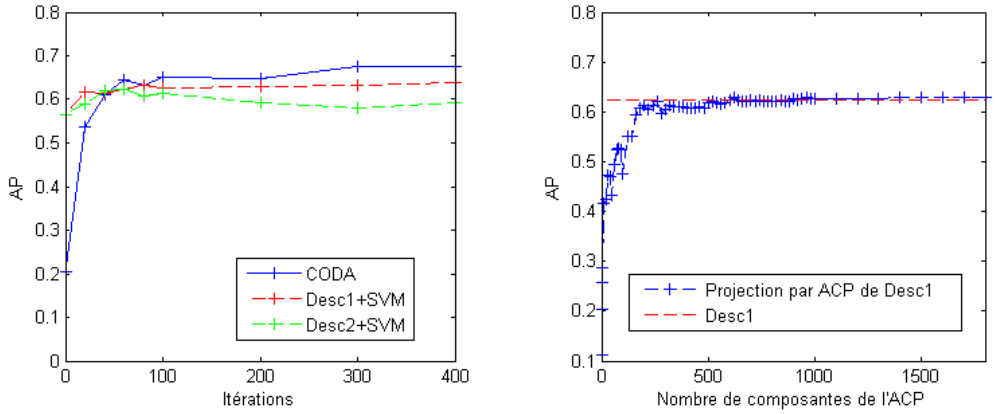


FIGURE 7.9 – Gauche : convergence de l'architecture CODA, que ce soit en tant que classifieur ou les couches intermédiaires. Droite : Dimensionalité de la carte de descripteurs. Cette figure présente l'évolution de la performance après une étape de réduction de dimension sur le descripteur Desc1. Etude menée sur un seul fold.

La comparaison entre autoencodeur convolutionel discriminant et non discriminant est illustrée en Tab. 7.2. L'autoencodeur dit *positif* est un autoencodeur convolutionel appris uniquement pour diminuer l'erreur de reconstruction des positifs. L'autoencodeur dit *mélange* est un autoencodeur convolutionel appris pour diminuer l'erreur de reconstruction des positifs et des négatifs, ayant pour but de les plonger dans l'espace latent.

Pour ce qui est de l'autoencodeur positif, il est notable que pour le Desc1 les résultats sont moins bons en AP, identiques pour les taux FPPI élevés, mais meilleurs pour les taux de FPPI faibles (0.01FPPI). Cependant, étant donné que le rappel à 0.01 FPPI est un des indicateurs les moins fiables (forts écarts types), il semblerait que l'autoencodeur positif est moins bon que l'autoencodeur CODA. Nous retrouvons d'ailleurs ce comportement lorsque nous comparons le HOG+SVM et Desc1+SVM. Cependant, comme montré en Tab. 7.3, en moyenne sur les 10 folds, le HOG+SVM est équivalent pour cet indicateur. Enfin, en ce qui concerne le descripteur dans l'espace latent (Desc2), la discrimination apporte un gain en performance certain.

Pour l'autoencodeur mélange, les performances sont clairement plus mauvaises et ce quels que soient les indicateurs. Le mélange des positifs et des négatifs ne crée pas de filtres de convolution qui peuvent extraire des caractéristiques discriminantes. De plus, la projection dans l'espace latent ne rend le mélange des caractéristiques que plus effectif.

Ainsi, l'aspect discriminant permet d'obtenir de meilleurs descripteurs.

Détecteur	Type d'apprentissage	AP	Rappel @ 0.01 FPPI	Rappel @ 0.1 FPPI	Rappel @ 1 FPPI
SVM (1er étage)	—	60.0%	9.8%	27.3%	77.0%
HOG+SVM suivi Desc1+SVM	discriminant	63.9%	6.9%	31.9%	76.5%
HOG+SVM suivi Desc1+SVM	positif	59.9%	15.1%	32.9%	75.1%
HOG+SVM suivi Desc1+SVM	mélange	46.6%	6.1%	18.6%	61.4%
HOG+SVM suivi Desc2+SVM	discriminant	59.4%	7.2%	29.9%	75.7%
HOG+SVM suivi Desc2+SVM	positif	54.0%	7.5%	21.7%	71.4%
HOG+SVM suivi Desc2+SVM	mélange	35.6%	3.1%	11.2%	44.0%

TABLE 7.2 – Comparaison de Desc1 et Desc2, appris avec des autoencodeurs discriminants, positifs ou mélanges. Expérience effectuée sur VeDAI, sur le fold 1.

Les résultats des différentes chaînes algorithmiques sont présentés en Tab. 7.3. Desc1 est légèrement plus efficace que l'utilisation d'un HOG, tout en encodant une information différente. Leur fusion permet d'obtenir un résultat significativement meilleur.

À l'inverse, Desc2 ne permet pas d'obtenir de meilleures performances, il y a donc une dégradation de l'information entre la carte des descripteurs et la projection dans l'espace latent. Cependant, la fusion permet de gagner quelques points, sans pour atteindre des performances équivalentes à celles de Desc1.

La concaténation de Desc1 et Desc2 ne permet pas d'augmenter les performances atteintes par Desc1. Ce résultat est attendu du fait que Desc2 n'est qu'une forme réduite de Desc1 et a des performances inférieures.

Enfin, les résultats indiquent que la chaîne Desc1+SVM est pratiquement équivalente à utiliser l'architecture CODA totale, en conséquence la partie description de l'architecture semble apporter l'information supplémentaire qui permet d'atteindre de meilleures performances.

Détecteur	mAP	Rappel à 0.01 FPPI	Rappel à 0.1 FPPI	Rappel à 1 FPPI
SVM (1er étage)	58.9±3.5	13.2±5.1	30.4±3.9	72.1±4.1
HOG+SVM suivi Desc1+SVM	60.3±3.7	12.2±5.9	34.6±4.8	71.3±3.8
HOG+SVM suivi Desc2+SVM	57.9±6.5	9.9±5.5	29.8±3.5	71.3±4.2
HOG+SVM suivi [Desc1 Desc2]+SVM	60.3±4.4	11.7±5.1	32.0±5.8	73.8±4.2
HOG+SVM fusion Desc1+SVM	**66.0±3.7**	18.7±7.8	**44.2±2.6**	77.5±4.8
HOG+SVM fusion Desc2+SVM	63.9±4.1	18.0±7.5	39.1±3.8	76.0±5.1
HOG+SVM fusion [Desc1 Desc2]+SVM	65.7±4.1	18.3±8.1	42.1±3.8	**78.1±5.6**
SVM fusion CODA	65.8±3.7	**20.0±5.2**	41.6±3.8	77.5±5.2

TABLE 7.3 – Résultats sur la base VeDAI de l'algorithme SVM-CODA

7.4.3 Temps de traitement

Un des points durs rencontrés lors des expériences sur les CODA est le temps d'apprentissage. Tab. 7.4 présente le temps d'apprentissage sur une itération, en sachant que nous avons effectué toutes nos expériences avec un minimum de 400 itérations afin d'avoir l'assurance de la convergence. Les temps sont donnés pour une implémentation Matlab, mono processeur. La complexité théorique est linéaire en nombre de neurones et en nombre de filtres, ce qui est retrouvé expérimentalement.

La Tab. 7.5 donne un aperçu de l'influence de la dimension du descripteur sur le temps d'apprentissage. La complexité théorique est linéaire en fonction du nombre de neurones, mais les résultats montrent que si la différence est effective, le coût fixe est assez élevé (aux alentours de 100 ms minimum).

Il est notable que les temps d'apprentissage sont longs dès que le nombre de noyaux et de neurones devient élevé (plus de 5 jours pour 400 itérations). Une implémentation par GPU permettrait de gagner en temps de traitement.

	nombrede neurones	4 noyaux	8 noyaux	16 noyaux
TE 1 itération (s)	100	116±21	230±23	492±30
TE 1 itération (s)	400	269±	688±52	1339±94
TE par fenêtre (ms)	400	30±4	60±6	130±8
TE par fenêtre (ms)	100	71±6	181±14	353±25

TABLE 7.4 – CODA : temps d'éxécution (TE), par itération en secondes et par fenêtre en millisecondes, pour un même ensemble de descripteurs. Résultats calculés sur 10 itérations successives.

Taille descripteur	2240	2560	2688	3136
TE par fenêtre (ms)	282±8	343±16	388±20	365±24

TABLE 7.5 – Temps d'exécution par fenêtre en millisecondes en fonction de la taille du descripteur traité. Résultats obtenus par moyenne sur 10 itérations, pour différents descripteurs.

7.5 Conclusions

Dans ce chapitre, nous avons développé une extension de l'autoencodeur discriminant afin d'apprendre les descripteurs en même temps que le classifieur. Nous avons pu voir que cette approche permet de créer des descripteurs complémentaires des histogrammes de gradients et permettent d'améliorer les performances.

