



HAL
open science

Analyse fine 2D/3D de véhicules par réseaux de neurones profonds

Florian Chabot

► **To cite this version:**

Florian Chabot. Analyse fine 2D/3D de véhicules par réseaux de neurones profonds. Robotique [cs.RO]. Université Clermont Auvergne [2017-2020], 2017. Français. NNT : 2017CLFAC018 . tel-01708264

HAL Id: tel-01708264

<https://theses.hal.science/tel-01708264>

Submitted on 13 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : D.U : 2820

EDSPIC : 802

UNIVERSITE CLERMONT AUVERGNE

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

T H È S E

Présentée par

FLORIAN CHABOT

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPÉCIALITÉ : VISION POUR LA ROBOTIQUE

Analyse fine 2D/3D de véhicules par réseaux de neurones profonds

soutenue publiquement le 28 Juin 2017 devant le jury :

Mme. Catherine ACHARD	MCF, HDR - ISIR	Examinatrice
M. Stéphane CANU	PR - LITIS	Président
M. Mohamed CHAOUCH	DR - CEA LIST	Encadrant
M. Thierry CHATEAU	PR - UCA	Directeur
M. Christophe GARCIA	PR - LIRIS	Rapporteur
M. Frédéric JURIE	PR - GREYC	Rapporteur
Mme. Céline TEULIÈRE	MCF - UCA	Encadrante
M. Jaonary RABARISOA	IR - CEA LIST	Encadrant, invité

*À Charlotte,
À ma famille,
À mes amis.*

Remerciements

Je remercie Frédéric Jurie et Christophe Garcia pour avoir accepté de rapporter ce manuscrit de thèse. Merci également à Catherine Achard et Stéphane Canu pour leur présence lors de ma soutenance.

Je tiens à remercier Mohamed Chaouch pour son soutien indéfectible tout au long de ces trois années de thèse. J'ai énormément appris à ses côtés et rien n'aurait été possible sans ses nombreuses qualités humaines et scientifiques. Je remercie Jaonary Rabarisoa pour avoir apporté un point de vue extérieur sur ces travaux de recherche.

Un grand merci à mon encadrante Céline Teulière et mon directeur de thèse Thierry Chateau pour leur gentillesse, leur soutien et leur réactivité.

Mes remerciements vont également à toute l'équipe du LVIC au CEA, en particulier l'équipe d'analyse de scènes pour les nombreux échanges scientifiques que nous avons pu avoir.

Je remercie de tout coeur mes parents, pour m'avoir toujours soutenu et avoir toujours cru en moi.

Merci à mes amis, pour m'avoir permis de penser à autre chose que ma thèse durant ces trois ans. En particulier, merci à Pierrot, Fredo, Quentin, Pierrick, Thomas et Roxane.

Enfin, je remercie infiniment Charlotte pour avoir partagé les bons et les moins bons moments. C'est sans nul doute, sa patience et son soutien qui m'ont permis d'aboutir à ce travail.

Résumé

Les travaux développés dans cette thèse s'intéressent à l'analyse fine des véhicules à partir d'une image. Nous définissons le terme d'analyse fine comme un regroupement des concepts suivants : la détection des véhicules dans l'image, l'estimation de leur point de vue (ou orientation), la caractérisation de leur visibilité, leur localisation 3D dans la scène et la reconnaissance de leur marque et de leur modèle. La construction de solutions fiables d'analyse fine de véhicules laisse place à de nombreuses applications notamment dans le domaine du transport intelligent et de la vidéo surveillance.

Dans ces travaux, nous proposons plusieurs contributions permettant de traiter partiellement ou complètement cette problématique. Les approches mises en oeuvre se basent sur l'utilisation conjointe de l'apprentissage profond et de modèles 3D de véhicule. Dans une première partie, nous traitons le problème de reconnaissance de marques et modèles en prenant en compte la difficulté de la création de bases d'apprentissage. Dans une seconde partie, nous investiguons une méthode de détection et d'estimation du point de vue précis en nous basant sur l'extraction de caractéristiques visuelles locales et de la cohérence géométrique. La méthode utilise des modèles mathématiques uniquement appris sur des données synthétiques. Enfin, dans une troisième partie, un système complet d'analyse fine de véhicules dans le contexte de la conduite autonome est proposé. Celui-ci se base sur le concept d'apprentissage profond multi-tâches.

Des résultats quantitatifs et qualitatifs sont présentés tout au long de ce manuscrit. Sur certains aspects de l'analyse fine de véhicules à partir d'une image, ces recherches nous ont permis de dépasser l'état de l'art.

Mots clés : Vision par ordinateur, Réseaux de neurones convolutifs, Modèles 3D

Abstract

In this thesis, we are interested in fine-grained analysis of vehicle from an image. We define fine-grained analysis as the following concepts : vehicle detection in the image, vehicle viewpoint (or orientation) estimation, vehicle visibility characterization, vehicle 3D localization and make and model recognition. The design of reliable solutions for fine-grained analysis of vehicle open the door to multiple applications in particular for intelligent transport systems as well as video surveillance systems.

In this work, we propose several contributions allowing to address partially or wholly this issue. Proposed approaches are based on joint deep learning technologies and 3D models. In a first section, we deal with make and model classification keeping in mind the difficulty to create training data. In a second section, we investigate a novel method for both vehicle detection and fine-grained viewpoint estimation based on local appearance features and geometric spatial coherence. It uses models learned only on synthetic data. Finally, in a third section, a complete system for fine-grained analysis is proposed. It is based on the multi-task concept.

Throughout this report, we provide quantitative and qualitative results. On several aspects related to vehicle fine-grained analysis, this work allowed to outperform state of the art methods.

Keywords : Computer vision, Convolutional neural networks, 3D models

Table des figures

1.1	Différentes applications des travaux réalisés dans cette thèse.	2
2.1	Analogie entre les réseaux de neurones et la biologie	9
2.2	Schéma classique de l'apprentissage supervisé pour la classification d'objets.	10
2.3	Intérêt des réseaux de neurones	12
2.4	Le neurone artificiel	13
2.5	Le perceptron multicouche	14
2.6	Un petit réseau convolutif pour la reconnaissance de chiffre manuscrit	18
2.7	Illustration de la convolution	19
2.8	Illustration du Max pooling	20
2.9	Trois fonctions d'activation	21
2.10	Taux d'erreur de différentes architectures sur ImageNet pour la classification d'objets.	23
2.11	Deux architectures classiques : AlexNet [Krizhevsky 2012] et VGG16 [Simonyan 2014].	25
2.12	Le module d'inception	26
2.13	L'Architecture GoogLeNet.	26
2.14	Connexion résiduelle	27
2.15	ResNet	27
3.1	Exemples de détections d'objets renvoyées par un détecteur multi-classes	30
3.2	Schéma classique de détection d'objets par fenêtre glissante	33
3.3	Illustration de la <i>selective search</i>	36
3.4	Processus de génération de propositions d'objets 3D de [Chen 2016].	37
3.5	Les structures d'objets par parties.	39
3.6	Détecteur multi-vues basé sur un ensemble de composantes DPM	42
3.7	Processus d'estimation de la pose fine d'un véhicule avec un modèle déformable 3D [Lin 2014b]	45
3.8	Des exemples de classes provenant de la base de données CompCar [Yang 2015] pour la reconnaissance de marques et modèles de véhicules.	47

3.9	Exemple de courbes rappel/précision	49
4.1	Une solution intuitive pour la reconnaissance de marques et modèles par utilisation de modèles 3D.	58
4.2	Principe du calcul des contours saillants.	61
4.3	Exemple d'un rendu contours d'un modèle 3D de véhicule.	61
4.4	Comparaison qualitative de l'algorithme de canny [Canny 1986] et de l'algorithme de [Wang 2008].	62
4.5	Ensemble du système de reconnaissance de marques et modèles de véhicules 2D/3D.	64
4.6	Exemples de la base de données synthétique.	67
4.7	Visualisation des descripteurs communs aux couleurs	68
4.8	Visualisation des caractéristiques extraites sur différentes couches du CNN invariant aux couleurs et aux contours.	70
4.9	Courbes de rang	72
4.10	Exemples de résultats obtenus pour la classification de marques et modèles	74
5.1	Les sorties du système de détection de points de vue précis.	78
5.2	Modèle de caméra sténopé	79
5.3	Illustration du point de vue grossier et de la visibilité des parties.	82
5.4	Le déroulement général de l'algorithme	83
5.5	Architecture du CNN pour la détection de parties	84
5.6	Génération de patchs positifs et négatifs pour l'apprentissage du modèle d'apparence.	84
5.7	Construction du modèle de cohérence spatial géométrique.	87
5.8	Sélection des hypothèses pertinentes par cohérence spatiale pour une échelle l , un point de vue grossier donné v et un modèle 3D m	90
5.9	Illustration de la notion de configuration de parties	90
5.10	Fonctionnement de l'accumulateur pour le filtrage global	91
5.11	La matrice de confusion pour l'estimation du point de vue grossier	95
5.12	Des images de résultats en ajoutant des masques d'occultation	97
5.13	Images de résultats	99
6.1	Les sorties de l'approche Deep MANTA	103
6.2	Illustration du réseau de proposition d'objet	106

6.3	Quatre exemples provenant de la base de gabarits 3D et de celle de formes 3D.	107
6.4	Un exemple d'un modèle 2D/3D de véhicule	108
6.5	Vue d'ensemble de l'approche Deep MANTA	111
6.6	Le processus d'annotation semi-automatique	116
6.7	Courbes de rappel/précision et rappel/similarité angulaire sur l'ensemble de test de KITTI	121
6.8	Courbes de rappel/précision 3D	122
6.9	Images de résultats sur KITTI	127
6.10	Images de résultats sur KITTI	128
6.11	Images de résultats sur KITTI	129
6.12	Exemple de résultats pour l'analyse 3D d'objets multi-classes (véhicules, piétons, cyclistes...) par extension du Deep MANTA sur la base KITTI [Geiger 2012].	130
6.13	Exemple de résultats pour l'analyse 3D de véhicules et la reconnaissance de marques et modèles par extension du Deep MANTA.	131

Liste des tableaux

2.1	Comparaison de quatre réseaux de référence.	28
4.1	Performances sur les images de test (couleurs et contours) de CompCar (rang 1 / rang 5).	71
4.2	Performances sur les images de test (couleurs et contours) de 2DCar (rang 1 / rang 5).	71
5.1	Les résultats sur la <i>3D Object Dataset</i>	94
6.1	Résultats pour la détection 2D de véhicules sur deux ensembles de validation	119
6.2	Résultats pour l'estimation de l'orientation des véhicules sur deux ensembles de validation	120
6.3	Résultats pour la détection 2D de véhicules et l'orientation sur l'ensemble de test de KITTI	120
6.4	Mise en avant des performances de notre approche de raffinement pour la détection de véhicules	121
6.5	Performances de la localisation 3D à 1 mètre	122
6.6	Performances de la localisation 3D à 2 mètre	122
6.7	Localisation de parties, visibilité et gabarit 3D	123
6.8	L'influence du nombre de tâches apprises ainsi que des différents paramètres de régularisation.	125

Table des matières

1	Introduction	1
2	Les réseaux de neurones profonds	7
2.1	Introduction	7
2.2	Apprentissage supervisé et réseaux de neurones	8
2.2.1	Apprentissage supervisé	9
2.2.2	Différenciation des réseaux de neurones	11
2.3	Réseaux de neurones : les bases	12
2.3.1	Le neurone formel	12
2.3.2	Le perceptron multicouches	13
2.3.3	Apprentissage du perceptron multicouches	13
2.3.3.1	Rétropropagation pour la couche de sortie	15
2.3.3.2	Rétropropagation pour les couches cachées	16
2.3.3.3	Règle du delta généralisé	17
2.4	Les réseaux de neurones convolutifs profonds	17
2.4.1	Différents modules d'un réseau de neurones convolutif	18
2.4.2	Les architectures neuronales classiques	23
2.5	Conclusion	28
3	État de l'art	29
3.1	Introduction	29
3.2	Détection d'objets	30
3.2.1	Détection par fenêtre glissante	31
3.2.2	Détection par proposition d'objets	34
3.3	Modèles de représentation d'objets	37
3.3.1	Représentation globale	38
3.3.2	Représentation par parties	38
3.3.3	Représentation multi-vues	40
3.3.4	Représentation de la visibilité	42
3.4	Analyse 3D des objets	43
3.5	Catégorisation et Classification fine	45
3.6	Métriques d'évaluation	48

3.6.1	Évaluation de la détection	48
3.6.2	Évaluation de l'estimation du point de vue	50
3.6.3	Évaluation de la localisation 3D	51
3.6.4	Évaluation de la classification fine	51
3.7	Positionnement de la thèse	52
4	Classification fine 2D/3D de véhicules	55
4.1	Introduction	55
4.1.1	Solution intuitive	57
4.1.2	Solution proposée	58
4.2	Contours synthétiques et contours d'images réelles	60
4.2.1	Génération de contours synthétiques par rendu 3D	60
4.2.2	Détection de contours dans une image réelle	60
4.3	Modèle de reconnaissance de marques et modèles de véhicules	63
4.3.1	Descripteurs communs aux images couleurs et aux contours	63
4.3.2	Classifieur synthétique	65
4.4	Expérimentations	66
4.4.1	Bases de données	66
4.4.2	Résultats	67
4.4.2.1	Visualisation des caractéristiques communes.	67
4.4.2.2	Évaluation sur CompCar	69
4.4.2.3	Évaluation sur 2DCar	69
4.5	Conclusion	73
5	Détection de parties de véhicules pour l'estimation du point de vue précis	75
5.1	Introduction	75
5.2	Estimation de la pose par mise en correspondance 2D/3D	79
5.3	Modèle multi-vues pour l'estimation du point de vue	81
5.3.1	Modèle d'apparence	82
5.3.2	Modèle géométrique de cohérence spatiale	85
5.3.2.1	Composante de contexte global de boîte	86
5.3.2.2	Composante de contexte local de partie	86
5.3.2.3	Apprentissage des composantes géométriques	88
5.4	Estimation du point de vue précis	88
5.4.1	Détection d'hypothèses de parties	88

5.4.2	Sélection des hypothèses	89
5.4.2.1	Filtrage global	89
5.4.2.2	Filtrage local	91
5.4.3	Raffinement de la pose	92
5.5	Expérimentations	93
5.6	Conclusion	98
6	Approche MANy-Task (MANTA) : Analyse complète 2D/3D des véhicules dans la scène	101
6.1	Introduction	101
6.2	Réseau de Proposition d'objets	105
6.3	L'approche Deep MANTA	106
6.3.1	Bases de données de formes 3D et de gabarits 3D	107
6.3.2	Modèle 2D/3D de véhicule	107
6.3.3	Deep MANTA : le réseau	109
6.3.4	Deep MANTA : inférence	110
6.4	Apprentissage du Deep MANTA	112
6.4.1	Fonctions de perte Many-task	113
6.4.2	Annotation semi-automatique	115
6.5	Expérimentations	117
6.5.1	Détection de véhicules 2D et estimation de leurs orientations	118
6.5.2	Localisation 3D	121
6.5.3	Localisation des parties, leur visibilité et le gabarit 3D . . .	123
6.5.4	ManyTasks et paramètres de régularisation	124
6.6	Extensions de Deep MANTA	124
6.7	Conclusion	126
7	Conclusion et perspectives	133
	Bibliographie	137

Introduction

L'une des nombreuses facultés humaines est la capacité remarquable de comprendre et d'analyser l'environnement. À partir des signaux fournis par son système oculaire, l'homme est capable de décrire les objets qui l'entourent de manière très précise et très rapidement. On peut notamment souligner la capacité de l'homme à reconnaître les objets et les localiser tout en les caractérisant finement : leurs formes, leurs couleurs, leurs orientations... L'un des nombreux objectifs des chercheurs en vision par ordinateur est de construire des systèmes informatiques "intelligents" capables d'analyser des images de manière aussi performante que l'homme. Pour être fiables, ces algorithmes d'analyse doivent s'adapter aux changements d'apparence des objets liés au contexte dans lequel ils sont observés. Par exemple, l'apparence d'un objet peut varier en fonction de la luminosité de l'environnement ou il peut être partiellement caché par un autre objet. C'est la prise en compte de ces contraintes, naturellement gérées par le cerveau humain, que le système doit intégrer pour espérer se comporter comme un véritable système intelligent.

Contexte de la thèse

Les travaux réalisés au cours de cette thèse s'inscrivent dans le contexte automobile. Nous nous intéressons ici à un seul type d'objet : le véhicule et en particulier la voiture. Même si nous nous sommes focalisés sur cet objet d'intérêt, les algorithmes présentés sont facilement transférables à d'autres objets pour des applications de surveillance vidéo ou de robotique. Néanmoins, l'analyse fine de véhicules par l'image, c'est-à-dire la détection et la caractérisation fine, reste un sujet important par le nombre d'applications qu'elle engendre.

On peut citer parmi elles des applications de vidéo surveillance comme le comptage de véhicules ou le péage automatique. Dans ces deux applications, la



FIGURE 1.1 – Différentes applications des travaux réalisés dans cette thèse. (a) assistance à la conduite (freinage automatique), (b) parking automatique, (c) comptage de véhicules, (d) conduite autonome sur autoroute.

détection de véhicules par vision est nécessaire. Dans le cadre du péage automatique, le système doit être en plus capable de caractériser quel type de véhicules (voiture, van, camion, ...) se présente à la borne de péage afin d'évaluer le prix que son conducteur doit payer. On peut imaginer d'autres applications de vidéosurveillance faisant appel à la reconnaissance très fine de véhicules comme par exemple lors des pics de pollution. Dans ce cas particulier, réussir à reconnaître la marque, le modèle, la version et l'année de construction des véhicules permettrait une meilleure régulation lors de la mise en place de la circulation différenciée.

D'autres types d'applications sont inhérents à l'embarcation de système d'analyse fine au sein même d'un véhicule : ce sont les applications de transport intelligent. On peut citer parmi elles, l'assistance à la conduite, le parking automatique et la conduite autonome. Dans ce contexte, l'estimation de l'orientation et de la localisation 3D des autres véhicules présents autour du véhicule intelligent est nécessaire. Cela permet de prédire la trajectoire et la vitesse appropriées à la situation. Les performances en constante évolution de ces systèmes permettront certainement dans les années à venir d'intégrer dans les véhicules l'un des "Saint Graal" de l'intelligence artificielle par vision : un système de conduite sans conducteur. La Figure 1.1 illustre différentes applications possibles de ces travaux.

Problématiques

Dans cette thèse, nous proposons des méthodes d'analyse fine de véhicules à partir d'une caméra monoculaire. En d'autres termes, l'entrée des algorithmes développés est une image. Nous n'avons pas investigué des solutions d'analyse fine de véhicules en utilisant des informations temporelles (séquences d'images)

ou des informations 3D (capteurs RGB-D ou stéréoscopique). Les problématiques abordées sont les suivantes :

La détection de véhicules. Elle consiste à identifier les véhicules dans l'image c'est-à-dire proposer un ensemble de rectangles (boîtes englobantes) contenant des véhicules. Cette tâche n'est pas vraiment considérée comme faisant partie intégrante de l'analyse fine mais est nécessaire pour espérer tendre vers une caractérisation plus précise des véhicules : si on ne détecte pas les objets, les analyser plus finement est impossible. La détection d'objets est une problématique déjà bien étudiée et les principaux verrous sont multiples : la variation de taille des objets dans les images, leurs variations d'apparence suivant les points de vue sous lesquels ils sont observés et enfin leur visibilité.

La caractérisation de la visibilité. Elle permet de savoir si les véhicules sont partiellement cachés par d'autres objets (occultation) ou s'ils sortent partiellement de l'image (troncature). Réussir à prédire quelles parties des objets ne sont pas visibles permet d'analyser l'objet dans son environnement. Apporter une description de cette visibilité peut également permettre d'améliorer la détection de véhicules en prenant en compte cette contrainte.

L'estimation de l'orientation (ou du point de vue). Cette orientation correspond aux trois angles de rotation qui doivent être appliqués à l'objet pour l'observer dans l'image sous ce point de vue. En pratique, suivant les conditions d'acquisition des images, ces trois angles ne sont pas forcément nécessaires pour caractériser l'orientation du véhicule. Typiquement, si la caméra est fixée sur un support et que les objets se trouvent sur le plan du sol, deux angles sur les trois sont déjà connus. L'orientation correspond alors souvent à un seul angle de rotation compris entre 0 et 360 degrés et appelé azimut. Par exemple, un véhicule vu de face a pour azimut 0 degré et un véhicule vu de derrière a un azimut de 180 degrés.

La reconnaissance de marques et modèles. Elle a pour but de ne plus considérer les véhicules comme un seul type d'objets mais également de discriminer différentes sous-catégories à partir d'une description beaucoup plus fine de chacun d'eux. C'est une tâche difficile à réaliser du fait de la grande ressemblance qui peut exister entre différents modèles de véhicules. En d'autres termes, comment trouver ce qui va permettre de discriminer deux véhicules très

proches en apparence ?

La localisation 3D dans la scène. Elle fournit la position 3D (par rapport à la caméra ayant capté l'image) de chaque véhicule présent dans l'image. En utilisant comme entrée du système une image, réussir à localiser précisément les véhicules en 3D est particulièrement complexe : une image ne fournit pas d'informations de profondeur permettant d'inférer la localisation 3D des objets.

Contributions et organisation du manuscrit

Les approches proposées dans ces travaux présentent deux points communs : l'apprentissage profond (réseaux de neurones convolutifs) et l'utilisation de modèles 3D de véhicules. Dans les cinq dernières années, l'apprentissage profond a largement été utilisé par la communauté de vision par ordinateur pour résoudre de nombreuses tâches. Cette intérêt s'explique par sa capacité à extraire des informations très pertinentes sur les images et ainsi permettre l'entraînement de modèles très performants. Par ces aspects, l'apprentissage profond semble être adapté pour la construction de systèmes d'analyse fine. Néanmoins, les très bons résultats des méthodes utilisant l'apprentissage profond sont intimement corrélés au nombre de données utilisées pour leur apprentissage. En effet, pour être robustes et généralisables, ces modèles ont besoin d'être appris sur un grand nombre d'images annotées. Bien que des bases de données de taille importante existent, il reste difficile d'en créer pour répondre à un problème particulier.

En partant de cette observation, la contribution générale de cette thèse est d'utiliser des modèles 3D de véhicules afin de construire des bases de données pour l'analyse fine de véhicules. Dans nos travaux, les modèles 3D sont utilisés pour créer de toutes pièces des bases de données synthétiques (par rendu 3D) ou pour annoter de manière automatique des images réelles. Dans les deux cas, les modèles 3D permettent de générer un très grand nombre de données (images et/ou annotations) sans effort d'annotation.

Ce document est organisé de la manière suivante. Dans le chapitre 2 nous présentons le principe de l'apprentissage supervisé ainsi que les réseaux de neurones profonds que nous avons utilisés tout au long de nos travaux. Dans le chapitre 3,

une vue d'ensemble des approches en lien avec l'analyse fine d'objets est proposée.

Les chapitres 4,5,6 présentent les algorithmes développés et nos contributions. Dans le chapitre 4, un système de reconnaissance de marques et modèles de véhicule est proposé et tente de répondre à la question suivante : étant donné une base de données constituée de modèles 3D, serions-nous capables de construire un système de reconnaissance de marques et modèles de véhicules permettant de reconnaître des véhicules dans une image ? Dans ce chapitre, nous proposons un extracteur de caractéristiques visuelles robuste au biais qui peut exister entre des images synthétiques et des images réelles. Le chapitre 5 s'intéresse à la détection de véhicules et à l'estimation de leur orientation. Cette approche est basée sur l'utilisation exclusive de modèles 3D afin d'apporter une description visuelle et géométrique du véhicule. Elle utilise des caractéristiques locales permettant d'être robuste aux occultations. Le chapitre 6 détaille un système permettant l'analyse complète (2D et 3D) de véhicule dans une scène routière dans le cadre de la conduite autonome en exploitant le concept de réseau de neurones multi-tâches. Les modèles 3D sont utilisés pour retrouver les informations 3D des véhicules (orientation et localisation 3D) dans une image. Enfin le chapitre 7 expose les perspectives et les conclusions de ces recherches.

Publications

Les travaux présentés dans cette thèse ont fait l'objet de plusieurs publications :

Conférences internationales

1. **F. Chabot**, M. Chaouch, J. Rabarisoa, C. Teulière, T. Chateau. *Deep MANTA : A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image*. CVPR, 2017.
2. **F. Chabot**, M. Chaouch, J. Rabarisoa, C. Teulière, T. Chateau. *Deep Edge-color invariant features for 2D/3D car fine-grained classification*. IV, 2017.
3. **F. Chabot**, M. Chaouch, J. Rabarisoa, C. Teulière, T. Chateau. *Accurate 3D car pose estimation*. ICIP, 2016.
4. C. Bernay-Angeletti, **F. Chabot**, C. Aynaud, R. Aufrere, R. Chapuis. *A Top-Down Perception Approach for Vehicle Pose Estimation*. ROBIO, 2015.

Conférences nationales

5. **F. Chabot**, M. Chaouch, J. Rabarisoa, C. Teulière, T. Chateau. *Descripteurs profonds communs aux couleurs et aux contours pour la reconnaissance fine 2D/3D de véhicules*. RFIA, 2016.
6. **F. Chabot**, M. Chaouch, J. Rabarisoa, C. Teulière, T. Chateau. *Détection de pose de véhicule pour la reconnaissance de marque et modèle*. ORASIS, 2015.

Les réseaux de neurones profonds

Sommaire

2.1	Introduction	7
2.2	Apprentissage supervisé et réseaux de neurones	8
2.2.1	Apprentissage supervisé	9
2.2.2	Différenciation des réseaux de neurones	11
2.3	Réseaux de neurones : les bases	12
2.3.1	Le neurone formel	12
2.3.2	Le perceptron multicouches	13
2.3.3	Apprentissage du perceptron multicouches	13
2.3.3.1	Rétropropagation pour la couche de sortie	15
2.3.3.2	Rétropropagation pour les couches cachées	16
2.3.3.3	Règle du delta généralisé	17
2.4	Les réseaux de neurones convolutifs profonds	17
2.4.1	Différents modules d'un réseau de neurones convolutif	18
2.4.2	Les architectures neuronales classiques	23
2.5	Conclusion	28

2.1 Introduction

Ce chapitre a pour objectif de présenter les réseaux de neurones convolutifs profonds (*Deep Convolutional Neural Networks* CNN) utilisés tout au long de cette thèse. De manière générale, les réseaux de neurones encodent une fonction mathématique à appliquer sur un signal d'entrée (dans notre cas, une image) et permettant de prédire un signal de sortie. Cette fonction est une composition de plusieurs fonctions non-linéaires. Ces réseaux sont inspirés très sommairement du fonctionnement du cerveau humain. Ils sont constitués d'un grand nombre de neurones artificiels (introduits pour la première fois en 1943 par [McCulloch 1943]) connectés

entre eux et modélisant le fonctionnement des neurones biologiques. La Figure 2.1 illustre l'analogie entre le cerveau humain et les réseaux de neurones. Ces réseaux existent depuis longtemps [McCulloch 1943, Rosenblatt 1958, Minsky 1969], mais leur étude a stagné à partir de la fin des années 90 jusqu'à une explosion de popularité depuis 2012. Ils ont depuis surpassé beaucoup de méthodes dans les tâches de vision par ordinateur (classification, détection, segmentation...). Cet engouement récent, notamment dans le domaine de la vision, s'explique de plusieurs manières. Premièrement, d'énormes bases de données publiques annotées sont actuellement disponibles (MSCOCO [Lin 2014a], YouTube-8M [Abu-El-Haija 2016], ImageNet [Deng 2009]) et rendent ainsi possible l'apprentissage de réseaux de neurones complexes. Par exemple, la base ImageNet [Deng 2009] contient environ 14 millions d'images correspondant à 22 000 classes d'objets. Ainsi, ce type de base a permis à [Krizhevsky 2012] de remporter la compétition ImageNet en classification d'objets (1,2 millions d'images correspondant à 1000 classes) en proposant une architecture neuronale contenant 60 millions de paramètres. C'est ce grand nombre de paramètres qui différencie les réseaux de neurones modernes de ceux des années 90. La seconde raison ayant relancé l'étude des réseaux neuronaux est la capacité des machines modernes à effectuer beaucoup de calculs dans un temps raisonnable notamment grâce à l'utilisation des cartes graphiques (GPU). Cela permet de construire des réseaux de neurones de plus en plus complexes et performants et cela de plus en plus rapidement. Enfin, la multiplication des *framework* de *Deep Learning* (Caffe [Jia 2014], Tensorflow [Abadi 2016], Torch [Collobert 2002]...) permet une démocratisation de ces méthodes d'apprentissage automatique.

Dans ce qui suit, nous rappellerons rapidement les objectifs de l'apprentissage supervisé et l'intérêt des réseaux de neurones par rapport aux approches classiques d'apprentissage. Ensuite, nous détaillerons le fonctionnement général de ces réseaux. Enfin, nous présenterons un type de réseau de neurones particulier appelé réseau de neurones convolutif (CNN) utilisé tout au long de cette thèse.

2.2 Apprentissage supervisé et réseaux de neurones

L'apprentissage automatique (en anglais, *machine learning*) est un vaste sujet de recherche. Nous pouvons distinguer différentes familles d'apprentissage (supervisé, semi-supervisé, non-supervisé, par renforcement...). L'apprentissage automatique a pour objectif de construire des modèles mathématiques permettant de

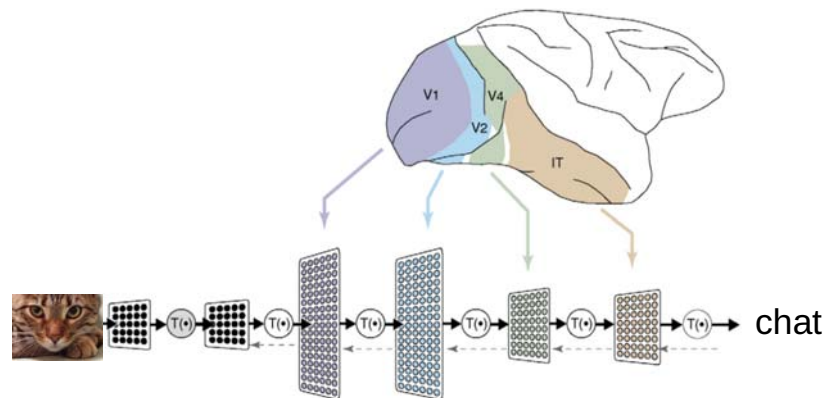


FIGURE 2.1 – Analogie entre les réseaux de neurones et la biologie. Un signal d'entrée (ici une image) active des réponses neuronales en série pour aboutir à une réponse (ici "chat"). Source [DiCarlo 2007]

prédire une sortie étant donné un signal d'entrée. Les réseaux de neurones sont des outils permettant d'apprendre ces modèles. Dans ce qui suit, nous contextualisons l'apprentissage automatique dans le cadre de la vision par ordinateur bien que ce type de modèle puisse être appliqué à d'autres types de données que des images.

2.2.1 Apprentissage supervisé

Le type d'apprentissage automatique le plus utilisé est l'apprentissage dit "supervisé" qui permet à la machine d'apprendre ses paramètres en utilisant une base de données annotée. Par exemple, dans le cadre de la classification d'images, un modèle entraîné grâce à l'apprentissage supervisé prédit le type d'objet (sa classe) présent dans l'image. En vision par ordinateur, cette base est généralement constituée d'un certain nombre d'images et de leurs sorties associées (la vérité terrain). Durant l'apprentissage, chaque image de cette base est présentée au modèle qui va mettre à jour ses paramètres pour produire la sortie désirée. La mise à jour de ces paramètres est effectuée en utilisant la notion de minimisation de risque : quand un exemple d'apprentissage est présenté au modèle, la sortie prédite par celui-ci est comparée à la sortie désirée. L'erreur entre la sortie désirée et celle prédite est alors calculée. L'objectif de l'apprentissage supervisé est de trouver les paramètres du modèle qui minimisent cette erreur sur tous les exemples de la base d'apprentissage. Un modèle ainsi appris permet par la suite, lors de la phase de test, de prédire

la sortie associée à une image qu'il n'a pas vue lors de la phase d'apprentissage. C'est ce que l'on appelle la généralisation du modèle.

L'apprentissage supervisé en vision par ordinateur se divise généralement en deux étapes. La première est l'extraction de caractéristiques visuelles sur les images de la base d'apprentissage. L'extraction des caractéristiques a pour objectif de fournir une description discriminante dans un espace réduit comparé à l'espace de l'image qui a une dimension trop grande. En d'autres termes, elle permet de transformer l'image en un vecteur compact la caractérisant. La seconde étape est l'utilisation de ces caractéristiques pour construire des modèles mathématiques, comme les classifieurs linéaires, permettant de séparer les images dans l'espace des caractéristiques. L'apprentissage de classifieurs linéaires sur des vecteurs de caractéristiques visuelles est un problème bien formulé et résoluble notamment avec les Séparateurs à Vaste Marge (SVM) [Cortes 1995]. La Figure 2.2 illustre schématiquement le processus classique de l'apprentissage supervisé. Une fois le modèle appris, il peut être utilisé sur une nouvelle image dont la classe est inconnue. Le vecteur de caractéristiques visuelles est extrait sur cette image et le modèle prédit sa classe.

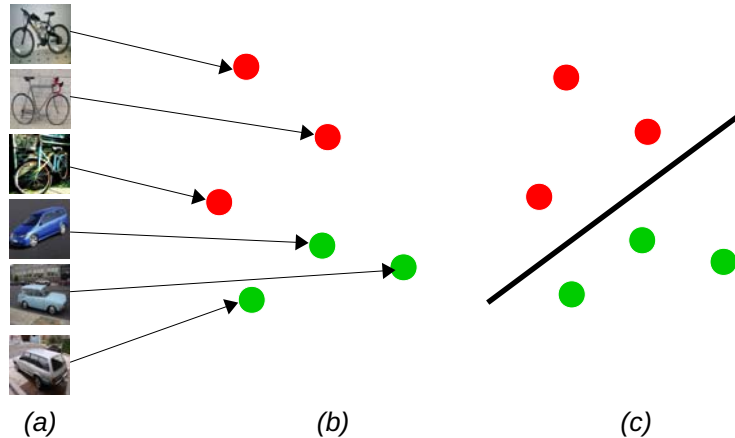


FIGURE 2.2 – Schéma classique de l'apprentissage supervisé pour la classification d'objets. (a) Une base d'images correspondant à deux classes (vélo et voiture). (b) les caractéristiques extraites de ces images. Le vecteur caractéristique de chaque image est ici schématisé par un point 2D pour la lisibilité du schéma mais est généralement de plus grande dimension. (c) La séparation apprise grâce à un classifieur linéaire permettant de discriminer les deux classes.

Il existe de nombreux types de caractéristiques visuelles comme les histo-

grammes de gradients orientés (HOG) [Dalal 2005], les transformations de caractéristiques visuelles invariantes à l'échelle (SIFT) [Lowe 1999], les motifs binaires locaux (LBP) [Ojala 1994] ou encore les caractéristiques Haar [Viola 2001]. Ces approches permettent d'extraire des caractéristiques bas niveau, c'est-à-dire basées sur des primitives de l'image comme les gradients ou les contours. Elles sont conçues "à la main" et calculables pour n'importe quelle image. Ces algorithmes d'extraction sont complètement externes à l'apprentissage d'un classifieur et sont calculés au préalable sur les images.

2.2.2 Différenciation des réseaux de neurones

Les réseaux de neurones font partie des outils d'apprentissage automatique et sont notamment utilisés pour l'apprentissage supervisé. Ils s'inscrivent dans une logique légèrement différente des approches "classiques" d'apprentissage pour la vision. En effet, nous avons pu voir précédemment que le processus d'apprentissage inclut une étape d'extraction de caractéristiques visuelles sur l'image. Celle-ci se base le plus souvent sur des algorithmes orientés traitement d'images. Bien que très pertinentes pour certaines tâches de vision, ces caractéristiques peuvent s'avérer inefficaces suivant la nature du problème à résoudre. Cela nous amène à nous poser plusieurs questions. Qu'est-ce qui caractérise vraiment les objets ? Les contours ? Les couleurs ? Comment réussir à trouver une représentation réduite d'une image la plus pertinente et la plus discriminante possible ?

C'est sur ce point que les réseaux de neurones marquent une rupture technologique. En effet, plutôt que d'extraire des caractéristiques visuelles "manuellement", ils permettent d'utiliser directement l'image en entrée et d'apprendre quelles caractéristiques visuelles sont les plus pertinentes pour un problème donné. La Figure 2.3 illustre cette différence. Contrairement aux caractéristiques de type HOG [Dalal 2005] ou SIFT [Lowe 1999] qui sont des représentations de l'image bas niveau, les réseaux de neurones permettent, par une succession de fonctions non-linéaires, d'extraire des caractéristiques de plus en plus haut niveau et pertinentes. C'est cette succession de fonctions qui confère le nom de *Deep Learning* (apprentissage profond) aux approches basées sur les réseaux de neurones modernes. Plus il y a de fonctions qui se succèdent plus le réseau est dit "profond" et plus les caractéristiques visuelles extraites sont de haut niveau. Un réseau est considéré comme profond, s'il est constitué d'au moins cinq fonctions successives.

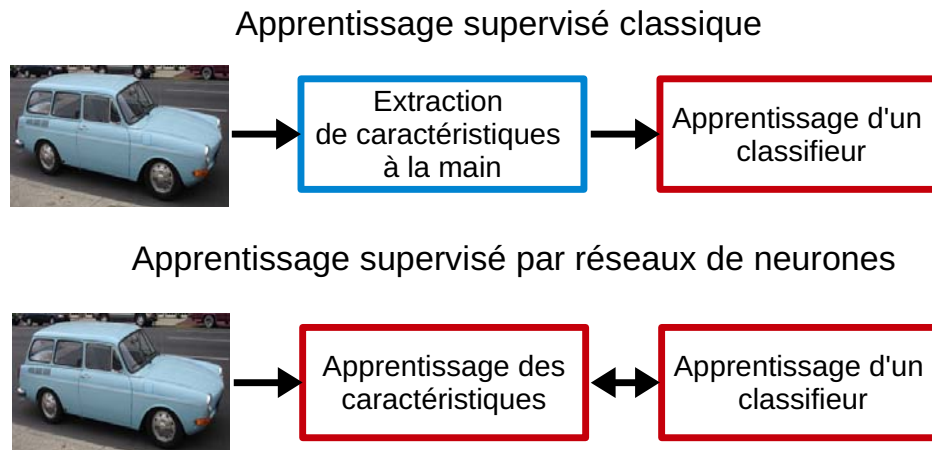


FIGURE 2.3 – Intérêt des réseaux de neurones pour l'apprentissage des caractéristiques visuelles discriminantes par rapport aux approches d'apprentissage supervisé classiques.

2.3 Réseaux de neurones : les bases

Nous proposons ici d'exposer les bases des réseaux neuronaux. Nous présentons dans un premier temps l'entité de base d'un réseau de neurones : le neurone formel. Ensuite, nous présentons le perceptron multicouches puis l'algorithme permettant de l'apprendre : la rétropropagation du gradient.

2.3.1 Le neurone formel

Le neurone formel (ou artificiel), initialement introduit par [McCulloch 1943], est une modélisation mathématique du neurone biologique. Il consiste en une fonction mathématique à appliquer à un signal et renvoyant une valeur d'activation. En considérant un signal d'entrée $\mathbf{x} = [x_1, \dots, x_n]^T$, le neurone artificiel renvoie la valeur d'activation y :

$$y = f\left(b + \sum_i w_i x_i\right) \quad (2.1)$$

Dans cette formulation, les w_i sont communément appelés les poids et b est appelé le biais. La fonction f est nommée fonction d'activation ou plus rarement fonction de transfert. Dans le neurone formel initial, cette fonction était la fonction *sign* renvoyant ainsi une valeur binaire en sortie du neurone. Les poids, le biais et la fonction d'activation caractérisent le neurone formel. La Figure 2.4 illustre le fonctionnement de celui-ci.

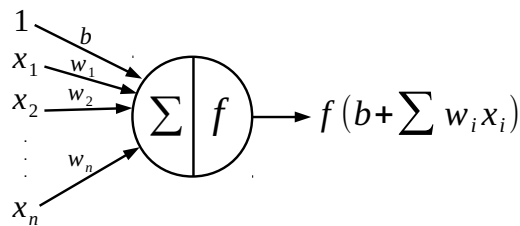


FIGURE 2.4 – Le neurone artificiel.

En 1958, [Rosenblatt 1958] introduit un algorithme d'apprentissage automatique basé sur le neurone artificiel et appelé perceptron : c'est le premier réseau neuronal bien que constitué d'un seul neurone. Le perceptron est un modèle simple et adapté pour des problèmes linéairement séparables. Pour résoudre des problématiques plus complexes comme par exemple le problème XOR, le perceptron simple ne suffit plus [Minsky 1969]. De ce fait, un réseau de neurones plus complexe a été introduit : le perceptron multicouches (PMC).