L'étude du nombre de filtres et du nombre d'itérations nous permet d'espérer des résultats meilleurs, cependant, une implémentation plus optimale (en particulier en utilisant des GPU) serait essentielle pour pouvoir effectuer des simulations dans des temps raisonnables. De même, une réduction du temps de traitement permettrait d'étudier les paramètres tels que la taille des filtres, le type de ré-échantillonnage ou encore la taille des différents voisinages.

Enfin, les descripteurs appris grâce aux CODA sont plus performants, mais il semblerait que l'apprentissage des non linéarités du problème par le classifieur (résultats de l'AED) est plus effective. En effet, les résultats obtenus par CODA sont inférieurs à ceux d'une architecture HOG+AED.

Développer de nouvelles approches, tant sur les descripteurs que sur les classifieurs permet de gagner en performance. Cependant, si le taux de détection est fortement amélioré par les différentes méthodes proposées, il reste assez bas. Il semble donc difficile, en conservant une logique purement basée sur une approche d'opposition entre les véhicules et toutes les autres images, d'atteindre des performances beaucoup plus élevées, du moins sans exemples supplémentaires. Ainsi, des approches utilisant les caractéristiques de l'image de test semblent indiquées comme pistes d'amélioration.

Chapitre 8

Caractérisation du fond d'une image

Dans les chapitres précédents, nous avons étudié la problématique de détection dans le but de trouver un objet dans une image. Toutes les techniques présentées permettent d'obtenir des taux de détection élevés, mais uniquement pour des taux de faux positifs par image également élevés, ainsi que présenté en Fig. 8.1. En conséquence, la difficulté principale du problème réside dans les nombreux faux positifs, qui sont difficilement discernables des objets pour les différents algorithmes, que ce soit à cause des descripteurs ou des classifieurs, chacun exprimant des hypothèses sur le fond ou les objets à détecter.

Dans ce chapitre, nous présentons des travaux portant sur la caractérisation de fond [1] des images de test pour améliorer la détection. Au lieu de se baser sur les caractéristiques de l'objet à détecter, nous travaillons ici sur les caractéristiques du fond, dans une image de test. Le but de ces expériences est d'utiliser les a priori de régularité du fond afin de faire baisser le taux de faux positifs. Étant donné qu'une méthode de détection de fond ne nous affirme pas que ce qui est détecté est un véhicule, il faut la coupler avec une méthode de détection. Dans tout le chapitre, nous avons utilisé un détecteur HOG+SVM à 12 racines, en travaillant sur la classe 'slv'.

1. Ces travaux ont été réalisés en collaboration avec Pierre-Henri Abbo au cours de son stage de fin d'étude.

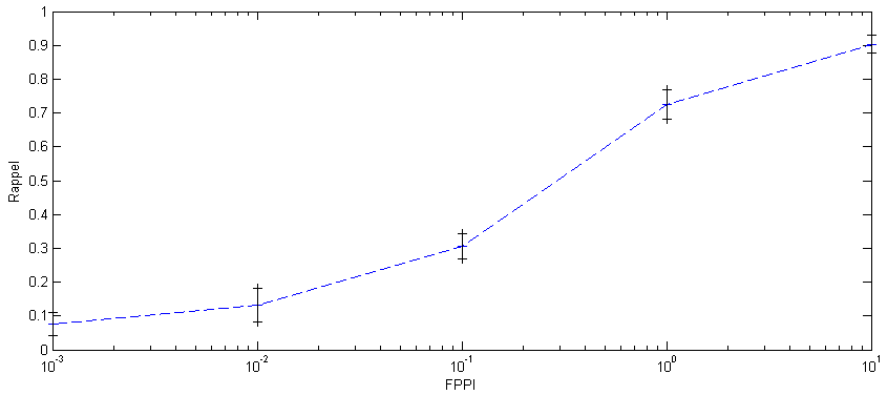


FIGURE 8.1 – Rappel en fonction du FPPI, configuration HOG+SVM à 12 racines.

FIGURE 8.2 – VeDAI : quelques fonds caractéristiques présentant des redondances.

8.1 Étude des faux positifs et analyse des fonds

Les faux positifs des différentes méthodes contiennent un certain nombre de structures qui pourraient être éliminées au vu de leur redondance dans l'image, comme illustré par les fonds présentés en Fig. 8.2. Les faux positifs peuvent se séparer en différentes grandes catégories, comme présenté en Fig. 8.3. Nous trouvons bien sûr les quelques exemples proches de l'ellipse de détection (Proche) ou d'autres classes de véhicules (Hors classe). Le pourcentage de ces faux positifs est faible, comme analysé dans le Chap. 4. Les ombres des véhicules (Ombre), un peu plus éloignées du centre du véhicule, forment un autre ensemble de faux positifs, mais celui-ci reste également faible.

Les structures rectangulaires d'une part et les velux et cheminées d'autre part représentent une forte ressemblance avec les véhicules et ils constituent logiquement deux grands ensembles de faux positifs. La différence entre ces deux ensembles réside dans le fait que les structures rectangulaires sont souvent isolées dans l'image, tandis que les velux et cheminées sont assez redondants, les toitures industrielles étant assez uniformes et les différentes maisons d'un même quartier résidentiel ayant des fenêtres identiques.

Les arbres et les craquelures sont des textures qui peuvent avoir de forts gradients, qui peuvent donner dans l'espace des descripteurs des primitives proches de celles des véhicules. Nous en retrouvons logiquement dans les faux positifs. Cependant, si il y a parfois des forêts, les faux positifs bien notés sont sur les arbres isolés. À l'inverse, les craquelures sont des faux positifs redondants dans une image.

Enfin, les marquages au sol, comme les places de parking par exemple, sont fortement contrastés (blanc sur noir) et ont souvent la propriété d'être de la même largeur (approximativement) que les véhicules à détecter.

Afin de diminuer le nombre de faux positifs, nous pouvons estimer que les structures telles que les marquages au sol, les velux, certains arbres et les craquelures, sont des structures redondantes dans les images. Nous avons donc souhaité développer une approche qui repère les éléments qui constituent le fond, directement sur l'image à traiter et les élimine lors d'une fusion des scores.

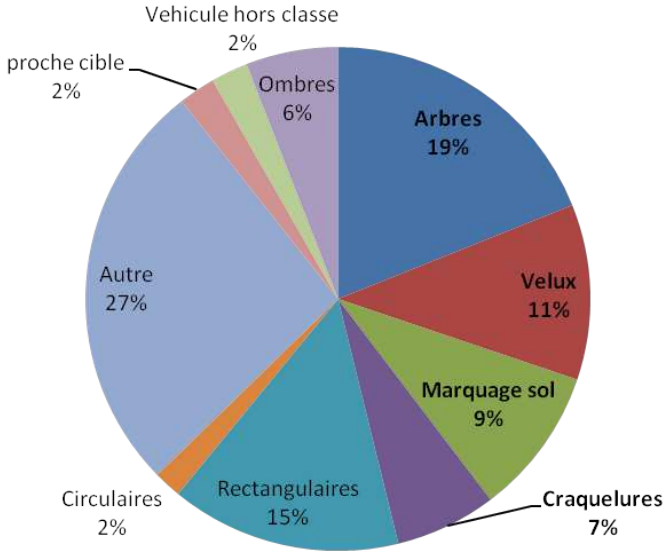


FIGURE 8.3 – Analyse des différents types de faux positifs. Statistiques obtenues en moyenne sur les 10 folds pour les 250 faux positifs les mieux notés.

8.2 Expériences préliminaires

8.2.1 Saillance par résidu spectral

Dans un premier temps, nous avons testé une méthode de calcul de saillance par résidu spectral [74]. La méthode part du fait qu'une image naturelle présente un spectre de Fourier qui varie en $1/f$. Les auteurs introduisent la notion de résidu spectral, défini ainsi :

$$R(f) = L_{moy}(f) - L(f) \tag{8.1}$$

avec L_{moy} le spectre logarithmique moyen de l'image et $L(f)$ le spectre logarithmique de l'image. Intuitivement, il s'agit de supprimer les fréquences caractéristiques du fond pour se concentrer sur les parties caractéristiques de l'image. L'image est ensuite reconstruite à partir du résidu spectral, par transformée de Fourier inverse. Nous obtenons alors une carte de saillance, habituellement lissée par une gaussienne.

La Fig. 8.4 présente deux résultats obtenus par l'algorithme de saillance par résidu spectral. Nous observons que sur des images simples, la saillance fonctionne très bien. En revanche, sur des images plus complexes, les véhicules ne sont absolument pas détectés.

Les performances exhaustives sont présentées en Tab. 8.1. Ce tableau montre qu'utiliser un a priori de saillance dégrade les performances. Ce résultat est attendu d'après les images présentées dans la Fig. 8.4. En effet, comme la saillance rehausse particulièrement des parties de l'image qui ne sont pas des véhicules et ne remonte les notes des véhicules que dans des cas où une méthode HOG+SVM effectue déjà de bonnes performances, l'algorithme n'apporte pas d'information discriminante.

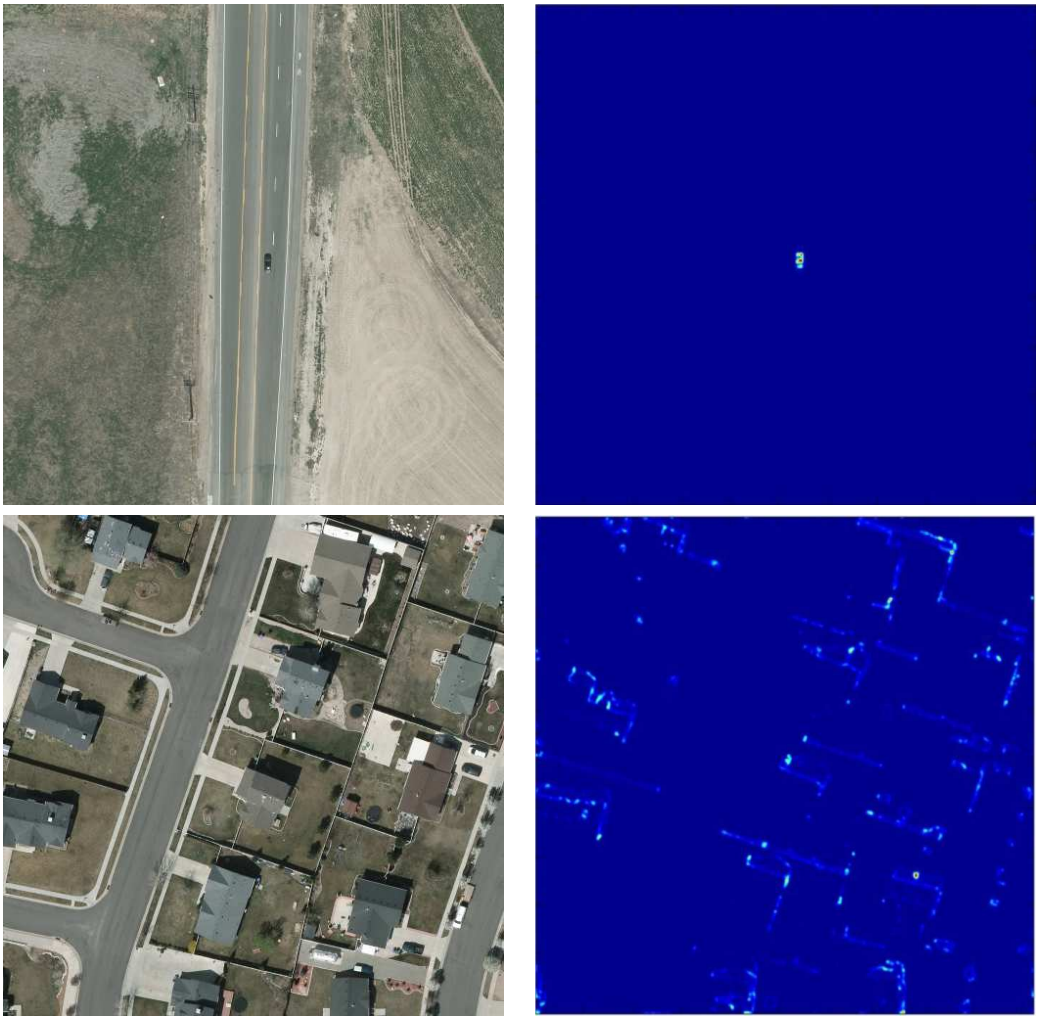


FIGURE 8.4 – Analyse de deux images, une sur fond simple et une sur fond complexe. À gauche, l'image d'origine, à droite l'image de saillance calculée par résidu spectral.

	mAP	0.01 FPPI	0.1 FPPI	1 FPPI	10 FPPI
HOG+SVM	72.9	20.8	45.0	72.5	90.5
perte	-1.9	-7.1	-3.1	-1.8	-0.8

TABLE 8.1 – Pertes de performances en utilisant un a priori de saillance par résidu spectral.

8.2.2 Apprentissage par SVM-OneClass

Une autre solution envisagée a été d'utiliser un SVM-OneClass pour apprendre les caractéristiques du fond. Une fois le SVM appris, chacun des descripteurs est noté par sa marge, qui nous donne une note de proximité au fond. Les premiers résultats sont présentés en Fig. 8.5. Nous pouvons voir que le SVM-OneClass ne fait pas ressortir les structures qu'un observateur humain pourrait considérer comme naturellement saillantes et, plus particulièrement, il considère la plupart du temps les véhicules comme des élé-

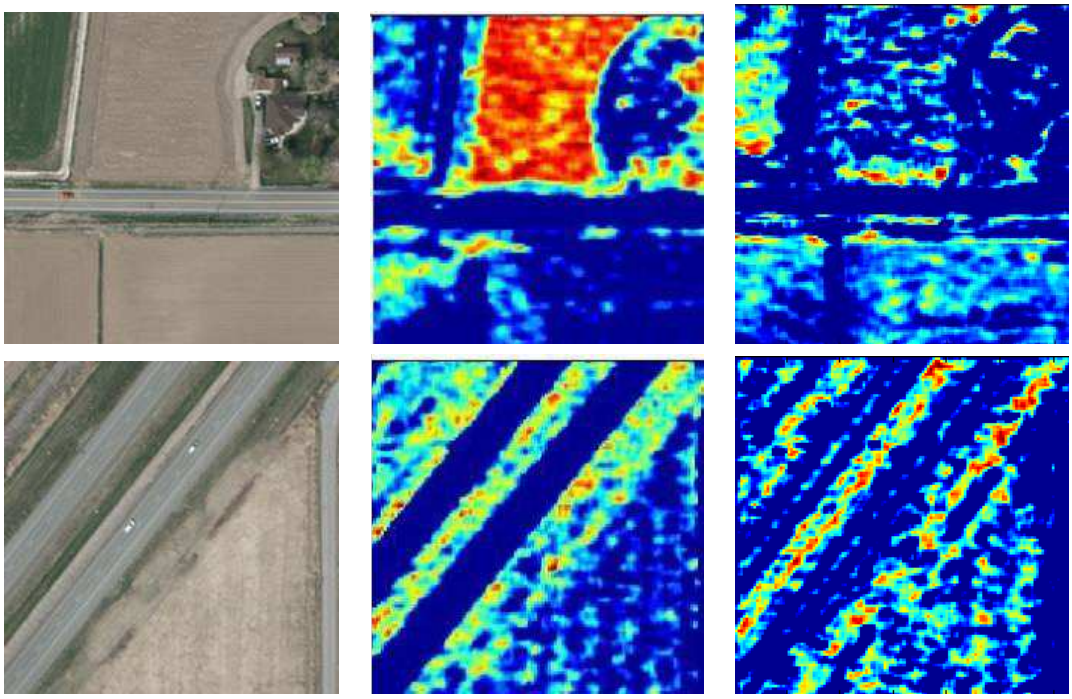


FIGURE 8.5 – Analyse de deux images. Chaque image représente la marge obtenue après une classification par SVM-OneCLass, utilisant un noyau linéaire (au centre) et un noyaux RBF (à droite).

ments de fond. En raison de ces résultats préliminaires, nous n'avons pas effectué plus d'analyses.

8.3 Clustering de fond

8.3.1 Algorithme

Afin d'éliminer les structures redondantes dans une image à traiter, la méthode suivante a été conçue, puis testée. Les descripteurs sont tout d'abord densément extraits de l'image. Ensuite, une étape de clustering est appliquée à ces classifieurs, afin d'obtenir une base représentative des éléments de fonds. Finalement, chaque descripteur obtient un score qui correspond à sa plus proche distance à cette base d'éléments de fond. Ainsi, un descripteur proche sera considéré comme du fond, tandis qu'un descripteur éloigné sera considéré comme un élément saillant.