2.3.2 Le perceptron multicouches

Un perceptron multicouches est constitué de trois types de couches :

- Une couche d'entrée qui correspond aux données d'entrée $\mathbf{x} = [x_1, \dots, x_n]^T$. Cette couche ne contient pas de neurones.
- Une couche de sortie constituée de K neurones et produisant les sorties du réseau $\mathbf{y} = [y_1, \dots, y_K]^T$, c'est-à-dire les valeurs de sortie associées aux données d'entrée \mathbf{x} .
- Des couches cachées constituées chacune de plusieurs neurones. Ces couches permettent la transformation non-linéaire du signal d'entrée vers le signal de sortie.

Dans le cadre du PMC, tous les neurones d'une couche sont connectés aux neurones de la couche précédente. Le schéma 2.5 illustre un perceptron multicouches constitué de trois couches cachées.

2.3.3 Apprentissage du perceptron multicouches

Pour connaître les paramètres de ce réseau, c'est-à-dire les poids et les biais de chaque neurone formel qui le constitue, il faut utiliser une base de données d'apprentissage fournissant des données d'entrée et les sorties désirées leur correspondant. Le calcul de ces paramètres est appelé phase d'apprentissage et se

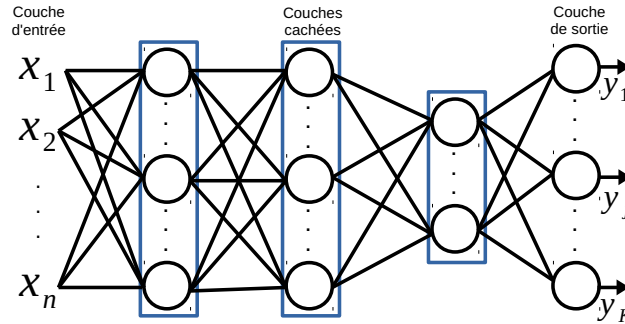


FIGURE 2.5 – Un exemple de perceptron multicouches constitué de trois couches cachées. Chaque cercle représente un neurone formel. Les neurones dans le même rectangle font partie de la même couche cachée.

fait grâce à l’algorithme de rétropropagation du gradient introduit la même année par [Rumelhart 1986, Lecun 1986]. Avant l’apprentissage, les paramètres du réseau sont inconnus et initialisés avec des valeurs aléatoires. Pour l’apprentissage, il faut considérer un sous-ensemble de la base d’apprentissage (appelé *batch*) composé de N exemples $\{(\mathbf{x}^t, \mathbf{r}^t)\}_{t=1}^N$ avec $\mathbf{x}^t = [x_1^t, \dots, x_n^t]^\top$ une donnée d’entrée et $\mathbf{r}^t = [r_1^t, \dots, r_K^t]^\top$ la sortie désirée lui correspondant. Ainsi, $(\mathbf{x}^t, \mathbf{r}^t)$ est l’exemple t du *batch*. Nous formalisons ici l’apprentissage du PMC pour un problème multi-classes (K classes). En d’autres termes, $r_j^t = 1$ si \mathbf{x}^t appartient à la classe j , et $r_j^t = 0$ sinon.

Nous considérons ici, un PMC à K neurones de sortie. Les paramètres d’un neurone formel j seront notés respectivement $w_{j,i}$ et b_j pour ses poids et son biais. Le poids $w_{j,i}$ correspond au poids de connexion du neurone j au neurone i de la couche précédente. La fonction d’activation f utilisée pour chaque neurone formel du PMC est la fonction différentiable sigmoïde soit :

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

Nous verrons dans la section 2.4 que cette fonction d’activation peut être de différentes natures. Nous noterons la réponse du neurone formel j , $y_j^t = f(a_j^t) = f(\sum_{i=1}^R w_{j,i} y_i^t + b_j)$ avec R le nombre de neurones de la couche précédente. En passant la donnée \mathbf{x}^t dans le PMC, l’erreur observée sur le neurone j de la couche de sortie s’écrit :

$$e_j^t = \mathcal{L}(r_j^t, y_j^t) \quad (2.3)$$

avec y_j^t la valeur renvoyée par le neurone de sortie j . \mathcal{L} est appelée fonction de perte et représente ce que le réseau essaie de minimiser sur l’ensemble des données.

Cette fonction est différente suivant le problème à résoudre. Dans le cas du PMC, la fonction de perte est définie comme suit :

$$\mathcal{L}(r_j^t, y_j^t) = r_j^t - y_j^t \quad (2.4)$$

L'erreur quadratique observée pour la donnée \mathbf{x}^t sur les K neurones de la couche de sortie s'écrit :

$$E^t = \frac{1}{2} \sum_{j=1}^K (e_j^t)^2 \quad (2.5)$$

et l'erreur quadratique moyenne sur l'ensemble du *batch* est définie par :

$$E = \frac{1}{N} \sum_{t=1}^N E^t \quad (2.6)$$

La mise à jour des poids du réseau se fait par descente de gradient stochastique (SGD) de l'erreur quadratique moyenne. En d'autres termes, le poids $w_{j,i}$ du neurone j est mis à jour en lui ajoutant le terme $-\alpha \Delta w_{j,i}$. La mise à jour du biais b_j du neurone j est effectuée en lui ajoutant le terme $-\alpha \Delta b_j$. α est appelé taux d'apprentissage (*learning rate*). Il permet de pondérer la mise à jour des paramètres du réseau. Les deux termes $\Delta w_{j,i}$ et Δb_j sont définis comme suit :

$$\Delta w_{j,i} = \frac{\partial E}{\partial w_{j,i}} = \frac{1}{N} \sum_{t=1}^N \frac{\partial E^t}{\partial w_{j,i}} \quad \Delta b_j = \frac{\partial E}{\partial b_j} = \frac{1}{N} \sum_{t=1}^N \frac{\partial E^t}{\partial b_j} \quad (2.7)$$

Dans ce qui suit, nous détaillons la rétropropagation du gradient suivant le type de couche considéré (couche de sortie ou couche cachée).

2.3.3.1 Rétropropagation pour la couche de sortie

En reprenant l'équation 2.7 et grâce à la décomposition en chaîne des dérivées partielles, nous pouvons écrire :

$$\frac{\partial E^t}{\partial w_{j,i}} = \frac{\partial E^t}{\partial e_j^t} \frac{\partial e_j^t}{\partial y_j^t} \frac{\partial y_j^t}{\partial a_j^t} \frac{\partial a_j^t}{\partial w_{j,i}} \quad \frac{\partial E^t}{\partial b_j} = \frac{\partial E^t}{\partial e_j^t} \frac{\partial e_j^t}{\partial y_j^t} \frac{\partial y_j^t}{\partial a_j^t} \frac{\partial a_j^t}{\partial b_j} \quad (2.8)$$

Ces dérivées partielles peuvent individuellement s'exprimer comme suit :

$$\frac{\partial E^t}{\partial e_j^t} = \frac{\partial}{\partial e_j^t} \frac{1}{2} \sum_{j=1}^K (e_j^t)^2 = e_j^t \quad (2.9)$$

$$\frac{\partial e_j^t}{\partial y_j^t} = \frac{\partial}{\partial y_j^t} (r_j^t - y_j^t) = -1 \quad (2.10)$$

$$\frac{\partial y_j^t}{\partial a_j^t} = \frac{\partial}{\partial a_j^t} \frac{1}{1 + e^{-a_j^t}} = \frac{e^{-a_j^t}}{(1 + e^{-a_j^t})^2} = y_j^t(1 - y_j^t) \quad (2.11)$$

$$\frac{\partial a_j^t}{\partial w_{j,i}} = \frac{\partial}{\partial w_{j,i}} \sum_{i=1}^R w_{j,i} y_i^t + b_j = y_i^t \quad (2.12)$$

$$\frac{\partial a_j^t}{\partial b_j} = \frac{\partial}{\partial b_j} \sum_{i=1}^R w_{j,i} y_i^t + b_j = 1 \quad (2.13)$$

Ce qui permet d'obtenir :

$$-\alpha \Delta w_{j,i} = \frac{\alpha}{N} \sum_{t=1}^N \delta_j^t y_i^t \quad -\alpha \Delta b_j = \frac{\alpha}{N} \sum_{t=1}^N \delta_j^t \quad (2.14)$$

en posant

$$\delta_j^t = e_j^t y_j^t (1 - y_j^t) \quad (2.15)$$

2.3.3.2 Rétropropagation pour les couches cachées

Dans le cas de la couche cachée précédant la couche de sortie, l'erreur e_j^t du neurone caché j est inconnue. Pour ce type de couche, la dérivée partielle de l'erreur quadratique de l'équation 2.7 s'écrit comme suit :

$$\frac{\partial E^t}{\partial w_{j,i}} = \frac{\partial E^t}{\partial y_j^t} \frac{\partial y_j^t}{\partial a_j^t} \frac{\partial a_j^t}{\partial w_{j,i}} \quad \frac{\partial E^t}{\partial b_j} = \frac{\partial E^t}{\partial y_j^t} \frac{\partial y_j^t}{\partial a_j^t} \frac{\partial a_j^t}{\partial b_j} \quad (2.16)$$

La dérivée partielle $\frac{\partial E^t}{\partial y_j^t}$ peut s'exprimer ainsi :

$$\begin{aligned} \frac{\partial E^t}{\partial y_j^t} &= \frac{\partial}{\partial y_j^t} \frac{1}{2} \sum_k (e_k^t)^2 = \sum_k e_k^t \frac{\partial e_k^t}{\partial y_j^t} = \sum_k e_k^t \frac{\partial e_k^t}{\partial a_k^t} \frac{\partial a_k^t}{\partial y_j^t} \\ &= \sum_k e_k^t \frac{\partial (r_k^t - y_k^t)}{\partial a_k^t} \frac{\partial (\sum_l w_{k,l} y_l^t + b_k)}{\partial y_j^t} \\ &= \sum_k e_k^t (-y_k^t (1 - y_k^t)) w_{k,j} \end{aligned} \quad (2.17)$$

Nous obtenons donc :

$$-\alpha\Delta w_{j,i} = \frac{\alpha}{N} \sum_{t=1}^N \delta_j^t y_i^t \quad -\alpha\Delta b_j = \frac{\alpha}{N} \sum_{t=1}^N \delta_j^t \quad (2.18)$$

avec :

$$\delta_j^t = y_j^t(1 - y_j^t) \sum_k \delta_k w_{k,j} \quad (2.19)$$

Le terme δ_k est le gradient défini dans 2.15. L'équation 2.19 se généralise à toutes les couches cachées. Le gradient δ_j^t d'un neurone j d'une couche cachée, utilise le gradient de la couche qui le suit.

2.3.3.3 Règle du delta généralisé

La mise à jour des paramètres du réseau se fait en utilisant la règle du delta généralisé. Nous présentons ici la mise à jour standard des poids par descente de gradient stochastique (*Stochastic gradient descent* SGD) avec *momentum*. Pour une itération τ , permettant de passer dans le réseau un *batch* de données, la mise à jour se fait comme suit :

Pour les poids :

$$w_{j,i}(\tau) = w_{j,i}(\tau - 1) - \alpha\Delta w_{j,i}(\tau) + \beta\Delta w_{j,i}(\tau - 1) \quad (2.20)$$

Pour les biais :

$$b_j(\tau) = b_j(\tau - 1) - \alpha\Delta b_j + \beta\Delta b_j(\tau - 1) \quad (2.21)$$

avec β appelé *momentum* (valeur entre 0 et 1) permettant de donner une inertie à la descente de gradient en prenant en compte les corrections appliquées à l'itération précédente $\tau - 1$. Il existe d'autres règles que la SGD avec *momentum* pour l'optimisation des réseaux de neurones : ADADELTA [Zeiler 2012], ADAGRAD [Duchi 2011], ADAM [Kingma 2015], RMSProp [Tieleman 2012]. Ces règles d'optimisation permettent d'adapter le taux d'apprentissage aux paramètres afin de mieux converger tout en étant plus rapide.

2.4 Les réseaux de neurones convolutifs profonds

Le premier réseau de neurones convolutif (CNN) a été introduit à la fin des années 80 par [LeCun 1989]. C'est le premier réseau de neurones pour la reconnaissance d'images. Ce réseau permettait la reconnaissance de chiffres manuscrits.

L'idée est de passer l'image dans une succession de filtres convolutifs apportant une description réduite et pertinente de l'image. Ces caractéristiques sont, par la suite, envoyées à un perceptron multicouches composé de couches cachées et d'une couche de sortie complètement connectées permettant la classification du chiffre présent dans l'image. Les filtres de convolution et les couches complètement connectées sont appris simultanément. La Figure 2.6 illustre l'architecture d'un petit réseau convolutif. Les CNN sont un type particulier de réseaux de neurones applicables facilement à des images pour capter spatialement de l'information. De plus, de par leur structure convolutive, ils permettent de prendre en entrée des données de grande dimension ce qui est une limite du perceptron multicouches. Une image à trois canaux (RGB) de taille 224×224 pixels représente un vecteur d'entrée de taille 150 528 pour un perceptron multicouches. Cela implique 150 528 poids à apprendre pour chaque neurone de la couche cachée connectée aux entrées, ce qui est compliqué à apprendre. Les CNN peuvent être vus comme un assemblage de modules en série permettant l'extraction de caractéristiques de manière hiérarchique à partir des pixels d'une image.

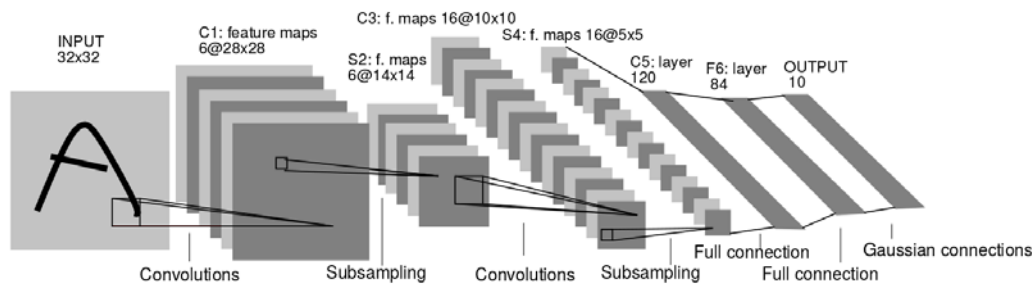


FIGURE 2.6 – Un petit réseau convolutif pour la reconnaissance de chiffres manuscrits. Source [Lecun 1998]

2.4.1 Différents modules d'un réseau de neurones convolutif

Nous présentons ici les différents modules utilisés dans les CNN : les convolutions, l'agglomération (*pooling*), les fonctions d'activation, le dropout, la batch normalization et les fonctions d'erreur classiques utilisées pour l'apprentissage.

La convolution. La pièce maîtresse d'un CNN est la couche convolutive. La sortie en résultant est appelée carte de caractéristiques. Une couche convolutive est constituée de plusieurs filtres (ou noyaux) de convolution à appliquer sur une

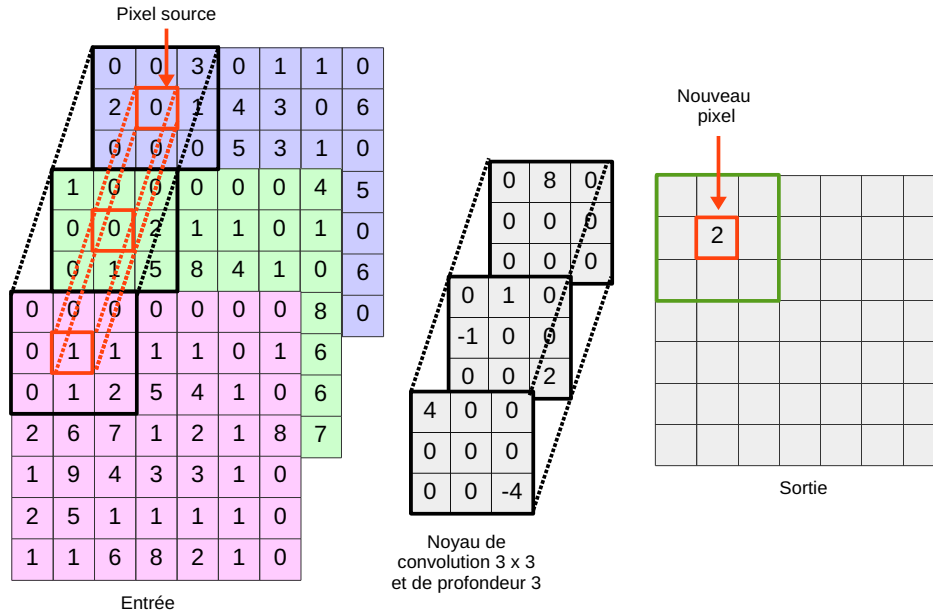


FIGURE 2.7 – Illustration de la convolution. Étant donné une entrée, un filtre de convolution (ou noyau de convolution) est appliqué pour chaque position. La profondeur du noyau dépend de la profondeur de l’entrée sur laquelle il est appliqué : dans cet exemple l’entrée a trois canaux donc la profondeur du noyau est de trois. Le résultat pour une position donnée correspond à la somme de la multiplication des éléments du noyau par ceux de l’entrée : dans cet exemple $2 \times 4 + 5 \times 2 = 2$. Dans le cadre des CNN, la sortie d’une convolution est appelée carte de caractéristiques. Le nombre de cartes de caractéristiques dépend du nombre de filtres appliqués sur l’entrée.

matrice d’entrée (une image ou une carte de caractéristique précédente). Soit une entrée I de taille $W \times H \times C$ et un noyau de convolution G de taille $K \times K \times C$. La sortie de la convolution de I par G s’écrit I' et est de taille $(W - \frac{K-1}{2}) \times (H - \frac{K-1}{2})$. La valeur de I' à la position (i, j) se calcule comme suit :

$$I'_{i,j} = \sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} \sum_{c=0}^{C-1} G_{k_1,k_2,c} I_{\frac{K-1}{2}+k_1,j \quad \frac{K-1}{2}+k_2,c} \quad (2.22)$$

La Figure 2.7 rappelle le fonctionnement de la convolution. En pratique, une valeur (appelé biais) associée au filtre de convolution, est ajoutée en chaque position de la sortie du filtre. Durant l’apprentissage, ce sont les valeurs des poids et des biais des neurones composant ces filtres qui sont appris. Ces filtres permettent d’extraire des caractéristiques locales contrairement au perceptron

multicouches où les réponses résultantes d'une couche cachée sont extraites sur la globalité des données (du fait des couches complètement connectées). Un filtre d'une couche convolutive est appliqué à toutes les positions de la matrice d'entrée, c'est pour cela que l'on parle de poids partagés. L'idée sous-jacente est que des caractéristiques semblables peuvent être trouvées à des endroits différents dans l'image.

Le pooling. La couche pooling (ou agglomération) permet de rajouter de l'invariance spatiale lors de l'extraction de caractéristiques tout en réduisant la dimension des entrées. Elle peut être de différentes natures mais les types de *pooling* les plus utilisés sont le *Max Pooling* (illustré dans la Figure 2.8) et l'*Average pooling*. Le *Max Pooling* renvoie l'élément maximum sur une fenêtre de calcul. L'*Average pooling* permet de renvoyer la moyenne des éléments sur une fenêtre de calcul.

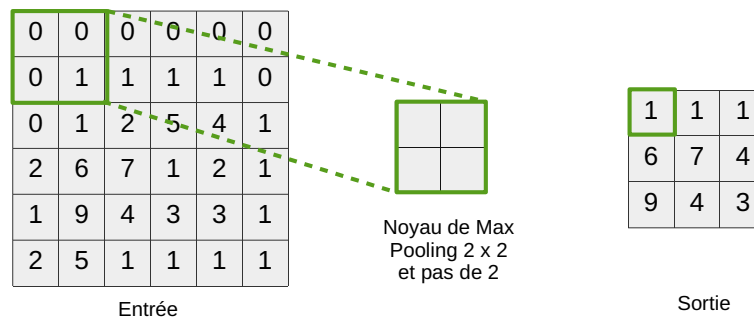


FIGURE 2.8 – Illustration du Max Pooling. Dans cet exemple, le noyau de pooling est de taille 2×2 et est appliqué tous les deux pixels ($stride = 2$). Le maximum des quatre éléments sur une fenêtre de l'entrée est gardé.

Les fonctions d'activation. Il existe différentes fonctions d'activation permettant la non-linéarité dans les différentes couches des CNN. Parmi les plus connues :

- La fonction sigmoïde utilisée notamment dans le perceptron multicouches original (voir section 2.3)

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.23)$$

- La tangente hyperbolique

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.24)$$

- La fonction d'unité de rectification linéaire (ReLU).

$$f(x) = \max(0, x) \quad (2.25)$$

Celle-ci est certainement la plus utilisée dans les CNN profonds car elle permet une optimisation plus facile. Elle a pour avantage de fournir des réponses parcimonieuses (*sparse*) et permet de réduire les problèmes de disparition de gradient. En effet, la sigmoïde et la tangente hyperbolique ont pour inconvénient de renvoyer des gradients très petits lorsque la valeur absolue de l'entrée est grande. Le réseau a alors du mal à mettre à jour ses paramètres lors de la phase d'apprentissage. La fonction ReLU renvoie quant à elle, un gradient constant pour une entrée grande permettant ainsi d'apprendre plus rapidement (en particulier les réseaux d'une certaine profondeur). Il existe d'autres fonctions d'activation de la même famille que les ReLU comme les LReLU [Maas 2013], les PReLU [He 2015] et les eLU [Clevert 2016].

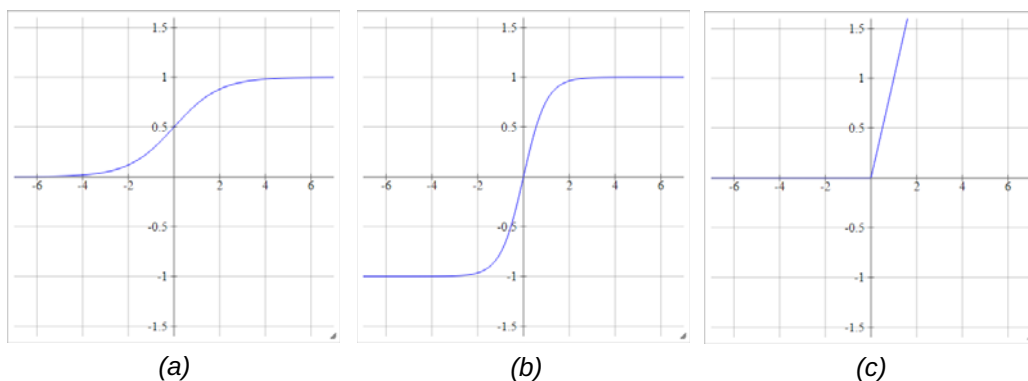


FIGURE 2.9 – Trois fonctions d'activation. (a) La sigmoïde, (b) La tangente hyperbolique, (c) La fonction ReLU

Le dropout. Pour éviter le sur-apprentissage (*overfitting*), la couche de dropout a été introduite [Srivastava 2014]. Cette couche est utilisée pendant l'apprentissage. Elle permet de désactiver aléatoirement des neurones durant les différentes itérations de l'apprentissage. En d'autres termes, le dropout permet au réseau d'apprendre des sous-réseaux contenant moins de paramètres et donc moins sujets au sur-apprentissage. Cette manière de faire permet d'apprendre des paramètres plus génériques qui ne se focalisent pas sur des détails de la base d'apprentissage. Une fois l'apprentissage terminé, tous les neurones sont réactivés.

La batch normalization. Cette technique a été introduite par [Ioffe 2015] afin d'apprendre les CNN de manière plus rapide et efficace. Elle part de l'observation suivante : pendant l'apprentissage, la distribution des entrées des différentes couches du réseau change à chaque itération. Cela induit une adaptation permanente des paramètres du CNN à ces différentes distributions ce qui augmente le temps d'apprentissage. L'idée de la batch normalization est de normaliser les entrées de chaque couche afin que les distributions de celles-ci soient de moyenne nulle et de variance unitaire. Durant l'apprentissage, les couches de batch normalization apprennent des paramètres (un facteur d'échelle et un biais) permettant d'ajuster cette normalisation : ces paramètres permettent d'appliquer une transformation sur la distribution normalisée. Autrement dit, durant l'apprentissage, si le réseau considère que la distribution normalisée n'est pas adaptée pour une couche donnée, il apprend les paramètres permettant de l'ajuster.

Les fonctions de perte. Il existe plusieurs fonctions de perte (ou fonctions d'objectif) utilisables pour l'apprentissage des réseaux de neurones. Ces fonctions sont dépendantes de la tâche que le réseau doit effectuer (classification, régression...). Nous listons ici, les fonctions de perte les plus utilisées.

- La fonction de perte *Softmax*, généralement utilisée pour l'optimisation de réseau de classification d'images. Elle permet la maximisation de la probabilité qu'a une entrée d'appartenir à une classe plutôt qu'à une autre. Soit un vecteur de sortie prédit par le CNN $\mathbf{y}^* = [y_1^*, \dots, y_K^*]^\top$ et la classe désirée i . La perte *Softmax* est définie par :

$$E(\mathbf{y}^*, i) = -\log \left(\frac{e^{y_i^*}}{\sum_j e^{y_j^*}} \right) \quad (2.26)$$

- La fonction de perte par entropie croisée sigmoïde, permettant une régression sur des probabilités. Soit y^* la probabilité prédite par le CNN et y la probabilité désirée, la fonction de perte par entropie croisée est définie par :

$$E(y^*, y) = -y \log(y^*) + (y - 1) \log(1 - y^*) \quad (2.27)$$

- La perte euclidienne, utilisée pour des problématiques de régression sur des valeurs réelles. Soit \mathbf{y}^* le vecteur de réels prédit par le CNN et \mathbf{y} le vecteur de réels désiré, la perte euclidienne est définie par :

$$E(\mathbf{y}^*, \mathbf{y}) = \frac{1}{2} \|\mathbf{y}^* - \mathbf{y}\|^2 \quad (2.28)$$

- La perte L1 lisse, introduite par [Girshick 2015] qui permet une meilleure optimisation pour les problèmes de régression :

$$E(\mathbf{y}^*, \mathbf{y}) = \sum_i \text{smooth}_{L1}(y_i^* - y_i) \quad (2.29)$$

avec

$$\text{smooth}_{L1}(x) = \begin{cases} \frac{1}{2}x^2, & \text{si } |x| < 1 \\ |x| - \frac{1}{2}, & \text{sinon} \end{cases} \quad (2.30)$$

2.4.2 Les architectures neuronales classiques

Nous présentons ici les architectures de réseaux convolutifs profonds utilisées couramment dans les travaux de recherche en vision par ordinateur. Il est important de remarquer que les architectures proposées dans la littérature ont une forte tendance à devenir de plus en plus profonde avec les années. Autrement dit, il semble que, plus le réseau est profond, plus les performances sont bonnes (cela est illustré dans la Figure 2.10). Néanmoins, cette profondeur implique de faire face à certaines difficultés notamment en termes de temps de calcul et d'optimisation durant l'apprentissage. C'est pourquoi la communauté reste très active sur la problématique de conception d'architectures neuronales.

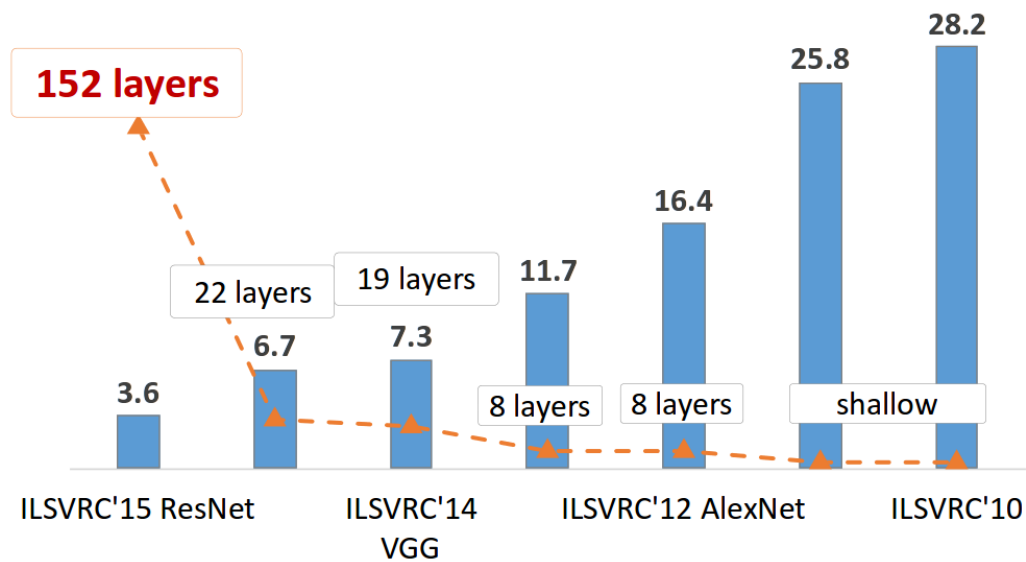


FIGURE 2.10 – Taux d'erreur de différentes architectures sur ImageNet pour la classification d'objets. Au fil des années, l'erreur de classification diminue, notamment grâce à la profondeur grandissante des architectures. Source [He 2016].

Des architectures standards sont abondamment utilisées en vision pour deux raisons principales. La première est qu'elles permettent d'inter-comparer facilement les méthodes basées sur les CNN. En d'autres termes, bien que certains travaux se focalisent sur l'étude des architectures neuronales, la majorité des méthodes de vision réutilisent des CNN déjà appris et les modifient pour concevoir de nouvelles architectures répondant à des tâches particulières. Ainsi, les architectures standards de CNN sont des *baseline* permettant de valider ou non la pertinence d'une idée. La seconde raison, est en lien avec la difficulté d'apprendre les réseaux profonds du fait de leur grand nombre de paramètres et du manque de données d'apprentissage. Une pratique commune est l'utilisation de CNN déjà appris sur d'immenses bases de données pour ensuite les adapter à une tâche spécifique. C'est ce que l'on appelle le *fine-tuning*. Cette pratique permet d'apprendre des réseaux profonds en utilisant une initialisation des poids et des biais déjà très pertinente et générique. L'adaptation de ces paramètres est ensuite réalisée durant la phase d'apprentissage de la tâche spécifique que l'on souhaite réaliser. Cela a pour incidence une vitesse d'apprentissage bien plus rapide et une convergence quasiment garantie.

AlexNet. Cette architecture est celle proposée par [Krizhevsky 2012] et ayant permis la recrudescence de l'étude des réseaux de neurones à partir de 2012, notamment, grâce à la victoire lors de la compétition de classification d'images ImageNet. Cette architecture utilise cinq couches de convolution et trois couches de *pooling*. La taille des noyaux de convolution est variable (11×11 , 5×5 , 3×3) en fonction de la couche considérée. La fonction d'activation utilisée entre chaque couche est la fonction ReLU. Après le passage de l'image dans les couches de convolution, de pooling et d'activation, une carte de caractéristiques est obtenue. Celle-ci est envoyée dans un perceptron multicouches composé de deux couches cachées et d'une couche de sortie. La Figure 2.11 (a) illustre l'architecture de l'AlexNet.

VGG. Ce CNN a été introduit par [Simonyan 2014]. Au lieu d'utiliser une seule convolution par niveau de profondeur comme AlexNet, cette architecture utilise des séquences de convolution. De plus, les filtres convolutifs sont de plus petite taille que dans AlexNet (noyau de taille 3×3). La Figure 2.11 (b) illustre l'architecture du VGG.

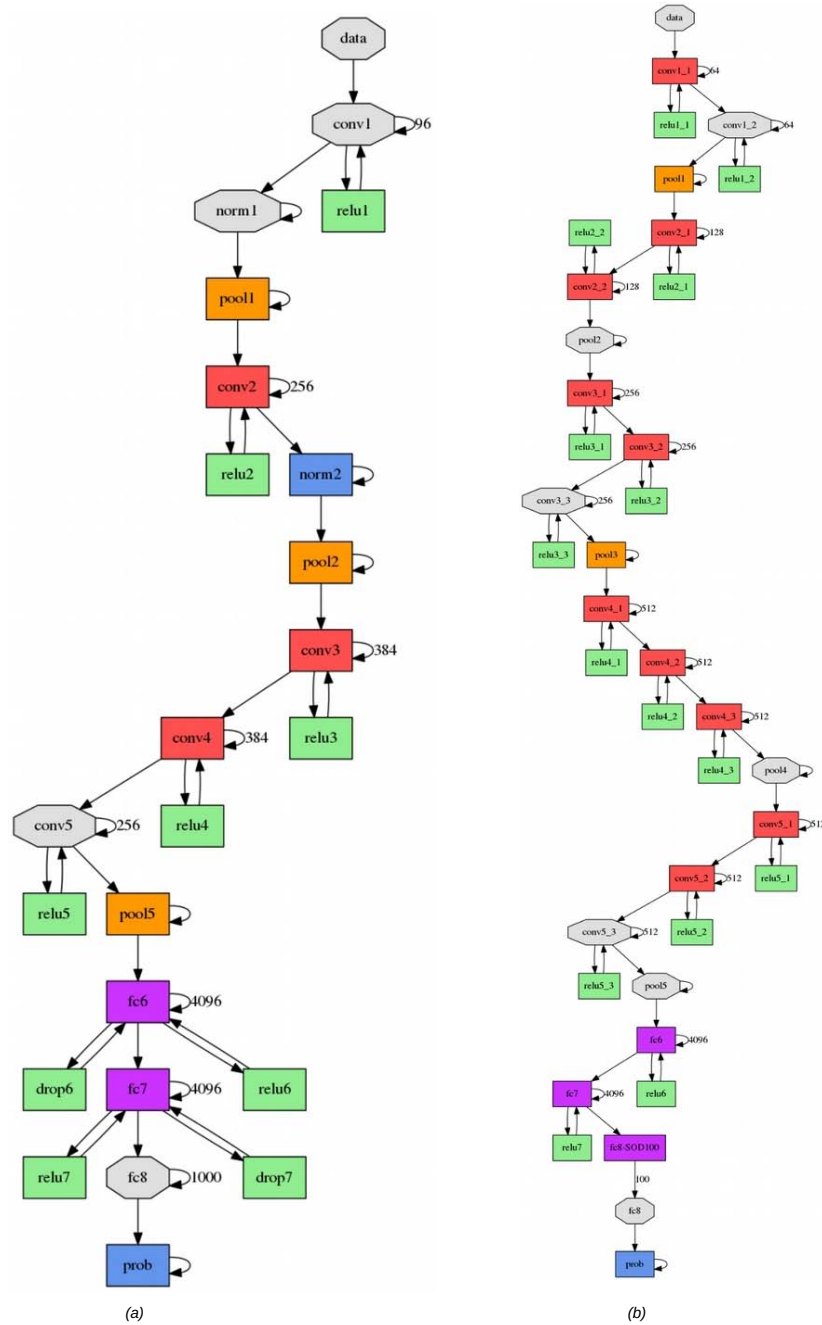


FIGURE 2.11 – Deux architectures classiques : (a) AlexNet [Krizhevsky 2012] et (b) VGG16 [Simonyan 2014]. Le VGG est constitué de plus de couches et est donc plus profond.

GoogLeNet. Cette architecture neuronale convolutive a été créée par [Szegedy 2015] et permet une réduction du temps de calcul par rapport à l'architecture VGG présentée précédemment. Pour cela, le GoogLeNet est composé de plusieurs couches appelées couches d'inception. Elles sont composées de plusieurs modules de convolution de taille 1×1 , 3×3 et 5×5 , exécutés en parallèle sur la carte de caractéristiques résultant de la couche précédente. Des filtres additionnels permettent de réduire la dimension des cartes de caractéristiques ce qui permet un gain important de temps de calcul. La Figure 2.12 illustre une couche d'inception et la Figure 2.13 représente l'architecture globale du GoogLeNet. D'autres modules d'inception ont par la suite été proposés notamment Inception V2 et V3 [Szegedy 2016].

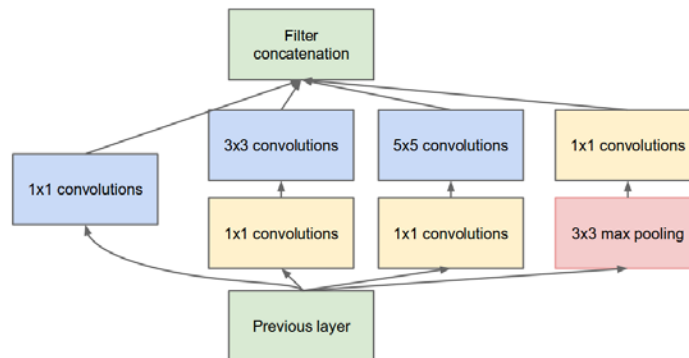


FIGURE 2.12 – Le module d'inception. Source [Szegedy 2015].

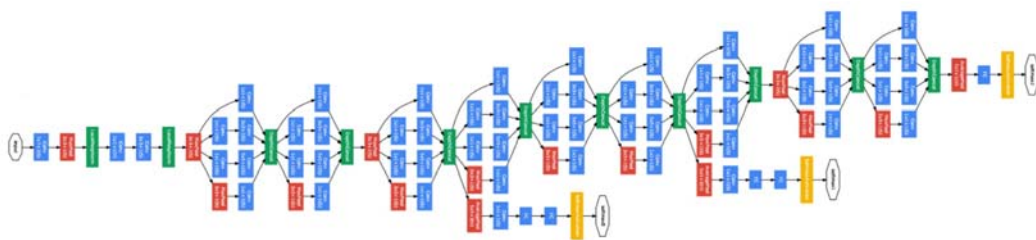


FIGURE 2.13 – L'Architecture GoogLeNet. Source [Szegedy 2015]

ResNet. Ce CNN a été introduit par [He 2016]. Il permet l'apprentissage de réseaux très profonds (plus de 150 couches). La difficulté à apprendre des réseaux aussi profonds est notamment liée à la rétropropagation du gradient. Plus le réseau est profond, plus le gradient est faible pour la mise à jour des poids des

couches de plus bas niveau (les premières couches). Ainsi une architecture trop profonde ne met pas vraiment à jour ces couches. L'idée développée dans ResNet est l'utilisation de connexions résiduelles permettant une meilleure optimisation des réseaux très profonds. Une connexion résiduelle permet de passer l'entrée dans deux filtres de convolution mais également de passer directement cette entrée aux couches suivantes. Cela est réalisé en additionnant le résultat des deux couches de convolution et l'entrée comme illustré dans la Figure 2.14. Avec cette architecture, les auteurs démontrent l'intérêt d'apprendre des réseaux très profonds de par leurs performances et proposent une manière pour les apprendre efficacement. La Figure 2.15 illustre l'architecture d'un réseau résiduel.

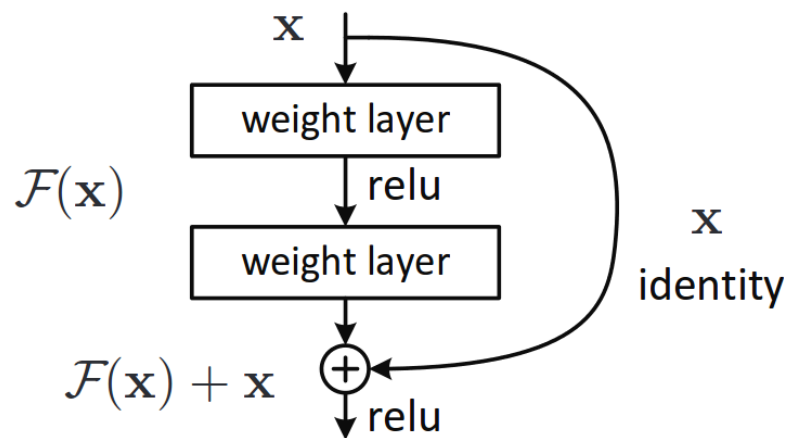


FIGURE 2.14 – Connexion résiduelle. Source [He 2016]

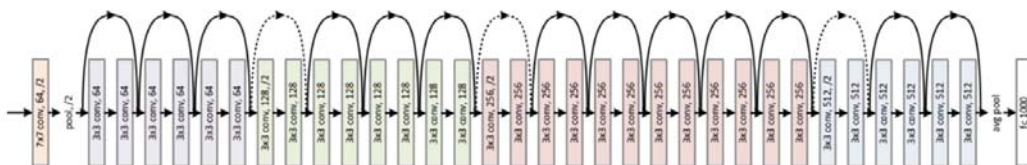


FIGURE 2.15 – L'architecture du ResNet (34 couches). Source [He 2016]

Dans le tableau 2.1, nous proposons un comparatif des paramètres des quatre architectures convolutives présentées ci-dessus.

	AlexNet	VGG16	GoogLeNet	ResNet50
Taille des filtres	3,5,11	3	1,3,5,7	1, 3, 7
Profondeur des filtres	3 - 256	3 - 512	3 - 1024	3 - 2048
Nombre de filtres par couche	96 - 384	64 - 512	64 - 384	64 - 2048
Nombre de couches convolutives	5	16	21	49
Nombre de couches FC	3	3	1	1
Nombre de paramètres	61M	138M	7M	25M

TABLE 2.1 – Comparaison de quatre réseaux de référence.