La technique de clustering appliquée est l'algorithme des K-moyennes [2]. C'est un algorithme itératif sur les données, qui converge vers un minimum local. Il minimise la distance intra-cluster D définie telle que :

$$D = \sum_{i=1}^{K} \sum_{X \in C_i} ||X - M_i||^2 \tag{8.2}$$

avec C_i le cluster i, X un élément de ce cluster et M_i son barycentre. Un des choix les plus importants est le nombre de clusters. Différents critères existent dans la littérature sur

la manière de le fixer, nous avons choisi le critère de Ball-Hall [3]. Le critère de Calinski-Harabasz [9], plus utilisé dans le clustering, s'est avéré peu efficace en grandes dimensions. Nous définissons la distance intra-cluster ou *Within Group Scatter Sum* (WGSS) du cluster C_k comme étant :

$$WGSS^k = \sum_{X_i \in C_k} ||\mu_k - X_i||^2 \tag{8.3}$$

Cette quantité donne la dispersion dans un cluster. Le critère de Ball-Hall pour K clusters se définit tel que :

$$C_{Ball-Hall}(K) = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{Card(C_k}) WGSS^k \tag{8.4}$$

Le clustering s'arrête lorsque le critère se stabilise.

La distance d'un descripteur au cluster le plus proche est ensuite calculée comme suit :

$$D(X_i) = \min_{j \in 1..K} (||X_i - M_j||^2) \tag{8.5}$$

puis convertie en score normalisé sur l'image :

$$sc(X_i) = \frac{(D(X_i) - \mu(D(X)))}{\sigma(D(X))} \tag{8.6}$$

avec $\mu(D(X))$ la moyenne des distances sur l'image et $\sigma(D(X))$ l'écart type des distances sur l'image. Enfin, la fusion avec la méthode de détection de véhicules est effectuée soit par une fusion linéaire, soit par une fusion multiplicative.

8.3.2 Résultats qualitatifs

L'une des difficultés principales est la détermination du nombre optimal de clusters.

La Fig. 8.6 présente l'évolution de la distance intra-cluster (WGSS) en fonction du nombre de clusters. Trois parties sont à distinguer dans la courbe. En premier lieu, une forte décroissance est observée, due au fait que les structures principales de l'image sont segmentées. Puis une décroissance plus faible est observable, où les clusters sont plus proches les uns des autres mais la segmentation fait encore la différence. Enfin, une décroissance linéaire termine la courbe. Cette dernière partie correspond à de la sur-segmentation. Le critère de Ball-Hall a une décroissance forte lorsque K est faible, puis le critère est presque constant lorsqu'une sur-segmentation intervient.

La Fig. 8.7 montre les différents résultats de segmentation en fonction du nombre de clusters. Pour 2 clusters, les deux principales structures que sont l'eau et la terre sont bien segmentées. Nous observons de nombreux points saillants, certains d'entre eux font partie de structures encore assez grandes. Pour 20 clusters, en revanche, le ponton et une partie de la route ont des scores qui ont diminués. Finalement, ajouter plus de clusters, en passant à 40, ne donne pas vraiment plus d'information, nous observons une forte sur-segmentation de l'eau et des champs par exemple, sans pour autant que les scores ne changent franchement.

8.3.3 Résultats quantitatifs

Une analyse des faux positifs nous montre en Tab. 8.3 que l'algorithme permet effectivement de réduire le taux de faux positifs tels que les craquelures, les marquages

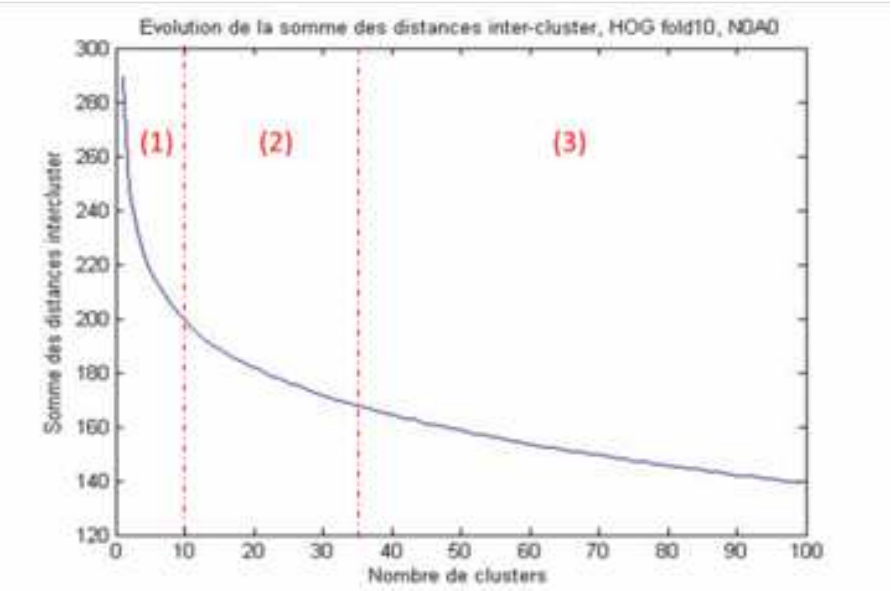


FIGURE 8.6 – Décroissance de la distance intra-cluster (WGSS) en fonction du nombre de clusters, en effectuant un algorithme de K-moyennes.

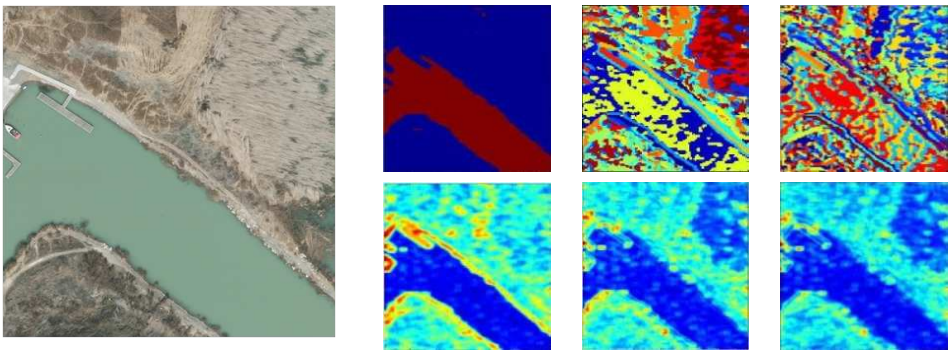


FIGURE 8.7 – Illustration du résultat des K-moyennes sur une image, en utilisant respectivement 2, 20 et 40 clusters. En haut, chaque couleur de l'image représente un indice de cluster, en bas, l'image représente les scores par K-moyennes associés. La couleur rouge indique un élément éloigné des clusters, le bleu indique un élément proche des clusters. À gauche, image d'origine.

	mAP	0.001 FPPI	0.01 FPPI	0.1 FPPI	1 FPPI	10 FPPI
HOG+SVM	68.4±2.4	10.1±6.4	19.0±5.1	40.8±4.9	69.4±2.6	89.1±1.9
HOG+SVM + K-moyennes	68.7±2.4	12.6±6.8	20.6±5.4	40.3±4.4	69.1±3.3	89.5±2.5
HOG+SVM + K-moyennes (gain)	0.3±2.4	2.5±3.2	1.5±3.3	-0.4±2.3	-0.4±1.7	0.2±0.5
HOG+SVM x K-moyennes	65.2±2.4	13.4±6.8	16.5±5.4	38.1±3.1	69.6±2.9	80.0±2.9

TABLE 8.2 – Résultats de la fusion d'un algorithme par K-moyennes et de la technique de détection de référence. Fusion par somme pondérée et par multiplication.

au sol ou encore les velux/cheminées. Le nombre de faux positifs n'ayant aucune structure précise diminue également, leur proximité avec d'autres éléments du fond étant fort

Catégorie	Véhicule			Motif de fond répétitif		
Sous-Catégorie	Proche	Hors classe	Ombres	Craquelures	Marquages sol	Velux-cheminées
HOG+SVM	2%	2%	6%	7%	9%	11%
HOG+SVM + K-moyennes	1%	4%	11%	1,5%	4,5%	5,8%
Catégorie	Motifs spécifiques			Autres		
Sous-Catégorie	Arbres isolés	Cercles	Rectangles	—		
HOG+SVM	19%	2 %	14,8	27%		
HOG+SVM + K-moyennes	28%	4%	24%	15,5%		

TABLE 8.3 – Analyse des 250 faux positifs à plus haut score, avec et sans la méthode d'apprentissage de fond.

probable. Logiquement, les autres types de faux positifs augmentent, tels que les motifs spécifiques, ce qui est également attendu du point de vue de la méthode.