2.5 Conclusion

Dans ce chapitre, nous avons introduit les réseaux de neurones et plus particulièrement les réseaux neuronaux convolutifs profonds (CNN) utilisés abondamment tout au long des travaux de recherche présentés dans cette thèse. Ces modèles permettent d'apprendre, grâce à l'apprentissage supervisé, les caractéristiques visuelles discriminantes à extraire sur les images afin de résoudre des problématiques de classification et/ou de régression. De ce fait, ils sont particulièrement adaptés aux problématiques de l'analyse fine.

Aujourd'hui, les CNN sont au centre de l'attention, en particulier en vision par ordinateur, de par les résultats impressionnants qu'ils produisent sur différentes tâches. Cependant, ils ne sont pas parfaits et leur étude au sein de la communauté est très active, notamment sur les problématiques de complexité calculatoire, d'optimisation pendant l'apprentissage et de choix des hyper-paramètres. De plus, pour être appris efficacement, les CNN ont besoin d'être entraînés sur un grand nombre de données pour éviter le sur-apprentissage. Ces données ne sont pas toujours disponibles pour résoudre un problème spécifique.

CHAPITRE 3

État de l'art

Sommaire

3.1	Introduction	29
3.2	Détection d'objets	30
3.2.1	Détection par fenêtre glissante	31
3.2.2	Détection par proposition d'objets	34
3.3	Modèles de représentation d'objets	37
3.3.1	Représentation globale	38
3.3.2	Représentation par parties	38
3.3.3	Représentation multi-vues	40
3.3.4	Représentation de la visibilité	42
3.4	Analyse 3D des objets	43
3.5	Catégorisation et Classification fine	45
3.6	Métriques d'évaluation	48
3.6.1	Évaluation de la détection	48
3.6.2	Évaluation de l'estimation du point de vue	50
3.6.3	Évaluation de la localisation 3D	51
3.6.4	Évaluation de la classification fine	51
3.7	Positionnement de la thèse	52

3.1 Introduction

Ce chapitre a pour objectif de présenter les travaux en lien avec notre problématique, à savoir l'analyse fine d'objets dans une scène. Nous définissons l'analyse fine comme une caractérisation précise des objets dans une image. Ce terme regroupe les concepts suivants : la localisation des objets dans l'image et dans la scène 3D, l'estimation de leur point de vue, la caractérisation de leur visibilité et la

reconnaissance de sous-catégories (classification fine). Nous avons divisé cet état de l'art en plusieurs sections. La première se réfère aux techniques de détection d'objets, brique nécessaire pour l'analyse fine d'objets. La seconde s'intéresse aux différentes manières de représenter les objets. Ces représentations influent directement sur le degré de finesse que l'on souhaite atteindre. La troisième dresse un éventail des approches d'analyse 3D des objets. Dans une quatrième section, nous détaillons les travaux en lien avec la sous-catégorisation et la classification fine. Enfin, nous présentons les différentes métriques d'évaluation permettant de quantifier les performances de ces algorithmes, puis le positionnement de nos travaux par rapport à cet état de l'art.

3.2 Détection d'objets

La détection d'objets est une des problématiques les plus étudiées en vision par ordinateur. Son objectif est de trouver dans une image d'entrée, des régions (boîtes englobantes) contenant des objets. On peut diviser la détection d'objets en deux catégories : les détecteurs à une classe et les détecteurs multi-classes. Un détecteur à une classe se concentre sur la détection d'un seul type d'objet. Le détecteur doit être capable de décider si une région de l'image correspond à l'objet ou à du fond. Dans le cadre d'un détecteur multi-classes, le détecteur doit être capable de séparer le fond et les objets, tout en séparant les objets entre eux (c'est-à-dire décider de quel type d'objets il s'agit). Il existe de nombreuses bases de données publiques permettant d'entraîner et d'évaluer des détecteurs. On peut citer parmi elles, PascalVOC [Everingham 2010], SUN [Choi 2010], KITTI [Geiger 2012], MSCOCO [Lin 2014a] et ILSVRC [Deng 2009]. La figure 3.1 illustre l'objectif de la détection d'objets.

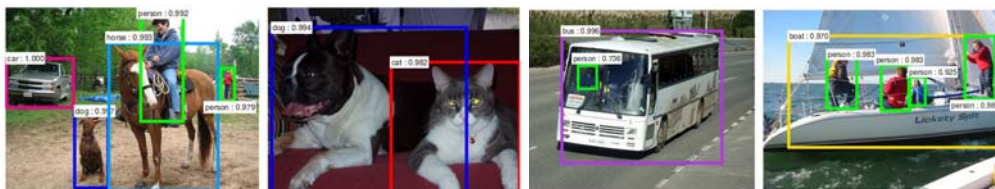


FIGURE 3.1 – Exemples de détections d'objets renvoyées par un détecteur multi-classes. Chaque couleur de boîte englobante correspond à une classe d'objets. Source [Ren 2015].

Les défis de la détection. Un détecteur d'objets doit faire face à plusieurs difficultés. La première est le temps de calcul : un bon détecteur est un détecteur qui maximise les performances tout en étant le plus rapide possible à exécuter. Cette notion de temps de calcul est particulièrement importante dans la conception de détecteurs multi-classes de par le plus grand nombre de classes d'objets à détecter. La seconde concerne l'apparence variable des objets. Celle-ci peut varier suivant plusieurs facteurs : l'occultation, le point de vue sous lequel les objets sont observés et la taille des objets dans l'image. Cette variabilité d'apparence est également une difficulté pour les systèmes de classification d'images. Une troisième difficulté est en lien avec la variation d'apparence des régions qui ne correspondent pas à des objets (le fond). Un détecteur doit être capable de décider si une région correspond à du fond, et cela, même dans des images provenant d'environnements complexes.

Le principe général de la détection. Un détecteur consiste en un modèle permettant de discriminer des régions dans l'image. Il peut être appris, par exemple, grâce à l'apprentissage supervisé. La détection consiste à extraire des régions dans l'image d'entrée et à leur appliquer ce modèle. Cette opération permet de renvoyer un score de confiance sur la région extraite modélisant la probabilité que la région contienne ou pas un objet. La manière d'extraire les régions est discriminante pour la rapidité et l'efficacité du détecteur : si le modèle est appliqué à toutes les régions possibles dans l'image, le temps de calcul devient énorme.

Nous proposons de détailler deux techniques standards pour l'extraction des régions sur lesquelles va être appliqué le modèle. En particulier, nous développons ici le fonctionnement de la détection par fenêtre glissante (*sliding window*) et de la détection par proposition d'objets.

3.2.1 Détection par fenêtre glissante

Les approches de type fenêtre glissante utilisent un modèle de classification préalablement appris qui consiste en une fenêtre 2D de taille fixe permettant de discriminer le fond des objets. L'idée générale de ce type d'approche est d'appliquer cette fenêtre sur toutes les positions de l'image d'entrée à tester, d'où l'appellation "glissante". Autrement dit, cette fenêtre vient parcourir l'image et produit en chaque position de celle-ci un score de confiance.

Le schéma classique de détection par fenêtre glissante, illustré dans la figure 3.2, est le suivant :

- Étant donné une image en entrée, une pyramide d'image est calculée. Ce calcul consiste à redimensionner l'image en utilisant plusieurs facteurs d'échelle et plusieurs ratios largeur/hauteur. L'ensemble de ces images forme la pyramide et chaque image correspond à un niveau de celle-ci. Il est nécessaire de calculer une pyramide d'images car la fenêtre à appliquer sur l'image est de taille fixe : le modèle de détection est généralement appris pour une seule échelle et un seul ratio largeur/hauteur. Or, les objets présents dans une image peuvent être de différentes tailles et présenter des ratios variables. De cette manière, les objets présents dans l'image correspondent à la taille du modèle au moins pour un niveau de la pyramide.
- Des caractéristiques visuelles sont extraites sur les différents niveaux de la pyramide ce qui permet d'obtenir une carte de caractéristiques pour chaque niveau pyramidal.
- Un modèle préalablement appris est appliqué en chaque position des différentes cartes de caractéristiques, renvoyant un score de confiance pour chaque position et chaque niveau de la pyramide. Cette étape permet de sélectionner des boîtes candidates (celles avec les plus hauts scores de confiance).
- Dans les systèmes de détection, il est très fréquent que des boîtes candidates soient agglomérées autour du même objet. En d'autres termes, plusieurs boîtes peuvent correspondre au même objet. Pour supprimer ces détections redondantes, un algorithme appelé suppression des non-maxima (NMS) est appliqué. L'idée de cet algorithme repose sur le fait que des détections ne peuvent pas se recouvrir spatialement au-delà d'un certain seuil (le seuil de recouvrement, *overlap* en anglais). Si des détections se recouvrent de manière trop importante, la détection avec le meilleur score de confiance est gardée, les autres sont supprimées.

Le schéma par fenêtre glissante a été largement utilisé pour la détection d'objets. Les différentes méthodes qui l'utilisent se différencient par la nature du modèle à appliquer sur la pyramide et les caractéristiques visuelles utilisées (caractéristiques de formes [Ferrari 2010, Ferrari 2007], HOG [Dalal 2005, Felzenszwalb 2010], caractéristiques profondes [Sermanet 2014, Szegedy 2013]...)

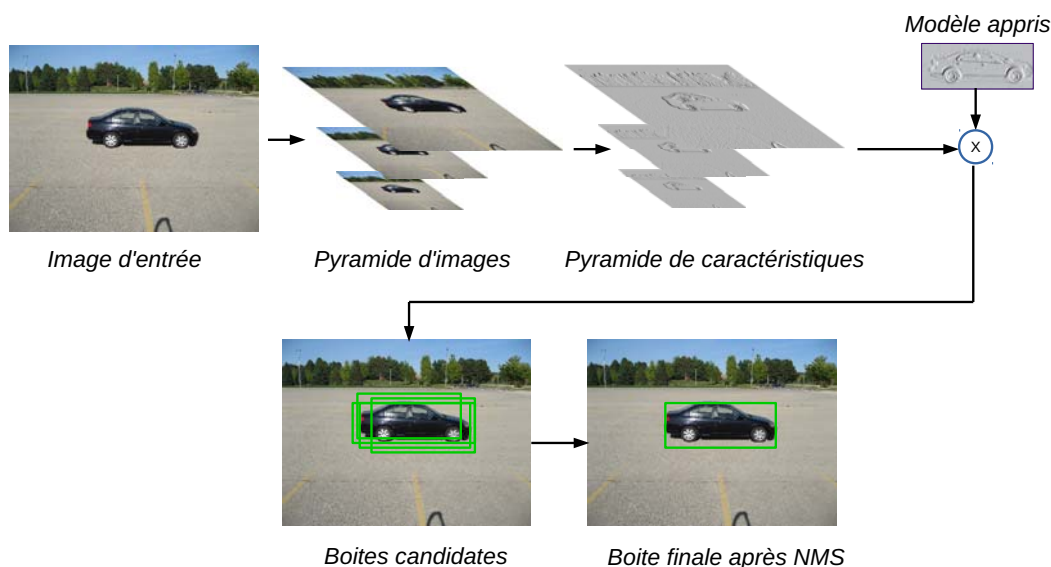


FIGURE 3.2 – Schéma classique de détection d'objets par fenêtre glissante. Une pyramide d'images est créée à partir d'une image d'entrée. Des caractéristiques visuelles sont calculées sur cette pyramide d'image. Un modèle est appliqué en chaque position et chaque niveau de pyramide renvoyant des boîtes candidates. Une suppression des non maxima est alors appliquée (NMS) pour supprimer les boîtes correspondant au même objet.

ainsi que le modèle de représentation des objets choisi (un aperçu des différentes manières de représenter les objets sera exposé dans la section 3.3).

La fenêtre glissante est devenue incontournable, notamment après la publication de [Viola 2001]. Les auteurs y introduisent une approche basée sur le principe du *boosting* [Freund 1997]. L'idée est de scanner l'image avec des classifieurs dits "faibles" appelés filtres de Haar. La somme des réponses de ces classifieurs faibles permet la prise de décision du détecteur. Cette méthode a été longtemps considérée comme l'état de l'art en détection de visages. L'avantage majeur de celle-ci est le temps de calcul. En effet, les caractéristiques renvoyées par les filtres de Haar sont très rapides à calculer notamment grâce aux images intégrales. De plus, dans cette méthode, les classifieurs faibles sont utilisés en cascade. En d'autres termes, ils permettent, dans un premier temps, de supprimer rapidement des régions qui ne contiennent pas d'objets. Pour les régions les plus problématiques, l'agrégation des classifieurs faibles permet de construire des classifieurs de plus en plus robustes. Des extensions basées sur le *boosting* ont également été introduites notamment pour résoudre le problème de détection multi-classes

[Torralba 2004, Torralba 2007]. Ces approches proposent de partager des caractéristiques visuelles entre les classes d'objets à détecter.

En 2005, les auteurs de [Dalal 2005] proposent de nouvelles caractéristiques visuelles : les HOG. Ces caractéristiques, basées sur les gradients de l'image ont permis de faire un bond en avant dans les performances des systèmes de détection d'objets. Les auteurs proposent d'utiliser les HOG et les séparateurs à vaste marge (SVM) pour séparer les exemples d'apprentissage dans l'espace des caractéristiques. Les HOG et les SVM ont également été utilisés dans l'approche DPM [Felzenszwalb 2010] (*Deformable Part models*) où le modèle de détection est basé sur des représentations locales et globales des objets. Cette méthode est restée quelques années à l'état de l'art en détection de personnes. Le modèle utilisé dans le DPM sera expliqué plus en détail dans la section 3.3.2.

Des approches basées fenêtre glissante et CNN ont également été introduites. Parmi elles, on peut citer [Garcia 2002] et plus récemment [Sermanet 2014, Szegedy 2013]. Dans [Sermanet 2014], les auteurs proposent de transformer un réseau de neurones standard en remplaçant les couches complètement connectées par des couches convolutives. Cela permet l'application du réseau de neurones sur n'importe quelle taille d'image. Cette manière de faire est très intéressante notamment pour passer dans le réseau différents niveaux d'une pyramide d'images. Le réseau complètement convolutif est entraîné pour renvoyer des scores de confiance pour chaque classe d'objets ainsi que les quatre coins de leur boîte englobante. Dans [Szegedy 2013], les auteurs proposent d'apprendre un réseau de régression permettant de renvoyer les masques des boîtes englobantes des objets.

3.2.2 Détection par proposition d'objets

Une alternative à la recherche exhaustive des objets (la fenêtre glissante) est l'utilisation d'algorithmes de proposition d'objets. Récemment, la proposition d'objets a permis d'améliorer les performances et le temps d'exécution dans les systèmes de détection d'objets. L'objectif de ces méthodes est de proposer des boîtes avec une forte probabilité d'être un objet d'intérêt. Ces boîtes sont ensuite extraites et envoyées à un classifieur pour la décision finale. Ces méthodes réduisent le temps d'exécution car elles diminuent considérablement l'espace de recherche par rapport aux approches de recherche exhaustive de type fenêtre glissante : le modèle permettant de détecter les objets n'est pas appliqué sur toutes les positions de l'image (ni sur tous les niveaux de la pyramide) mais seulement

sur un ensemble réduit de régions. Dans ce qui suit, nous présentons les méthodes existantes permettant la génération de propositions d'objets.

Un des algorithmes pour la proposition d'objets 2D est la *selective search* introduit par [Uijlings 2013]. Comme d'autres approches [Carreira 2010, Endres 2010], cette méthode se base sur une segmentation de l'image à différentes résolutions. En utilisant la méthode de segmentation introduite par [Felzenszwalb 2004], la *selective search* segmente l'image d'entrée sur plusieurs échelles. Cela produit un premier ensemble de régions d'intérêts. Les auteurs de [Uijlings 2013] introduisent ensuite un calcul de similarité entre régions basé sur des informations de couleurs, de textures, de tailles et d'inclusions. Cette similarité permet de fusionner les régions redondantes (trop similaires) et de renvoyer un ensemble de propositions d'objets pertinent. La figure 3.3 illustre l'algorithme de *selective search*. Cette méthode a été utilisée par la suite dans deux travaux de références pour la détection d'objets par réseau de neurones convolutif : le RCNN [Girshick 2014] et le Fast-RCNN [Girshick 2015]. Dans le RCNN [Girshick 2014], les propositions d'objets provenant de la *selective search* sont extraites dans l'image et redimensionnées à taille fixe. Ces régions sont ensuite envoyées à un CNN pour déterminer leur classe. Cette approche est très coûteuse en temps de calcul (pour l'apprentissage et pour le test) car chaque région passe dans toutes les couches du CNN. C'est pour cette raison que le Fast-RCNN [Girshick 2015] a été introduit. Il permet d'extraire les régions provenant de la *selective search* sur une carte de caractéristiques profonde. En d'autres termes, l'image d'entrée entière est passée dans un réseau de neurones convolutif fournissant une carte de caractéristiques à basse résolution (du fait des *Pooling* successifs). Les propositions d'objets sont alors extraites sur cette carte et envoyées à un classifieur consistant généralement en deux couches cachées (complètement connectées) et une couche de sortie permettant la classification de l'objet (classe de l'objet ou fond). Dans ces deux approches, une fonction supplémentaire est apprise par le réseau permettant de transformer les propositions d'objets originelles produites par la *selective search* afin que celles-ci collent au mieux à l'objet. Cette fonction est appelée régression sur les boîtes.

Afin d'encore réduire le temps de calcul et gagner en performances, le réseau de proposition d'objets [Ren 2015] (*Region Proposal Network* RPN) a été introduit. Les auteurs proposent la création d'un seul CNN capable de proposer des objets d'intérêt, de les extraire sur une carte de caractéristiques et de classifier

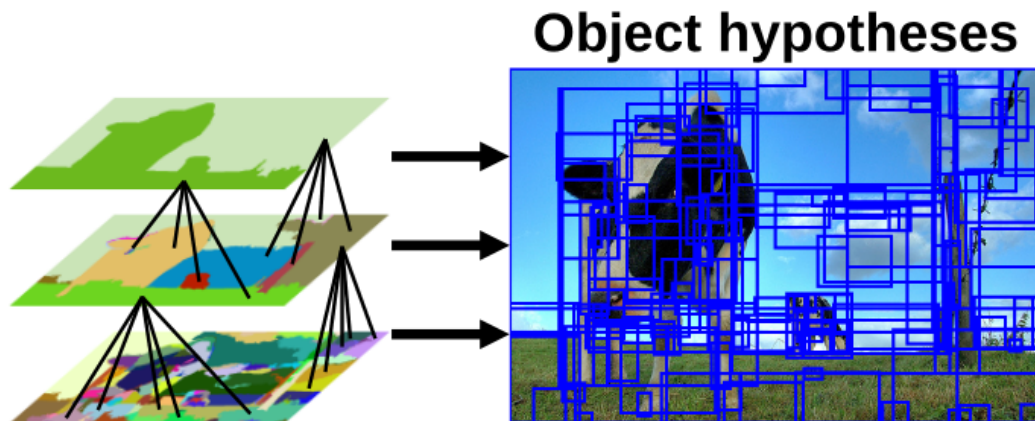


FIGURE 3.3 – Illustration de la *selective search*. À gauche, des cartes de segmentation à différentes résolutions. À droite, les différentes propositions d'objets renvoyées par l'algorithme après la fusion des régions. Source du schéma [Uijlings 2013].

chaque région. Cette méthode pour la détection d'objets, a été largement utilisée et modifiée [Yang 2016, Xiang 2017, Kong 2016, Dai 2016] de par ses très bonnes performances pour la détection multi-classes et sa rapidité d'exécution. Un travail très récent [He 2017] l'utilise même pour la segmentation d'instances et l'estimation de la pose de personnes. L'approche de [Ren 2015] sera expliquée plus en détail dans le chapitre 6 qui utilise ce type de réseau de proposition d'objets. Récemment, d'autres algorithmes de détection par CNN ont été introduits [Liu 2016, Redmon 2016]. Ils se basent sur le concept *one-shot* c'est-à-dire qu'ils n'utilisent plus d'étape d'extraction des propositions d'objets sur les cartes de caractéristiques. Cela permet d'économiser encore du temps de calcul lors de l'utilisation du CNN sur l'image. Un article très intéressant [Huang 2016] fournit une analyse très poussée des différents détecteurs de l'état de l'art basés CNN [Liu 2016, Ren 2015, Dai 2016] en testant différentes architectures convolutives.

Dans le cadre de la conduite autonome, deux travaux ont proposé de générer des propositions d'objets directement en 3D [Chen 2015, Chen 2016]. Dans ces deux approches, l'espace 3D est discrétisé (en localisation 3D par rapport à la caméra et en orientation) permettant de générer des boîtes 3D sur l'ensemble du plan du sol supposé planaire. L'approche 3DOP [Chen 2015] (*3D Object Proposal*) utilise des images stéréo permettant le calcul d'une carte de disparité. De cette manière, un

score de confiance est attribué à chaque boîte 3D en utilisant des heuristiques sur les points 3D contenus dans ces boîtes. Un score important est attribué aux boîtes 3D qui répondent à un certain nombre de critères : la densité du nuage de points 3D qu'elles contiennent mais également des contraintes géométriques sur celui-ci. Dans l'approche Mono3D [Chen 2016], les auteurs proposent une génération de propositions d'objets 3D en utilisant des images monoculaires. Cette méthode projette des boîtes 3D dans l'image afin de récupérer la boîte 2D associée. En se basant sur une carte de segmentation sémantique (obtenue grâce à l'algorithme de [Zheng 2015]), une carte de contour, et des informations de contexte, un score de confiance est attribué à chaque boîte 3D. Dans ces deux approches, les propositions 3D sont générées sur le plan du sol puis sont projetées en 2D et envoyées à un détecteur d'objets de type Fast-RCNN [Girshick 2015] présenté plus haut. La figure 3.4 illustre le processus de génération de propositions d'objets 3D de Mono3D [Chen 2016].

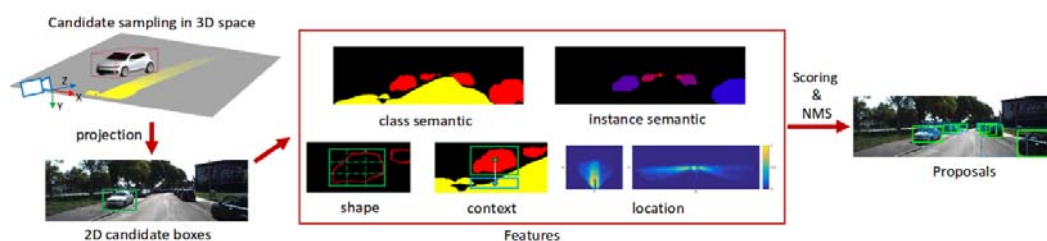


FIGURE 3.4 – Processus de génération de propositions d'objets 3D de [Chen 2016]. Des boîtes 3D sont générées sur le plan du sol. Ces boîtes 3D sont ensuite projetées dans l'image fournissant ainsi des boîtes englobantes 2D. Un score de confiance est associé à chaque boîte en se basant sur la forme de l'objet qu'elle contient, une carte de segmentation et des informations de contexte. Source [Chen 2016].

3.3 Modèles de représentation d'objets

Dans cette section, nous détaillons les diverses manières de représenter les objets c'est-à-dire les différentes façons de les appréhender. Nous appuyons sur le fait que ces modèles de représentation sont décorrélés de l'extraction de caractéristiques visuelles. En d'autres termes, un modèle de représentation peut être utilisé avec différentes caractéristiques visuelles. La communauté a proposé énormément de modèles de représentation pour résoudre de nombreuses tâches allant de la clas-

sification d'objets à la détection en passant par la reconnaissance fine. Ces modèles sont conçus pour répondre aux difficultés propres à chacune de ces tâches. Nous les avons catégorisés de la manière suivante : la représentation globale, la représentation par parties, la représentation multi-vues et la représentation par la visibilité.

3.3.1 Représentation globale

Un premier ensemble de représentations des objets est la représentation globale. Ce type de modèle prend en compte l'apparence générale des objets. Autrement dit, un objet est représenté par un vecteur de caractéristiques calculé sur la totalité de l'objet. Cette représentation a notamment été utilisée dans les travaux de [Dalal 2005] pour la détection de piétons par fenêtre glissante. Les architectures neuronales classiques s'apparentent également à ce type de représentation. Par exemple, pour la classification d'objets, les CNN de référence [Krizhevsky 2012, Szegedy 2015, Simonyan 2014] permettent d'extraire des caractéristiques sur l'ensemble de l'image contenant l'objet. Ces caractéristiques encodent l'objet dans sa globalité et permettent de séparer de manière efficace les différentes classes.

3.3.2 Représentation par parties

Une autre famille de représentation est la représentation par parties. Elle se base sur une idée plutôt intuitive : un objet est constitué d'un ensemble de parties et il existe des relations entre celles-ci. On peut effectivement penser qu'une représentation des objets par parties permet d'apporter des informations supplémentaires et notamment de la cohérence spatiale entre les différentes zones de l'objet. De plus, cela peut permettre la gestion des objets trop occultés du fait de leurs représentations locales. Ces approches peuvent être divisées en trois catégories : le modèle de forme implicite (*Implicit Shape Model* ISM), le modèle de constellation et le modèle pictural. Ces différentes structures de parties sont illustrées dans la figure 3.5. Dans ces trois types d'approches, un objet est représenté par un graphe (ou structure) de parties. Ce qui différencie ces trois modèles ce sont les connexions qui relient les parties entre elles.

Dans le modèle de forme implicite (*Implicit Shape Model* ISM), initialement introduit par [Leibe 2004], les parties de l'objet ne sont pas connectées entre

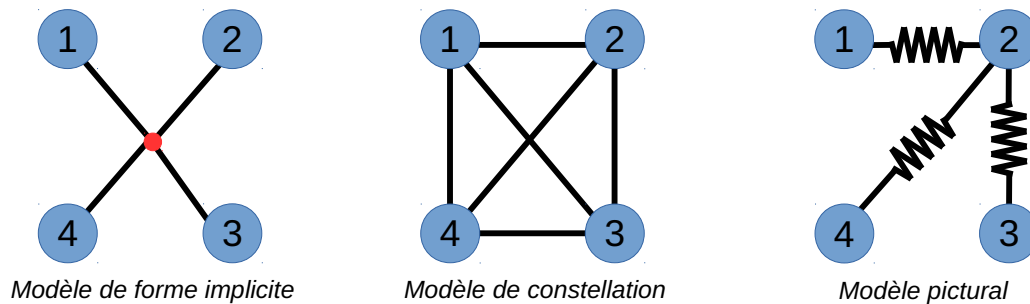


FIGURE 3.5 – Les structures d’objets par parties. Chaque noeud (bleu) représente une partie de l’objet. Le modèle de forme implicite (ou modèle en étoile) introduit des relations entre les parties et le centre de l’objet. Le modèle de constellation encode toutes les relations inter-parties. Dans le modèle pictural, les ressorts représentent une déformation possible entre parties.

elles mais sont liées au centre de l’objet qu’elles représentent. Ce modèle permet d’introduire une relation spatiale entre les parties d’un objet et le centre de celui-ci. En d’autres termes, pendant l’apprentissage d’un tel modèle, la position relative du centre de l’objet par rapport à chaque partie est encodée. Dans [Leibe 2004], des parties de l’objet sont extraites de manière non-supervisée en utilisant des détecteurs de points d’intérêt de Harris [Harris 1988]. Ces parties sont ensuite mises en correspondance avec des représentants de parties contenus dans un dictionnaire en se basant sur l’apparence. Enfin, chaque partie vote pour le centre de l’objet : une concentration de vote sur une position de l’image indique la présence d’un objet. Ce type d’approche a, par la suite, été étendu notamment pour la détection de piétons [Seemann 2007, Wohlhart 2012].

Le modèle de constellation se caractérise par un graphe de parties complètement connectées. Contrairement au modèle ISM, toutes les parties de l’objet sont connectées entre elles. Cela induit de ne considérer que peu de parties pour garantir un petit nombre de connexions inter-parties et ainsi garder un temps d’exécution raisonnable. En revanche, les modèles de constellation permettent d’introduire de fortes contraintes spatiales entre les différentes parties des objets. Dans [Burl 1996], les modèles de constellations sont utilisés pour la détection de visages. Dans cette méthode, l’arrangement spatial des parties est modélisé par un modèle probabiliste de formes. Toujours sur la même idée, les auteurs de [Weber 2000] proposent l’apprentissage d’un modèle de constellation de manière

non-supervisée et ceux de [Fergus 2003] s'intéressent à un modèle de constellation utilisant des caractéristiques visuelles invariantes à l'échelle. Dans la littérature plus récente, les auteurs de [Chen 2014] proposent une constellation de parties robustes aux non-détections pour la détection d'animaux.

Le modèle pictural introduit par [Fischler 1973], permet de pénaliser la déviation de localisation des parties par rapport à l'objet complet. Les relations de connexions spatiales entre les parties de l'objet et le centre de celui-ci peuvent être vues comme des ressorts. Un ressort associé à une partie est considéré au repos quand la partie est localisée sur sa position moyenne. La force appliquée sur le ressort est d'autant plus grande que la partie est loin de sa position moyenne. Ce type de modélisation a été abondamment utilisé [Felzenszwalb 2005, Andriluka 2009, Yang 2013, Pishchulin 2013, Sun 2012, Kiefel 2014] notamment pour la détection de personnes car elle permet d'appréhender de manière plus souple les objets déformables. Le modèle pictural le plus connu est le modèle de parties déformables (*Deformable parts model* DPM) introduit par [Felzenszwalb 2010]. Un DPM utilise le principe de la fenêtre glissante pour la tâche de détection. Cette approche utilise trois types de modèles : un modèle global de l'objet appelé *root*, un modèle de parties et un modèle de déformations de parties. Le *root* capture l'apparence globale de l'objet (comme dans un détecteur classique par fenêtre glissante de type [Dalal 2005]). Le modèle de parties capture l'apparence des parties de l'objet. Les caractéristiques visuelles utilisées sont les HOG [Dalal 2005] pour ces deux modèles d'apparence. Enfin, le modèle de déformations de parties a pour objectif d'encoder la localisation des parties par rapport au centre de l'objet. Le modèle de parties et de déformations sont appris de manière "latente", c'est-à-dire sans utiliser d'annotations sur les parties des objets. Cette méthode a été considérée, pendant quelques années, comme l'approche de référence pour la tâche de détection de personnes. Cependant, sa principale limitation concerne l'extension à la détection multi-classes. Pour détecter plusieurs classes d'objets, il faut apprendre autant de modèles DPM que de classes, ce qui induit un temps de calcul important lors du test.

3.3.3 Représentation multi-vues

Pour représenter plus finement les objets et prendre en compte les variations d'apparence liées au point de vue, des travaux se sont intéressés à la représentation

multi-vues. Celle-ci permet de caractériser les objets en fonction du point de vue (ou orientation) sous lequel ils sont observés. La description des objets est alors plus riche et permet d'apporter de l'information sur le point de vue de ceux-ci. L'étude de cette représentation a été poussée en avant par la création de *Benchmark* se concentrant sur la problématique de l'estimation de point de vue : *3D Object Dataset* [Savarese 2007], Pascal3D+ [Xiang 2014] et EPFL [Ozuysal 2009]. Certains travaux ont modélisé la représentation multi-vues comme un problème multi-classes. Plus précisément, la sphère de point de vue est discrétisée et chaque intervalle d'angle correspond à une classe de point de vue. Pour donner un exemple concret, la base de données *3D Object Dataset* [Savarese 2007] fournit pour chaque objet une classe de point de vue sur huit points de vue possibles. Dans la base Pascal3D+ [Xiang 2014], la métrique d'évaluation du point de vue permet d'évaluer les performances pour plusieurs ensembles de discrétisation (4, 8, 16 ou 24 classes de points de vue). L'idée sous-jacente de ce type d'approches est de considérer des détecteurs d'objets indépendants pour chaque point de vue possible [Su 2009, Stark 2010] car l'apparence d'un objet varie suivant l'angle sous lequel il est observé. L'un des premiers travaux sur ce sujet est celui de [Thomas 2006] combinant le modèle de forme implicite [Leibe 2004] et le système de détection multi-vues de [Ferrari 2004]. Toujours avec la même idée, des méthodes basées sur le DPM [Felzenszwalb 2010] ont été proposées. Celles-ci permettent d'apprendre des détecteurs d'objets composés de plusieurs composantes DPM de point de vue pour chaque objet [Gu 2010, Lopez-Sastre 2011, Xiang 2014, Geiger 2011, Pepik 2012]. La Figure 3.6 illustre le système de [Lopez-Sastre 2011], basé sur ce concept. Les auteurs de [Payet 2011] utilisent une collection de modèles de formes de véhicules. Chacun de ces modèles est spécifique à un point de vue. La mise en correspondance de ces modèles avec les contours de l'image permet l'estimation de l'orientation des objets. D'autres méthodes utilisent des représentations parcimonieuses pour classifier le point de vue [Zhang 2013b, Zhang 2013a].

Avec les progrès récents dans le domaine du *Deep learning*, des travaux ont introduit des détecteurs multi-vues basés sur les CNN [Ghodrati 2014, Tulsiani 2015, Su 2015]. Les auteurs de [Ghodrati 2014] comparent différentes caractéristiques visuelles pour l'estimation du point de vue, et prouvent la pertinence des caractéristiques profondes extraites de couches convolutives apprises sur ImageNet. Dans [Tulsiani 2015], les auteurs introduisent un CNN multi-classes pour la classifica-

tion jointe d'objets et de point de vue discret. Dans [Su 2015], des rendus photo-réalistes de modèles 3D sont utilisés pour apprendre un CNN multi-classes pour l'estimation du point de vue discret.

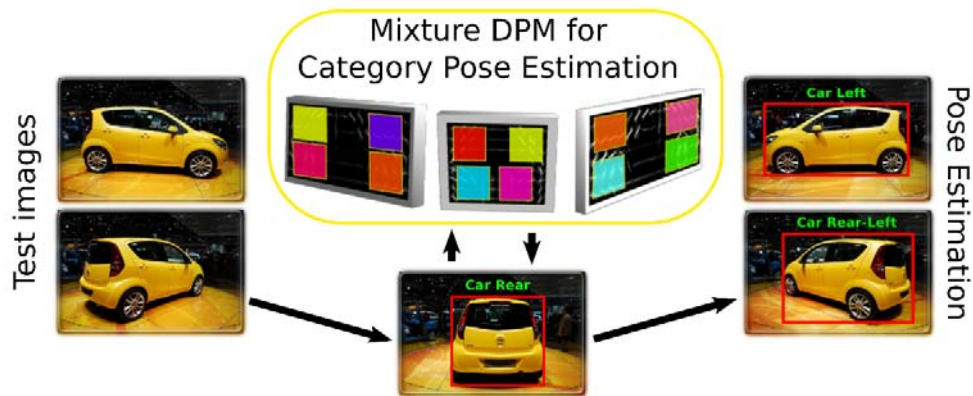


FIGURE 3.6 – Détecteur multi-vues basé sur un ensemble de composantes DPM. Source [Lopez-Sastre 2011].

3.3.4 Représentation de la visibilité

Une autre manière de représenter les objets consiste à caractériser la visibilité de ceux-ci. On peut noter deux applications majeures liées à cette caractérisation. La première concerne l'amélioration des performances de la détection ou de la classification d'objets : réussir à modéliser les occultations dans de tels systèmes est un atout pour reconnaître des objets vus partiellement. La seconde, propre à l'analyse fine, est de pouvoir prédire exactement quelle partie de l'objet est occultée, remplaçant ainsi l'objet dans son environnement.

Pour la détection de personnes, [Wang 2009] propose la combinaison d'un modèle global et local permettant de découvrir des régions occultées. Cette approche basée sur des descripteurs HOG-LBP [Ahonen 2006] permet ainsi d'améliorer les performances de détection en se basant sur les parties visibles des personnes. Avec la même idée, [Wu 2005] introduit un détecteur modélisant les occultations de personnes avec des caractéristiques basées sur la silhouette. [Zia 2013b] propose un classifieur de point de vue précis d'objet prenant en compte les occultations par utilisation de particules de formes. Les auteurs de [Wojek 2011] proposent un modèle permettant la détection de demi-personnes. [Gao 2011] introduit un modèle

de détection basé sur la segmentation en modélisant les parties d'un objet par des variables binaires permettant de savoir si une partie est visible ou non. Avec la même idée, les auteurs de [Ranjan 2016] utilisent une variable binaire pour classifier la visibilité de points de contrôle sur les visages. Dans [Xiang 2015], les auteurs se basent sur plusieurs sous-modèles caractérisant chacun un type d'occlusion. D'autres approches introduisent des sous-modèles de l'objet basés sur les occultations [Pepik 2013, Bourdev 2009, Girshick 2011].

3.4 Analyse 3D des objets

Les sections précédentes ont permis de présenter les méthodes pour la détection d'objets et les différentes manières de les caractériser (caractérisation plus ou moins fine). Dans cette section, nous abordons les différents travaux en lien avec l'analyse 3D. Celle-ci est devenue un sujet important notamment grâce à de nouvelles bases de données comme KITTI [Geiger 2012], Pascal3D+ [Xiang 2014] et ObjectNet3D [Xiang 2016]. Nous entendons par le terme analyse 3D, la caractérisation fine de l'objet dans la scène : le point de vue précis (non discret) et la localisation 3D par rapport à la caméra. Dans un souci de cohérence par rapport au sujet de cette thèse, la majeure partie des travaux présentés ci-dessous sont des approches utilisant simplement une image RGB en entrée de leur méthode.

Un bon nombre de méthodes utilise des modèles 3D d'objets d'intérêts pour l'analyse 3D dans une image. Ces modèles 3D peuvent être utilisés pour apprendre des caractéristiques d'apparence et/ou des caractéristiques géométriques. Des travaux utilisent notamment des bases d'images de rendus 3D non photo-réalistes [Stark 2010, Zia 2011, Pepik 2012] ou photoréalistes [Peng 2015, Su 2015] pour apprendre des détecteurs ou des classifieurs. L'utilisation de bases synthétiques permet la génération automatique de données sans effort important d'annotation. Leur utilisation est particulièrement pertinente pour la construction de modèles géométriques précis [Liebelt 2010, Zia 2011, Pepik 2012].

Les auteurs de [Lim 2013] utilisent des modèles 3D de mobilier afin de les aligner sur les objets présents dans les images. Dans ce travail, l'objet à analyser est connu. L'alignement est effectué en extrayant des caractéristiques HOG sur l'image et sur des points d'intérêt du modèle 3D. Un score d'alignement basé

sur des correspondances locales, géométriques et globales doit être minimisé. Le travail de [Liebelt 2010] propose d'apprendre des modèles d'apparence sur des images réelles et des contraintes géométriques sur des modèles 3D CAD. Les auteurs de [Liebelt 2008] utilisent des modèles 3D pour générer facilement une base d'apprentissage par rendu leur permettant d'apprendre un grand nombre de points de vue discrets. Les auteurs de [Yoruk 2013] utilisent des primitives de contours 3D caractérisées par leurs positions et leurs orientations 3D. Ces primitives sont vues par les auteurs comme la généralisation 3D des HOG. Ces primitives sont ensuite mises en correspondance avec des HOG extraits dans l'image.

Les auteurs de [Zia 2011, Zia 2013b, Zia 2013a, Lin 2014b, Zhu 2015] s'intéressent à l'analyse 3D d'objets en se basant sur des modèles 3D déformables. Pour cela, une base de données de modèles 3D d'un objet d'intérêt est utilisée. Sur chacun d'entre eux, des sommets 3D sont annotés définissant des formes 3D. En utilisant une analyse en composantes principales (ACP) sur ces formes 3D, un modèle 3D déformable est construit. Dans [Zia 2011, Zia 2013b, Zia 2013a], un détecteur de parties (appris sur des *patches* synthétiques et basé sur les forêts aléatoires (*Random forest*) et les caractéristiques visuelles *ShapeContext* [Belongie 2000]) est utilisé pour détecter les parties visibles d'un véhicule dans une image. Cela fournit une carte de scores pour chacune d'elles. Un ensemble de particules du modèle 3D déformable est alors généré en faisant varier l'azimut, l'élévation et les paramètres de formes du modèle. Ces particules sont projetées dans l'image et la meilleure particule est celle qui maximise le score sur les parties. La principale limitation de ces méthodes concerne la génération des particules. Celle-ci est effectuée par "force brute" induisant un temps de calcul important. Dans [Lin 2014b], les auteurs proposent d'utiliser le DPM [Felzenszwalb 2010] pour détecter les véhicules et des parties grossières de ceux-ci. Un modèle de régression est ensuite utilisé sur chaque partie grossière pour localiser des points de contrôle plus précis. L'alignement d'un modèle déformable est ensuite effectué en utilisant ces points 2D et ceux du modèle déformable 3D. La figure 3.7 illustre le déroulement de cette méthode.

Les auteurs de [Fidler 2012] proposent la construction d'un modèle de cube 3D pour caractériser un objet. Cette approche est une extension du DPM [Felzenszwalb 2010] en 3D : chaque face du modèle de cube 3D est un modèle de type DPM. La détection de ces faces permet la prédiction des boîtes englobantes 3D des objets. Dans [Xiang 2015], les auteurs introduisent des sous-catégories d'objets

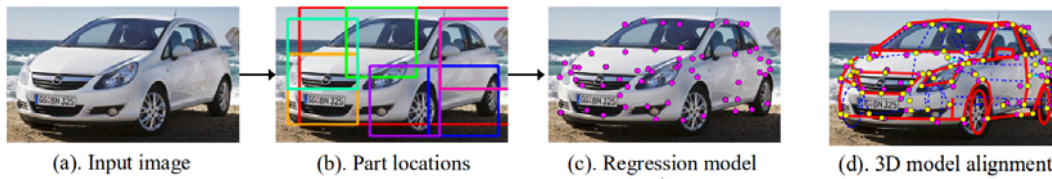


FIGURE 3.7 – Processus d'estimation de la pose fine d'un véhicule avec un modèle déformable 3D [Lin 2014b]. Un détecteur de parties grossières est appliqué sur l'image d'entrée. Des points d'intérêt sont ensuite localisés par régression puis un modèle 3D déformable est mis en correspondance avec ces points.

appelées *3D voxel pattern*. Ces sous-catégories correspondent aux types de véhicules, leurs points de vue et différents niveaux d'occultation. Le processus de détection de cette approche permet d'associer à chaque boîte englobante 2D le *3D voxel pattern* lui correspondant. La localisation 3D des objets est ensuite effectuée en projetant le *3D voxel pattern* dans l'image. Dans [Xiang 2012] est introduit le modèle d'agencement par aspect (*Aspect Layout Model ALM*) appris grâce à une base de données de modèles 3D. Les auteurs de [Glasner 2011] utilisent un modèle de forme implicite (ISM) [Leibe 2004] appris sur des modèles 3D reconstruits grâce à un algorithme de *Structure from Motion* (SFM) [Snavely 2006, Furukawa 2010] (permettant la reconstruction 3D d'un objet grâce à plusieurs images de celui-ci). D'autres approches utilisent le SFM afin de reconstruire des modèles 3D pour l'analyse 3D d'objets comme par exemple [Hejrati 2012, Yoruk 2013]. Dans [Zhu 2014], les contours de l'objet sont utilisés pour aligner un modèle 3D de l'objet en se basant sur la silhouette.

3.5 Catégorisation et Classification fine

La dernière problématique que nous proposons d'étudier dans cette thèse est la reconnaissance fine d'objets. Beaucoup de systèmes de reconnaissance fine ont été proposés pour divers types d'objets : les oiseaux [Zhang 2014, Krause 2015], les chiens [Liu 2012], les fleurs [Nilsback 2008], les visages [Taigman 2014]... Par exemple, dans le cadre de la reconnaissance fine de chiens, l'objectif est de reconnaître la race de l'animal. Pour la reconnaissance fine de visages, le système doit être capable de prédire les identités des personnes. Dans notre cas, nous nous intéressons à l'objet rigide "véhicule". Pour cet objet, la reconnaissance fine

consiste à retrouver la marque et le modèle d'un véhicule contenu dans une image.

Intuitivement, les deux difficultés principales de la reconnaissance fine de véhicules sont :

- La similarité inter-classe : deux véhicules de marques et modèles différents peuvent être très semblables en terme d'apparence.
- La variance intra-classe : un même véhicule n'a pas le même aspect suivant son point de vue, sa couleur...

En partant de ces observations, la majorité des travaux sur la reconnaissance de marques et modèles de véhicules prennent en compte les variations d'apparence liées au point de vue.

Les auteurs de [Ramnah 2014] utilisent des modèles de courbes 3D pour reconnaître la marque et le modèle des véhicules. Durant une phase préliminaire, ils reconstruisent la silhouette 3D de plusieurs véhicules en se basant sur des images de ceux-ci sous plusieurs points de vue et en utilisant l'approche proposée par [Laurentini 1994]. Ensuite, les contours de chaque image ayant permis les reconstructions des silhouettes 3D, sont calculés. Ces contours permettent de texturer les silhouettes 3D. Une phase de filtrage est ensuite utilisée pour supprimer des contours apparentés à du bruit. Étant donné une image de véhicule en entrée, le point de vue de celui-ci est calculé, en utilisant une approche de calcul de point de vue telle que [Xiang 2012, Ozuysal 2009]. Chaque modèle de courbes 3D est alors projeté dans l'image et un score de mise en correspondance des courbes 3D projetées avec les contours de l'image est calculé. Ce score permet de renvoyer la marque et le modèle du véhicule présent dans l'image.

Les auteurs de [Prokaj 2009] se basent sur un calcul de la pose du véhicule à partir d'une séquence d'images. Ensuite, ils proposent de mettre en correspondance des modèles 3D texturés avec le véhicule dans l'image. La mise en correspondance utilise des caractéristiques de type SIFT [Lowe 1999]. Ces caractéristiques sont extraites sur chaque modèle 3D recalé dans le point de vue du véhicule de l'image. Elles sont également extraites sur le véhicule de l'image. Un score de bon appariement est alors calculé permettant de retrouver le modèle 3D correspondant au véhicule de l'image.

Dans [Lin 2014b], les auteurs proposent de résoudre le problème de la reconnaissance de marques et modèles de véhicules en utilisant des parties caractéristiques de véhicules. Ces parties sont préalablement détectées et des caractéristiques de type HOG [Dalal 2005] sont calculées sur des *patches* centrés sur ces parties. Le vecteur caractéristique est alors une concaténation de tous ces *patches* et permet la reconnaissance fine de la marque et du modèle du véhicule dans l'image. Cette idée de reconnaissance fine par l'utilisation de parties caractéristiques est partagée par les auteurs de [Stark 2012, Krause 2013].

Les auteurs de [Mottaghi 2015] proposent une approche hiérarchique de type "grossier à fin" (*coarse-to-fine* en anglais). Elle permet de capturer hiérarchiquement des concepts de plus en plus fins (point de vue grossier puis point de vue continu puis catégorie de l'objet puis sous-catégorie fine de l'objet). Cette approche utilise à la fois des caractéristiques HOG et des caractéristiques provenant de CNN. Les HOG permettent la capture de l'apparence globale et les CNN permettent la capture de l'apparence locale.



FIGURE 3.8 – Des exemples de classes provenant de la base de données CompCar [Yang 2015] pour la reconnaissance de marques et modèles de véhicules.

Enfin, les auteurs de [Yang 2015] ont introduit une grande base de données pour la reconnaissance de marques et modèles de véhicules. Cette base est constituée de 36 451 images d'entraînement et 15 626 images de test correspondant à 431 classes de marques et modèles de véhicules. Ils fournissent également des résultats de classification sur cette base en utilisant les CNN. Ils soulèvent ainsi une propriété intéressante des réseaux de neurones convolutifs : les CNN sont capables d'apprendre

des caractéristiques discriminantes et multi-vues pour la reconnaissance fine. En d'autres termes, ils montrent que la recherche du point de vue des véhicules pour ensuite reconnaître leur marque et leur modèle n'est pas nécessaire. La figure 3.8 illustre des images provenant de la base CompCar. Il existe également des méthodes basées sur les CNN et le concept de *metric learning* [Song 2016, Sohn 2016]. Dans celles-ci, les CNN sont utilisés pour séparer les différentes classes, dans l'espace des caractéristiques, en se basant sur la notion de distance par similarité et dissimilarité inter-classes.

3.6 Métriques d'évaluation

Nous proposons ici de détailler certaines métriques standards d'évaluation utilisées dans l'état de l'art et dans nos travaux. En particulier nous expliquons ici les métriques permettant l'évaluation de la détection d'objets, de l'estimation de l'orientation, de la localisation 3D et de la classification. Dans la suite du document, d'autres métriques seront utilisées pour évaluer des tâches spécifiques mais celles-ci seront détaillées dans les chapitres concernés.

3.6.1 Évaluation de la détection

La métrique la plus souvent utilisée pour évaluer les performances d'un détecteur (2D) d'objets est la courbe rappel/précision utilisée notamment dans la compétition Pascal VOC [Everingham 2010]. Cette courbe permet le calcul d'une valeur de performance appelée précision moyenne (*Average Precision AP*) correspondant à l'aire sous la courbe.

Pour calculer cette courbe, les régions renvoyées par le détecteur sont classées suivant trois catégories :

- Les vrais positifs (VP) qui correspondent au nombre de régions détectées qui sont effectivement des objets.
- Les faux positifs (FP) (ou fausses alarmes) qui correspondent au nombre des régions qui ont été détectées mais qui ne sont pas des objets.
- Les faux négatifs (FN) (ou non-détections) qui correspondent au nombre d'objets qui n'ont pas été détectés alors qu'ils auraient dû l'être.

Le critère utilisé pour classer une détection dans une de ces trois catégories est le critère de recouvrement (*overlap*) avec les boîtes de la vérité terrain : si la détection recouvre suffisamment une boîte de la vérité terrain, elle est classée dans les VP, sinon elle est classée dans les FP. Les FN sont les boîtes de la vérité terrain qui n'ont aucune détection les recouvrant suffisamment.

Un détecteur d'objets fournit un score de confiance à chacune des régions détectées. Ainsi, pour un seuil de confiance donné, le rappel et la précision se calculent comme suit :

$$rappel = \frac{VP}{VP + FN} \quad (3.1)$$

$$precision = \frac{VP}{VP + FP} \quad (3.2)$$

La courbe de précision/rappel est alors obtenue en faisant varier le seuil de confiance pour obtenir différents couples rappel/précision. L'AP correspond à l'aire sous cette courbe. La figure 3.9 illustre l'apparence des courbes rappel/précision.

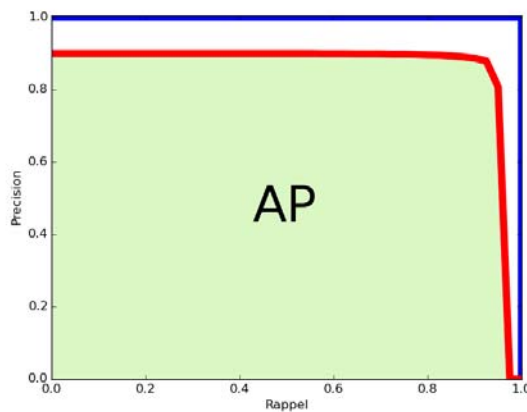


FIGURE 3.9 – Exemple de courbes rappel/précision. La courbe bleu correspond à un détecteur parfait. La courbe rouge est un exemple d'une courbe rappel/précision d'un détecteur réel. L'aire sous cette courbe est appelée AP (*Average precision*) et constitue une bonne métrique d'évaluation quantitative d'un détecteur.

3.6.2 Évaluation de l'estimation du point de vue

MPPE. La métrique MPPE (*mean precision in pose estimation*) permet d'évaluer la précision de l'estimation du point de vue d'un objet. Elle est utilisée dans le cas où les points de vue sont considérés comme des classes (points de vue discrets). Dans le cadre d'un détecteur multi-vues (c'est-à-dire qui permet de détecter les objets et d'estimer leurs points de vue), la MPPE est calculée comme suit : pour chaque objet correctement détecté (voir section précédente 3.6.1), la classe de point de vue estimée par le système est comparée à celle de la vérité terrain. Si ces deux classes sont similaires, le point de vue de cette détection est considéré correct. Dans le cas contraire, le point de vue est considéré comme incorrect. Ce processus est effectué sur l'ensemble de la base de test et sur chacune des bonnes détections permettant de remplir un tableau 2D appelé matrice de confusion. Chaque ligne et chaque colonne de cette matrice correspond à une classe de point de vue. De cette manière, la cellule i, j de la matrice correspond au nombre moyen de points de vue estimés comme la classe i pour une vérité terrain de classe j . En d'autres termes, chaque cellule de la diagonale de cette matrice correspond au taux de bonne estimation pour chaque point de vue : la MPPE est la valeur de la moyenne de cette diagonale.

CVP. La métrique CVP (*continuous viewpoint estimation*) permet l'évaluation du point de vue continu (précis). Comme dans la métrique MPPE, CVP est calculée sur les objets correctement détectés. Dans cette métrique, un calcul de différence angulaire (en valeur absolue) entre l'angle de la vérité terrain et celui renvoyé par le système est effectué. Si cette différence est inférieure à un certain seuil (typiquement 10 degrés), le point de vue précis estimé est considéré correct. CVP fournit le pourcentage de points de vue précis correctement estimés sur l'ensemble des bonnes détections de la base de test.

AOS. La métrique AOS (*average orientation similarity*) permet d'évaluer l'estimation du point de vue précis de manière conjointe avec la détection d'objets. Elle a été introduite par [Geiger 2012]. Cette métrique est calculée en utilisant une courbe rappel/similarité angulaire. Son calcul est assez similaire au calcul de l'AP pour l'évaluation de la détection. Plus précisément, étant donné un ensemble de détection pour un seuil de confiance donné, le rappel est calculé de la même manière que pour la détection (voir 3.6.1). Ensuite, pour chaque objet correctement détecté

(les Vrais Positifs), la distance cosinus s entre l'angle estimé et l'angle de la vérité terrain est calculée :

$$s = \frac{1 + \cos(\theta - \theta^*)}{2} \quad (3.3)$$

où θ est l'angle de l'objet de la vérité terrain et θ^* celui renvoyé par le système à évaluer. Dans le cas des faux positifs, cette distance angulaire vaut 0. La similarité moyenne sur l'ensemble des détections est alors calculée. La courbe de rappel/similarité angulaire est tracée en faisant varier le score de confiance. l'AOS correspond à l'aire sous cette courbe.

3.6.3 Évaluation de la localisation 3D

La métrique la plus utilisée pour évaluer la localisation 3D des objets est l'ALP (*Average Localization Precision*) proposée par [Xiang 2015]. Son processus de calcul est assez semblable à l'AOS définie dans la section précédente à ceci près que la similarité angulaire est remplacée par une mesure de précision de la localisation 3D. En d'autres termes, l'ALP se déduit d'une courbe rappel/précision 3D. Pour un score de confiance donné, le rappel est calculé de la même manière que pour la détection d'objets (recouvrement des boîtes 2D de vérité terrain avec les boîtes 2D des détections). Pour chaque Vrai Positifs, la distance entre les coordonnées 3D du centre de l'objet détecté et le centre 3D de la vérité terrain associée est calculée. Si cette distance est inférieure à un certain seuil (typiquement 1 mètre ou 2 mètres), l'objet est considéré bien localisé. La courbe rappel / précision 3D est alors calculée en faisant varier le score de confiance. De manière similaire aux métriques présentées plus haut, l'ALP correspond à l'aire sous cette courbe.

3.6.4 Évaluation de la classification fine

La métrique utilisée pour évaluer la classification (dans notre cas la classification fine) est certainement la plus simple. De manière générale, étant donné une image d'un objet, un système de classification renvoie un score de probabilité pour chaque classe possible. Les classes sont alors triées de la plus probable à la moins probable. L'image est considérée comme correctement classifiée si le score le plus fort correspond à la classe réelle de l'objet dans l'image. Ainsi, sur une base d'images de test, la métrique mesure le rapport entre le nombre d'images correctement classifiées par le système et le nombre total d'images de la base. Cette

métrique peut également être calculée en utilisant la notion de rang. Dans cette variante, une image est considérée comme correctement classifiée si la classe réelle de l'objet se trouve dans les K premiers scores. Dans ce cas, l'image est correctement classifiée pour le rang K .

3.7 Positionnement de la thèse

Nous avons présenté dans ce chapitre différentes méthodes en lien avec les problématiques de cette thèse. De manière générale, les méthodes de l'existant se focalisent sur un seul aspect de l'analyse fine d'objet. Dans nos travaux, nous proposons d'aboutir à un système complet, permettant de gérer simultanément la majorité des aspects de l'analyse fine. Pour parvenir à ce système, des algorithmes spécifiques à certaines tâches ont été développés en amont. En particulier, nous avons investigué :

- L'utilisation de modèles 3D pour les problématiques de détection de véhicules (voir 3.2), d'analyse 3D (voir 3.4) et de classification fine (voir 3.5). Ces modèles 3D sont le fil conducteur de nos travaux et sont utilisés pour fournir des caractéristiques d'apparence (dans les chapitres 4 et 5) et géométriques (dans le chapitre 5). Dans le chapitre 6, les modèles 3D sont utilisés pour annoter automatiquement des données réelles et retrouver les informations 3D des véhicules dans une image.
- L'utilisation de la géométrie des objets pour l'analyse fine, notamment via des modèles de représentation par parties. Dans le chapitre 5, l'utilisation conjointe de modèles de formes implicites et de constellations (voir 3.3.2) nous permet de construire des modèles géométriques robustes pour la détection de véhicules et l'estimation du point de vue.
- La prise en compte de la visibilité des objets (voir 3.3.4). Dans les chapitres 5 et 6, une représentation par parties des véhicules est adoptée. Dans le chapitre 5, cette représentation permet d'être robuste aux occultations. Dans le chapitre 6, la visibilité de chaque partie de véhicules est prédite permettant ainsi d'apporter une information supplémentaire caractérisant plus finement les véhicules dans la scène.

- Les approches de type proposition d'objets pour la détection de véhicules (voir 3.2.2). Ce type d'approche est utilisé dans le chapitre 6 pour apprendre un CNN multi-tâches pour l'analyse complète 2D/3D de scènes routières dans le contexte de la conduite autonome.

Classification fine 2D/3D de véhicules

Sommaire

4.1	Introduction	55
4.1.1	Solution intuitive	57
4.1.2	Solution proposée	58
4.2	Contours synthétiques et contours d'images réelles	60
4.2.1	Génération de contours synthétiques par rendu 3D	60
4.2.2	Détection de contours dans une image réelle	60
4.3	Modèle de reconnaissance de marques et modèles de véhicules	63
4.3.1	Descripteurs communs aux images couleurs et aux contours	63
4.3.2	Classifieur synthétique	65
4.4	Expérimentations	66
4.4.1	Bases de données	66
4.4.2	Résultats	67
4.4.2.1	Visualisation des caractéristiques communes	67
4.4.2.2	Évaluation sur CompCar	69
4.4.2.3	Évaluation sur 2DCar	69
4.5	Conclusion	73

4.1 Introduction

Dans ce chapitre, nous proposons de traiter l'une des problématiques de l'analyse fine de véhicules : la classification fine. Plus précisément, nous voulons construire un système capable de renvoyer la marque et le modèle d'un véhicule à partir d'une image de celui-ci. Un tel système implique de créer des classifieurs capables de décrire et de séparer des classes souvent très semblables en termes d'apparence. En effet, des véhicules de marques et de modèles différents peuvent

être très proches visuellement ce qui ajoute une difficulté pour les démarquer. En plus de cette similarité inter-classes, l'apparence des véhicules varie suivant plusieurs autres facteurs. Par exemple, le point de vue du véhicule a une incidence sur son apparence : une voiture vue de face ou de derrière ne présente pas les mêmes caractéristiques visuelles. La couleur modifie également la manière dont un véhicule est perçu mais n'est pas une caractéristique suffisante pour la classification fine : un même modèle de véhicule peut être de différentes couleurs. Enfin, l'environnement peut introduire du bruit sur les images de véhicules à cause de leurs propriétés spéculaires. L'objectif de la classification fine est de trouver des descripteurs discriminants suffisamment invariants à toutes ces perturbations.

Il existe plusieurs travaux portant sur la reconnaissance fine de marques et modèles de véhicules. Ces approches traitent les différentes difficultés identifiées ci-dessus. Par exemple, [Ramnah 2014, Prokaj 2009] utilisent en amont un détecteur de poses de véhicules pour pallier les variations d'apparence liées au point de vue. D'autres méthodes se concentrent sur des descripteurs de parties locales pour discriminer les marques et les modèles de voitures [Lin 2014b, Krause 2013]. Plus récemment, les auteurs de [Yang 2015] ont fourni une nouvelle base de données pour la reconnaissance fine de voitures. Des approches basées sur les CNN sont testées sur celle-ci afin de fournir de premiers résultats. Une conclusion intéressante de leurs travaux est que les CNN peuvent apprendre des descripteurs multi-vues pour la reconnaissance fine. En d'autres termes, les auteurs de [Yang 2015] mettent en avant la capacité des CNN à être invariants au point de vue pour la classification fine.

Cependant, apprendre des classifieurs robustes implique de disposer de suffisamment de données pour l'apprentissage, notamment si l'entraînement est réalisé avec les CNN. De plus, lorsqu'un nouveau modèle de véhicule est mis sur le marché, apprendre à le reconnaître implique de collecter beaucoup d'images de celui-ci puis de recommencer l'apprentissage. C'est pourquoi nous nous intéressons ici à l'utilisation de modèles 3D pour apprendre un classifieur permettant, par la suite, de classifier des images. L'utilisation de modèles 3D pour la classification fine de véhicules présente plusieurs avantages. Le premier est la possibilité de générer de grosses bases de données synthétiques sans effort d'annotations pour entraîner des classifieurs robustes. En effet, même s'il existe des bases de données importantes pour la reconnaissance de marques et modèles [Krause 2013, Yang 2015] il reste

difficile de collecter beaucoup d'images du même véhicule sous différentes vues sans l'aide d'un expert. Grâce aux modèles 3D, la difficulté est moindre : il suffit d'un modèle 3D pour générer un grand nombre d'images de ce véhicule. De cette manière, une marque et un modèle peuvent très facilement être rajoutés comme classe d'un classifieur. De plus, l'utilisation de modèles 3D permet de pallier l'ambiguïté sur la version et l'année du véhicule : toutes les images générées à partir du modèle 3D correspondront exactement à la marque, au modèle, à la version et à l'année du véhicule. Cela n'est pas le cas dans des bases de données réelles. En effet, les versions et les années des différentes marques et modèles sont en général concaténées pour former une seule classe. Utiliser des modèles 3D permettrait d'aller dans un niveau de précision ultra-fin sur le type de véhicules tout en garantissant un grand nombre de données.

4.1.1 Solution intuitive

Comme dit précédemment, nous nous intéressons à l'utilisation de modèles 3D pour entraîner un classifieur fin qui pourra être appliqué à des images réelles. Pour apprendre une classe de véhicules, il suffirait alors d'utiliser le modèle 3D lui correspondant. Autrement dit, étant donné une image réelle en entrée, ce classifieur permettrait de trouver la marque et le modèle de la voiture et ainsi le modèle 3D correspondant. Nous disposons d'une base de données de modèles 3D non-texturés qui permet de générer des rendus contours non-photoréalistes. C'est un processus de rendu très simple qui permet de se concentrer sur la forme du véhicule (contours) sans avoir besoin de générer des conditions d'éclairage complexes ou des rendus couleurs. L'intuition fondamentale de ce travail est qu'une partie significative des contours 3D peut être trouvée dans des images de contours calculés sur des images réelles avec des détecteurs de contours basés sur les gradients.

Une approche intuitive (illustrée dans la figure 4.1) consiste à générer des exemples d'apprentissage par rendus contours 3D d'une base de modèles 3D. Un classifieur de marques et modèles de véhicules basé CNN est alors appris en utilisant ces exemples synthétiques. Pour classifier un véhicule réel lors du test, la carte de contours de l'image est calculée. Cette carte est ensuite envoyée au classifieur appris sur les données synthétiques. Malheureusement, ce type d'approche ne fonctionne pas (démontré dans les expérimentations). La principale raison de cet échec

réside dans la notion de sur-apprentissage (*overfitting*) et d'adaptation de domaine. En effet, le CNN appris sur des données synthétiques de contours sur-apprend des structures qui ne sont pas présentes dans des images de contours réelles. En cela, le CNN appris n'est pas généralisable aux données réelles.

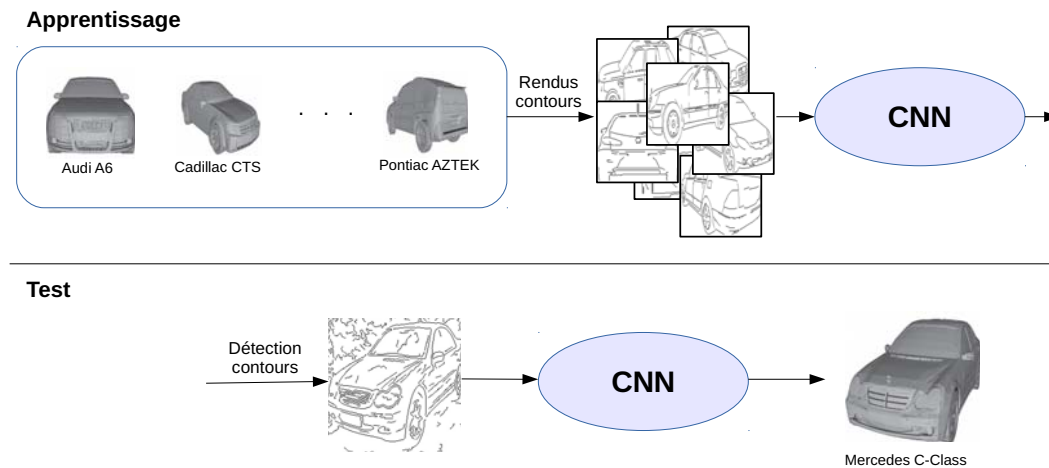


FIGURE 4.1 – Une solution intuitive pour la reconnaissance de marques et modèles par utilisation de modèles 3D. L'apprentissage d'un CNN sur des données synthétiques générées par rendus contours sur des modèles 3D. Une carte de contours calculée sur une image réelle de test est ensuite envoyée au CNN pour prédire sa marque et son modèle. Une telle approche donne de mauvaises performances.

4.1.2 Solution proposée

Pour pallier ce problème, nous avons investigué une solution hybride. Plus précisément, nous proposons dans un premier temps d'apprendre des caractéristiques visuelles discriminantes pour la reconnaissance de marques et modèles de véhicules en utilisant des données réelles et un CNN. Dans un second temps, ces caractéristiques sont utilisées pour apprendre un classifieur de marques et modèles de véhicules sur des images synthétiques de contours. Ce classifieur peut ensuite être utilisé pour reconnaître la marque et le modèle d'un véhicule dans une image couleur réelle et pour lui associer le modèle 3D correspondant.

L'originalité de l'approche réside dans les propriétés des caractéristiques visuelles apprises sur les images réelles. Ces caractéristiques sont invariantes à la couleur et aux contours. Autrement dit, les caractéristiques profondes extraites

d'une image couleur et de sa carte de contours sont très semblables.

Ces caractéristiques présentent plusieurs intérêts :

- Elles permettent d'extraire des structures de contours présentes dans une image réelle et discriminantes pour la reconnaissance fine. De cette manière, utiliser ces caractéristiques pour apprendre un classifieur sur des images de contours synthétiques contourne le problème du sur-apprentissage de contours 3D.
- Elles produisent une description très riche des véhicules car elles ont été apprises sur deux domaines très différents (couleurs et contours). En d'autres termes, l'utilisation jointe d'images couleurs et de contours durant l'apprentissage permet d'améliorer les performances par rapport à un CNN appris seulement sur des images de contours.

Notre apprentissage se divise en deux étapes. Dans un premier temps, nous apprenons des descripteurs invariants aux couleurs et aux contours en utilisant un CNN et la base de données CompCar [Yang 2015]. Ici nous considérons deux domaines : images couleurs et images de contours. Notre méthode permet de trouver directement une représentation commune pour les deux domaines. Cette manière de faire est différente des méthodes d'adaptation de domaine comme [Ganin 2015, Long 2015b, Tzeng 2014, Massa 2015] qui consistent à trouver la transformation entre les données d'apprentissage et les données de test en minimisant généralement la distance entre les distributions des caractéristiques des deux domaines. Avec ce travail, nous montrons la capacité des réseaux de neurones à trouver la représentation commune entre deux types de données qui sont naturellement très éloignés l'un de l'autre. Dans un second temps, ces descripteurs invariants aux couleurs et aux contours sont calculés sur des images de contours synthétiques et utilisés pour apprendre un classifieur synthétique linéaire de marques et modèles de véhicules. Nous montrons que ce classifieur donne de très bons résultats lorsqu'il est appliqué à des images couleurs réelles.

La section 4.2 rappelle le principe du rendu non photo-réaliste de type contours et des différents algorithmes de détection de contours dans une image. La section 4.3 explique comment les descripteurs invariants aux couleurs et aux contours sont définis et appris. L'utilisation de ces descripteurs pour apprendre un classifieur fin sur des modèles 3D est ensuite détaillée. La section 4.4 donne les résultats montrant

l'efficacité de nos modèles de classification.

4.2 Contours synthétiques et contours d'images réelles

Le travail présenté dans ce chapitre se base sur (1) des données synthétiques obtenues par rendus non-photoréalistes de type contours et (2) des cartes de contours obtenues à partir d'images réelles.

4.2.1 Génération de contours synthétiques par rendu 3D

L'objectif du rendu 3D de type contours est de dessiner dans une fenêtre de rendu certaines arêtes du modèle 3D. Les arêtes à dessiner sont les arêtes connectant deux faces dont les normales forment un angle supérieur à un certain seuil (par exemple 30 degrés). Ces arêtes sont appelées arêtes saillantes (ou arêtes franches). Ce type de rendu est peu dépendant de la qualité du modèle 3D car les contours saillants sont relativement invariants à la résolution du maillage. La figure 4.2 illustre le principe d'arêtes saillantes et la figure 4.3 montre un exemple de rendu non-photoréaliste de type contours.

4.2.2 Détection de contours dans une image réelle

Nombreux sont les algorithmes de détection de contours dans une image et il existe différentes manières d'aborder cette problématique. Un premier ensemble de méthodes propose une vision orientée traitement d'images [Canny 1986, Wang 2008, Arbelaez 2011] et un second ensemble de méthodes propose des approches basées apprentissage automatique [Xie 2015, P. Dollar 2013]. Ces dernières requièrent une base d'apprentissage annotée afin d'apprendre à classifier quels pixels de l'image correspondent à des contours (à l'aide de Forêts aléatoires [P. Dollar 2013] ou des CNN [Xie 2015]). Dans cette thèse nous nous sommes particulièrement intéressés aux algorithmes de détection de contours par traitement d'image. Le plus connu d'entre eux est l'algorithme de Canny [Canny 1986]. Il se base sur l'orientation et la magnitude du gradient dans une image. Dans cet algorithme, l'image est tout d'abord lissée grâce à un noyau gaussien (afin d'éliminer le bruit). Ensuite, le gradient est calculé en utilisant les filtres de convolution de Sobel

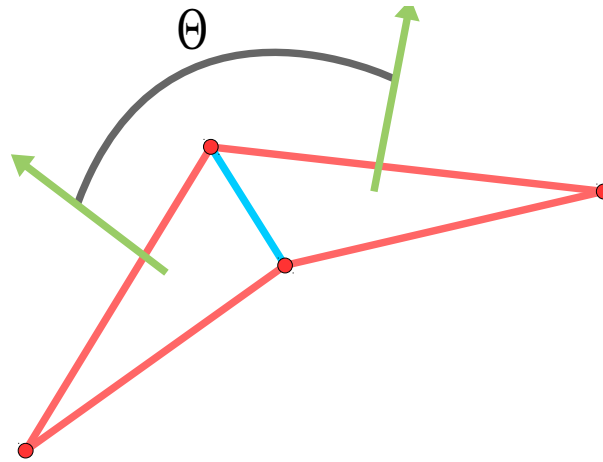


FIGURE 4.2 – Principe du calcul des contours saillants. Nous avons représenté deux faces d'un modèle 3D. Les normales à ces faces sont représentées en vert. L'angle θ correspond à l'angle entre ces deux normales. Si cet angle est supérieur à un certain seuil, l'arête entre les deux faces (en bleu) est considérée comme saillante. Dans ce cas elle est dessinée dans la fenêtre de rendu 3D.



FIGURE 4.3 – Exemple d'un rendu contours d'un modèle 3D de véhicule (dans cet exemple, il s'agit d'une Audi A6).

[Sobel 1968]. Une étape de suppression des non-maxima est alors effectuée dans la direction du gradient. Enfin, un seuillage sur la valeur absolue de la magnitude du gradient est appliqué. La carte de contours renvoyée par l’algorithme du Canny est une carte binaire. Cet algorithme est très sensible au bruit car la valeur absolue de la magnitude du gradient n’est pas suffisante pour caractériser un contour. En partant de cette observation, les auteurs de [Wang 2008] proposent un détecteur de contours basé sur une carte de saillance. Celle-ci est calculée grâce à la carte du gradient en utilisant sa distribution spatiale. Cela permet de mettre en avant les pixels qui n’ont pas forcément un gradient très élevé mais qui appartiennent tout de même à des contours. De plus, contrairement à l’algorithme du Canny qui propose un seuillage du gradient directement sur les pixels de la carte du gradient, les auteurs de [Wang 2008] proposent un seuillage par segment. Autrement dit, des segments sont créés à partir de la carte de saillance et se voient attribuer un score de saillance. Le seuillage s’effectue alors sur ces segments. La figure 4.4 illustre qualitativement les résultats de l’approche proposée par [Wang 2008]. Dans nos travaux, nous avons utilisé ce détecteur de contours pour sa simplicité et sa résistance au bruit. Néanmoins il pourrait être intéressant d’investiguer d’autres algorithmes.

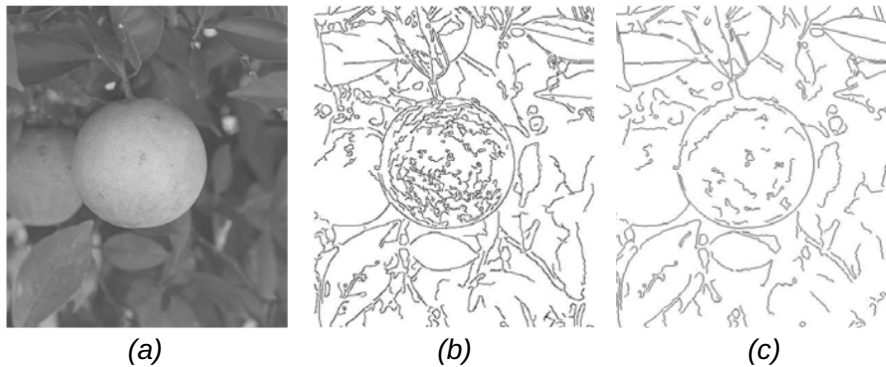


FIGURE 4.4 – Comparaison quantitative de l’algorithme de Canny [Canny 1986] et de l’algorithme de [Wang 2008]. (a) l’image initiale en niveau de gris, (b) la carte de contours calculée par Canny, (c) la carte de contours calculée par [Wang 2008]. Nous constatons une plus grande robustesse par rapport au bruit dans cet algorithme par rapport à celui de canny. Source [Wang 2008].

4.3 Modèle de reconnaissance de marques et modèles de véhicules

Dans cette section, le classifieur de marques et modèles de véhicules est présenté. L'objectif est d'apprendre un classifieur sur des images synthétiques qui pourra être testé sur des images couleurs réelles. Nous avons construit, dans un premier temps, une base de données appelée 2D3DCar composée de données d'apprentissage et de test. La base d'apprentissage (appelée 3DCar) se compose de N modèles 3D de voitures. La base de test (appelée 2DCar) se compose d'images réelles couleurs qui correspondent exactement aux modèles de 3DCar. La base 2D3DCar est illustrée dans la Figure 4.5 (c).

L'apprentissage est divisé en deux phases. Dans un premier temps, nous utilisons la base CompCar [Yang 2015] composée de M classes, afin d'apprendre les descripteurs discriminatifs pour la reconnaissance fine mais aussi communs aux deux domaines (couleurs et contours). Dans un second temps, nous utilisons ces descripteurs pour apprendre un classifieur uniquement sur des images de contours synthétiques générées avec la base de données 3DCar. La génération de ces données est effectuée en réalisant des rendus contours de chaque modèle 3D de la base 3DCar (fournie par [Zia 2013a]) suivant plusieurs points de vue et plusieurs paramètres intrinsèques de caméra. La Figure 4.5 illustre l'ensemble du système.

4.3.1 Descripteurs communs aux images couleurs et aux contours

L'objectif principal de cette étape est de trouver une représentation commune dans l'espace des descripteurs entre des cartes de contours et des images couleurs. De plus, ces descripteurs doivent être discriminants pour reconnaître la marque et le modèle des véhicules.

En utilisant CompCar [Yang 2015], nous générons une carte de contours pour chacune des images en utilisant la détection de contours proposée par [Wang 2008] décrit dans 4.2.2. Ainsi, nous obtenons de CompCar deux bases de données : la base de données couleurs $\{X_{Color}, Y_{Color}\}$ et la base de données contours $\{X_{Edge}, Y_{Edge}\}$ où X correspond aux données et $Y \in [1, \dots, M]$ aux étiquettes des classes. Les réseaux de neurones convolutifs profonds peuvent être définis par deux ensembles de paramètres : les paramètres d'extraction de caractéristiques θ_f (typi-

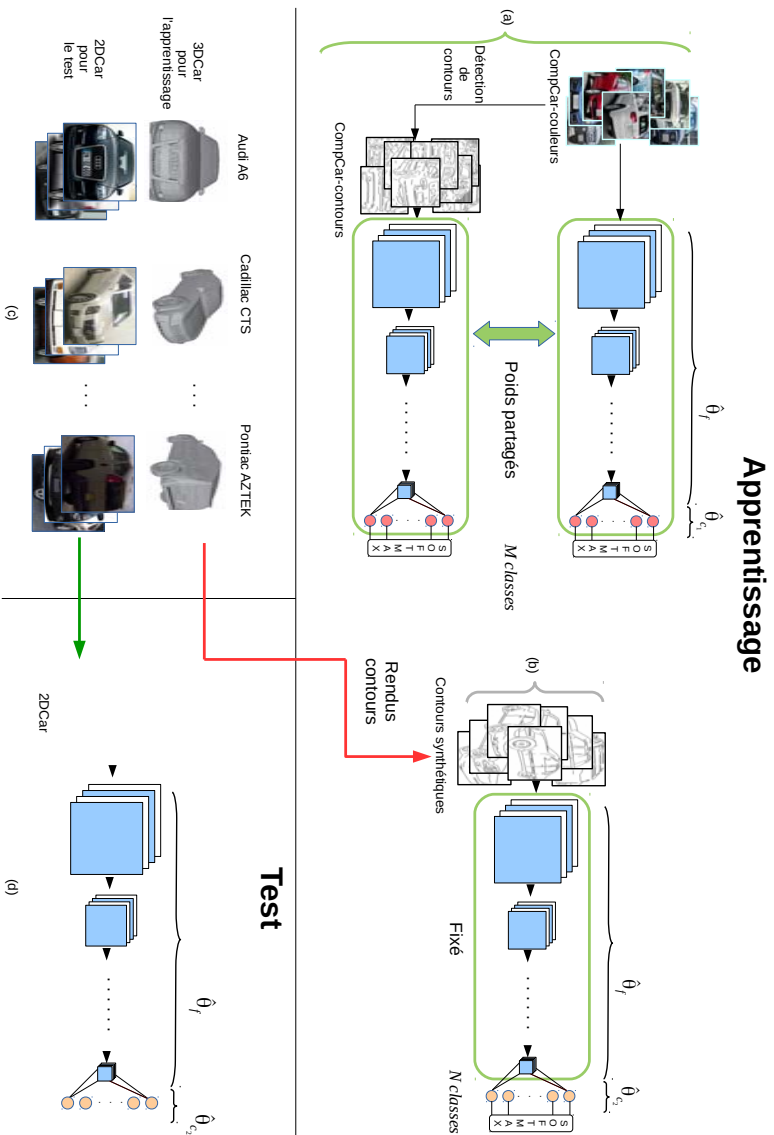


FIGURE 4.5 – Ensemble du système de reconnaissance de marques et modèles de véhicules 2D/3D. (a) Apprentissage des descripteurs communs aux couleurs et aux contours sur la base CompCar : il en résulte les paramètres d'extraction de caractéristiques communes $\hat{\theta}_f$ et les paramètres de classification $\hat{\theta}_{ct}$. (b) Apprentissage du classifieur fin synthétique sur des images de contours synthétiques générées avec la base 3D Car. Dans cette étape, les paramètres de classification $\hat{\theta}_{cs}$ sont appris en fixant $\hat{\theta}_f$. (c) La base de données 3D2D Car. (d) Test du réseau sur les images de la base 2D Car.

quement les paramètres des couches convolutives) et les paramètres de classification θ_{c_1} (typiquement les couches complètement connectées d'un perceptron multi-couches prenant en entrée les caractéristiques extraites par les couches convolutives).

Nous définissons un réseau de neurones complètement partagé entre les deux types de données (les paramètres de ce réseau sont les mêmes pour les deux domaines). La Figure 4.5 (a) illustre cette phase d'apprentissage. L'optimisation des paramètres du réseau (θ_f, θ_{c_1}) est effectuée en minimisant la fonction de coût suivante :

$$\begin{aligned} (\hat{\theta}_f, \hat{\theta}_{c_1}) = \underset{(\theta_f, \theta_{c_1})}{\operatorname{argmin}} & \mathcal{L}(\theta_f, \theta_{c_1}, X_{Color}, Y_{Color}) \\ & + \mathcal{L}(\theta_f, \theta_{c_1}, X_{Edge}, Y_{Edge}) \end{aligned} \quad (4.1)$$

où \mathcal{L} correspond à la fonction de coût classique Softmax abondamment utilisée pour optimiser des problématiques de classification. Aucun coefficient de pondération n'est utilisé pour ne pas favoriser un domaine par rapport à un autre. Cette manière de faire peut être assimilée aux réseaux de neurones siamois [Chopra 2005] où l'objectif est de minimiser la distance (norme L2) dans l'espace des descripteurs entre deux images similaires. Néanmoins, contrairement aux réseaux siamois, nous essayons de réduire la distance entre des images représentant le même véhicule (sur les deux domaines) tout en apprenant des descripteurs discriminants pour la reconnaissance fine. Contrairement aux méthodes d'adaptation de domaine comme [Ganin 2015, Long 2015b, Tzeng 2014, Massa 2015] nous n'essayons pas de mettre en correspondance un type de données avec un autre, mais nous proposons de trouver directement la représentation commune qui peut être pertinente pour les deux types de données.