Les résultats sur les indicateurs de performances sont présentés en Tab. 8.2. Dans le cas de la fusion additive, la mAP s'améliore légèrement ainsi que les taux de détection pour des taux très faibles de FPPI. Cependant, pour des taux de FPPI élevés, la méthode dégrade les performances. Ainsi, la méthode permet d'abaisser efficacement le nombre de faux positifs à haut score, mais en ce qui concerne les faux positifs à bas score, la méthode a tendance à ne pas être efficace. Pour ce qui est de la fusion multiplicative, l'amélioration sur les taux de FPPI bas est encore présente, de même que la dégradation sur les taux élevés. En revanche la mAP est finalement dégradée.

8.4 Conclusions

Dans ce chapitre, nous avons proposé d'utiliser l'image testée lors de la détection pour en retirer des informations relatives au fond et ainsi baisser le nombre de faux positifs à haut score. Nous avons tout d'abord étudié brièvement deux méthodes simples, une utilisant la saillance par résidu spectral, puis une autre utilisant un SVM-OneClass. Ces deux méthodes ne donnant pas de résultats probants, nous avons développé une méthode basée sur le clustering de l'image en différentes structures, afin d'en extraire les points singuliers.

Cette technique permet effectivement de baisser le nombre de faux positifs à haut score, mais elle rehausse certains négatifs à bas score. Une étude plus poussée de la fusion des scores (utilisation en cascade, fusion bayésienne ou encore utilisation de fonctions de croyance pour quantifier l'ignorance) permettrait certainement d'améliorer les résultats.

Chapitre 9

Conclusions et perspectives

Cette thèse présente nos travaux sur la Détection et Reconnaissance de véhicules faiblement résolus dans des images aériennes. Après avoir implanté et évalué des méthodes classiques de Détection et Reconnaissance, nous avons proposé différentes techniques visant à prendre en compte les spécificités de notre problématique. Nous avons développé une approche découplant les fonds des véhicules, en utilisant deux modèles distincts. Dans un second temps, nous avons développé un modèle d'apprentissage de variétés discriminantes, les autoencodeurs discriminants et leurs différentes variations. Enfin, nous avons proposé une méthode reposant sur des propriétés statistiques des images pour faciliter la détection des véhicules. Dans un premier temps, nous reprenons chaque contribution présentée dans cette thèse et nous complétons par ses limitations et ses développements possibles. Dans un second temps, nous proposons quelques perspectives et pistes possibles sur le sujet. La liste des publications effectuées au cours de cette thèse est située en Annexe E.

9.1 La base de données VeDAI

9.1.1 Contributions

Au cours de notre étude, nous avons conçu et mis en place une base de données permettant d'évaluer des algorithmes de Détection sur des images aériennes, dans lesquelles les véhicules sont faiblement résolus. Cette base de données est fournie avec à la fois des annotations riches, mais également un protocole de test rigoureux et un ensemble de performances d'algorithmes de référence, qui permettent à la communauté de proposer d'autres algorithmes et de les comparer efficacement.

9.1.2 Limitations et pistes d'amélioration

Si la base de données VeDAI permet d'avoir un bon point de repère pour l'évaluation d'algorithmes de Détection et Reconnaissance et contient beaucoup d'images annotées, elle reste très en deçà de ce qui peut être proposé par des grands challenges comme Imagenet (14 millions d'images). Malgré la grande disparité des fonds proposés, seuls une ville et ses alentours ont été annotés, les images présentent ainsi des similarités non négligeables, qui permettent d'assurer une certaine robustesse statistique d'un ensemble de fond à un autre ensemble de fond. Pour les véhicules, certaines classes ne présentent que peu de modèles différents (en particulier les tracteurs). Enfin, les vues sont uniquement verticales.

Afin de disposer d'une meilleure analyse, il serait intéressant d'évaluer l'apport de plus d'exemples dans cette base, afin de connaître l'impact de l'ajout d'exemples lors d'un apprentissage. De plus, il serait possible d'agrémenter la base avec différentes sources afin de comparer l'évolution des performances en fonction des capteurs et contextes étudiés. Enfin, il reste à étudier le cas où les images ne sont pas uniquement en vue verticale.

9.2 La détection autoencodeur contre ACP

9.2.1 Contributions

Étant donné que nous travaillons sur les véhicules faiblement résolus dans un environnement ouvert, nous avons proposé une approche basée sur le découplage entre l'apprentissage du fond et celui des véhicules. Cette méthode utilise un autoencodeur comme modèle génératif de véhicules et une Analyse en Composantes Principales pour modéliser les fonds. Cette méthode fonctionne quels que soient les descripteurs choisis et améliore les résultats d'une détection par des méthodes de l'état de l'art dans des conditions identiques. Nous avons démontré par là que modéliser l'objet d'intérêt différemment de son environnement permet de gagner en performance.

9.2.2 Limitations et pistes d'amélioration

La détection AE contre ACP permet d'obtenir de bon résultats. Cependant, cette méthode nécessite de nombreux exemples afin de fonctionner mieux qu'une méthode par SVM. Enfin, elle sous-entend que les fonds de la base disponible et les fonds étudiés lors de la phase de tests ont des propriétés statistiques identiques, ce qui est le cas pour VeDAI, mais peut ne pas être le cas dans une application industrielle. L'apprentissage de fond doit alors être effectué de nouveau.

Les non linéarités sont apprises par la méthode AE contre ACP. Cependant, un découpage des non linéarités principales par un changement dans la structure du modèle permet d'avoir de meilleurs résultats. L'étude sur une base de données ayant également des vues obliques, où le nombre de racines risque d'être élevé, serait à effectuer. De plus, la modélisation du fond se base sur les mêmes descripteurs que ceux des véhicules. Il pourrait être intéressant d'utiliser des représentations différentes.

9.3 L'autoencodeur discriminant

9.3.1 Contributions

Nous avons conçu et développé une méthode de classification par autoencodeur discriminant. Le principe est d'apprendre à reconstruire correctement une classe de positifs (dans notre cas des véhicules), tout en reconstruisant mal l'ensemble des négatifs (fonds). Nous avons également proposé une extension du concept en multi-classes, qui permet d'effectuer de la classification équivalente à plusieurs autoencodeurs discriminants, mais en utilisant jusqu'à deux fois moins d'opérations. Cet algorithme permet d'obtenir de bonnes performances en Détection ainsi qu'en Reconnaissance, équivalentes à celles d'un SVM non linéaire. De plus tant en phase d'apprentissage qu'en phase de test, sa complexité est plus faible que ce dernier, permettant un passage à l'échelle (augmentation du nombre d'exemples d'apprentissage) et une détection plus rapide.

9.3.2 Limitations et pistes d'amélioration

L'autoencodeur discriminant est une technique qui permet d'obtenir des résultats de classification identiques à un SVM non linéaire, mais son utilisation requiert (tout comme un SVM non linéaire) un certain nombre de validations croisées afin de fixer ses paramètres (nombres de neurones, marge). Expérimentalement, ces paramètres s'avèrent plutôt robustes (la plage de variation optimale est assez large), mais des techniques plus efficaces pourraient être envisagées, comme par exemple une optimisation conjointe de la marge et de la rétro-propagation de l'erreur.

Enfin, les autoencodeurs discriminants pourraient être utilisés dans un contexte d'apprentissage de métrique, sur des problématiques d'appariement (par exemple [76]). De plus, les autoencodeurs utilisés n'utilisent qu'une seule couche. L'impact du nombre de couches cachées doit également être évalué.

9.4 L'autoencodeur discriminant convolutionel

9.4.1 Contributions

Nous avons également conçu une première extension de l'autoencodeur discriminant, qui est l'autoencodeur discriminant convolutionel. Il permet à la fois d'apprendre un classifieur efficace, mais également d'apprendre des descripteurs qui extraient des données pertinentes à partir des informations disponibles.

9.4.2 Limitations et pistes d'amélioration

La limitation principale des autoencodeurs discriminants convolutionels est leur coût en temps de calcul sans architecture processeur dédiée, tant pour l'apprentissage que pour le test. De plus, tout comme pour les autoencodeurs discriminants, les paramètres de marge, nombre de neurones et nombre de filtres sont à ajuster, chaque paramétrage entrainant un coût supplémentaire en calcul. Une extension multi-classes reste à évaluer. Enfin, pour atteindre de meilleures performances, il semble tout indiqué d'utiliser plus de couches cachées, afin de se rapprocher de l'apprentissage profond (*Deep Learning* en anglais).

9.5 Apprentissage en ligne des caractéristiques du fond

9.5.1 Contributions

Nous avons proposé d'utiliser les propriétés statistiques de l'image de test afin de diminuer le taux de fausses alarmes. Nous avons montré que l'utilisation d'algorithmes simples comme de la saillance ou un SVM OneClass ne semblent pas apporter de résultats intéressants. Nous avons développé une approche fondée sur la redondance de certains éléments dans une image de test afin de détecter les éléments appartenant au fond.