4.3.2 Classifieur synthétique

Une fois les descripteurs communs aux couleurs et aux contours appris, nous les utilisons pour entraîner un classifieur seulement sur des images de contours synthétiques $\{X_{synth}, Y_{synth}\}$ générées avec les modèles 3D de la base de données 3DCar avec $Y \in [1, \dots, N]$. Nous fixons les paramètres d'extraction de descripteurs $\hat{\theta}_f$ appris précédemment et nous entraînons de nouveaux paramètres de classification θ_{c_2} correspondant aux N labels de nos modèles 3D. La nouvelle fonction de

coût à minimiser est définie comme suit :

$$\hat{\theta}_{c_2} = \underset{\theta_{c_2}}{\operatorname{argmin}} \mathcal{L}(\hat{\theta}_f, \theta_{c_2}, X_{synth}, Y_{synth}) \quad (4.2)$$

La Figure 4.5 (b) illustre cette étape d'apprentissage. Procéder en deux étapes présente des avantages. Fixer les paramètres d'extraction de descripteurs permet d'apprendre cette deuxième étape très rapidement (très peu de paramètres à optimiser) et ajouter une classe à ce classifieur est très simple de par l'utilisation de modèles 3D.

4.4 Expérimentations

4.4.1 Bases de données

Dans ces expérimentations, nous considérons deux bases de données : CompCar et 2D3DCar. La base de données CompCar fournit 36 451 images d'entraînement et 15 626 images de test correspondant à $M = 431$ classes. La base 3DCar est composée de $N = 36$ modèles 3D et la base 2DCar est composée de 403 images (à peu près 10 images par marque et modèle). Nous avons généré 3 456 rendus contours pour chaque modèle 3D. Le nombre total d'exemples dans la base de données synthétique est donc de 124 416 exemples. Pour obtenir ces exemples, nous avons fait varier les paramètres suivants :

- l'azimut des véhicules entre 0 et 360 degrés avec un pas de 1 degré.
- l'élévation des véhicules entre 0 et 10 degrés avec un pas de 1 degré.
- la position des véhicules par rapport à la caméra entre 4 à 10 mètres en z (avec un pas de 2 mètres), de 1 à 1.50 mètres en y (avec un pas de 0.25 mètres) et de -0.5 à 0.5 mètres en x avec un pas de 0.5 mètres).

Comme les images résultantes sont rognées avec leurs boîtes englobantes 2D, nous n'avons pas fait varier les paramètres de focale de la caméra. Les effets de perspective liés à la valeur de la focale sont simulés grâce à la variation en z .

Du bruit a été ajouté aux images de rendu en utilisant des images de fond de PASCAL VOC 2007 [Everingham 2010]. Pour cela, le détecteur de contours de [Wang 2008] est appliqué sur ces images de fond. Les cartes de contours résultantes sont composées aléatoirement avec les rendus contours des modèles 3D. Pour plus de réalisme, du bruit est ajouté de cette manière à l'extérieur et à l'intérieur des véhicules. La figure 4.6 montre certaines images de notre base synthétique obtenues avec ce procédé.

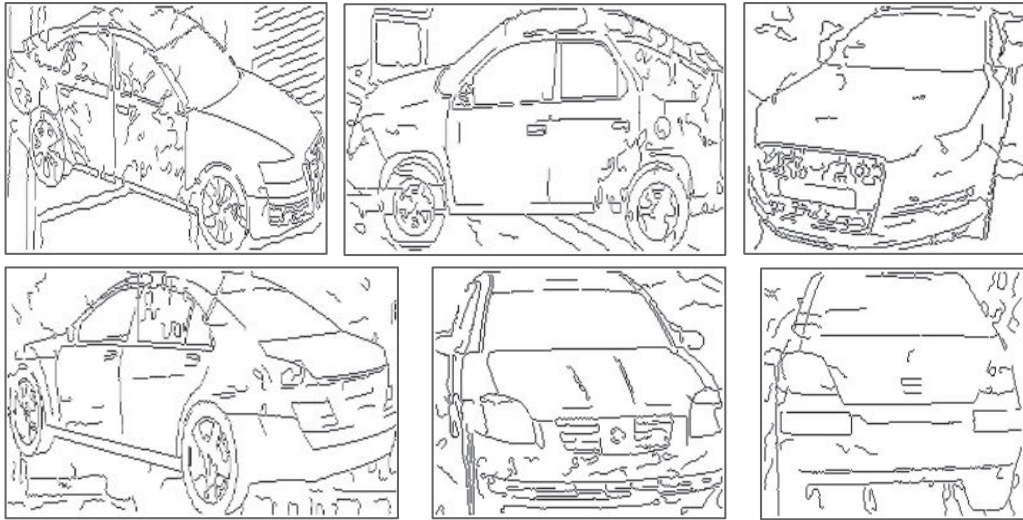


FIGURE 4.6 – Exemples de la base de données synthétique. Les rendus contours 3D sont enrichis avec des images de contours provenant de PASCAL VOC 2007.

4.4.2 Résultats

Sept réseaux profonds ont été entraînés pour évaluer, comparer et valider notre nouvelle approche. Nous définissons chaque réseau par \mathcal{N}_{dte}^{dtf} où dte et dtf correspondent aux données d'apprentissage utilisées pour le classifieur et pour le calcul des descripteurs respectivement (couleurs+contours : ce , couleurs : c , contours : e et contours synthétiques : s). Nous avons appris tous les réseaux en utilisant l'architecture GoogLeNet proposée par [Szegedy 2015]. Pour l'apprentissage des descripteurs invariants aux contours et aux couleurs nous avons utilisé une initialisation utilisant des poids pré-appris (fine-tuning) du googLeNet sur la base ImageNet. Nous détaillons le type de données d'apprentissage pour chaque réseau appris dans le Tableau 4.1 et le Tableau 4.2. La métrique d'évaluation utilisée est la métrique de performance standard pour la classification : le pourcentage d'exemples de la base de test correctement classifiés.

4.4.2.1 Visualisation des caractéristiques communes.

Couleurs / Synthétique. Nous proposons ici de visualiser les descripteurs invariants aux couleurs et aux contours. En utilisant le réseau \mathcal{N}_{ce}^{ce} défini dans la section 4.3.1, nous avons extrait les descripteurs sur les images de contours synthétiques et les images couleurs réelles (venant de 3DCar et 2DCar) et nous

avons tracé leurs distributions en utilisant la réduction de dimension t-SNE [Van der Maaten 2008] (*t-Distributed Stochastic Neighbor Embedding*) dans la Figure 4.7. Cet algorithme permet de réduire des vecteurs à haute dimension pour pouvoir les visualiser (en 2D). Il est basé sur la minimisation de la divergence de Kullback-Leibler [Kullback 1951] entre la distribution basse dimension recherchée et celle à haute dimension. Nous pouvons voir que les deux domaines (couleurs réelles et contours synthétiques) sont très proches dans l'espace des descripteurs tout en étant discriminants pour la reconnaissance de marques et modèles de véhicules. Cela montre que les descripteurs entraînés sur des contours extraits d'images réelles et sur des images couleurs se généralisent aux images de contours synthétiques. De plus, cela prouve le pouvoir de généralisation pour la reconnaissance fine : même si les descripteurs sont appris sur une base de données avec des marques et modèles spécifiques (les classes de CompCar), ils peuvent être appliqués à d'autres marques et modèles de voitures tout en restant discriminants pour ces nouvelles classes (les classes de 2D3DCar).

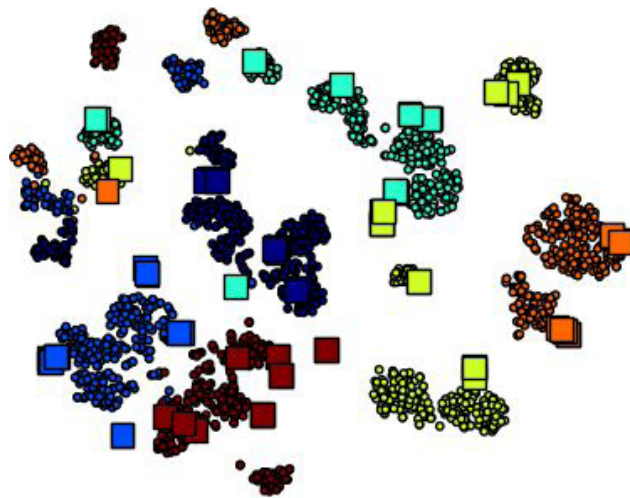


FIGURE 4.7 – Visualisation des descripteurs communs aux couleurs et aux contours appris avec le réseau \mathcal{N}_{ce}^{ce} . Les descripteurs ont été calculés pour six classes (six couleurs sur la figure) sur les images couleurs de la base 2DCar (□) et sur les images de contours synthétiques générées avec la base 3DCar (○). La projection de ces caractéristiques à haute dimension dans un espace 2D se fait via l'algorithme de t-SNE [Van der Maaten 2008].

Couleurs / Contours réels. Nous avons également analysé qualitativement les sorties des différentes couches d'extraction des descripteurs du réseau \mathcal{N}_{ce}^{ce} . Pour cela, nous passons successivement dans ce réseau une image couleur de CompCar puis sa carte de contours calculée avec [Wang 2008]. La Figure 4.8 illustre les sorties de différentes couches convolutives du CNN. Nous observons que plus la couche observée est profonde, plus les sorties correspondant à ces deux images sont semblables. Cela confirme qu'un espace commun de caractéristiques visuelles existe entre ces deux domaines.

4.4.2.2 Évaluation sur CompCar

Nous avons évalué et comparé les performances du réseau \mathcal{N}_{ce}^{ce} (représenté dans la Figure 4.5 (a)) appris sur les deux domaines (couleurs réelles et contours réels) pour reconnaître la marque et le modèle d'un véhicule. Le Tableau 4.1 donne les résultats pour la reconnaissance fine sur les images de tests de CompCar (couleurs et contours). Nous pouvons voir clairement que nos descripteurs communs appris avec le réseau \mathcal{N}_{ce}^{ce} sont très performants sur les deux types de données alors que \mathcal{N}_c^c et \mathcal{N}_e^e échouent complètement sur le type de données sur lequel ils n'ont pas été appris. De plus, il est très intéressant de noter que le réseau \mathcal{N}_{ce}^{ce} fournit de meilleurs performances que le réseau \mathcal{N}_e^e lorsque ces deux réseaux sont testés sur des images de contours réelles (8% de plus pour \mathcal{N}_{ce}^{ce}). En d'autres termes, le réseau \mathcal{N}_e^e , appris uniquement sur des images de contours réels est moins performant sur ce même type de données. Cela prouve la pertinence de l'apprentissage conjoint sur deux domaines. Les informations de couleurs permettent de mieux apprendre les paramètres du réseau pour obtenir un classifieur de cartes de contours plus performant. Une autre explication possible de ce gain est le fait d'avoir utilisé un réseau pré-entraîné sur des images couleurs.

4.4.2.3 Évaluation sur 2DCar

Nous avons également évalué les performances du classifieur synthétique \mathcal{N}_s^{ce} (section 4.3.2) sur des images réelles couleurs. Le Tableau 4.2 donne les résultats pour la reconnaissance de marques et modèles de véhicules en utilisant des images de contours synthétiques pour l'apprentissage. Le test est fait sur la base 2DCar. Nous pouvons voir que notre classifieur \mathcal{N}_s^{ce} qui utilise les descripteurs de \mathcal{N}_{ce}^{ce} augmente considérablement le score de classification en comparaison avec les autres réseaux. Les résultats montrent également qu'apprendre directement un ré-

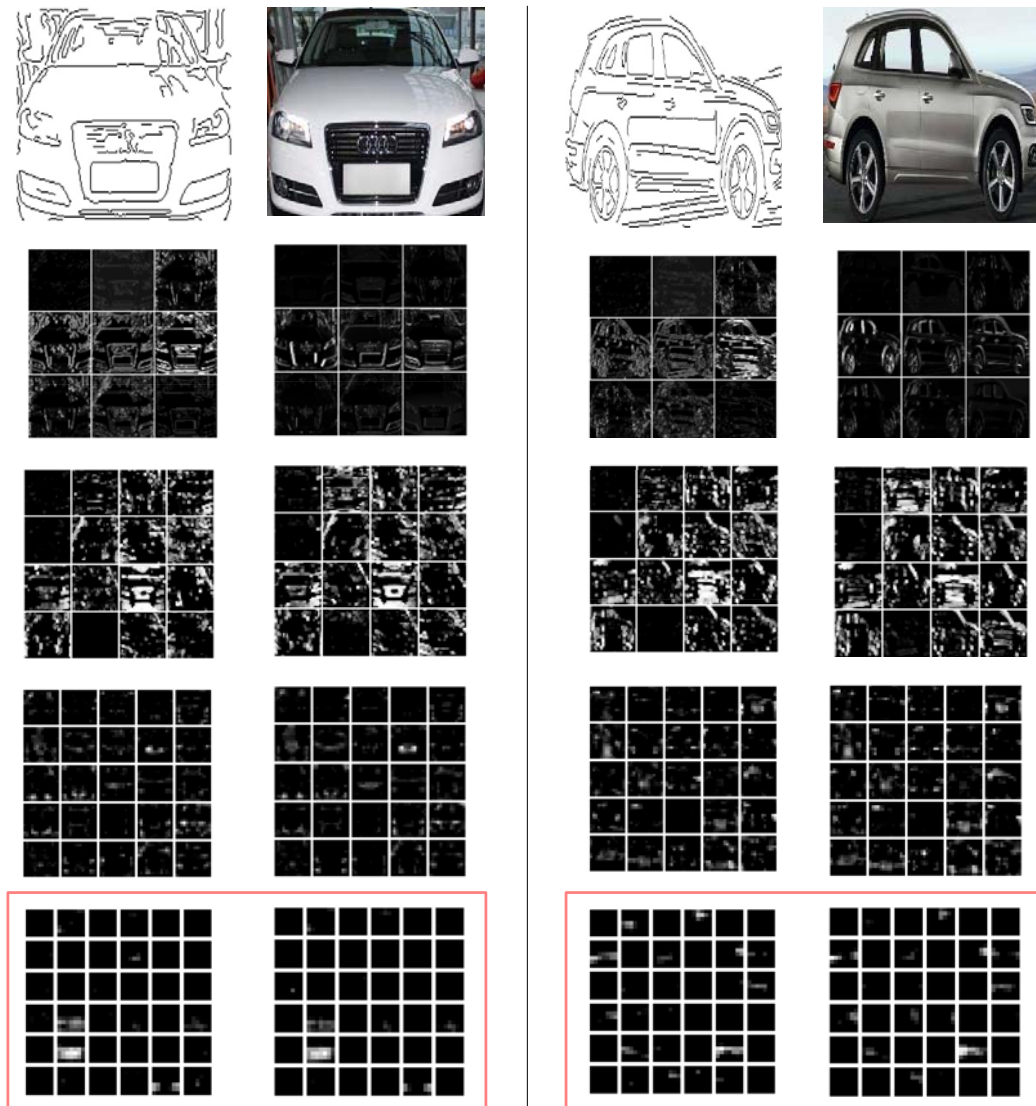


FIGURE 4.8 – Visualisation des caractéristiques extraites sur différentes couches du CNN invariant aux couleurs et aux contours. Nous donnons au CNN \mathcal{N}_{ce}^{ce} une image de couleurs et l'image de contours lui correspondant. Dans cette visualisation, la seconde ligne correspond à la sortie de la première couche convolutive du réseau (caractéristiques bas-niveau). La dernière ligne correspond à la sortie de la dernière couche du réseau (caractéristiques haut niveau). Les autres lignes correspondent à des sorties de couches intermédiaires. Nous pouvons remarquer que plus la couche considérée est profonde (de haut-niveau), plus les caractéristiques sur les deux domaines sont semblables.

Réseau	\mathcal{N}_{ce}^{ce}	\mathcal{N}_c^c	\mathcal{N}_e^e
Paramètres des descripteurs et du classifieur	Appris sur CompCar-color et CompCar-edge	Appris sur CompCar-color	Appris sur CompCar-edge
Performance sur CompCar-color	95.9 / 99.4	96.2 / 99.5	46.7 / 70.6
Performance sur CompCar-edge	92.0 / 98.6	3.8 / 12.0	84.5 / 96.4

TABLE 4.1 – Performances sur les images de test (couleurs et contours) de CompCar (rang 1 / rang 5).

seau complet sur des images synthétiques (\mathcal{N}_s^s) n'est pas efficace sur les images réelles lors du test (couleurs et contours). Ce réseau ne peut pas se généraliser à ce type d'images car il se concentre sur des structures synthétiques qui ne peuvent pas être trouvées dans les images réelles. Nous avons également appris des classifieurs en utilisant les SVM et structured-SVM et les résultats sont similaires. La Figure 4.9 montre les courbes de rang pour ces réseaux et la Figure 4.10 illustre des résultats renvoyés par notre réseau \mathcal{N}_s^{ce} .

Réseau	\mathcal{N}_s^{ce}	\mathcal{N}_s^c	\mathcal{N}_s^e	\mathcal{N}_s^s
Paramètres des descripteurs	fixés avec \mathcal{N}_{ce}^{ce}	fixés avec \mathcal{N}_c^c	fixés avec \mathcal{N}_e^e	Appris sur synthétiques
Paramètres classifieur	Appris sur synthétiques	Appris sur synthétiques	Appris sur synthétiques	
Performance sur 2DCar-color	83.6 / 96.0	42.7 / 77.7	53.6 / 83.6	4.7 / 18.1
Performance sur 2DCar-edge	69.7 / 95.2	28.5 / 64.8	56.6 / 88.6	10.2 / 29.3

TABLE 4.2 – Performances sur les images de test (couleurs et contours) de 2DCar (rang 1 / rang 5).

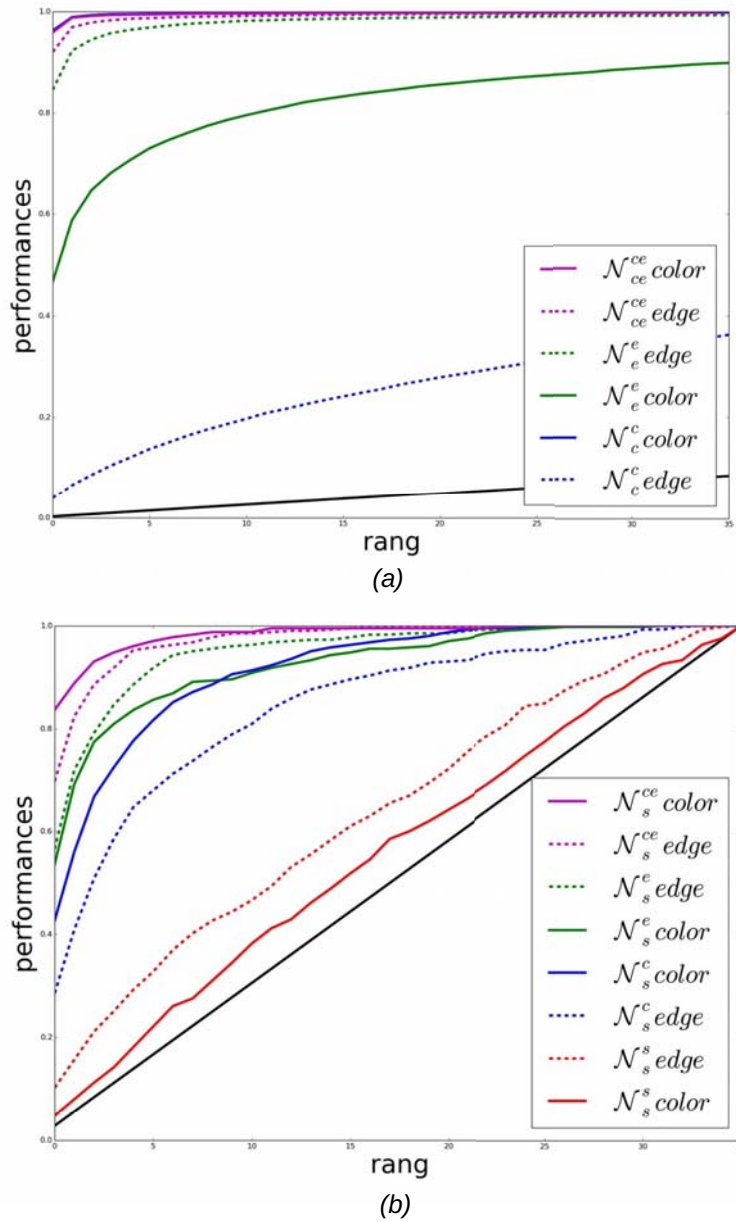


FIGURE 4.9 – Courbes de rang. L'axe des abscisses représente le rang $[1, \dots, N]$ l'axe des ordonnées la performance de reconnaissance de marques et modèles de véhicules. Les courbes de même couleur sont associées au même réseau mais testées sur différents types de données : contours (lignes pointillées) et couleurs (lignes entières). Les courbes magenta correspondent au réseau appris avec notre approche et la courbe noire au hasard. (a) Courbes de rang sur CompCar et (b) courbes de rang sur 2DCar.

4.5 Conclusion

L'approche proposée dans ce chapitre permet de calculer des descripteurs communs aux images couleurs et images contours. Ces descripteurs sont très discriminants pour la reconnaissance de marques et modèles de véhicules sur les deux domaines. Une fois ces descripteurs appris, nous les utilisons pour entraîner un classifieur fin sur des images de contours synthétiques pouvant être appliqué à des images couleurs réelles. La puissance de l'approche réside dans le fait qu'il est facile de générer pour l'entraînement beaucoup d'images synthétiques sans effort d'étiquetage. Nous donnons des résultats prouvant l'efficacité de la méthode. Ce travail montre la capacité des CNN à extraire des descripteurs invariants aux domaines et démontre leurs pouvoirs de généralisation pour la reconnaissance fine sur plusieurs domaines.







Requête	Rang 1	Rang 2	Rang 3	Rang 4	Rang 5
Citroen C4	 Citroen C4	 Toyota Matrix	 Audi Q7	 Mercedes R-Class	 Nissan Maxima
Chrysler Pacifica	 Nissan Altima	 Toyota RAV4	 Chrysler Pacifica	 Lexus LS430	 Chrysler PTCruiser
Mercedes C-Class	 Mercedes C-Class	 Nissan Altima	 Mercedes E400T	 Jaguar S-Type	 Mercedes R-Class
Alpha 147	 Alpha 147	 Peugeot 307	 Mercedes R-Class	 Audi Q7	 Lexus LS430

FIGURE 4.10 – Exemples de résultats obtenus pour la classification de marques et modèles de véhicule en utilisant le réseau \mathcal{N}_s^{ce} . Le réseau de neurones convolutif prend en entrée une image couleurs réelle de la base 2DCar (première colonne) et produit un score de confiance de classification pour chaque modèle 3D de la base 3DCar.

Détection de parties de véhicules pour l'estimation du point de vue précis

Sommaire

5.1	Introduction	75
5.2	Estimation de la pose par mise en correspondance 2D/3D	79
5.3	Modèle multi-vues pour l'estimation du point de vue	81
5.3.1	Modèle d'apparence	82
5.3.2	Modèle géométrique de cohérence spatiale	85
5.3.2.1	Composante de contexte global de boîte	86
5.3.2.2	Composante de contexte local de partie	86
5.3.2.3	Apprentissage des composantes géométriques	88
5.4	Estimation du point de vue précis	88
5.4.1	Détection d'hypothèses de parties	88
5.4.2	Sélection des hypothèses	89
5.4.2.1	Filtrage global	89
5.4.2.2	Filtrage local	91
5.4.3	Raffinement de la pose	92
5.5	Expérimentations	93
5.6	Conclusion	98

5.1 Introduction

Nous abordons dans ce chapitre la détection et l'estimation de point de vue précis de véhicule, qui est une autre problématique de la reconnaissance

fine. Un point de vue précis est caractérisé par les trois angles de rotation à appliquer à un objet initialement aligné dans le repère 3D de la caméra (vue canonique). Beaucoup de méthodes se sont focalisées sur la détection et l'estimation du point de vue grossier des objets, c'est-à-dire la prédiction d'intervalles d'angles assez larges. Ces méthodes proposent généralement de diviser la sphère de point de vue en 8 ou 16 intervalles et d'apprendre un détecteur multi-classes où chaque classe correspond à un point de vue discret [Lopez-Sastre 2011, Xiang 2014, Ozuysal 2009, Liebelt 2010, Pepik 2012]. Elles utilisent généralement des détecteurs basés sur le DPM [Lopez-Sastre 2011, Xiang 2014, Pepik 2012] ou les CNN [Tulsiani 2015]. Ces méthodes ont montré des résultats impressionnants [Pepik 2012, Tulsiani 2015], mais la notion de point de vue précis n'est pas abordée clairement car ces approches ne fournissent pas des angles de point de vue continus. L'utilisation de la géométrie 3D [Zia 2011, Zia 2013a, Pepik 2012] pour l'analyse de scène détaillée a reçu une attention particulière, apportant des informations plus fines pour décrire les objets. En utilisant un détecteur et un estimateur de point de vue grossier dans une étape d'initialisation, ces méthodes mettent ensuite en correspondance des modèles 3D [Lim 2013, Pepik 2015b, Aubry 2014, J.J.Lim 2014] ou des modèles 3D déformables [Zia 2011, Zia 2013b, Zia 2013a, Lin 2014b, Xiang 2012] avec l'objet dans l'image. Cette mise en correspondance est réalisée en minimisant une fonction de coût qui prend en compte des descripteurs d'apparence et de géométrie. Dans la lignée de ces approches, nous présentons une méthode qui détecte les véhicules dans une image et estime leurs points de vue précis en utilisant des modèles 3D non-texturés.

Notre approche est motivée par deux idées principales :

- La première est que des modèles 3D non-texturés peuvent être utilisés pour apprendre des caractéristiques visuelles locales à partir de rendus 3D non photo-réalistes de type contours. Ces caractéristiques visuelles peuvent également être trouvées dans des cartes de contours calculées sur des images réelles comme proposé par [Stark 2010]. La localité permet d'être moins sensible à la différence entre des images de contours synthétiques et des images de contours réelles permettant ainsi l'apprentissage de caractéristiques visuelles sur des données synthétiques. Contrairement au chapitre précédent qui calculait des caractéristiques visuelles globales pour la reconnaissance de marques et modèles de véhicules, l'approche abordée dans

ce chapitre se focalise sur la détection de parties. Nous avons considéré que localement, le biais visuel entre les contours d'une image synthétique et ceux d'une image réelle est négligeable ce qui implique de ne pas utiliser de méthode de type adaptation de domaine. De plus, les caractéristiques locales sont apprises grâce à des CNN peu profonds moins enclins au sur-apprentissage.

- La seconde idée est en lien avec la géométrie des objets rigides. La connaissance a priori de la géométrie de l'objet permet d'introduire des contraintes spatiales entre les différentes parties de celui-ci. Ces contraintes peuvent être apprises facilement et précisément en utilisant des modèles 3D. La localité permet d'être plus robuste aux occultations que les approches globales : dans notre méthode, même si plusieurs parties du véhicule ne sont pas détectées à cause des occultations, les parties visibles suffisent pour le détecter et calculer son point de vue précis.

Nous utilisons dans ce chapitre la même base de modèles 3D non-texturés évoquée dans le chapitre précédent pour apprendre des descripteurs d'apparence et de géométrie. Chaque modèle 3D a des sommets 3D annotés manuellement et correspondant à des parties de véhicules comme dans les travaux de [Zia 2013a]. L'algorithme présenté ici détecte les parties de véhicules visibles dans les images pour ensuite mettre en correspondance les sommets annotés des modèles 3D avec ces détections. Cette mise en correspondance permet de retrouver le point de vue précis de manière robuste [Levenberg 1944, Marquardt 1963]. Étant donné une image en entrée, les sorties de notre approche sont les suivantes : la boîte englobante du véhicule, la localisation de ses parties, son point de vue grossier et son point de vue précis. La figure 5.1 illustre les sorties de ce système.

Notre méthode permet de détecter les véhicules et d'estimer leurs poses précises conjointement, contrairement à beaucoup d'approches de la littérature qui utilisent séquentiellement un détecteur et un processus de raffinement. Généralement, ces méthodes commencent par une initialisation préalable en utilisant des détecteurs d'objets et de point de vue grossier de l'état de l'art pour ensuite estimer le point de vue précis. Dans notre approche, des parties des véhicules sont extraites en utilisant un détecteur de parties basé sur les CNN. Les détections sont ensuite filtrées en se basant sur des contraintes spatiales. L'estimation du point de vue continu se fait par la mise en correspondance de points 2D et 3D permettant

une grande précision. Un score final est introduit pour évaluer la qualité de la mise en correspondance et ainsi retrouver, la boîte de détection du véhicule, son point de vue grossier et fin.

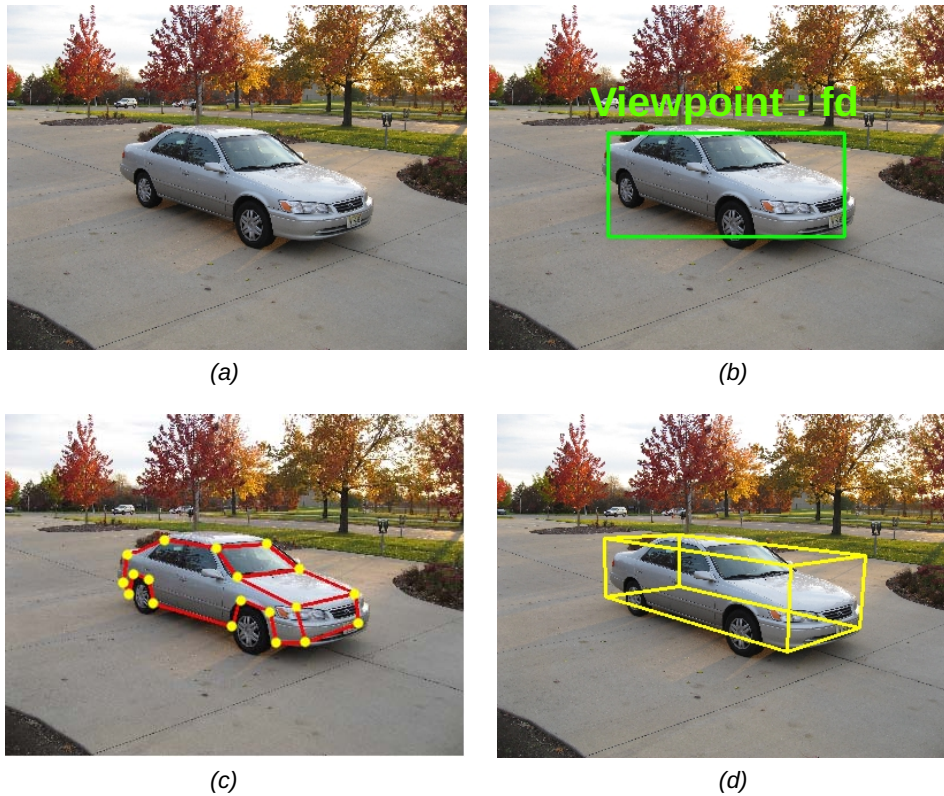


FIGURE 5.1 – Les sorties du système de détection de point de vue précis. Ce système prend en entrée une image (a) et trouve simultanément : (b) la boîte 2D du véhicule et le point vue grossier associé (dans cet exemple, le point de vue grossier correspond à l’azimut face-droit), (c) les parties visibles du véhicule (points jaunes) et (d) le point de vue précis.

Dans la section 5.2, nous rappelons le formalisme utilisé pour le calcul de pose par mise en correspondance de points 2D avec des points 3D. Dans les sections, 5.3 et 5.4, nous détaillons le modèle adopté et la méthode de reconnaissance de point de vue fin. Enfin, nous présentons des résultats expérimentaux et des comparaisons avec l’existant sur la base de données publique *3D Object dataset* [Savarese 2007].

5.2 Estimation de la pose par mise en correspondance 2D/3D

Nous rappelons ici le formalisme du modèle de caméra sténopé pour le calcul de la transformation perspective (calcul de pose) entre des points 3D dans le repère de la caméra et leurs correspondants 2D sur le plan de l'image. L'objectif du calcul de pose est de trouver l'orientation des objets (les trois angles) ainsi que sa position dans la scène 3D en utilisant des points caractéristiques de l'objet observés dans une image. Le modèle de caméra sténopé représente une caméra parfaite, c'est-à-dire (1) sans prise en compte des déformations de l'image liées aux propriétés physiques des lentilles réelles (distorsion) et (2) en considérant que tous les rayons lumineux passent par le même point. Dans notre cas, nous considérons que les images sont rectifiées, c'est-à-dire que la distorsion est préalablement corrigée. La figure 5.2 illustre un tel modèle.

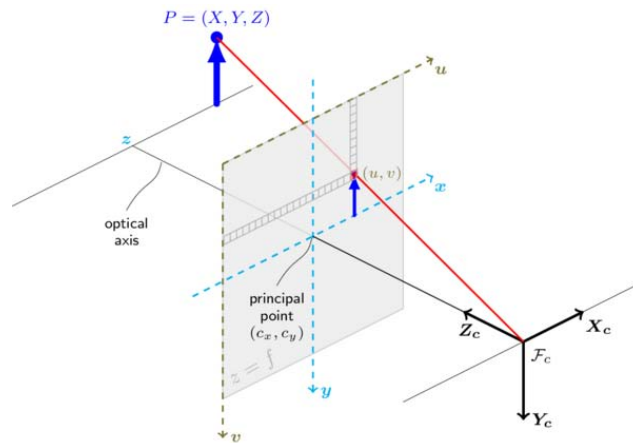


FIGURE 5.2 – Modèle de caméra sténopé. Source OpenCV.

Dans ce qui suit, les coordonnées euclidiennes d'un point 2D dans l'image s'écrivent $\mathbf{q} = [u, v]^T$ et les coordonnées euclidiennes de son correspondant dans le repère 3D $\mathbf{Q} = [X, Y, Z]^T$. Nous définissons les coordonnées homogènes de \mathbf{q} par $\tilde{\mathbf{q}} = [u, v, 1]^T$ et celles de \mathbf{Q} par $\tilde{\mathbf{Q}} = [X, Y, Z, 1]^T$. Ce type de coordonnées permet de modéliser simplement les transformations projectives. La relation qui lie $\tilde{\mathbf{q}}$ et $\tilde{\mathbf{Q}}$ dans un modèle de caméra sténopé, à un facteur d'échelle s près, est la suivante :

$$s\tilde{\mathbf{q}} = \mathbf{M}\tilde{\mathbf{Q}} \quad (5.1)$$

avec $\mathbf{M} = \mathbf{A}[\mathbf{R}|\mathbf{t}]$ la matrice de projection (de taille 3×4). \mathbf{R} est une matrice de rotation de taille 3×3 et \mathbf{t} est un vecteur de translation de taille 3×1 . Ces deux transformations correspondent aux paramètres extrinsèques de la caméra. \mathbf{A} est la matrice calibration de la caméra (paramètres intrinsèques) et est définie par :

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (5.2)$$

où f_x et f_y sont les distances focales de la caméra exprimées en pixels. (c_x, c_y) est le point principal. Ce point correspond aux coordonnées dans l'image de la projection du centre optique, généralement proche des coordonnées du centre de l'image. Cette matrice est obtenue par une étape de calibration de caméra.

Le calcul de pose par mise en correspondance 2D/3D a pour but d'estimer $\hat{\mathbf{R}}$ et $\hat{\mathbf{t}}$ étant donné un ensemble de N points 2D $\{\mathbf{q}_i\}_{i \in [1, \dots, N]}$, leurs correspondants en 3D $\{\mathbf{Q}_i\}_{i \in [1, \dots, N]}$ et une matrice de calibration \mathbf{A} . Cela se caractérise par la minimisation de la fonction d'erreur E :

$$\hat{\mathbf{R}}, \hat{\mathbf{t}} = \underset{\mathbf{R}, \mathbf{t}}{\operatorname{argmin}} E(\mathbf{R}, \mathbf{t}) \quad (5.3)$$

avec

$$E(\mathbf{R}, \mathbf{t}) = \sum_i \|\tilde{\mathbf{q}}_i - \mathbf{A}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{Q}}_i\| \quad (5.4)$$

Cette erreur à minimiser est appelée erreur de projection. Des algorithmes d'optimisation existent pour résoudre itérativement ce genre de problème comme l'algorithme classique de Levenberg-Marquardt [Levenberg 1944, Marquardt 1963]. D'autres travaux ont proposé une résolution directe et donc plus rapide de ce problème [Lepetit 2009, Gao 2003]. En considérant que les points dans l'image et les points 3D sont localisés de manière exacte, ces méthodes d'optimisation fonctionnent très bien. Dans la réalité, cela n'est pas le cas : la mise en correspondance est souvent parasitée par de mauvaises détections de points 2D dans l'image. C'est pourquoi, les méthodes d'optimisation sont souvent utilisées conjointement avec des algorithmes de tirage aléatoire itératif de type RANSAC (*Random Sample Consensus*) [Fischler 1981]. Pour chaque itération, l'idée du RANSAC est de considérer aléatoirement un sous ensemble des points 2D. La mise en correspondance 2D/3D est alors calculée en utilisant seulement ce sous ensemble. Les paramètres extrinsèques $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ sont alors estimés. Ils sont ensuite utilisés pour projeter

tous les points 3D dans l'image. Le nombre de points bien appariés est alors calculé sur l'ensemble des points 2D et des points 3D. Les paramètres extrinsèques qui maximisent le nombre de points bien appariés sont sauvegardés. Cette procédure, décrite dans l'algorithme 1, permet d'introduire de la robustesse dans la mise en correspondance.

Algorithm 1 Algorithme du RANSAC pour le calcul de pose

```

1: procédure RANSAC( $\{\mathbf{Q}_i\}, \{\mathbf{q}_i\}, \mathbf{A}$ )
2:    $nbInliersBest \leftarrow 0$ 
3:    $\mathbf{R} \leftarrow null$ 
4:    $\mathbf{t} \leftarrow null$ 
5:   for  $k := 0; k < numberOfIteration; k++$  do
6:      $\{\tilde{\mathbf{q}}_j\}, \{\tilde{\mathbf{Q}}_j\} \leftarrow RandomSubSet(\{\tilde{\mathbf{q}}_i\}, \{\tilde{\mathbf{Q}}_i\})$ 
7:      $\hat{\mathbf{R}}, \hat{\mathbf{t}} \leftarrow PoseComputation(\{\tilde{\mathbf{q}}_j\}, \{\tilde{\mathbf{Q}}_j\}, \mathbf{A})$ 
8:      $nbInliers \leftarrow ComputeInliers(\{\tilde{\mathbf{q}}_i\}, \{\tilde{\mathbf{Q}}_i\}, \mathbf{A}, \hat{\mathbf{R}}, \hat{\mathbf{t}})$ 
9:     if  $nbInlier > nbInlierBest$  then
10:        $nbInlierBest \leftarrow nbInlier$ 
11:        $\mathbf{R} \leftarrow \hat{\mathbf{R}}$ 
12:        $\mathbf{t} \leftarrow \hat{\mathbf{t}}$ 
13:   return  $\mathbf{R}, \mathbf{t}$ 

```

5.3 Modèle multi-vues pour l'estimation du point de vue

Notre modèle d'estimation de point de vue est défini par $\mathcal{M} = (\mathcal{M}_1, \dots, \mathcal{M}_N)$ où N est le nombre de points de vue grossiers et \mathcal{M}_v le modèle du point de vue grossier v . Chaque modèle \mathcal{M}_v est composé d'un modèle d'apparence de partie \mathcal{D}_v et d'un modèle de cohérence spatiale \mathcal{G}_v . Il est défini comme suit :

$$\mathcal{M}_v = (\mathcal{D}_v, \mathcal{G}_v)$$

Chaque modèle d'apparence \mathcal{D}_v est utilisé pour détecter les parties de véhicules visibles dans le point de vue grossier v fournissant ainsi des hypothèses de parties. Les notions de point de vue grossier et de visibilité de parties sont illustrées dans la Figure 5.3. Le modèle de cohérence spatiale permet de choisir les hypothèses

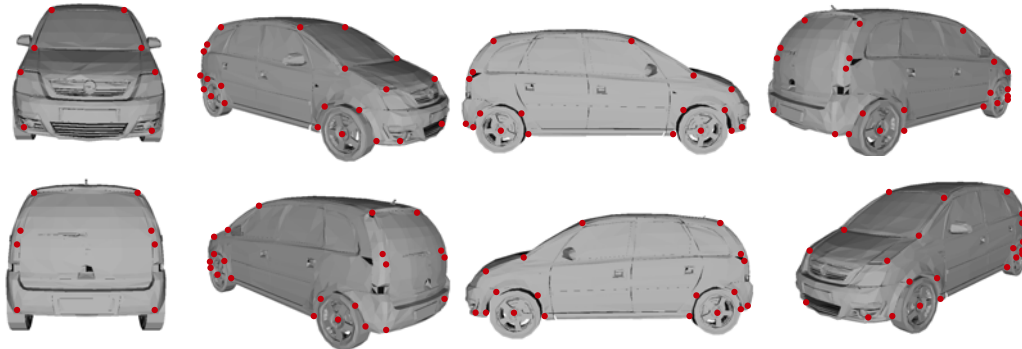


FIGURE 5.3 – Illustration du point de vue grossier et de la visibilité des parties. Ici, nous représentons huit points de vue grossiers. Les parties en rouge correspondent aux parties visibles dans ce point de vue. Cette visibilité est prédéfinie manuellement comme dans les travaux de [Zia 2011].

géométriquement cohérentes. Le modèle d'apparence \mathcal{D}_v et le modèle géométrique \mathcal{G}_v sont entraînés grâce aux modèles 3D. Ils sont détaillés dans les sections 5.3.1 et 5.3.2. La Figure 5.4 illustre de manière générale l'utilisation de notre modèle multi-vues.

5.3.1 Modèle d'apparence

Le rôle du modèle d'apparence est de détecter, pour chaque point de vue grossier v , un ensemble de points 2D dans l'image d'entrée correspondant aux parties 3D annotées et visibles dans v . Chaque \mathcal{D}_v consiste en un détecteur de parties multi-classes de $n_v + 1$ classes (n_v est le nombre de parties visibles dans v + une classe de fond). Ces détecteurs n'apprennent pas l'apparence globale du véhicule comme dans l'approche DPM [Felzenszwalb 2010] mais fournissent seulement des détections de parties locales. Chaque détecteur \mathcal{D}_v est appris avec un CNN inspiré du classifieur LeNet introduit par [Lecun 1998] pour la classification de chiffres manuscrits. Chaque CNN est composé de deux étapes d'extraction de caractéristiques formées par une couche de convolution, une fonction d'activation (Tangente hyperbolique) et une couche de Max Pooling. Les filtres de convolution sont entraînés avec une taille de noyau de 5×5 . La partie du CNN dédiée à la détection (couches complètement connectées) est formée par une couche cachée et une couche de sortie de taille $n_v + 1$. Nous utilisons la fonction de perte Softmax pour entraîner ces détecteurs. La figure 5.5 illustre l'architecture du CNN.

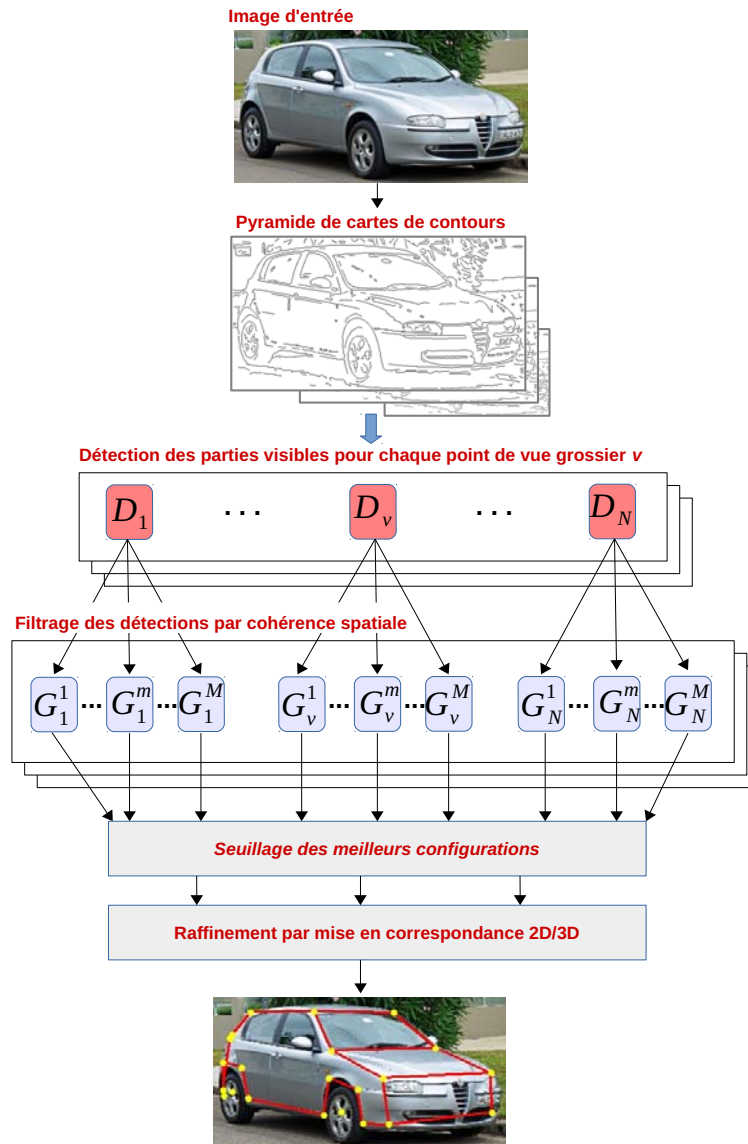


FIGURE 5.4 – Le déroulement général de l’algorithme. Étant donné une image d’entrée, une pyramide de cartes de contours est calculée. Les N modèles d’apparence (détecteur de parties visibles) sont ensuite appliqués à chaque niveau de la pyramide. Le filtrage des hypothèses de parties est effectué par le modèle de cohérence spatiale. Les meilleures configurations de parties sont gardées en appliquant un seuil. Enfin, un calcul de pose 2D/3D est effectué pour retrouver le point de vue précis.

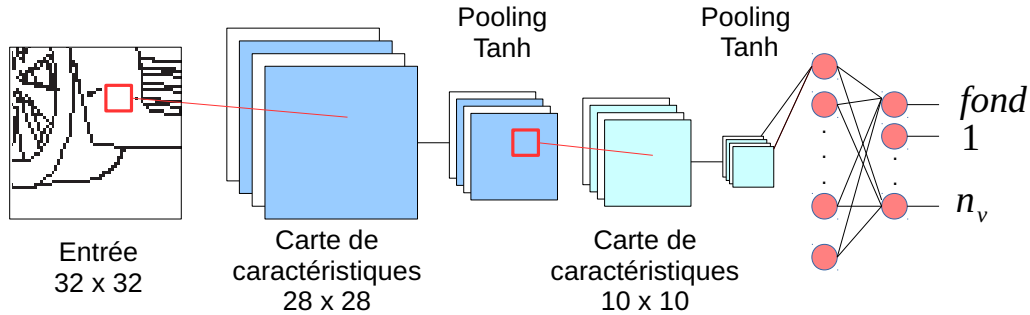


FIGURE 5.5 – Architecture du CNN pour un point de vue grossier v . Elle consiste en deux couches d'extraction de caractéristiques (Convolution / Max Pooling / Tanh) et deux couches complètement connectées.

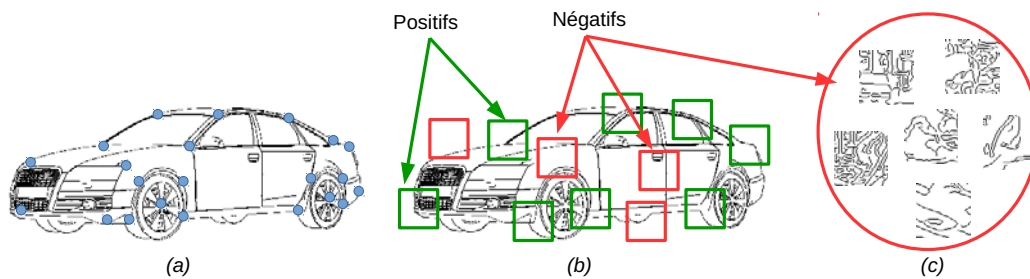


FIGURE 5.6 – Génération de patches positifs et négatifs pour l'apprentissage du modèle d'apparence. Le modèle 3D est projeté en rendu contours pour un certain jeu de paramètres extrinsèques de caméra (rotation et translation). Les paramètres de rotation ont pour seule contrainte d'être inclus dans les intervalles d'angles définis par le point de vue grossier v . Les parties 3D visibles dans v sont également projetées (points bleus) (a). Des patches positifs et négatifs sont extraits sur l'image de rendu (b). Des patches négatifs sont également extraits d'une base de données réelles sur laquelle a été appliqué un détecteur de contours (c).

Pour générer beaucoup d'exemples positifs pour l'apprentissage de ces modèles d'apparence sans un travail fastidieux d'annotation sur des images réelles, nous créons des bases d'images de parties en utilisant des modèles 3D. Des patches correspondant à des parties de véhicules sont extraits de rendus non photo-réalistes de type contours (voir section 4.2.1). Ces patches sont de taille 32×32 et sont centrés sur les projections des sommets 3D manuellement annotés. Ainsi, pour chaque modèle 3D de notre base de données, et pour chacun des N points de vues grossiers v , des exemples de parties positifs et négatifs sont générés pour plusieurs

paramètres extrinsèques de caméra autour du point de vue de référence v . Des exemples négatifs supplémentaires sont également extraits aléatoirement de la base PASCAL VOC 2007 et passés dans un détecteur de contour [Wang 2008]. La figure 5.6 illustre cette étape de génération de données pour l'apprentissage des modèles d'apparence.

5.3.2 Modèle géométrique de cohérence spatiale

Le modèle de cohérence spatiale géométrique encode la cohérence entre les parties du véhicule. Le fait que les véhicules soient des objets rigides permet d'introduire de fortes contraintes géométriques sur leurs parties. Notre modèle de cohérence spatiale géométrique est composé de plusieurs sous-modèles associés à chaque type de véhicule. Chaque sous-modèle correspond à un modèle 3D de notre base composée de M véhicules. Nous motivons ce choix par le fait que les parties d'un véhicule sont localisées différemment suivant son type : par exemple, les parties arrière d'un véhicule ne sont pas au même endroit pour un SUV ou une berline. Nous considérons que la variabilité de notre base de modèles 3D est suffisante pour représenter toutes les formes de véhicules possibles. En utilisant ces sous-modèles, nous n'avons pas à estimer des paramètres de forme contrairement aux approches basées sur des modèles 3D déformables [Zia 2011, Zia 2013b, Zia 2013a, Lin 2014b, Xiang 2012]. Dans notre méthode, l'espace des formes possibles est de dimension M , alors qu'un modèle déformable encode un espace de formes infini. Notre modèle de cohérence spatiale géométrique est défini comme suit :

$$\mathcal{G}_v = \{\mathcal{G}_v^m\}_{m \in \{1 \dots M\}} \quad (5.5)$$

où \mathcal{G}_v^m est le sous-modèle géométrique associé au point de vue grossier v et au modèle 3D m . Chaque sous-modèle est composé de deux composantes géométriques :

- la composante de contexte global de boîte \mathcal{B}_v^m
- la composante de contexte local de partie \mathcal{P}_v^m

Ainsi, le sous-modèle géométrique \mathcal{G}_v^m est défini par :

$$\mathcal{G}_v^m = (\mathcal{B}_v^m, \mathcal{P}_v^m) \quad (5.6)$$

La première composante \mathcal{B}_v^m encode la localisation de la boîte englobante du véhicule connaissant les parties visibles. La seconde composante \mathcal{P}_v^m encode la position relative entre chaque paire de parties. Nous avons constaté que l'utilisation de composantes globales et locales pour un filtrage des détections cascadié est essentielle pour obtenir une cohérence spatiale robuste et une reconnaissance efficace. Après la détection de parties, nous commençons par supprimer un grand nombre de fausses détections en utilisant la composante de contexte global. Ensuite, la composante de contexte local est utilisée pour sélectionner les parties du véhicule les plus pertinentes. \mathcal{P}_v^m est plus discriminante que \mathcal{B}_v^m car elle se base sur des connexions entre parties et est donc plus contrainte.

5.3.2.1 Composante de contexte global de boîte

La composante de contexte global de boîte \mathcal{B}_v^m pour un modèle 3D m et un point de vue grossier v où n_v parties sont visibles, est définie formellement par $\mathcal{B}_v^m = (\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n_v})$. \mathbf{B}_i est un vecteur à 8 dimensions spécifiant les déplacements extrêmes de la boîte englobante du véhicule (haut-gauche (*top-left*) et bas-droit (*bottom-right*)) relativement à la position de la partie i soit :

$$\mathbf{B}_i = [\Delta x_{min}^{tl}, \Delta x_{max}^{tl}, \Delta y_{min}^{tl}, \Delta y_{max}^{tl}, \Delta x_{min}^{br}, \Delta x_{max}^{br}, \Delta y_{min}^{br}, \Delta y_{max}^{br}] \quad (5.7)$$

Δx^{tl} , Δy^{tl} , Δx^{br} et Δy^{br} sont représentés dans la figure 5.7 (a). Comme dans le DPM [Felzenszwalb 2010], le composant de contexte global décrit la distribution des parties à l'intérieur de la boîte englobante de l'objet. La différence principale avec le DPM repose sur l'information spatiale exacte et illimitée donnée par les modèles 3D. Cette information permet d'apprendre la position de la boîte englobante sans utiliser de variables latentes. La figure 5.7 (b) illustre la composante de contexte global.

5.3.2.2 Composante de contexte local de partie

De manière similaire, la composante de contexte local de partie \mathcal{P}_v^m est définie par $\mathcal{P}_v^m = (\mathbf{P}_{1,2}, \mathbf{P}_{1,3}, \dots, \mathbf{P}_{n_v, (n_v-1)})$ où $\mathbf{P}_{i,j}$ est un vecteur de 4 dimensions qui encode la distance minimum et maximum ainsi que l'orientation minimum et maximum entre la partie i et la partie j (par rapport à l'axe horizontal), soit :

$$\mathbf{P}_{i,j} = [d_{min}, d_{max}, \theta_{min}, \theta_{max}] \quad (5.8)$$

La distance d et l'orientation θ sont représentés dans la figure 5.7 (c) et la figure 5.7 (d) illustre le composant de contexte local.

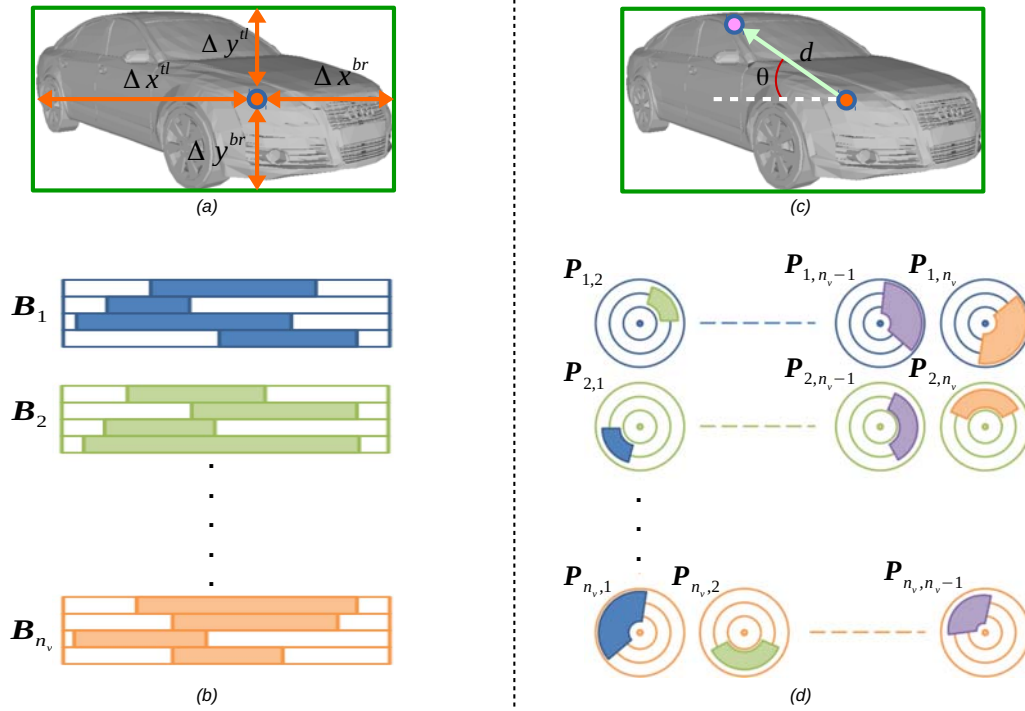


FIGURE 5.7 – Construction du modèle de cohérence spatial géométrique pour un point de vue grossier v et un modèle 3D m . (a) représente un exemple de modèle 3D projeté et les déplacements de boîte associés Δx^{tl} , Δy^{tl} , Δx^{br} , Δy^{br} pour une partie donnée. (b) illustre les extremums de ces déplacements après l'apprentissage du modèle de contexte global $\mathcal{B}_v^m = (\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n_v})$. Chaque ligne représente une partie visible. (c) illustre la distance d et l'orientation θ entre deux parties. (d) illustre les extremums de distance et d'orientation après l'apprentissage du modèle de contexte local $\mathcal{P}_v^m = (\mathbf{P}_{1,2}, \mathbf{P}_{1,3}, \dots, \mathbf{P}_{n_v, (n_v-1)})$. Chaque ligne correspond à une partie donnée et chaque colonne à la distribution possible d'une autre partie par rapport à elle.

5.3.2.3 Apprentissage des composantes géométriques

Pour apprendre les paramètres de ces deux modèles, les parties du modèle 3D m sont projetées avec plusieurs paramètres extrinsèques autour du point de vue grossier v considéré. Pour chaque paramètre, la boîte englobante est calculée ainsi que les coordonnées des parties visibles. Contrairement à des bases de données d'images réelles, notre base d'images synthétiques recouvre un grand nombre de points de vue possibles et fournit donc des modèles de cohérence spatiale très précis.

5.4 Estimation du point de vue précis

Dans cette section, nous détaillons l'algorithme de détection et d'estimation du point de vue précis. Premièrement, étant donné une image en entrée, nous calculons une pyramide de cartes de contours. Ensuite, pour chaque point de vue grossier v , le détecteur de partie D_v est appliqué pour détecter des hypothèses de parties. De ces hypothèses, les configurations les plus pertinentes géométriquement sont sélectionnées grâce au modèle de cohérence spatial. Enfin, nous calculons un score de mise en correspondance 2D/3D dans une étape de raffinement. La maximisation de ce score sur les différentes échelles, les points de vue grossiers, et les modèles 3D renvoie la boîte de détection du véhicule, son point de vue grossier et précis ainsi que les coordonnées des parties visibles. La Figure 5.4 illustre le déroulement de notre approche de détection et d'estimation du point de vue fin.

5.4.1 Détection d'hypothèses de parties

La première étape importante de l'algorithme est de fournir des hypothèses de parties pour chaque point de vue grossier v et chaque niveau d'échelle l . Une partie de véhicule est caractérisée par sa position dans l'image. Pour chaque échelle l et point de vue grossier v , le modèle d'apparence de parties \mathcal{D}_v est appliqué à la carte de contours de l'image. Pour permettre une détection rapide, \mathcal{D}_v est préalablement transformé en un CNN complètement convolutif : les couches complètement connectées sont remplacées par des convolutions comme dans [Sermanet 2014, Long 2015a]. Cela permet le passage de l'image complète dans \mathcal{D}_v et l'obtention d'une carte de scores pour chaque partie visible dans v . Ces cartes de scores sont 4 fois moins résolues que l'image d'origine du fait de la ré-

duction de dimension opérée dans le CNN. Cela signifie qu'une possible détection d'une partie est donnée tous les 4 pixels dans l'image d'entrée. L'ensemble des hypothèses de parties détectées pour un v et un l s'écrit :

$$H_v^l = \{\mathbf{h}_{i_k} = [u_{i_k}, v_{i_k}]^\top\}_{\substack{i \in \{1, \dots, n_v\} \\ k \in \{1, \dots, n_i\}}} \quad (5.9)$$

où n_i correspond au nombre d'hypothèses associées à la partie visible i . Ainsi, \mathbf{h}_{i_k} est la k -ème hypothèse de la partie i ayant pour coordonnées dans l'image $[u_{i_k}, v_{i_k}]^\top$.

5.4.2 Sélection des hypothèses

La sélection des hypothèses de parties pertinentes se fait via le modèle de cohérence spatiale \mathcal{G}_v . Cette étape peut être vue comme une étape de filtrage des fausses détections en introduisant des contraintes géométriques sur celles-ci. Cela consiste en deux filtrages successifs. Premièrement, le filtrage global élimine efficacement de fausses détections en groupant les parties qui appartiennent potentiellement au même véhicule. Deuxièmement, le filtrage local supprime les configurations de parties résultantes qui ne respectent pas la cohérence spatiale locale. La figure 5.8 illustre cette sélection des hypothèses pertinentes. Une configuration de parties est un ensemble d'hypothèses défini par :

$$C_{l,v} = \{\mathbf{h}_i, i \in \{1, \dots, n_v\}\} \quad (5.10)$$

Chaque hypothèse de $C_{l,v}$ est unique pour la partie i considérée soit $\mathbf{h}_i \in \{\mathbf{h}_{i_1}, \dots, \mathbf{h}_{i_{n_i}}\}$. Une configuration contient donc au plus n_v hypothèses (dans le cas où il existe au moins une hypothèse pour chaque partie). La figure 5.9 illustre la notion de configuration de parties.

5.4.2.1 Filtrage global

Un filtrage global est appliqué pour chaque point de vue grossier v et chaque modèle 3D m . Toutes les hypothèses de parties de H_v^l votent pour une boîte englobante en utilisant le composant de contexte global \mathcal{B}_v^m . Plus précisément, un accumulateur à 4 dimensions correspondant aux coins de la boîte englobante (*gauche, haut, droit, bas*) est défini. Chaque hypothèse k de la partie i de H_v^l incrémente cet accumulateur pour les 4 dimensions en utilisant sa coordonnée et le

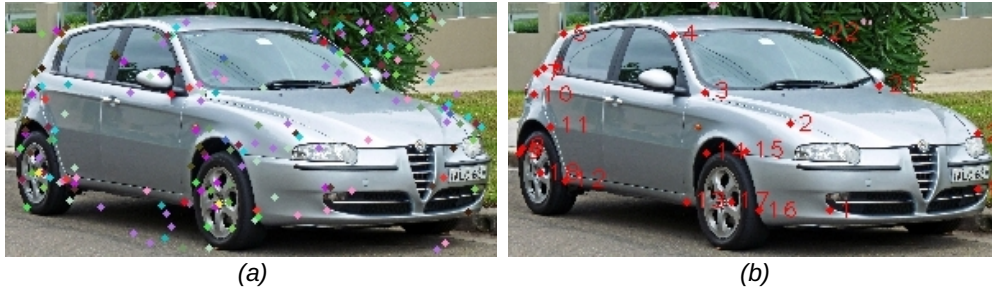


FIGURE 5.8 – Sélection des hypothèses pertinentes par cohérence spatiale pour une échelle l , un point de vue grossier donné v et un modèle 3D m . (a) Les hypothèses de parties visibles résultantes de la détection opérée par le modèle d'apparence \mathcal{D}_v . Chaque couleur représente une classe de parties. On observe de nombreuses fausses alarmes. (b) Le résultat de la sélection de parties par cohérence spatiale opérée par \mathcal{G}_v^m .

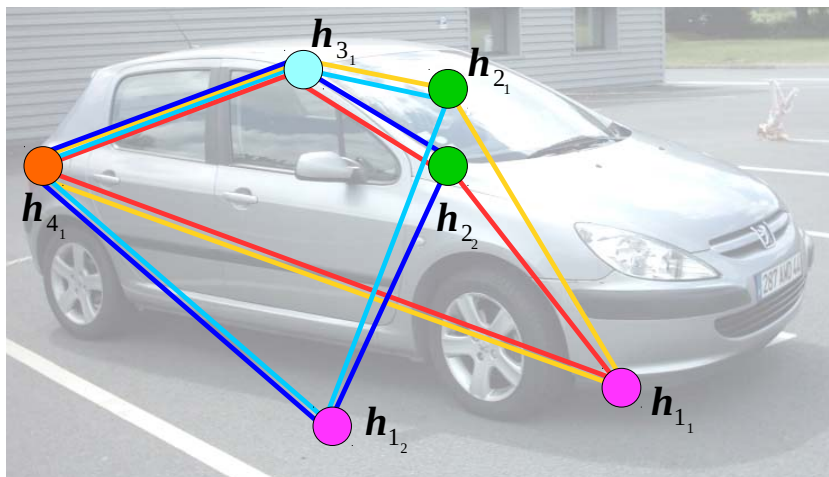


FIGURE 5.9 – Illustration de la notion de configuration de parties. Dans cet exemple, nous avons représenté quatre parties (points de couleur). Pour deux de ces parties, deux hypothèses ont été générées par le détecteur (deux points verts pour la partie 2 et deux points magenta pour la partie 1). Cela fournit quatre configurations de parties possibles, soit les graphes rouge, jaune, bleu et cyan.

vecteur des extremums \mathbf{B}_i de \mathcal{B}_v^m . À ce stade, nous proposons d'éliminer les hypothèses qui n'ont pas voté pour les maximums de l'accumulateur.

Les hypothèses ayant contribué au maximum local de l'accumulateur sont gardées. La figure 5.10 illustre le fonctionnement de l'accumulateur. En utilisant ces nouvelles hypothèses, un ensemble de configurations de parties est généré :

$$\mathbf{C}_{l,v}^m = \{C_{l,v}^m(k)\}_{k \in \{1..n_{conf}\}} \quad (5.11)$$

où n_{conf} est le nombre de configurations générées. Chaque configuration est globalement cohérente en termes de distribution spatiale dans sa boîte englobante.

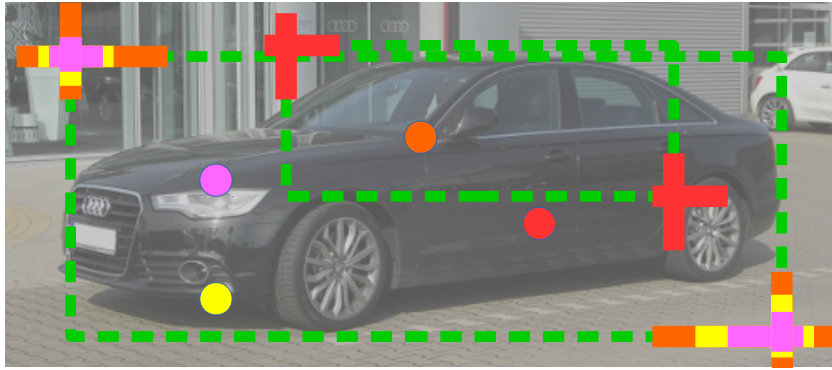


FIGURE 5.10 – Fonctionnement de l'accumulateur pour le filtrage global des hypothèses de parties. Chacun des quatre points représente une hypothèse de partie. Ces points sont représentés avec des couleurs différentes car ils correspondent chacun à une partie précise. Le point rouge est une fausse détection. L'utilisation du filtrage global permet de supprimer cette hypothèse. En utilisant le modèle de contexte géométrique global du point de vue grossier v , chacune des parties vote pour une boîte englobante à quatre degrés de liberté. Nous pouvons voir dans cet exemple que les trois points correspondant à de bonnes détections votent pour la même boîte, tandis que la fausse détection vote pour une autre boîte. Ces boîtes sont représentées en vert.

5.4.2.2 Filtrage local

Ce filtrage sélectionne les configurations qui ont une forte cohérence spatiale inter-parties. Dans cette étape, nous introduisons un score de cohérence spatiale associé à une configuration. Cette fonction de score Φ prend en compte l'apparence

et la géométrie et se définit comme suit :

$$\Phi(C_{l,v}^m) = \sum_{\substack{\mathbf{h}_i \in C_{l,v}^m, \mathbf{h}_j \in C_{l,v}^m \\ i \neq j}} \frac{(s(\mathbf{h}_i) + s(\mathbf{h}_j))c(\mathbf{h}_i, \mathbf{h}_j)}{2} \quad (5.12)$$

où $s(\mathbf{h})$ représente le score de confiance produit par le CNN pour l'hypothèse \mathbf{h} et c est une fonction de connexion booléenne définie par :

$$c(\mathbf{h}_i, \mathbf{h}_j) = \begin{cases} 1, & \text{si } \mathbf{h}_i \text{ et } \mathbf{h}_j \text{ sont connectées} \\ 0, & \text{sinon} \end{cases} \quad (5.13)$$

Le critère de connexion est donné par le modèle de cohérence local de partie \mathcal{P}_v^m : deux hypothèses \mathbf{h}_i et \mathbf{h}_j sont connectées si la longueur et l'orientation de l'arête qui les relie sont dans les limites définies par les paramètres d'extremum de $\mathbf{P}_{i,j}$.

Nous sélectionnons alors les meilleures configurations de parties en calculant ce score pour toutes les configurations de $\mathbf{C}_{l,v}^m$, puis en appliquant un seuil ϕ_v . Ce seuil, dépendant du point de vue grossier, permet de filtrer les configurations aberrantes. Sa valeur est fixée à $\frac{n_v(n_v-1)}{2}$ et permet de supprimer les configurations dont la moitié des connexions inter-parties ne sont pas géométriquement cohérentes. Un nouvel ensemble de configurations de parties $\hat{C}_{l,v}^m$ est alors défini :

$$\hat{C}_{l,v}^m = \{C_{l,v}^m(k) \in \mathbf{C}_{l,v}^m, \Phi(C_{l,v}^m(k)) \geq \phi_v\} \quad (5.14)$$

5.4.3 Raffinement de la pose

L'étape de raffinement permet de mettre en correspondance les parties 2D détectées dans l'image avec les parties 3D prédéfinies pour chaque modèle 3D m permettant ainsi de retrouver le point de vue précis. Cette étape attribue également un score de mise en correspondance pour chaque configuration de parties. La matrice de projection optimale et le modèle 3D associé sont obtenus en maximisant les scores sur tous les niveaux de la pyramide, tous les points de vue grossiers, et tous les modèles 3D. En utilisant un algorithme de mise en correspondance 2D/3D basé sur l'optimisation de Levenberg-Marquardt [Levenberg 1944, Marquardt 1963] et sur un tirage aléatoire de type RANSAC (voir section 5.2), nous calculons la matrice de projection $\mathbf{M}_{l,v}^m$ pour chaque configuration de parties $C_{l,v}^m \in \hat{C}_{l,v}^m$ qui met en correspondance les parties 2D avec les parties 3D du modèle m . Il est important de noter que si les paramètres intrinsèques de la caméra ne sont pas connus

(focale et centre optique), nous procédons à une estimation de ces paramètres via une pseudo-calibration de celle-ci. En d'autres termes, nous utilisons les parties 2D détectées et les parties 3D pour fournir une estimation de ces paramètres. Même si cette estimation n'est pas très précise, elle est suffisante pour l'estimation du point de vue précis. Après la mise en correspondance 2D/3D, tous les points 3D visibles du modèle m sont projetés dans l'image pour fournir la configuration de parties complète définie par :

$$\dot{C}_{l,v}^m = \{\mathbf{q}_i^m\}_{i \in \{1, \dots, n_v\}} \quad (5.15)$$

où $\tilde{\mathbf{q}}_i^m = \mathbf{M}_{l,v}^m \tilde{\mathbf{Q}}_i^m$ avec \mathbf{q}_i^m la projection de la partie 3D visible i du modèle 3D m , $\tilde{\mathbf{Q}}_i^m$. Chaque projection de partie se voit attribuer une probabilité d'être effectivement une partie en utilisant les cartes de scores renvoyées par les CNN. Le score de mise en correspondance global $\bar{\Phi}$ pour chaque configuration est alors défini par :

$$\bar{\Phi}(\dot{C}_{l,v}^m) = \frac{\Phi(\dot{C}_{l,v}^m)}{n_v(n_v - 1)} \quad (5.16)$$

La configuration finale $C_{f,l,v}^m$ est alors donnée par :

$$C_{f,l,v}^m = \underset{\dot{C}_{l,v}^m}{\operatorname{argmax}}(\bar{\Phi}(\dot{C}_{l,v}^m)) \quad (5.17)$$

Cette configuration fournit la meilleure échelle, le meilleur modèle 3D et le meilleur point de vue grossier ainsi que la projection des points 3D dans l'image (et donc le point de vue précis). La boîte englobante finale du véhicule est la boîte englobante des points 3D projetés.

5.5 Expérimentations

Cette section liste les différentes expérimentations que nous avons réalisées pour évaluer notre algorithme. Ces expérimentations se focalisent sur trois métriques d'évaluation : la détection de véhicules, l'estimation du point de vue grossier et la mise en correspondance 2D/3D permettant le calcul du point de vue précis. Pour comparer précisément notre méthode par rapport à d'autres approches de la littérature, nous avons pris soin de séparer les approches apprises sur des données réelles et celles apprises sur des données synthétiques. Les mesures de performances ont été réalisées sur la base de données *3D Object Dataset* [Savarese 2007] qui fournit 480 images de véhicules, leurs annotations de boîte englobante et de point de vue grossier (8 points de vue). De plus, les auteurs

[Zia 2013a] fournissent les vérités terrain des parties visibles des véhicules. Pour entraîner les modèles d'apparence et de géométrie, nous avons utilisé le même processus que dans le chapitre précédent pour la génération de données (voir 5.3.1). Ainsi, pour chaque détecteur de parties, nous avons utilisé 500K patches de parties positifs et 500K patches de négatifs.

Méthode	AP réelle	AP synth	MPPE réelle	MPPE synth	PARTS	CVPI	CVP
[Zia 2011]	-	90.4	-	84.0	74.2	73.3	-
[Zia 2013a]	-	57.1	97.1	66.9	81.5	95.7	61.9
[Stark 2010]	-	89.8	-	81.0	-	67.4	-
[Glasner 2011]	99.2	-	85.3	-	-	-	-
DPM-VOC+VP [Pepik 2012]	99.8	98.6	97.5	92.9	-	70.8	-
DPM-3D-Const [Pepik 2012]	-	94.3	-	84.9	-	-	-
La nôtre	-	95.6	-	91.0	82.8	-	79.1

TABLE 5.1 – Les résultats sur la *3D Object Dataset*. Ce tableau montre les résultats pour la détection d'objets (seconde et troisième colonnes), la détection du point de vue grossier (quatrième et cinquième colonnes). Pour ces deux tâches, nous considérons deux types de données pour l'apprentissage : réelles et synthétiques. La performance de la localisation des parties (sixième colonne) est le pourcentage de parties visibles qui sont bien localisées dans la base de données complète. Les deux dernières colonnes du tableau sont les résultats pour l'estimation du point de vue précis. CVPI liste les différentes méthodes d'estimation du point de vue précis qui utilisent une initialisation (calcul de la boîte 2D et du point de vue grossier au préalable). CVP donne les résultats pour les approches qui n'utilisent pas cette initialisation.

Détection des véhicules. Le processus standard d'évaluation de la détection d'objets est la précision moyenne (*average precision AP* présentée dans 3.6.1). Nous avons utilisé le critère de recouvrement utilisé dans PASCAL VOC : une détection est considérée correcte si elle recouvre une vérité terrain à plus de 0.5. La seconde colonne du tableau 5.1 montre les résultats de notre approche pour

cette tâche.

Estimation du point de vue grossier. Nous avons également évalué l'estimation du point de vue grossier en utilisant la métrique de précision moyenne pour l'estimation de pose (*mean precision in pose estimation* MPPE). La MPPE est calculée comme suit : si le véhicule est bien détecté, nous regardons si le point de vue grossier est le même que celui de la vérité terrain. C'est une métrique qui mesure la performance en termes de classification multi-classes du point de vue (voir 3.6.2). La quatrième colonne du tableau Table 5.1 montre les résultats de notre approche pour cette tâche. La figure 5.11 est la matrice de confusion associée à ces résultats.

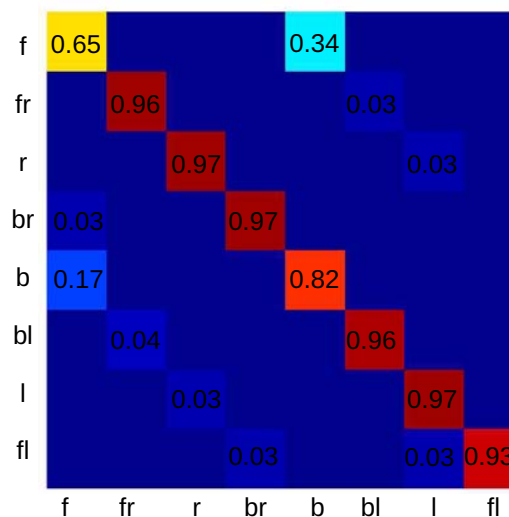


FIGURE 5.11 – La matrice de confusion pour l'estimation du point de vue grossier (la moyenne sur la diagonale correspond au MPPE = 90.4). La sphère de point de vue est divisée en $N = 8$ points de vue grossiers (face, face-droit, droit, arrière-droit, arrière, face-gauche, arrière-gauche, gauche).

Localisation des parties dans l'image. Nous avons également évalué la capacité de l'algorithme à correctement mettre en correspondance les parties pour pouvoir estimer le point de vue précis. Nous avons utilisé la métrique de [Zia 2011, Zia 2013a], qui mesure la qualité de la localisation des parties en considérant qu'une partie visible est bien localisée si sa distance à la vérité terrain est inférieure à un certain seuil. Comme dans [Zia 2011], ce seuil est de 20 pixels pour une boîte de taille 155 pixels de hauteur. Cette métrique est calculée quand

une détection correspond à une boîte de vérité terrain, même si le point de vue grossier estimé n'est pas le bon. La sixième colonne du tableau 5.1 donne les résultats pour la tâche de localisation des parties.

Point de vue précis. Enfin, nous donnons les résultats pour la tâche d'estimation du point de vue continu (voir métrique CVP 3.6.2). Un point de vue est considéré correct si la différence de son angle avec l'angle de la vérité terrain est inférieure à 10 degrés. Les angles de vérités terrain sont fournis par [Zia 2011] et ont été calculés en mettant en correspondance manuellement des modèles 3D sur 48 images de la base 3D Object. Les deux dernières colonnes du tableau 5.1 montrent les résultats de l'estimation de point de vue précis. Nous avons divisé les résultats en considérant les méthodes qui utilisent une initialisation de la boîte englobante et du point de vue grossier (CVPI) et les autres (CVP).

Analyse des résultats. Les expérimentations montrent que l'approche DPM-VOC+VP [Pepik 2012] est plus performante que les autres approches pour la détection de véhicule et l'estimation du point de vue grossier, en particulier quand cette méthode est apprise sur une base de données d'images réelles. En regardant de plus près les méthodes qui ont entraîné leurs modèles sur des bases de données synthétiques, nous pouvons voir que nous sommes très proches de l'approche DPM-VOC+VP pour ces deux tâches (3% de moins en détection et 1.9% de moins pour l'estimation de point de vue grossier). Ces performances légèrement moins bonnes sont liées à l'utilisation exclusive de caractéristiques locales sans utiliser d'informations visuelles globales (contrairement au DPM). Cependant, notre approche a été réfléchiée pour résoudre le problème des véhicules occultés, ce qui a orienté notre choix vers l'utilisation des caractéristiques visuelles locales. Pour illustrer la capacité de notre algorithme à gérer les occultations, nous avons généré synthétiquement des masques d'occultations sur les images. La figure 5.12 montre quelques images de résultats sur des images bruitées par ces masques. Nous pouvons constater que même si les parties du véhicule sont occultées, notre algorithme est capable de calculer la mise en correspondance 2D/3D et ainsi détecter le véhicule et estimer son point de vue précis.

De plus, une analyse par point de vue grossier, met en avant deux points de vue grossiers difficiles à classifier : la vue avant et la vue arrière. La matrice de confusion sur les huit points de vue grossiers, présentée dans la figure 5.11, montre clairement l'ambiguïté entre ces deux points de vues inverses. Ils sont en effet très

proches en termes d'apparence et de géométrie ce qui explique pourquoi notre algorithme se trompe. Dans ce contexte particulier, un détecteur d'apparence global comme DPM-VOC+VP [Pepik 2012] semble plus discriminant. L'avantage majeur de notre approche est qu'elle fournit la localisation précise des parties de véhicule dans l'image, ce qui n'est pas le cas dans beaucoup d'approches comparatives. Ainsi, nous pouvons voir dans la troisième ligne du tableau 5.1 que notre méthode augmente les performances de localisation des parties comparativement aux approches utilisant des modèles 3D déformables [Zia 2013a]. De la même manière, notre algorithme est comparable aux autres approches de l'état de l'art pour l'estimation du point de vue précis comme montré dans les deux dernières colonnes du tableau 5.1. Comparativement à des approches qui n'utilisent pas d'étape d'initialisation (CVP), notre approche augmente les performances pour l'estimation du point de vue précis. Si nous considérons toutes les approches pour cette tâche (CVP+CVPI), la méthode [Zia 2013a] basée sur une initialisation par DPM-VOC+VP [Pepik 2012] est la seule qui fournit une meilleure performance que la nôtre. Il faut retenir que notre modèle multi-vues d'estimation du point de vue va plus loin que la détection et l'estimation du point de vue grossier des véhicules. Il fournit également une description fine de son angle de vue ainsi que de sa forme. La figure 5.13 montre plusieurs exemples de résultats produits par notre modèle.