9.5.2 Limitations et pistes d'amélioration

La technique développée nécessite encore des améliorations, en particulier en ce qui concerne la fusion des scores. En effet, les scores obtenus par cette méthodes n'ont pas les mêmes propriétés que des scores données par une chaîne algorithmique de détection,

étant donné qu'ils correspondent à une quantification de l'appartenance aux structures redondantes de l'image et non à la catégorie d'objet à détecter.

Enfin, de même que pour la méthode AE contre ACP, il pourrait être intéressant d'utiliser une méthode de représentation différente de l'image pour cette technique de celle utilisée par les algorithmes de détection de véhicules.

9.6 Perspectives

En plus de la continuation des contributions présentées dans cette thèse, d'autres travaux connexes peuvent être abordés en lien avec les problématiques de Détection et Reconnaissance de véhicules en imagerie aérienne.

9.6.1 Apprentissage avec peu d'exemples

Dans le cadre des missions opérationnelles, il est possible de disposer de quelques clichés de reconnaissance, en nombre limité. Ces clichés pourraient être exploités en deux applications. Tout d'abord, en l'absence de véhicules, ils pourraient permettre une adaptation des modèles de fonds disponibles. À l'inverse, en présence d'un véhicule inconnu, il serait possible d'intégrer ces images afin d'obtenir un modèle complet le plus proche de la réalité. Ces deux problématiques sont deux facettes d'un même problème : la généralisation à partir de peu d'exemples. La base VeDAI peut servir d'évaluation de ce genre de méthode, par exemple en apprenant la classe 'car' et, à l'aide d'une seule image de tracteur, apprendre un détecteur de tracteur.

Les travaux effectués dans cette optique ont débuté avec [116]. Depuis, diverses techniques ont été développées à cette fin, comme [39], qui émet l'hypothèse que les objets ont des apparences semblables ou [4], qui apprend une classe d'objet entière en utilisant un seul exemple grâce aux entrainements précédents. Plus récemment, les travaux de [170] se basent sur la modélisation des similarités entre les classes d'objets et [143] utilise une architecture profonde fondée sur des RBM et des processus de Dirichlet hiérarchiques afin de modéliser les corrélations entre les descripteurs bas niveaux de différentes catégories d'objet.

9.6.2 Transfert de domaine entre images synthétiques et images réelles

Un des défis les plus importants dans l'apprentissage est la qualité et la quantité des données disponibles. Une manière efficace d'obtenir un grand nombre de données annotées automatiquement de manière fiable est d'utiliser des modèles de simulation et de synthèse. Cependant, aussi fidèles que puissent être les modèles physiques, les images de synthèses n'ont pas une représentativité tout à fait exacte de la réalité. Ainsi, transférer un apprentissage (*transfert learning*) effectué sur un grand nombre d'images de synthèse afin d'obtenir les meilleurs performances possibles sur des images réelles est un axe de recherche qui doit être creusé. De même, il pourrait être possible d'utiliser des images réelles afin d'affiner la modélisation offertes par les images synthétiques, comme dans les travaux de [100] qui évaluent les méthodes de classification en rapport avec des images synthétisées ou de [176] qui propose différentes techniques de génération de nouvelles données dans le but de faire de la classification.

9.6.3 Réseaux profonds (Deep Learning)

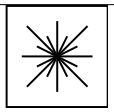
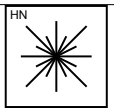
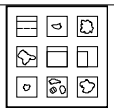
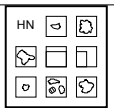
Les travaux utilisant les SVM donnent de bons résultats depuis des années pour la détection d'objets, mais un tournant est en train d'être pris avec les réseaux convolutionels qui battent systématiquement les approches classiques dans les challenges récents, comme nous l'avons évoqué tout au long de ce manuscrit ([153, 124, 150, 151, 57]). L'adaptation de telles architectures, conçues pour de la reconnaissance d'objets assez résolus, n'a pas encore été effectuée.

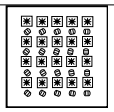
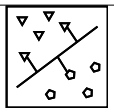
De plus, au cours de cette thèse nous avons développé des approches à base de réseaux de neurones multi-couches, mais nous n'avons pas utilisé d'architecture dite *profonde*, c'est-à-dire disposant de beaucoup de couches cachées, en raison principalement du temps imparti et du peu d'exemples d'apprentissage disponibles. Le formalisme développé (autoencodeurs discriminants ou convolutionels discriminants) s'étend à des réseaux plus profonds, en conséquence, il semblerait naturel de l'appliquer à des architectures comprenant plus de couches cachées. Cette application nécessiterait une implémentation plus efficace, à base d'hardwares massivement parallèles comme des GPU ou des supercalculateurs.

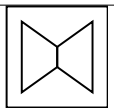
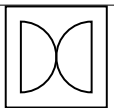
Annexe A

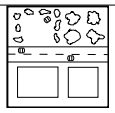
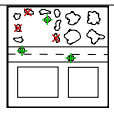
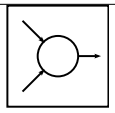
Table des symboles

Cette annexe présente les différents symboles utilisés dans les schémas de cette thèse, ainsi que leur signification.

	Extraction des descripteurs		Extraction des descripteurs difficiles, conservés après un premier classifieur
	Niveau de gris		Niveau de gris des fenêtres difficiles, conservées après un premier classifieur

	Template Matching		Mélange de Gaussiennes
	Séparateur à Vaste Marge		

	ACP suivie d'une reconstruction		Autoencodeur Classique
	Autoencodeur Discriminant		Autoencodeur Discriminant Convolutionel
	Carte de convolution, premier étage d'un Autoencodeur Discriminant Convolutionel		Carte de convolution suivie par la projection dans l'espace latent par un Autoencodeur Discriminant Convolutionel

	Parcours par fenêtre glissante		Résultats d'un algorithme de détection
	Fusion linéaire ou multiplicative		

Annexe B

Images VeDAI

Cette annexe présente différents exemples de fonds.


Eau
Chantier
Uni
Routes
Résidentiel Rural
Régulier
Zone Urbaine
Zone Industrielle

Annexe C

Présentation de la base Sagem et étude des performances de chaînes algorithmiques classiques

Annexe Confidentielle Sagem.

Annexe D

Étude des algorithmes par variétés génératives sur la base Sagem

Annexe Confidentielle Sagem.

Annexe E

Liste des publications

E.1 Journaux Internationaux

— S. Razakarivony, F. Jurie. *A novel target detection algorithm combining foreground and background manifold based models*, Machine Vision and Application Journal, Springer. Soumis, en cours d'évaluation.
— S. Razakarivony, F. Jurie. *Small Aerial Target Detection : A benchmark, Image and Vision Computing*, Elsevier. Soumis, en cours d'évaluation.

E.2 Conférences internationales

— S. Razakarivony, F. Jurie. *Discriminative Autoencoders for Small Targets Detection*, IEEE International Conference on Pattern Recognition 2014, Stockholm, Suède.
— S. Razakarivony, F. Jurie. *Small Target Detection combining Foreground and Background Manifolds*, IAPR International Conference on Machine Vision and Application 2013, Kyoto, Japon. Article nominé pour les MVA Awards 2013.

E.3 Conférence nationale

— S. Razakarivony, F. Jurie. *Autoencodeurs discriminants pour la détection de cibles faiblement résolues*, Reconnaissance de Formes et Intelligence Artificielle 2014, Rouen, France. Prix du meilleur article de la conférence.

Bibliographie

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1) :147–169, 1985.

[2] Michael R Anderberg. Cluster analysis for applications. Technical report, DTIC Document, 1973.

[3] Geoffrey H Ball and David J Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, DTIC Document, 1965.

[4] Evgeniy Bart and Shimon Ullman. Cross-generalization : Learning novel classes from a single example by feature replacement. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 672–679. IEEE, 2005.

[5] Thomas Batard, Michel Berthier, and Christophe Saint-Jean. Clifford–fourier transform for color image processing. In *Geometric Algebra Computing*, pages 135–162. Springer, 2010.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[7] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4) :509–522, 2002.

[8] Roberto Brunelli. Template matching techniques in computer vision, 2008.

[9] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1) :1–27, 1974.

[10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief : Binary robust independent elementary features. In *European Conference on Computer Vision*, pages 778–792. Springer, 2010.

[11] Peter Carbonetto, Gyuri Dorkó, Cordelia Schmid, Hendrik Kück, and Nando De Freitas. Learning to recognize objects with little supervision. *International Journal of Computer Vision*, 77(1-3) :219–237, 2008.

[12] Chih-Chung Chang and Chih-Jen Lin. LIBSVM : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.

[13] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details : Delving deep into convolutional nets. *arXiv preprint arXiv :1405.3531*, 2014.

[14] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class svm for learning in image retrieval. In *IEEE International Conference on Image Processing*, volume 1, pages 34–37. IEEE, 2001.

[15] Zezhi Chen, Nick Pears, Michael Freeman, and Jim Austin. Road vehicle classification using support vector machines. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 4, pages 214–218. IEEE, 2009.

[16] Ondrej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[17] Pierre Comon. Independent component analysis. *Higher-Order Statistics*, pages 29–38, 1992.

[18] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6) :681–685, 2001.

[19] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1) :38–59, 1995.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

[21] James Coughlan, Alan Yuille, Camper English, and Dan Snow. Efficient deformable template detection and localization without user initialization. *Computer Vision and Image Understanding*, 78(3) :303–319, 2000.

[22] David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 10–17. IEEE, 2005.

[23] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *European Conference on Computer Vision*, page 22, 2004.

[24] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, pages 1–2, 2004.

[25] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314, 1989.

[26] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[27] Ingrid Daubechies et al. *Ten lectures on wavelets*, volume 61. SIAM, 1992.

[28] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *International Conference on Machine Learning*, pages 233–240. ACM, 2006.

[29] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1) :1–38, 1977.

[30] Santosh K Divvala, Alexei A Efros, and Martial Hebert. How important are deformable parts in the deformable parts model ? In *European Conference on Computer Vision*, pages 31–40, 2012.

[31] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection : A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009.

[32] David L Donoho and Carrie Grimes. Hessian eigenmaps : Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10) :5591–5596, 2003.

[33] David Leigh Donoho and Carrie Grimes. *When does ISOMAP recover the natural parameterization of families of articulated images ?* Department of Statistics, Stanford University, 2002.

[34] Hassen Drira, Rim Slama, Boulbaba Ben Amor, Mohamed Daoudi, Anuj Srivastava, et al. Une nouvelle approche de reconnaissance de visages 3d partiellement occultés. In *Actes de la conférence RFIA 2012*, 2012.

[35] Didier Dubois and Henri Prade. An introduction to possibilistic and fuzzy logics. In *Readings in Uncertain Reasoning*, pages 742–761. Morgan Kaufmann Publishers Inc., 1990.

[36] Markus Enzweiler and Dariu M Gavrila. Monocular pedestrian detection : Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12) :2179–2195, 2009.

[37] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge - a retrospective.

[38] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal voc challenge. *International Journal of Computer Vision*, 88(2) :303–338, 2010.

[39] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples : An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1) :59–70, 2007.

[40] P. Felzenszwalb, R. Girshick, D. Mcallester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9) :1627–1645, 2009.

[41] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9) :1627–1645, 2010.

[42] Pedro F. Felzenszwalb, Ross B. Girshick, and David A. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2241–2248. IEEE, 2010.

[43] R. Feraud, O.J. Bernier, J.E. Viallet, and M. Collobert. A fast and accurate face detector based on neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 :42–53, 2001.

[44] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–264. IEEE, 2003.

[45] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *European Conference on Computer Vision*, pages 14–28. Springer, 2006.

[46] Sanja Fidler and Ales Leonardis. Towards scalable representations of object categories : Learning a hierarchy of parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[47] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1) :67–92, 1973.

[48] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2) :179–188, 1936.

[49] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination : consistency properties. Technical report, DTIC Document, 1951.

[50] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[51] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression : a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2) :337–407, 2000.

[52] Dennis Gabor. Theory of communication. part 1 : The analysis of information. *Journal of the Institution of Electrical Engineers-Part III : Radio and Communication Engineering*, 93(26) :429–441, 1946.

[53] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *Decision Forests for Computer Vision and Medical Image Analysis*, pages 143–157. Springer, 2013.

[54] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11) :2188–2202, 2011.

[55] Dariu M Gavrila and Vasanth Philomin. Real-time object detection for smart vehicles. In *International Conference on Computer Vision*, volume 1, pages 87–93. IEEE, 1999.

[56] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. `http://people.cs.uchicago.edu/~rbg/latent-release5/`, 2009.

[57] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv :1311.2524*, 2013.

[58] Joshua Gleason, Ara V Nefian, Xavier Bouyssounousse, Terry Fong, and George Bebis. Vehicle detection from aerial imagery. In *IEEE International Conference on Robotics and Automation*, pages 2065–2070. IEEE, 2011.

[59] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[60] Chunhui Gu, Pablo Andres Arbelaez, Yuanqing Lin, Kai Yu, and Jitendra Malik. Multi-component models for object detection. In *European Conference on Computer Vision*, pages 445–458. Springer, 2012.

[61] Chunhui Gu, Joseph J Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1030–1037. IEEE, 2009.

[62] Jihun Ham, Daniel D Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on Machine learning*, page 47. ACM, 2004.

[63] Peter JB Hancock, Roland J Baddeley, and Leslie S Smith. The principal components of natural images. *Network : computation in neural systems*, 3(1) :61–70, 1992.

[64] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[65] Hedi Harzallah, Frédéric Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *International Conference on Computer Vision*, pages 237–244. IEEE, 2009.

[66] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14 :1771–1800, 2002.

[67] Geoffrey E Hinton, Peter Dayan, and Michael Revow. Modeling the manifolds of images of handwritten digits. *Neural Networks*, 8(1) :65–74, 1997.

[68] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786) :504–507, 2006.

[69] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv :1207.0580*, 2012.

[70] Erik Hjelmås and Boon Kee Low. Face detection : A survey. *Computer Vision and Image Understanding*, 83(3) :236–274, 2001.

[71] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European Conference on Computer Vision*, pages 340–353. Springer, 2012.

[72] Adam Hoover, Gillian Jean-Baptiste, Xiaoyi Jiang, Patrick J Flynn, Horst Bunke, Dmitry B Goldgof, Kevin Bowyer, David W Eggert, Andrew Fitzgibbon, and Robert B Fisher. An experimental comparison of range image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7) :673–689, 1996.

[73] Harold Hotelling. Analysis of a complex statistical variable into principal components. *Journal of educational psychology*, 24(6) :417, 1933.

[74] Xiaodi Hou and Liqing Zhang. Saliency detection : A spectral residual approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[75] Paul VC Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654.

[76] Gary B Huang, Marwan Mattar, Tamara Berg, Eric Learned-Miller, et al. Labeled faces in the wild : A database forstudying face recognition in unconstrained environments. In *Workshop on Faces in'Real-Life'Images : Detection, Alignment, and Recognition*, 2008.

[77] Jing Huang, S Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–768. IEEE, 1997.

[78] Tommi Jaakkola, David Haussler, et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

[79] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[80] Xiaoyi Jiang, Cyril Marti, Christophe Irniger, and Horst Bunke. Distance measures for image segmentation evaluation. *EURASIP Journal on Applied Signal Processing*, 2006 :209–209, 2006.

[81] Thorsten Joachims. Making large-scale support vector machine learning practical. `http://svmlight.joachims.org/`, 1999.

[82] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[83] Aniruddha Kembhavi, David Harwood, and Larry S Davis. Vehicle detection using partial least squares. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6) :1250–1265, 2011.

[84] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3) :226–239, 1998.

[85] Uwe Knauer and Udo Seiffert. Cascaded reduction and growing of result sets for combining object detectors. In *Multiple Classifier Systems*, pages 121–133. Springer, 2013.

[86] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited : a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

[87] Teuvo Kohonen. Improved versions of learning vector quantization. In *International Joint Conference on Neural Networks*, pages 545–550. IEEE, 1990.

[88] M.A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *American Institute of Chemical Engineers Journal*, 37(2) :233–243, 1991.

[89] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Computer Vision Workshops, IEEE International Conference on*, pages 554–561. IEEE, 2013.

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[91] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1) :1–27, 1964.

[92] Stephane Lafon and Ann B Lee. Diffusion maps and coarse-graining : A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9) :1393–1403, 2006.

[93] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows : Object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[94] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2169–2178. IEEE, 2006.

[95] Quoc V Le. Building high-level features using large scale unsupervised learning. In *ICASSP*, pages 8595–8598. IEEE, 2013.

[96] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[97] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–50. Springer, 1998.

[98] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, 2004.

[99] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3) :259–289, 2008.

[100] Wenbin Li and Mario Fritz. Recognizing materials from virtual examples. In *European Conference on Computer Vision*, pages 345–358. Springer, 2012.

[101] Joerg Liebelt and Cordelia Schmid. Multi-view object class detection with a 3d geometric model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1688–1695. IEEE, 2010.

[102] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. A note on platt's probabilistic outputs for support vector machines. *Machine learning*, 68(3) :267–276, 2007.