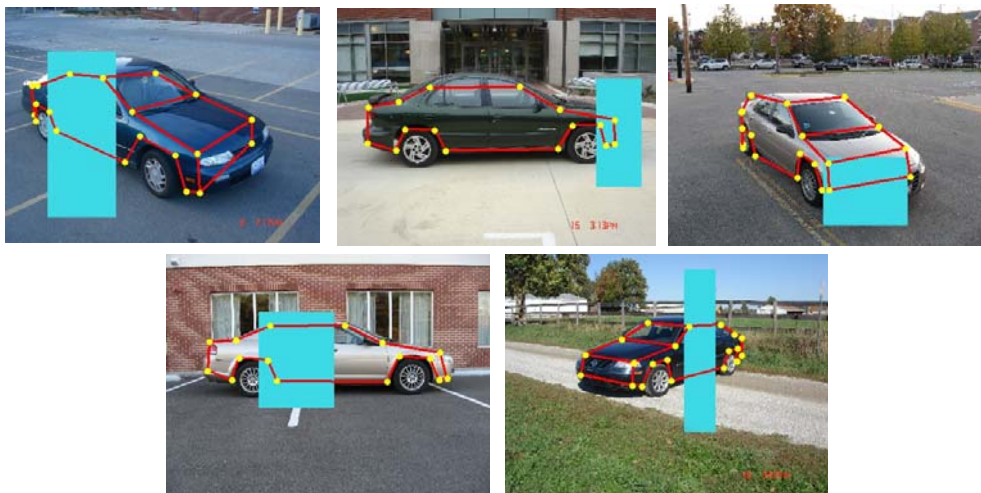


FIGURE 5.12 – Des images de résultats en ajoutant des masques d'occultation. Les points jaunes correspondent aux points 3D projetés après la phase de raffinement.

5.6 Conclusion

Dans ce chapitre, nous avons présenté une méthode permettant de détecter les véhicules ainsi que d'estimer leurs points de vue grossiers et précis. Additionnellement, elle fournit la localisation de parties du véhicule apportant ainsi une description plus fine de celui-ci. Elle se base sur l'utilisation de caractéristiques d'apparences et géométriques entraînées avec des données synthétiques générées grâce à un processus de rendu 3D de type contours. Par rapport à l'état de l'art, notre approche est compétitive sur la base *3D Object Dataset* [Savarese 2007] pour les tâches de détection, d'estimation du point de vue précis et la localisation de parties de véhicules.

Cependant, cette approche a deux limites majeures. La première concerne la robustesse de cette méthode qui est directement dépendante de la qualité des contours trouvés dans les images. En d'autres termes, cette méthode fonctionne très bien pour des véhicules très bien résolus. La seconde limitation est en lien avec un problème bien connu du calcul de pose par mise en correspondance 2D/3D : les points coplanaires. En effet, si tous les points 2D détectés dans l'image correspondent à des points 3D de l'objet qui sont sur le même plan 3D, le calcul de pose ne sera pas précis. Fort de ces observations, le chapitre suivant propose une évolution de cette approche en palliant les limitations énoncées ci-dessus.

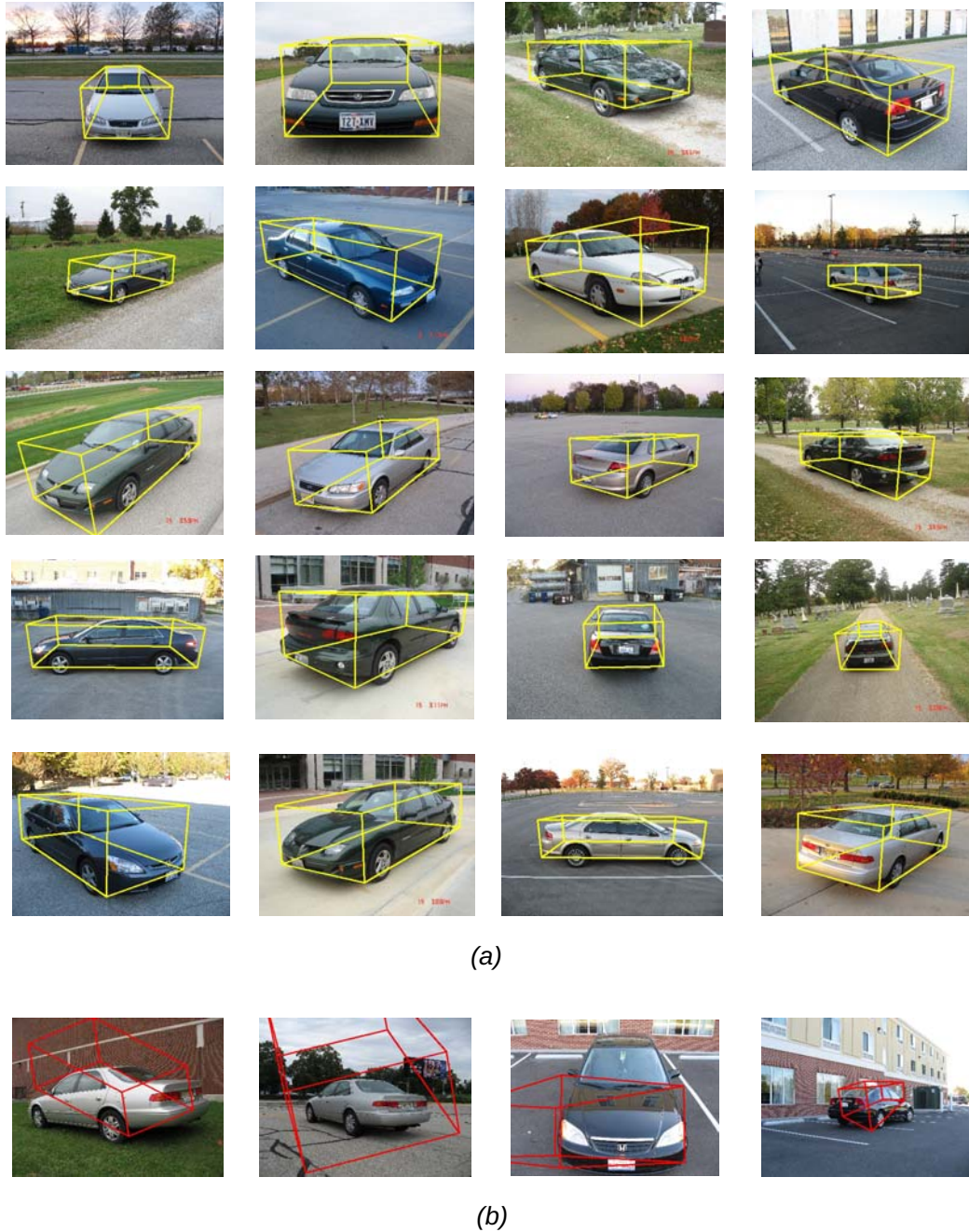


FIGURE 5.13 – Des images de résultats sur *3D Object Dataset*. La boîte représentée correspond à la projection de la boîte 3D du modèle 3D choisi. (a) des exemples qui fonctionnent (b) des exemples qui ne fonctionnent pas.

Approche MANy-TAsk (MANTA) : Analyse complète 2D/3D des véhicules dans la scène

Sommaire

6.1	Introduction	101
6.2	Réseau de Proposition d'objets	105
6.3	L'approche Deep MANTA	106
6.3.1	Bases de données de formes 3D et de gabarits 3D	107
6.3.2	Modèle 2D/3D de véhicule	107
6.3.3	Deep MANTA : le réseau	109
6.3.4	Deep MANTA : inférence	110
6.4	Apprentissage du Deep MANTA	112
6.4.1	Fonctions de perte Many-task	113
6.4.2	Annotation semi-automatique	115
6.5	Expérimentations	117
6.5.1	Détection de véhicules 2D et estimation de leurs orientations	118
6.5.2	Localisation 3D	121
6.5.3	Localisation des parties, leur visibilité et le gabarit 3D	123
6.5.4	ManyTasks et paramètres de régularisation	124
6.6	Extensions de Deep MANTA	124
6.7	Conclusion	126

6.1 Introduction

Dans ce chapitre, nous présentons une méthode complète d'analyse fine 2D et 3D de véhicules à partir d'une image provenant d'une caméra monoculaire.

Nous nous intéressons ici au véhicule autonome, bien que cette méthode soit transposable à de multiples applications. Dans ce contexte, l'analyse d'images provenant d'une caméra monoculaire est nécessaire du fait que bon nombre de véhicules sont aujourd'hui équipés avec ce type de capteur. Les objets présents dans ces images sont souvent difficiles à détecter et à analyser de par les grandes variations d'échelle et les occultations. Néanmoins, pour répondre aux besoins d'un véhicule sans conducteur, il est nécessaire d'analyser ces images finement et de manière robuste pour prédire les situations dangereuses. Outre la détection de véhicules dans l'image, réussir à extraire certaines de leurs caractéristiques de manière plus fine peut s'avérer d'une grande utilité. Un premier exemple de caractéristiques importantes concernent la localisation 3D et l'estimation de l'orientation des véhicules détectés. Ces informations, utilisées conjointement avec des informations temporelles, sont nécessaires pour estimer la vitesse et la direction de ces véhicules. Un autre exemple concerne la localisation de parties de ces véhicules. Réussir à localiser correctement les phares arrières et donc les clignotants d'une voiture permet de donner des informations sur les intentions de son conducteur. Enfin, prédire des propriétés de visibilité pour chaque partie permet de savoir si le véhicule est caché par un obstacle de l'environnement. Tout en s'inspirant de certains concepts présentés dans les chapitres précédents, nous proposons une méthode permettant d'analyser les véhicules dans une scène complexe. Nous proposons ici une approche permettant la résolution de différents problèmes au sein d'un même algorithme : la détection de véhicules, l'estimation précise de leurs orientations, la localisation de parties, l'estimation de propriétés de visibilité, l'estimation de gabarit et la localisation 3D de véhicules. Ce sont ces nombreuses tâches prédites par notre approche qui lui confèrent le nom de Deep MANTA (MANy-TAsk). La Figure 6.1 illustre les sorties de notre système.

La première originalité de cette approche est d'encoder l'information 3D des véhicules par des parties caractéristiques. L'idée sous-jacente est que les informations 3D des véhicules peuvent être retrouvées car ce sont des objets rigides avec une géométrie bien définie. Contrairement au chapitre 5 qui se focalisait sur la détection des parties visibles de véhicules dans l'image, cette nouvelle approche permet de localiser toutes les parties du véhicule : même si les parties ne sont pas visibles du fait de l'auto-occultation, de l'occultation ou de la troncature, elles sont tout de même localisées dans l'image. Toujours en opposition avec la méthode du chapitre 5 qui permettait de détecter les parties visibles par classification de

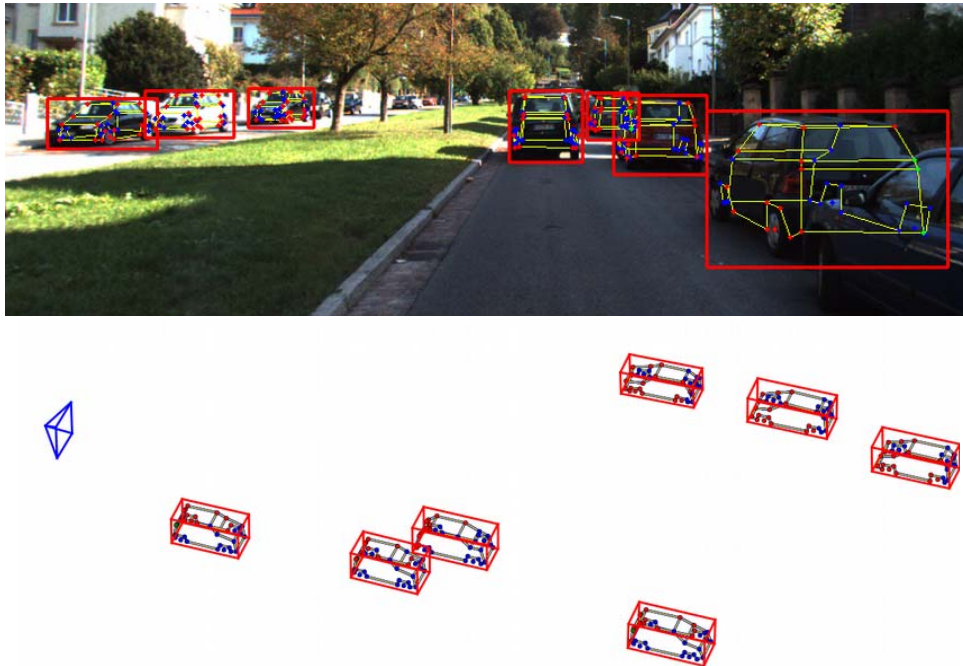


FIGURE 6.1 – Les sorties de l’approche Deep MANTA. *En haut* : les boîtes 2D des véhicules détectés, la localisation des parties et leur visibilité. Les points rouges sont les parties visibles, les points verts sont les parties occultées et les points bleus les parties auto-occultées. *En bas* : les boîtes 3D des véhicules dans la scène et la localisation 3D de leurs parties. La caméra est représentée en bleu.

patches, l’approche Deep MANTA retrouve les parties par régression sur leurs coordonnées. La localisation de toutes les parties des véhicules permet de réaliser une mise en correspondance 2D/3D très robuste. En effet, nous avons pu voir dans le chapitre 5 que si les parties détectées sont coplanaires ou s’il y a trop de mauvaises détections, cela empêche une mise en correspondance 2D/3D fiable pour l’estimation de l’orientation. Tout comme dans le chapitre précédent, nous utilisons une base de modèles 3D de véhicules où certains sommets 3D sont annotés. Ces sommets correspondent à des parties de véhicules (le centre des roues, les phares,...) et définissent la forme 3D pour chaque modèle 3D. L’idée principale est de retrouver les projections de ces points 3D (formes 2D) pour chaque véhicule détecté dans l’image d’entrée. Ensuite, le meilleur modèle 3D est sélectionné pour chaque détection. Une mise en correspondance 2D/3D est appliquée entre les formes 2D détectées et les formes 3D sélectionnées afin de retrouver l’orientation et la localisation 3D de chaque véhicule. Dans ce chapitre,

nous utilisons des modèles 3D à taille réelle pour permettre une localisation 3D précise lors de la mise en correspondance 2D/3D.

La seconde originalité est l'introduction d'un réseau de neurones profond à raffinement et *many-task* appelé le réseau Deep MANTA. Ce réseau renvoie un ensemble de détections, les formes 2D, la visibilité des parties et le gabarit 3D. L'architecture de ce réseau présente deux contributions. La première est associée à la partie "détection" du réseau. Inspiré du Réseau de proposition d'objets (RPN) introduit dans [Ren 2015], le réseau MANTA est capable de proposer un ensemble de boîtes candidates grossières qui sont par la suite itérativement raffinées par des passages multiples dans le réseau ce qui fournit des détections précises. La seconde contribution concerne le concept de *many-task*. Cela signifie qu'un seul vecteur de caractéristiques peut être utilisé pour prédire de multiples sorties. En particulier, nous optimisons conjointement six tâches : la proposition d'objets, la détection, la régression des boîtes, la localisation des parties, la visibilité des parties et la prédiction du gabarit 3D.

Enfin, la dernière originalité de ce travail est la génération semi-automatique et intelligente d'annotations fines à partir d'une base de données faiblement annotée. Ces annotations seront utilisées pour l'apprentissage de notre méthode. Les réseaux de neurones requièrent beaucoup de données (des exemples et les annotations associées) pour être correctement appris. De plus, il est presque impossible d'annoter manuellement des parties de véhicules qui ne sont pas observables dans les images. Pour résoudre ce problème, nous proposons une manière d'étiqueter semi-automatiquement des images réelles en utilisant des modèles 3D. Les informations 3D contenues dans les modèles 3D (géométrie, visibilité, ...) sont automatiquement projetées dans les images réelles ce qui fournit une importante base de données sans effort d'annotation. En opposition aux deux chapitres précédents, le réseau Deep MANTA est entraîné sur des images réelles mais annotées en utilisant des modèles 3D. Ce processus réunit ainsi le meilleur des deux espaces : des images 2D réelles pour apprendre des caractéristiques visuelles discriminantes et généralisables lors du test, ainsi que des modèles 3D pour fournir les annotations nécessaires.

La section suivante rappelle le principe du Réseau de Proposition d'objets (RPN). Dans la section 6.3 l'approche Deep MANTA est détaillée. L'apprentis-

sage du réseau est ensuite expliqué dans la section 6.4. Enfin, dans la section 6.5, nous montrons que notre méthode améliore les performances de détection, d'estimation de l'orientation et de la localisation 3D des véhicules en l'évaluant sur la base KITTI [Geiger 2012] dédiée au véhicule autonome.

6.2 Réseau de Proposition d'objets

Du fait de la popularité récente des réseaux de neurones convolutifs et des méthodes de propositions d'objets pour la détection d'instances, des travaux se sont focalisés sur l'apprentissage de réseaux de proposition d'objets [Ren 2015, Yang 2016, Xiang 2017, Kong 2016]. Ce type de réseau (*Region Proposal Network*), initialement introduit par [Ren 2015], permet de générer des propositions d'objets directement dans le réseau neuronal et d'extraire ces propositions sur des cartes de caractéristiques profondes. Le RPN utilise des ancres (*anchors*) : une ancre est une boîte 2D prédéfinie caractérisée par une échelle et un rapport largeur/hauteur. Le principe de fonctionnement du RPN est le suivant. Une image complète est passée à travers différentes couches de Convolution/Activation/Pooling jusqu'à une certaine profondeur. Cela fournit une carte de caractéristiques basse résolution. Cette carte de caractéristiques est envoyée dans le RPN qui consiste en un petit réseau convolutif. La sortie de ce réseau est une carte de scores de même résolution que la carte de caractéristiques en entrée. En chaque position de la carte (u, v) , un vecteur de A scores est donné où A correspond au nombre d'ancres. En d'autres termes, en chaque pixel de la carte de scores renvoyée par le RPN, un score de confiance est attribué à chaque ancre prédéfinie qui représente la probabilité que l'ancre contienne effectivement un objet. Typiquement, le Faster-RCNN utilise trois échelles et trois rapports largeur/hauteur ce qui donne $A = 9$ ancres. Les propositions de régions (ancres) ayant renvoyé les plus hauts scores sont extraites sur la carte de caractéristiques basse résolution et envoyées à un classifieur comme dans le Fast-RCNN [Girshick 2015]. La Figure 6.2 illustre le fonctionnement du RPN. Cette approche est peu coûteuse en temps de calcul et présente de très bonnes performances en détection d'objets.

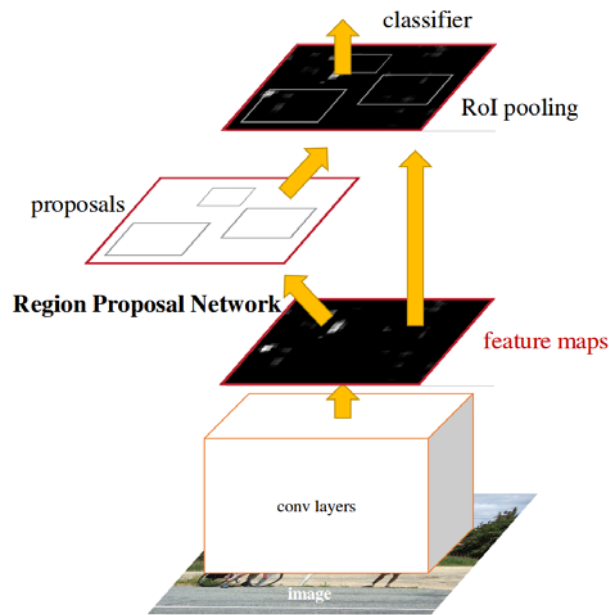


FIGURE 6.2 – Illustration du réseau de proposition d’objets (*Region Proposal Network* (RPN)). Source du schéma [Ren 2015]

6.3 L’approche Deep MANTA

Nous détaillons ici l’approche Deep MANTA permettant l’analyse simultanée 2D et 3D de véhicules à partir d’une image. Cette méthode se divise en deux étapes. Durant la première étape, l’image d’entrée est passée dans le réseau de neurones Deep MANTA renvoyant un ensemble de détections (boîtes 2D) et leurs scores de confiance associés. Pour chacune de ces détections, le réseau fournit également des informations sur la géométrie des véhicules (coordonnées des parties, gabarits 3D) et les propriétés de visibilité de chaque partie. Dans la seconde étape, dite d’inférence, les informations 3D de chaque véhicule détecté (orientation fine et localisation 3D) sont retrouvées. Cette étape utilise les sorties du Deep MANTA ainsi qu’une base de formes 3D et une base de gabarits 3D. Ces deux bases encodent la variabilité des véhicules en termes de dimensions, de types et de formes. Elles sont présentées dans la section 6.3.1. Dans la section 6.3.2, nous présentons le modèle 2D/3D choisi pour caractériser un véhicule dans une image. Le fonctionnement du réseau Deep MANTA est détaillé dans la section 6.3.3 et l’inférence dans la section 6.3.4.

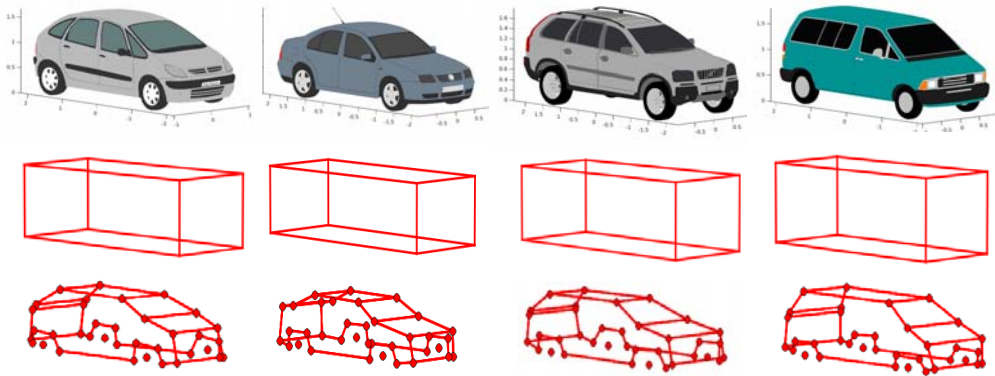


FIGURE 6.3 – Quatre exemples provenant de la base de gabarits 3D et de celle de formes 3D. À chaque modèle 3D m (première ligne) est associé un gabarit 3D \bar{t}_m^{3d} (seconde ligne) et une forme 3D \bar{S}_m^{3d} (troisième ligne).

6.3.1 Bases de données de formes 3D et de gabarits 3D

Tout comme dans le chapitre 5, nous utilisons une base de modèles 3D constituée de M modèles de différents types de véhicules (Sedan, SUV,...). Pour chaque modèle 3D m , N sommets sont annotés (parties 3D). Ces points 3D correspondent à des parties d'intérêt de véhicules. Étant donné un modèle 3D m , nous définissons sa forme 3D (alignée dans une vue canonique) par $\bar{S}_m^{3d} = (\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N)$ avec $\mathbf{Q}_k = [X_k, Y_k, Z_k]^\top$ correspondant aux coordonnées 3D de la partie k . Contrairement au chapitre 5 les modèles 3D utilisés sont à taille réelle : cette condition est nécessaire pour estimer la localisation 3D des véhicules dans la scène par mise en correspondance 2D/3D. Nous définissons également le gabarit 3D associé au modèle 3D m par $\bar{t}_m^{3d} = (w_m, h_m, l_m)$ où w_m, h_m, l_m sont respectivement la largeur, la hauteur et la longueur du modèle 3D. La Figure 6.3 montre quelques exemples de la base de formes 3D $\{\bar{S}_m^{3d}\}_{m \in \{1, \dots, M\}}$ et de celle de gabarits 3D $\{\bar{t}_m^{3d}\}_{m \in \{1, \dots, M\}}$.

6.3.2 Modèle 2D/3D de véhicule

Nous représentons chaque véhicule d'une image par un modèle 2D/3D. Ce modèle est formellement défini par les caractéristiques suivantes :

$$(B, B^{3d}, S, S^{3d}, V) \quad (6.1)$$

$B = (c_x, c_y, w, h)$ est la boîte 2D du véhicule avec (c_x, c_y) son centre et w, h sa largeur et sa hauteur respectivement. $B^{3d} = (c_x, c_y, c_z, \theta, t)$ est sa boîte englobante 3D

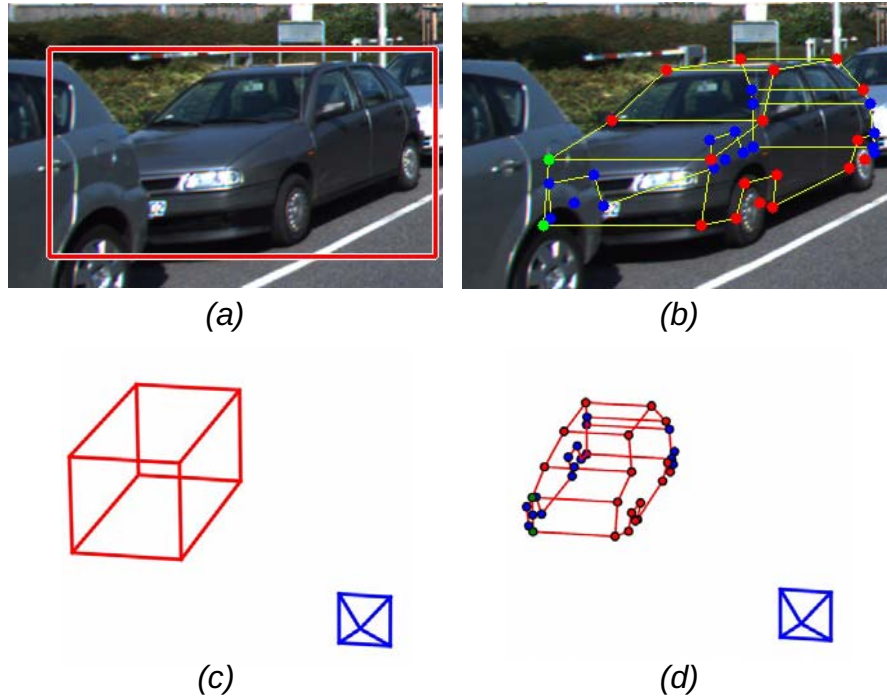


FIGURE 6.4 – Un exemple d’un modèle 2D/3D de véhicule. (a) la boîte englobante 2D B , (b) les coordonnées 2D des parties S et leurs visibilitées V : parties visibles (rouge), parties occultées (vert) et parties auto-occultées (bleu). (c) la boîte englobante 3D B^{3d} et (d) la forme 3D associée S^{3d} . La caméra est représentée en bleu.

caractérisée par les coordonnées 3D de son centre dans le repère caméra (c_x, c_y, c_z) , son orientation θ et son gabarit 3D $t = (w, h, l)$ (ses dimensions 3D réelles). $S = \{\mathbf{q}_k = [u_k, v_k]^T\}_{k \in \{1, \dots, N\}}$ est la forme 2D du véhicule, c’est-à-dire l’ensemble des coordonnées de ses parties dans l’image. $S^{3d} = \{\mathbf{Q}_k = [X_k, Y_k, Z_k]^T\}_{k \in \{1, \dots, N\}}$ est l’ensemble des coordonnées 3D des parties du véhicule dans le repère de la caméra. $V = \{v_k\}_{k \in \{1, \dots, N\}}$ est le vecteur de visibilité des parties où v_k est la classe de visibilité de la partie k . Quatre classes de visibilité sont définies :

- visible si la partie est observée dans l’image,
- occultée si la partie est occultée par un autre objet,
- auto-occultée si la partie est occultée par le véhicule,
- tronquée si la partie est en dehors de l’image.

La Figure 6.4 est un exemple d’un modèle 2D/3D de véhicule.

6.3.3 Deep MANTA : le réseau

Le réseau Deep MANTA a pour objectif de détecter les véhicules en utilisant un raffinement itératif de propositions d'objets. Il permet également de prédire d'autres propriétés comme la localisation de parties, leurs visibilitées et une mesure de similarité pour l'estimation du gabarit.

Architecture du réseau Deep MANTA et raffinement itératif des détections. L'image complète est donnée en entrée du réseau. Une succession de couches de Convolution/Activation/Pooling lui est appliquée. Cela produit une carte de caractéristiques à basse résolution. Cette résolution dépend du nombre de couches qui ont été appliquées à l'image. Classiquement, la résolution de la carte de caractéristiques est 16 fois plus petite que l'image originale. Cette carte de caractéristiques est passée dans un réseau de proposition d'objets. Cela renvoie un premier ensemble de K propositions d'objets $B_1 = \{B_{i,1}\}_{i \in \{1, \dots, K\}}$. Dans le Faster-RCNN classique [Ren 2015] ces propositions d'objets sont ensuite extraites sur la carte de caractéristiques de l'entrée du RPN et redimensionnées à une taille fixe en utilisant une couche de ROI Pooling introduite par [Girshick 2015]. Ces régions sont ensuite envoyées à un classifieur pour prendre la décision finale sur leur appartenance à une classe. Cette manière de faire présente plusieurs limites liées à la résolution très basse de cette carte de caractéristiques. Dans une telle carte, si les objets sont trop petits dans l'image initiale, une grosse partie de l'information est perdue : le classifieur aura des difficultés à leur associer une classe. De la même manière, si deux objets s'occulent et sont trop petits, l'information sur leur appartenance à une classe est fusionnée dans la carte de caractéristiques basse résolution. Pour pallier ce problème, nous proposons d'extraire les régions proposées par le RPN sur une carte de caractéristiques à haute résolution : cette carte correspond aux sorties des premières couches du réseau convolutif. Les régions extraites sur la carte à haute résolution et redimensionnées par ROI Pooling sont ensuite passées dans un réseau partageant des poids avec le réseau du premier niveau (RPN). Les boîtes englobantes de ces régions sont raffinées en appliquant une transformation (régression sur les boîtes). Un second ensemble de K objets $B_2 = \{B_{i,2}\}_{i \in \{1, \dots, K\}}$ est alors proposé. Cette opération est répétée une dernière fois pour fournir l'ensemble final de détection B_3 . Ces trois niveaux de raffinement sont illustrés dans la Figure 6.5. Cette manière de raffiner permet de garder une grande précision sur l'estimation des détections. Cela sera démontré

quantitativement dans les expérimentations de la section 6.5.

Prédictions Many-tasks. L'architecture du Deep MANTA renvoie un ensemble de boites englobantes $B_3 = \{B_{i,3}\}_{i \in \{1, \dots, K\}}$. Pour chacune de ces boites $B_{i,3}$, le réseau MANTA fournit également les coordonnées 2D des parties S_i par régression, leurs visibilitées V_i et un vecteur de similarité de gabarit 3D T_i . Ce vecteur de similarité est défini par $T_i = \{r_m\}_{m \in \{1, \dots, M\}}$ avec $r_m = (r_x, r_y, r_z)$ correspondant aux trois facteurs d'échelle à appliquer sur le gabarit 3D \bar{t}_m^{3d} (défini dans 6.3.1) pour correspondre au gabarit 3D réel du véhicule détecté i . Ce vecteur peut être vu comme la similarité entre le véhicule détecté et tous les gabarits 3D de la base de gabarits 3D $\{\bar{t}_m^{3d}\}_{m \in \{1, \dots, M\}}$. Toutes ces sorties sont calculées en utilisant le même vecteur de caractéristiques. Ce vecteur, correspondant à une région extraite, est passé dans une couche complètement connectée pour chaque tâche. En d'autres termes, nous montrons ici qu'une seule caractérisation d'une région d'intérêt peut être utilisée pour prédire le résultat de plusieurs tâches en même temps.

À ce stade de la méthode, une suppression des non-maxima (NMS) est appliquée pour supprimer les détections redondantes. Cela fournit un nouvel ensemble de K' détections et leurs propriétés associées $\{B_j, S_j, V_j, T_j\}_{j \in \{1, \dots, K'\}}$.

6.3.4 Deep MANTA : inférence

L'objectif de cette étape est de retrouver l'orientation et la localisation 3D de chaque véhicule détecté. Elle utilise les sorties du Deep MANTA, la base de formes 3D $\{\bar{S}_m^{3d}\}_{m \in \{1, \dots, M\}}$ et la base de gabarits 3D $\{\bar{t}_m^{3d}\}_{m \in \{1, \dots, M\}}$ définies dans 6.3.1 pour retrouver les informations 3D. Étant donné une détection j fournie par le réseau MANTA, l'inférence se divise en deux étapes. Dans la première étape, nous choisissons le gabarit 3D $c \in \{1, \dots, M\}$ dans la base de gabarits 3D $\{\bar{t}_m^{3d}\}_{m \in \{1, \dots, M\}}$ en utilisant le vecteur de similarité $T_j = \{r_m\}_{m \in \{1, \dots, M\}}$. Pour chaque exemple \bar{t}_m^{3d} de la base de gabarits 3D nous appliquons le facteur d'échelle r_m . L'ensemble des gabarits 3D résultants sont notés $\{t_m^{3d}\}_{m \in \{1, \dots, M\}}$. Le meilleur gabarit c est celui qui minimise la distance entre t_m^{3d} et \bar{t}_m^{3d} :

$$c = \operatorname{argmin}_{m \in \{1, \dots, M\}} d(\bar{t}_m^{3d}, t_m^{3d}). \quad (6.2)$$

Dans la deuxième étape, une mise en correspondance 2D/3D est appliquée en

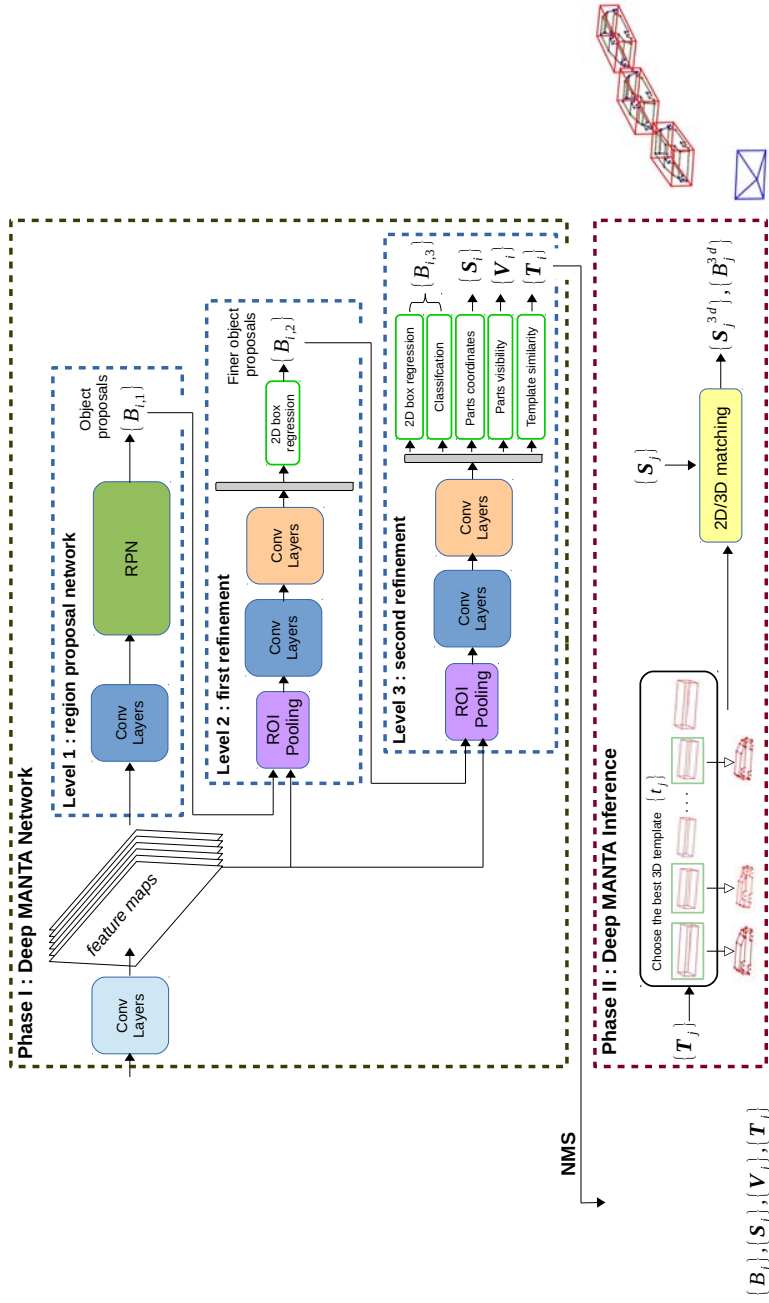


FIGURE 6.5 – Vue d'ensemble de l'approche Deep MANTA. L'image complète est passée dans le réseau Deep MANTA. Les couches de convolutions (*Conv layers*) de même couleur partagent les mêmes poids. Le réseau fournit un ensemble de propositions d'objets $\{B_{i,1}\}$ qui sont itérativement raffinées ($\{B_{i,2}\}$ puis les détections finales $\{B_{i,3}\}$). Les coordonnées des parties 2D $\{S_i\}$, la visibilité des parties $\{V_i\}$ et la mesure de similarité de gabarits 3D $\{T_i\}$ sont également renvoyées par le réseau. Une suppression des non-maxima (NMS) est alors appliquée. Cela supprime les détections redondantes et fournit le nouvel ensemble $\{B_j, S_j, V_j, T_j\}$. En utilisant ces sorties, l'étape d'inférence permet de choisir le meilleur gabarit 3D dans la base de gabarits 3D $\{\tilde{T}_m^{3d}\}_{m \in \{1, \dots, M\}}$ en utilisant le vecteur de similarité T_j . Puis, un calcul de pose 2D/3D est exécuté en utilisant la forme 3D associée.

utilisant la forme 3D \bar{S}_c^{3d} . Cette forme 3D est redimensionnée pour correspondre au gabarit $t_j = t_c^{3d}$. Ensuite, une estimation de pose est exécutée pour mettre en correspondance la forme 3D redimensionnée \bar{S}_c^{3d} avec la forme 2D S_j grâce à un algorithme standard de calcul de matrice de projection 2D/3D [Lepetit 2009]. Nous considérons ici que les paramètres intrinsèques de la caméra sont connus (contrairement au chapitre 5). Ces paramètres sont indispensables pour que le calcul de la localisation 3D soit précis. Cette dernière étape fournit la boîte englobante 3D B_j^{3d} et les coordonnées 3D des parties S_j^{3d} dans la scène. La dernière ligne de la Figure 6.5 illustre cette phase d'inférence.

6.4 Apprentissage du Deep MANTA

Dans cette section nous détaillons les différentes tâches à optimiser pendant l'apprentissage du Deep MANTA. Dans ce qui suit, nous considérons trois niveaux de raffinement $l \in \{1, 2, 3\}$ et cinq fonctions de perte : \mathcal{L}_{rpn} , \mathcal{L}_{det} , \mathcal{L}_{parts} , \mathcal{L}_{vis} et \mathcal{L}_{temp} . \mathcal{L}_{rpn} est la fonction de perte associée à la proposition d'objets (RPN) définie dans [Ren 2015]. \mathcal{L}_{det} est la fonction de perte associée à la détection permettant de discriminer les véhicules et le fond ainsi que régresser les boîtes englobantes. \mathcal{L}_{parts} est la perte correspondant à la localisation des parties. \mathcal{L}_{vis} est la perte en lien avec la visibilité des parties. \mathcal{L}_{temp} est la perte associée à la similarité des gabarits 3D. Elles seront détaillées dans la section suivante. Pour apprendre le réseau Deep MANTA du début à la fin, nous avons utilisé le framework du Faster-RCNN [Ren 2015] qui est basé sur les RPN. Étant donné une image d'entrée, l'optimisation du réseau se fait par la minimisation de la fonction de coût globale suivante :

$$\mathcal{L} = \mathcal{L}^1 + \mathcal{L}^2 + \mathcal{L}^3 \quad (6.3)$$

avec

$$\mathcal{L}^1 = \sum_k \mathcal{L}_{rpn}(k), \quad (6.4)$$

$$\mathcal{L}^2 = \sum_i \mathcal{L}_{det}^2(i) + \mathcal{L}_{parts}^2(i), \quad (6.5)$$

$$\mathcal{L}^3 = \sum_i \mathcal{L}_{det}^3(i) + \mathcal{L}_{parts}^3(i) + \mathcal{L}_{vis}(i) + \mathcal{L}_{temp}(i), \quad (6.6)$$

où k est l'indice d'une ancre dans un mini-batch et i est l'indice d'une proposition d'objet. Ces trois fonctions de perte correspondent aux trois niveaux de raffinement

du réseau Deep MANTA : plus le niveau de raffinement est élevé, plus le réseau apprend de tâches.