[103] Gang Liu and Robert M Haralick. Optimal matching problem in detection and recognition performance evaluation. *Pattern Recognition*, 35(10) :2125–2139, 2002.

[104] Alexander C Loui, Charles N Judice, and Sheng Liu. An image database for benchmarking of automatic face detection and recognition algorithms. In *IEEE International Conference on Image Processing*, pages 146–150, 1998.

[105] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2) :91–110, 2004.

[106] Lee B Lusted. Signal detectability and medical decision-making. *Science*, pages 1217–1219, 1971.

[107] Xiaoxu Ma and W Eric L Grimson. Edge-based rich representation for vehicle classification. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1185–1192. IEEE, 2005.

[108] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1038–1045. IEEE, 2009.

[109] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *International Conference on Computer Vision*, pages 89–96. IEEE, 2011.

[110] Stephan G. Mallat. A theory for multiresolution signal decomposition - the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 :674–693, 1989.

[111] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[112] Vladimir Y Mariano, Junghye Min, Jin-Hyeong Park, Rangachar Kasturi, David Mihalcik, Huiping Li, David Doermann, and Thomas Drayer. Performance evaluation of object detection algorithms. In *IAPR International Conference on Pattern Recognition*, volume 3, pages 965–969. IEEE, 2002.

[113] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning*, pages 52–59. Springer, 2011.

[114] Alexis Mignon and Frédéric Jurie. Pcca : A new approach for distance learning from sparse pairwise constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[115] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *European Conference on Computer Vision*, pages 69–82. Springer, 2004.

[116] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 464–471. IEEE, 2000.

[117] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49(1) :8–30, 1961.

[118] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object detection. In *International Conference on Computer Vision*, pages 786–793. IEEE, 1995.

[119] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4) :349–361, 2001.

[120] Stefan Munder and Dariu M Gavrila. An experiment study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11), 2006.

[121] Kevin Murphy, Antonio Torralba, Daniel Eaton, and William Freeman. Object detection and localization using local and global features. *Toward Category-Level Object Recognition*, page 382, 2006.

[122] M-E Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454. IEEE, 2006.

[123] X Niyogi. Locality preserving projections. In *Neural information processing systems*, volume 16, page 153, 2004.

[124] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[125] Wanli Ouyang and Xiaogang Wang. Joint deep learning for pedestrian detection. In *International Conference on Computer Vision*, pages 2056–2063. IEEE, 2013.

[126] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1) :15–33, 2000.

[127] Mattis Paulin, Jerome Revaud, Zaid Harchaoui, Florent Perronnin, Cordelia Schmid, et al. Transformation pursuit for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[128] Marco Pedersoli, Andrea Vedaldi, and Jordi Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1353–1360, 2011.

[129] Alex Pentland. Viewbased and modular eigenspaces for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[130] P Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J Rauss. The feret database and evaluation procedure for face recognition algorithms. *Image and Vision Computing*, 16(5) :295–306, 1998.

[131] John C Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. Citeseer, 1999.

[132] Jean Ponce, Tamara L Berg, Mark Everingham, David A Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C Russell, Antonio Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29–48. Springer, 2006.

[133] Jean Ponce, Martial Hebert, Cordelia Schmid, and Andrew Zisserman. *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*. Springer, 2006.

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

[135] Vijay Raghavan, Peter Bollmann, and Gwang S Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3) :205–229, 1989.

[136] Nima Razavi, Juergen Gall, Pushmeet Kohli, and Luc Van Gool. Latent hough transform for object detection. In *European Conference on Computer Vision*, pages 312–325. Springer, 2012.

[137] Remi Ronfard, Cordelia Schmid, and Bill Triggs. Learning to parse pictures of people. In *European Conference on Computer Vision*, pages 700–714. Springer, 2002.

[138] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443. Springer, 2006.

[139] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3) :231–259, 2006.

[140] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1) :23–38, 1998.

[141] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[142] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme : a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3) :157–173, 2008.

[143] Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. Learning with hierarchical-deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8) :1958–1971, 2013.

[144] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *IEEE workshop on Applications of Computer Vision*, pages 138–142. IEEE, 1994.

[145] Lawrence K Saul and Sam T Roweis. Think globally, fit locally : unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4 :119–155, 2003.

[146] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3) :297–336, 1999.

[147] Paul Schnitzspan, Stefan Roth, and Bernt Schiele. Automatic discovery of meaningful object parts with latent crfs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 121–128. IEEE, 2010.

[148] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5) :1299–1319, 1998.

[149] Edgar Seemann, Mario Fritz, and Bernt Schiele. Towards robust pedestrian detection in crowded image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[150] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *IAPR International Conference on Pattern Recognition*, 2012.

[151] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat : Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv :1312.6229*, 2013.

[152] Glenn Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[153] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf : an astounding baseline for recognition. *arXiv preprint arXiv :1403.6382*, 2014.

[154] Michael Stark, Michael Goesele, and Bernt Schiele. Back to the future : Learning shape models from 3d cad data. In *British Machine Vision Conference*, page 5, 2010.

[155] Michael Stark, Jonathan Krause, Bojan Pepik, David Meger, James J Little, Bernt Schiele, and Daphne Koller. Fine-grained categorization for 3d scene understanding. *International Journal of Robotics Research*, pages 1543–1552, 2011.

[156] Uwe Stilla, Eckart Michaelsen, Uwe Soergel, Stefan Hinz, and HJ Ender. Airborne monitoring of vehicle activity in urban areas. *International Archives of Photogrammetry and Remote Sensing*, 35(B3) :973–979, 2004.

[157] K-K Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1) :39–51, 1998.

[158] Michael J Swain and Dana H Ballard. Color indexing. *International Journal of Computer Vision*, 7(1) :11–32, 1991.

[159] Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *Analysis and Modeling of Faces and Gestures*, pages 168–182. Springer, 2007.

[160] Franklin Tanner, Brian Colder, Craig Pullen, David Heagy, Michael Eppolito, Veronica Carlan, Carsten Oertel, and Phil Sallee. Overhead imagery research data set : an annotated data library and tools to aid in the development of computer vision algorithms. In *Proceedings of IEEE Applied Imagery Pattern Recognition Workshop*, pages 1–8, 2009.

[161] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500) :2319–2323, 2000.

[162] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1521–1528. IEEE, 2011.

[163] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images : A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11) :1958–1970, 2008.

[164] Antonio Torralba, Kevin P Murphy, and William T Freeman. Using the forest to see the trees : exploiting context for visual object detection and localization. *Communications of the ACM*, 53(3) :107–114, 2010.

[165] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance : A fast descriptor for detection and classification. In *European Conference on Computer Vision*, pages 589–600. Springer, 2006.

[166] Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, and Arnold W. M. Smeulders. Segmentation as selective search for object recognition. In *International Conference on Computer Vision*, pages 1879–1886. IEEE, 2011.

[167] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision*, pages 606–613. IEEE, 2009.

[168] Paul Viola and Michael J. Jones. Robust real time face detection. *International Journal of Computer Vision*, pages 137–154, 2004.

[169] Paul Viola, Michael J. Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2) :153–161, July 2005.

[170] Gang Wang, David Forsyth, and Derek Hoiem. Improved object categorization and detection using comparative object similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10) :2442–2453, 2013.

[171] X. Wang, T. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *International Conference on Computer Vision*, pages 32–39, 2009.

[172] Markus Weber, Max Welling, and Pietro Perona. Towards automatic discovery of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 101–108. IEEE, 2000.

[173] website. Utah agrc website, 2012.

[174] Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1) :77–90, 2006.

[175] Kilian Q Weinberger, Fei Sha, and Lawrence K Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *International Conference on Machine Learning*, page 106. ACM, 2004.

[176] Michael Weinmann, Juergen Gall, and Reinhard Klein. Material classification based on training data synthesized using a btf database. In *European Conference on Computer Vision*, pages 156–171. Springer, 2014.

[177] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *ESANN*, volume 99, pages 219–224, 1999.

[178] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database : Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492. IEEE, 2010.

[179] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, volume 97, pages 412–420, 1997.

[180] Tong Zhang and Frank J Oles. Text categorization based on regularized linear classification methods. *Information retrieval*, 4(1) :5–31, 2001.

[181] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *Society for Industrial and Applied Mathematics Journal on Scientific Computing*, 26(1) :313–338, 2004.

[182] Tao Zhao and Ram Nevatia. Car detection in low resolution aerial images. *Image and Vision Computing*, 21(8) :693–703, 2003.

[183] Long Zhu, Yuanhao Chen, Alan L. Yuille, and William T. Freeman. Latent hierarchical structural learning for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1062–1069, 2010.

[184] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *IAPR International Conference on Pattern Recognition*, volume 2, pages 28–31. IEEE, 2004.