6.4.1 Fonctions de perte Many-task

Ici nous détaillons chaque fonction de perte utilisée pour l'optimisation de la fonction globale du réseau définie précédemment. Dans la suite, chaque proposition d'objets à chaque niveau de raffinement l est indexée par i et est représentée par sa boîte $B_{i,l} = (c_{x_{i,l}}, c_{y_{i,l}}, w_{i,l}, h_{i,l})$. La boîte de la vérité terrain B la plus proche de $B_{i,l}$ est sélectionnée. La vérité terrain des parties S , la vérité terrain de la visibilité des parties V et celle du gabarit 3D t sont aussi sélectionnées (voir 6.3.2 pour les notations). Nous utiliserons P pour nous référer à la fonction de perte softmax et R pour nous référer à la fonction de perte SmoothL1 définie dans [Girshick 2015].

Perte du RPN. Cette fonction de perte fonctionne de la même manière que dans [Ren 2015]. Pour entraîner le RPN, chaque ancre k (caractérisée par sa boîte $B_k = (c_{x_k}, c_{y_k}, w_k, h_k)$) se voit attribuer une classe C_k . C_k vaut 1 si l'ancre recouvre suffisamment une boîte de vérité terrain, 0 sinon. Le vecteur de probabilités de classes retourné par le RPN pour l'ancre k se note C_k^* . Un vecteur cible de régression de boîte $\Delta_k = (\delta_x, \delta_y, \delta_w, \delta_h)$ est également défini comme suit :

$$\delta_x = (c_{x_k} - c_x)/w \qquad \delta_w = \log(w_k/w) \qquad (6.7)$$

$$\delta_y = (c_{y_k} - c_y)/h \qquad \delta_h = \log(h_k/h) \qquad (6.8)$$

Le vecteur de régression renvoyé par le RPN se note Δ_k^* . Ainsi, la fonction de détection est définie par :

$$\mathcal{L}_{rpn} = \lambda_{cls_{rpn}} \sum_k P(C_k^*, C_k) + \lambda_{reg_{rpn}} \sum_k C_k R(\Delta_k^* - \Delta_k) \qquad (6.9)$$

avec $\lambda_{cls_{rpn}}$ et $\lambda_{reg_{rpn}}$ les paramètres de régularisation, $\lambda_{cls_{rpn}}$ pour la classification des ancres et $\lambda_{reg_{rpn}}$ pour la régression. Lorsque l'apprentissage est suffisamment avancé, une liste des ancres avec une forte probabilité d'être un objet est renvoyée. Chacune de ces ancres se voit appliquer la transformation apprise par régression. Cela renvoie des propositions d'objets (boîtes 2D) qui sont extraites sur une carte de caractéristiques. Ces régions extraites sont utilisées pour minimiser les fonctions de perte expliquées ci-dessous.

Perte de détection. À chacune des propositions d'objet i au niveau de raffinement l est attribuée une classe $C_{i,l}$. $C_{i,l}$ vaut 1 si la proposition d'objets contient bien un véhicule et 0 sinon. Le critère de classification permettant l'association de la classe est le recouvrement (*overlap*) entre la boîte $B_{i,l}$ et celle de la vérité terrain B . Le vecteur de probabilités de classes retourné par le Deep MANTA pour la proposition d'objet se note $C_{i,l}^*$. Un vecteur cible de régression de boîte $\Delta_{i,l} = (\delta_x, \delta_y, \delta_w, \delta_h)$ est également défini comme suit :

$$\delta_x = (c_{x_{i,l}} - c_x)/w \quad \delta_w = \log(w_{i,l}/w) \quad (6.10)$$

$$\delta_y = (c_{y_{i,l}} - c_y)/h \quad \delta_h = \log(h_{i,l}/h) \quad (6.11)$$

Le vecteur de régression renvoyé par le Deep MANTA se note $\Delta_{i,l}^*$. Ainsi, la fonction de détection est définie par :

$$\mathcal{L}_{det}^l(i) = \lambda_{cls}P(C_{i,l}^*, C_{i,l}) + \lambda_{reg}C_{i,l}R(\Delta_{i,l}^* - \Delta_{i,l}) \quad (6.12)$$

avec λ_{cls} et λ_{reg} les paramètres de régularisation, λ_{cls} pour la classification de boîte et λ_{reg} pour la régression.

Perte des parties. En utilisant les parties de la vérité terrain $S = (\mathbf{q}_1, \dots, \mathbf{q}_N)$ et la boîte $B_{i,l}$ de la proposition d'objet i au niveau de raffinement l , les coordonnées des parties du véhicule normalisées $S_{i,l} = (\bar{\mathbf{q}}_1, \dots, \bar{\mathbf{q}}_N)$ sont calculées comme suit :

$$\bar{\mathbf{q}}_k = \left(\frac{u_k - c_{x_{i,l}}}{w_{i,l}}, \frac{v_k - c_{y_{i,l}}}{h_{i,l}} \right). \quad (6.13)$$

Les parties normalisées prédites sont notées $S_{i,l}^*$. Ainsi, la perte de la localisation des parties est donnée par :

$$\mathcal{L}_{parts}^l(i) = \lambda_{parts}C_{i,l}R(S_{i,l}^* - S_{i,l}) \quad (6.14)$$

avec λ_{parts} le paramètre de régularisation de cette perte.

Perte de visibilité. Cette fonction est seulement optimisée pour le dernier niveau de raffinement $l = 3$. Le vecteur de visibilité de la vérité terrain $V_i = V$ est associé à la proposition d'objet i . Le vecteur de probabilités de classes de visibilité retourné par le Deep MANTA est défini par V_i^* . La fonction de perte de la visibilité est donnée par :

$$\mathcal{L}_{vis}(i) = \lambda_{vis}C_{i,3}P(V_i^*, V_i) \quad (6.15)$$

avec λ_{vis} le paramètre de régularisation de la fonction de perte de la visibilité.

Perte de similarité de gabarit. Cette fonction est seulement optimisée pour le dernier niveau de raffinement $l = 3$. Au lieu d’optimiser directement les dimensions du gabarit 3D t , nous l’encodons comme un vecteur de similarité T en utilisant la base de gabarits 3D comme expliqué dans 6.3.3. Pour l’apprentissage, la fonction logarithme est appliquée à chaque élément de T pour une meilleure normalisation (les valeurs de similarité sont ainsi dans $[-1, 1]$). Ce vecteur de similarité est assigné à $T_i = T$ à la proposition d’objet i . Le vecteur de similarité de gabarit prédit est noté T_i^* . Ainsi, la fonction de perte pour la similarité de gabarit est définie ainsi :

$$\mathcal{L}_{temp}(i) = \lambda_{temp} C_{i,3} R(T_i^* - T_i) \quad (6.16)$$

avec λ_{temp} le paramètre de régularisation pour cette perte.

Il faut noter que si la proposition d’objet i n’est pas strictement positive ($C_{i,l} = 0$) les fonctions de perte associées à la régression des boîtes, à la localisation des parties, à la visibilité et à la similarité de gabarit sont nulles car optimiser des propriétés liées au véhicule sur des régions correspondant à du fond n’a pas de sens.

6.4.2 Annotation semi-automatique

Une annotation semi-automatique est utilisée pour fournir les annotations indispensables pour l’apprentissage du réseau Deep MANTA (les coordonnées des parties des véhicules, leur visibilité, les vecteurs de similarité de gabarit). Pour réaliser cette annotation semi-automatique, nous avons besoin :

- d’une base de données d’images réelles faiblement annotée fournissant les boîtes englobantes 3D pour chaque véhicule
- d’une base de données de modèles 3D à taille réelle. Nous avons utilisé une base de données composée de M modèles 3D de véhicules et pour chacun de ces véhicules, nous avons annoté N sommets 3D.

Le processus d’annotation semi-automatique se divise en plusieurs étapes et est illustré dans la Figure 6.6. Pour chaque véhicule dans l’image réelle et connaissant sa boîte 3D :

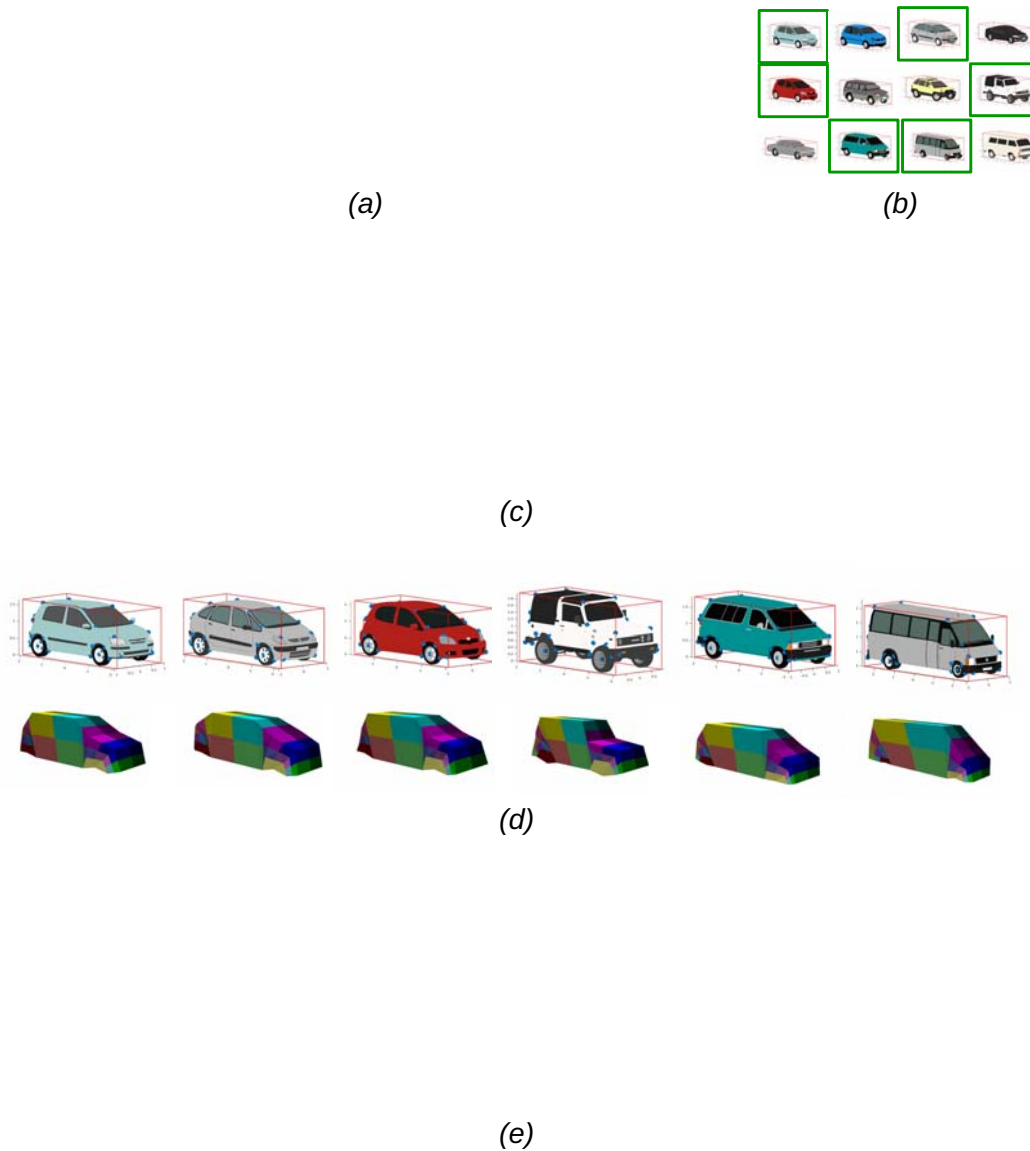


FIGURE 6.6 – Le processus d’annotation semi-automatique. (a) annotations grossières sur une image (boîtes englobantes 3D). (b) les modèles 3D correspondant le mieux aux véhicules dans l’image (en vert). (c) les projections de ces modèles 3D dans l’image. (d) les modèles de visibilité correspondants (chaque couleur représente une partie). (e) les annotations finales (localisation des parties et leurs visibilités). Points rouges : les parties visibles, points verts : les parties occultées, points bleus : les parties auto-occultées.

- Le modèle 3D de la base qui lui correspond le mieux en termes de gabarit 3D est choisi. Ce modèle 3D est celui dont les dimensions 3D sont les plus proches des dimensions 3D du véhicule dans l'image.
- Les parties 3D du modèle 3D choisi sont projetées dans l'image pour obtenir leurs coordonnées 2D.
- La visibilité de chaque partie est calculée en utilisant un modèle 3D de visibilité associé au modèle 3D choisi. Il consiste en un modèle 3D basse résolution où chacune des faces est associée à une classe de parties (voir Figure 6.6 (d)). Ce modèle est projeté dans l'image pour calculer les parties visibles, auto-occultées et tronquées. Une partie k est considérée visible si les faces qui lui correspondent sont suffisamment visibles dans l'image. Dans le cas contraire, cette partie est considérée auto-occultée ou tronquée (si elle est en dehors de l'image). Pour calculer les parties occultées, nous projetons les modèles 3D des autres véhicules de la scène ce qui produit des masques dans l'image. Nous analysons ensuite les parties qui étaient visibles avant la création de ces masques. Si une partie est masquée, elle est alors considérée comme occultée.

6.5 Expérimentations

Dans cette section nous évaluons notre approche Deep MANTA sur le *benchmark* KITTI [Geiger 2012] dédié à la détection d'objets et l'estimation de l'orientation dans le contexte de la conduite autonome. Ce *benchmark* propose 7 481 images pour l'apprentissage et 7 518 images pour le test. Pour chacune de ces images les matrices de calibration sont fournies. Pour chaque véhicule présent dans les images d'entraînement, KITTI propose plusieurs annotations : la boîte 2D, la position 3D dans la scène, l'orientation, les dimensions 3D, le niveau d'occultation et le niveau de troncature. Toutes ces annotations permettent d'annoter automatiquement les images d'apprentissage comme décrit dans 6.4.2 en utilisant une base de modèles 3D de véhicules. Nous avons utilisé une base de modèles 3D fournie par [Fidler 2012] et composée de $M = 103$ modèles 3D. Pour chacun de ces modèles nous avons manuellement annoté $N = 36$ points 3D.

Comme les vérités terrain ne sont pas données pour les images de test nous

avons utilisé des ensembles apprentissage/validation pour valider notre approche. Un ensemble apprentissage/validation consiste à séparer la base d'apprentissage fournie par KITTI en deux parties : l'une sera utilisée pour entraîner la méthode et l'autre pour la valider. Pour pouvoir nous comparer aux méthodes de l'état de l'art, nous avons utilisé deux ensembles d'apprentissage/validation : *val1* utilisé par [Xiang 2017, Xiang 2015] et *val2* utilisé par [Chen 2015, Chen 2016]. De cette manière nous avons pu évaluer la méthode pour des tâches qui ne sont pas initialement évaluées dans le *benchmark* KITTI.

Nous avons entraîné Deep MANTA sur deux architectures neuronales standards : le GoogLeNet [Szegedy 2015] et le VGG16 [Simonyan 2014]. L'optimisation de ces architectures est faite en utilisant une descente de gradient stochastique. Le Deep MANTA est initialisé en utilisant des poids pré-entraînés sur ImageNet. Nous avons utilisé 7 rapports hauteur/largeur et 10 échelles (comme proposé par [Xiang 2017]) pour le réseau de proposition d'objets ce qui donne un total de 70 ancres pour chaque position de la carte de caractéristiques. Pendant l'apprentissage, une proposition d'objet est considérée positive si son recouvrement avec une boîte de vérité terrain est supérieur ou égal à 0.7. Pour les expérimentations, les paramètres de régularisation λ sont mis à 1 excepté pour la fonction de perte de la localisation des parties où $\lambda_{parts} = 3$. Le choix de ces paramètres est discuté à la fin de cette section.

Nous présentons des résultats pour différentes tâches : la détection de véhicules 2D, l'estimation de l'orientation, la localisation 3D, la localisation des parties dans l'image, la prédiction de la visibilité des parties et l'estimation du gabarit 3D. Dans les résultats présentés ci-dessous, nous avons utilisé 200 propositions d'objets et un critère de recouvrement de 0.5 pour la suppression des non-maxima. Les résultats sont présentés pour les trois niveaux de difficulté (Simple, Modéré, Difficile) comme proposé par le benchmark KITTI [Geiger 2012]. Ces trois niveaux de difficulté prennent en compte la taille des véhicules, leur niveau d'occultation et de troncature.

6.5.1 Détection de véhicules 2D et estimation de leurs orientations

Nous avons utilisé la précision moyenne (*Average Precision AP*) avec un critère de recouvrement de 0.7 pour évaluer la détection 2D de véhicules (voir

3.6.1). Nous avons utilisé la similarité moyenne en orientation (*average orientation similarity* AOS) pour évaluer l'estimation de l'orientation des véhicules comme proposé par KITTI [Geiger 2012] (voir 3.6.2).

Le tableau 6.1 présente les résultats pour la détection et le tableau 6.2 ceux de l'estimation de l'orientation sur les deux ensembles d'apprentissage/validation utilisés. Le tableau 6.3 présente les résultats sur l'ensemble de test de KITTI et la Figure 6.7 montre les courbes rappel/précision et rappel/similarité angulaire sur ce même ensemble. Nous pouvons observer que notre méthode surpasse les approches de la littérature pour ces deux métriques (sur les deux ensembles de validation mais aussi sur l'ensemble de test). De plus, notre approche est plus rapide (en utilisant les mêmes ressources calculatoires que les autres méthodes, à savoir un GPU TITAN X). Cela est dû à la résolution de l'image d'entrée. Beaucoup d'approches de l'état de l'art basées sur la proposition d'objets [Xiang 2017, Chen 2016, Chen 2015] redimensionnent l'image d'entrée en appliquant un facteur d'échelle de 3 ou 3.5. Ces approches utilisent ce redimensionnement pour ne pas perdre d'informations dans la carte de caractéristiques réduite du fait de la profondeur des réseaux de neurones. Dans notre approche, grâce au raffinement, nous contourignons ce problème ce qui nous permet de mettre en entrée une image à sa taille initiale.

Methode	Type	Temps	AP		
			Simple	Modéré	Difficile
3DVP [Xiang 2015]	Mono	40 s	80.48 / -	68.05 / -	57.20 / -
Faster-RCNN [Ren 2015]	Mono	2 s	82.91 / -	77.83 / -	66.25 / -
SubCNN [Xiang 2017]	Mono	2 s	95.77 / -	86.64 / -	74.07 / -
3DOP [Chen 2015]	Stereo	3 s	- / 94.49	- / 89.65	- / 80.97
Mono3D [Chen 2016]	Mono	4.2 s	- / 95.75	- / 90.01	- / 80.66
Ours GoogLeNet	Mono	0.7 s	97.90 / 97.58	91.01 / 90.89	83.14 / 82.72
Ours VGG16	Mono	2 s	97.45 / 97.2	91.47 / 91.85	81.79 / 85.15

TABLE 6.1 – Résultats pour la détection 2D de véhicules (AP) sur deux ensembles de validation : *val1* / *val2*.

L'intérêt du raffinement itératif est montré dans le tableau 6.4 : nous comparons différentes variantes du Deep MANTA. La première ligne de ce tableau correspond au Deep MANTA appris sans la dernière étape de raffinement et où l'extraction

Chapitre 6. Approche MANy-Task (MANTA) : Analyse complète 2D/3D des véhicules dans la scène
120

Methode	Type	Temps	AOS		
			Simple	Modéré	Difficile
3DVP [Xiang 2015]	Mono	40 s	78.99 / -	65.73 / -	54.67 / -
SubCNN [Xiang 2017]	Mono	2 s	94.55 / -	85.03 / -	72.21 / -
3DOP [Chen 2015]	Stereo	3 s	- / 92.98	- / 87.34	- / 78.24
Mono3D [Chen 2016]	Mono	4.2 s	- / 93.70	- / 87.61	- / 78.00
Ours GoogLeNet	Mono	0.7 s	97.60 / 97.44	90.66 / 90.66	82.66 / 82.35
Ours VGG16	Mono	2 s	97.10 / 97.09	91.01 / 91.57	81.14 / 84.72

TABLE 6.2 – Résultats pour l’estimation de l’orientation des véhicules (AOS) sur deux ensembles de validation : *val1* / *val2*.

	AP			AOS		
	Simple	Modéré	Difficile	Simple	Modéré	Difficile
ACF-SC [Cadena 2015]	69.11	58.66	45.95	-	-	-
MDPM-un-BB [Felzenszwalb 2010]	71.19	62.16	48.43	-	-	-
DPM-VOC+VP [Pepik 2015a]	74.95	64.71	48.76	72.28	61.84	46.54
OC-DPM [Pepik 2013]	75.94	65.95	53.56	73.50	64.42	52.40
SubCat [Ohn-Bar 2015]	84.14	75.46	59.71	83.41	74.42	58.83
3DVP [Xiang 2015]	87.46	75.77	65.38	87.46	75.77	65.38
AOG [Li 2014]	84.80	75.94	60.70	33.79	30.77	24.75
Regionlets [Long 2014]	84.75	76.45	59.70	-	-	-
Faster R-CNN [Ren 2015]	86.71	81.84	71.12	-	-	-
3DOP [Chen 2015]	93.04	88.64	79.10	91.44	86.10	76.52
Mono3D [Chen 2016]	92.33	88.66	78.96	91.01	86.62	76.84
SDP + RPN [Yang 2016]	90.14	88.85	78.38	-	-	-
MS-CNN [Cai 2016]	90.03	89.02	76.11	-	-	-
SubCNN [Xiang 2017]	90.81	89.04	79.27	90.67	88.62	78.68
Deep MANTA GoogLeNet	95.77	90.03	80.62	95.72	89.86	80.39
Deep MANTA VGG16	96.40	90.10	80.79	96.32	89.91	80.55

TABLE 6.3 – Résultats pour la détection 2D de véhicules (AP) et l’orientation (AOS) sur l’ensemble de test de KITTI.

des régions d’intérêt se fait sur la carte de caractéristiques correspondant au 5ème niveau de convolution (comme dans le Faster-RCNN original [Ren 2015]). La seconde ligne du tableau correspond au Deep MANTA appris sans la dernière étape de raffinement et où l’extraction des régions d’intérêt se fait sur la carte de caractéristiques correspondant au premier niveau de convolution. Nous pouvons observer qu’extraire les régions sur le premier niveau de convolution augmente clairement

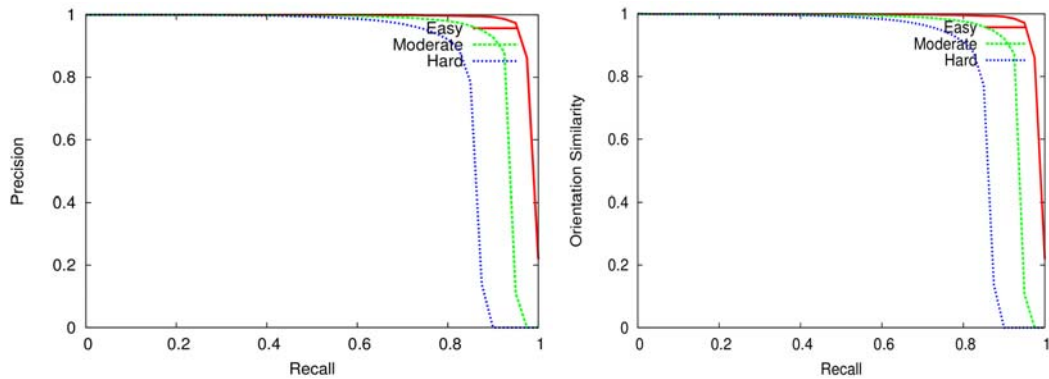


FIGURE 6.7 – Courbes de rappel/précision et rappel/similarité angulaire sur l’ensemble de test de KITTI

les performances de détection et d’estimation de l’orientation (environ 24% de plus pour le niveau de difficulté Modéré). La dernière ligne est le Deep MANTA présenté dans ce chapitre (utilisant le raffinement et une extraction de régions d’intérêt sur le premier niveau de convolution). Ces résultats confirment que notre architecture de raffinement est pertinente (environ 4% de performance en plus pour la détection et l’orientation pour le niveau de difficulté Modéré).

Methode	Raffinement	ROI Pooling sur	AP			AOS		
			Simple	Modéré	Difficile	Simple	Modéré	Difficile
Deep MANTA	Non	conv5	80.64	62.45	53.86	79.68	61.49	52.58
	Non	conv1	95.19	86.85	78.62	94.98	86.52	78.05
	Oui	conv1	97.58	90.89	82.72	97.44	90.66	82.35

TABLE 6.4 – Mise en avant des performances de notre approche de raffinement pour la détection de véhicules (AP) et l’estimation de l’orientation (AOS) sur l’ensemble de validation *val2*. Ces expérimentations montrent l’importance de notre approche par raffinement ainsi que celle du choix de la carte de caractéristiques sur laquelle extraire les régions.

6.5.2 Localisation 3D

Nous avons utilisé la métrique de précision moyenne de localisation (*Average Localization Precision* (ALP)) proposée par [Xiang 2015] pour évaluer la locali-

Methode	Type	Temps	1 mètre		
			Simple	Modéré	Difficile
3DVP [Xiang 2015]	Mono	40 s	45.61 / -	34.28 / -	27.72 / -
3DOP [Chen 2015]	Stereo	3 s	- / 81.97	- / 68.15	- / 59.85
Mono3D [Chen 2016]	Mono	4.2 s	- / 48.31	- / 38.98	- / 34.25
Deep MANTA GoogLenet	Mono	0.7 s	70.90 / 65.71	58.05 / 53.79	49.00 / 47.21
Deep MANTA VGG16	Mono	2 s	66.88 / 69.72	53.17 / 54.44	44.40 / 47.77

TABLE 6.5 – Performances de la localisation 3D (ALP) sur les deux ensembles de validation (*val1* / *val2*) pour une précision d’un mètre.

Methode	Type	Temps	2 mètres		
			Simple	Modéré	Difficile
3DVP [Xiang 2015]	Mono	40 s	65.73 / -	54.60 / -	45.62 / -
3DOP [Chen 2015]	Stereo	3 s	- / 91.46	- / 81.63	- / 72.97
Mono3D [Chen 2016]	Mono	4.2 s	- / 74.77	- / 60.91	- / 54.24
Deep MANTA GoogLenet	Mono	0.7 s	90.12 / 89.29	77.02 / 75.92	66.09 / 67.28
Deep MANTA VGG16	Mono	2 s	88.32 / 91.01	74.31 / 76.38	63.62 / 67.77

TABLE 6.6 – Performances de la localisation 3D (ALP) sur les deux ensembles de validation (*val1* / *val2*) pour une précision de deux mètres.

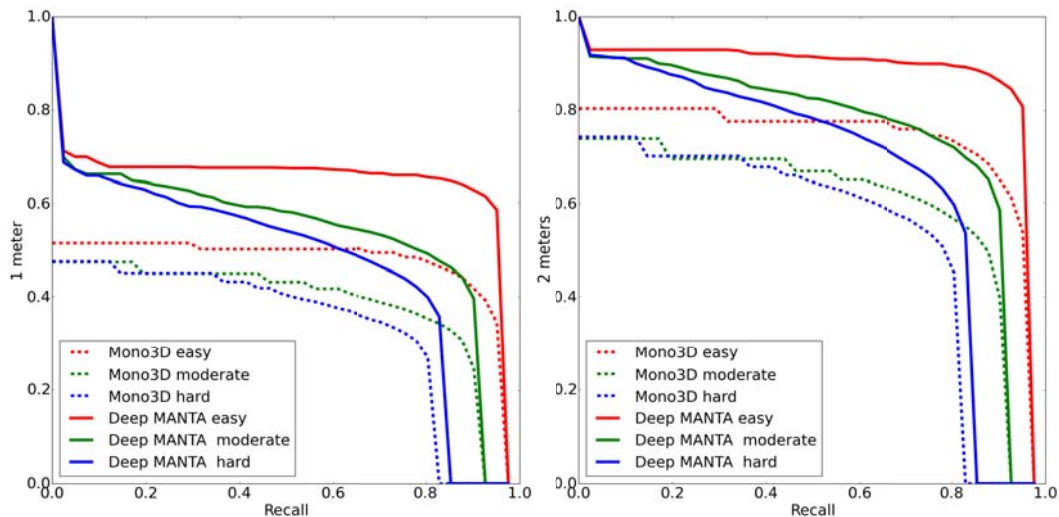


FIGURE 6.8 – Courbes de rappel/précision 3D pour une précision d’un mètre (gauche) et deux mètres (droite) sur la l’ensemble de validation *val2* utilisé par Mono3D [Chen 2016].

sation 3D des véhicules dans la scène (voir 3.6.2). Cette métrique consiste à remplacer la similarité d'orientation (AOS) par la précision de la localisation 3D. Une localisation 3D est correcte si sa distance à la vérité terrain est plus petite qu'un certain seuil. Les Tableaux 6.5 et 6.6 présentent les résultats de la localisation 3D pour une précision d'un mètre et de deux mètres respectivement sur les deux validations. Notre approche surpasse clairement les autres méthodes monoculaires [Chen 2016, Xiang 2015] (environ 16% de plus comparé à Mono3D [Chen 2016]). La Figure 6.8 présente les courbes rappel/précision 3D du Deep MANTA et de Mono3D [Chen 2016]. Comparé à 3DOP [Chen 2015], qui utilise des informations stéréo, les performances du Deep MANTA sont équivalentes pour une précision de 2 mètres mais moins bonnes à 1 mètre : notre approche utilise seulement une image monoculaire contrairement à 3DOP qui utilise des informations de profondeur.

6.5.3 Localisation des parties, leur visibilité et le gabarit 3D

Étant donné une bonne détection nous avons utilisé trois métriques pour évaluer la capacité de l'approche Deep MANTA à localiser correctement les parties en 2D, prédire leurs visibilités et estimer le gabarit 3D des véhicules. Pour la localisation 2D des parties, une partie est considérée bien localisée si sa distance normalisée à la partie de la vérité terrain est inférieure à un certain seuil (20 pixels). Ces distances sont calculées en utilisant une hauteur de boîte englobante fixe (155 pixels) comme proposé par [Zia 2013a]. De plus, une visibilité de partie est considérée correcte si elle a la même classe de visibilité que la partie de la vérité terrain. Enfin, nous avons évalué la prédiction du gabarit 3D en comparant les trois dimensions prédites (w, h, l) au gabarit 3D de la vérité terrain (w_{gt}, h_{gt}, l_{gt}) fourni par KITTI. Un gabarit 3D (w, h, l) est considéré correct si $|\frac{w_{gt}-w}{w_{gt}}| < 0.2$ et $|\frac{h_{gt}-h}{h_{gt}}| < 0.2$ et $|\frac{l_{gt}-l}{l_{gt}}| < 0.2$. Le tableau 6.7 montre les bonnes performances (dépassant les 90% pour les exemples "simples") de notre approche pour ces trois tâches.

Métrique	Simple	Modéré	Difficile
Localisation des parties	97.54	90.79	82.64
Visibilité des parties	92.48	85.08	76.90
Estimation du gabarit	94.04	86.62	78.72

TABLE 6.7 – Évaluation de la localisation des parties, de la visibilité des parties et du gabarit 3D sur l'ensemble de validation *val2*.

6.5.4 ManyTasks et paramètres de régularisation

Le tableau 6.8 montre les résultats du Deep MANTA avec différents paramètres de régularisation pour l'apprentissage du réseau. Ces chiffres ont également pour but de montrer les performances de notre approche en utilisant des réseaux appris avec moins de tâches. Dans le tableau 6.8, D correspond à la tâche de détection, P à la tâche de localisation des parties, V à la visibilité des parties et T à la tâche de similarité de gabarit. Avec ces notations, la première ligne du tableau 6.8 est le Deep MANTA appris seulement pour la tâche de détection ($\lambda_{parts} = \lambda_{vis} = \lambda_{temp} = 0$). Comme la localisation des parties et la similarité de gabarit ne sont pas apprises, l'orientation et la localisation 3D ne peuvent pas être prédites. La seconde ligne est le Deep MANTA appris sans la tâche de visibilité ($\lambda_{vis} = 0$) et avec $\lambda_{parts} = 3$. La troisième ligne est le Deep MANTA complet (toutes les tâches apprises) mais avec un paramètre de régularisation pour la localisation des parties $\lambda_{parts} = 1$. Enfin, la dernière ligne est le Deep MANTA avec $\lambda_{parts} = 3$ (le réseau présenté dans ce chapitre). Ces résultats sont intéressants pour différentes raisons. Premièrement, nous pouvons voir qu'augmenter le nombre de tâches apprises (c'est-à-dire enrichir la description du véhicule) n'affecte pas les performances (un peu meilleures pour la détection et l'orientation, mais un peu moins bonnes sur la localisation 3D). Cela montre l'intérêt du concept ManyTasks : un réseau de neurones est capable d'apprendre une représentation dans l'espace des caractéristiques qui peut être utilisée pour prédire beaucoup de tâches. Deuxièmement, nous pouvons observer que le paramètre λ_{parts} est très important pour la localisation 3D. Apprendre le Deep MANTA avec $\lambda_{parts} = 3$ augmente la localisation 3D de 6% pour une précision d'un mètre : durant l'apprentissage, cette régularisation permet d'équilibrer les différentes tâches à optimiser, pour obtenir des erreurs d'entraînement d'ordres équivalent.

6.6 Extensions de Deep MANTA

Nous proposons ici deux extensions possibles du Deep MANTA en cours de développement.

La première extension de Deep MANTA est de remplacer la tâche d'estimation du gabarit 3D par une tâche de reconnaissance de marques et modèles de véhicule (classification fine). En effet, la prédiction du gabarit 3D permet d'être générique à n'importe quel type de véhicule : dans KITTI, les marques et les modèles des

	AP	AOS	1 m	2 m
D	89.86	-	-	-
DPT / $\lambda_{parts} = 3$	89.73	89.39	58.37	78.11
DPVT / $\lambda_{parts} = 1$	89.58	89.27	51.47	73.93
DPVT / $\lambda_{parts} = 3$	90.54	90.23	57.44	77.58

TABLE 6.8 – L’influence du nombre de tâches apprises ainsi que des différents paramètres de régularisation. Ce tableau donne les résultats pour la détection de véhicules (AP), orientation (AOS), et la localisation 3D (ALP) à un mètre et à deux mètres. Les chiffres donnés sont moyennés sur les deux ensembles de validation et les trois niveaux de difficulté (Simple, Modéré, Difficile). Voir le texte pour les détails.

véhicules annotés ne sont pas fournis. Cependant, si nous disposons d’une base de données d’images où cette information est disponible, et que nous disposons des modèles 3D correspondants, Deep MANTA peut être adaptée pour prédire cette tâche. Nous avons donc construit une base de données en utilisant des voitures miniatures (à l’échelle 1 :43) correspondant à une sous-partie de la base de modèles 3D. Le Deep MANTA a ensuite été appris sur cette base. Un premier résultat qualitatif de cette extension est illustré dans la Figure 6.13.

La seconde extension concerne l’analyse fine multi-classes. Elle permettrait de détecter et d’analyser d’autres objets que les véhicules (les piétons, les cyclistes...). Nous avons donc commencé à investiguer cette problématique en modifiant le Deep MANTA. La première modification concerne la classification : après extraction sur la carte de caractéristiques, l’étape de classification est étendue au nombre de classes que l’on souhaite détecter. La seconde modification concerne la régression sur les parties. L’annotation semi-automatique des parties sur des objets déformables (par exemple les piétons) reste très difficile à faire. Ainsi, pour chaque classe, au lieu de régresser sur des parties, nous avons choisi de régresser sur huit points : les huit points de la boîte 3D projetés dans l’image. Même si ces points n’ont pas de réalité physique, les réseaux de neurones permettent de les localiser par extraction de caractéristiques visuelles globales. La Figure 6.12 illustre les premiers résultats de l’extension du Deep MANTA à la problématique multi-classes sur la base KITTI. Dans cette figure, les sorties correspondent à ce que le réseau

Deep MANTA a prédit (pas d'étape d'inférence).

6.7 Conclusion

Dans ce chapitre nous avons présenté une méthode permettant l'analyse 2D et 3D de véhicules à partir d'une caméra monoculaire. Elle se base sur un réseau de neurones profond basé sur un processus de raffinement itératif de propositions d'objets permettant de détecter les véhicules dans l'image de manière très performante et très rapide. Ce réseau permet également d'apporter une description plus fine des véhicules dans la scène : la localisation de leurs parties (visible ou non), la prédiction de la visibilité de chaque partie et l'estimation du gabarit 3D des véhicules. Pour apprendre ce réseau, une base de données a été créée en utilisant un processus de labellisation semi-automatique intelligent. Durant une phase d'inférence, la localisation 3D et l'orientation des véhicules sont retrouvées en utilisant une base de modèles 3D et les sorties du réseau de neurones.

Notre approche est plus rapide que les méthodes de l'état de l'art tout en étant plus performante : nous augmentons les performances de détection et d'estimation de l'orientation des véhicules de 1% par rapport aux approches de référence. Enfin, nos résultats sur la localisation 3D de véhicule montrent un gain de performance de 16% par rapport à l'approche de référence [Chen 2016].

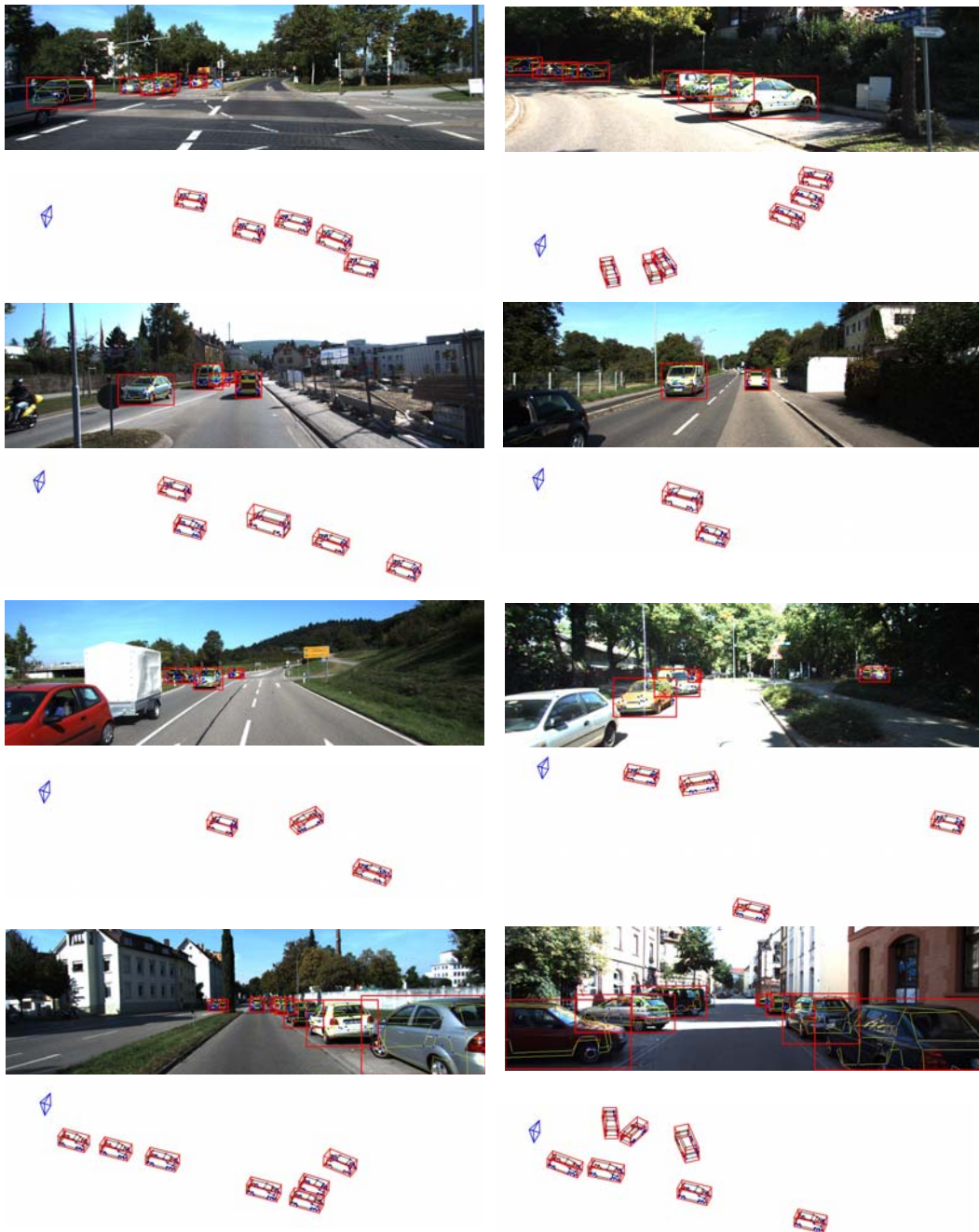


FIGURE 6.9 – Des images de résultats sur l’ensemble de test de KITTI. Pour chaque image, la détection 2D, la localisation des parties et leur visibilité sont représentées. Points rouges : les parties visibles, points verts : les parties occultées, points bleus : les parties auto-occultées. Sous chaque image, nous représentons les parties 3D et les boîtes englobantes 3D. La caméra est représentée en bleu.

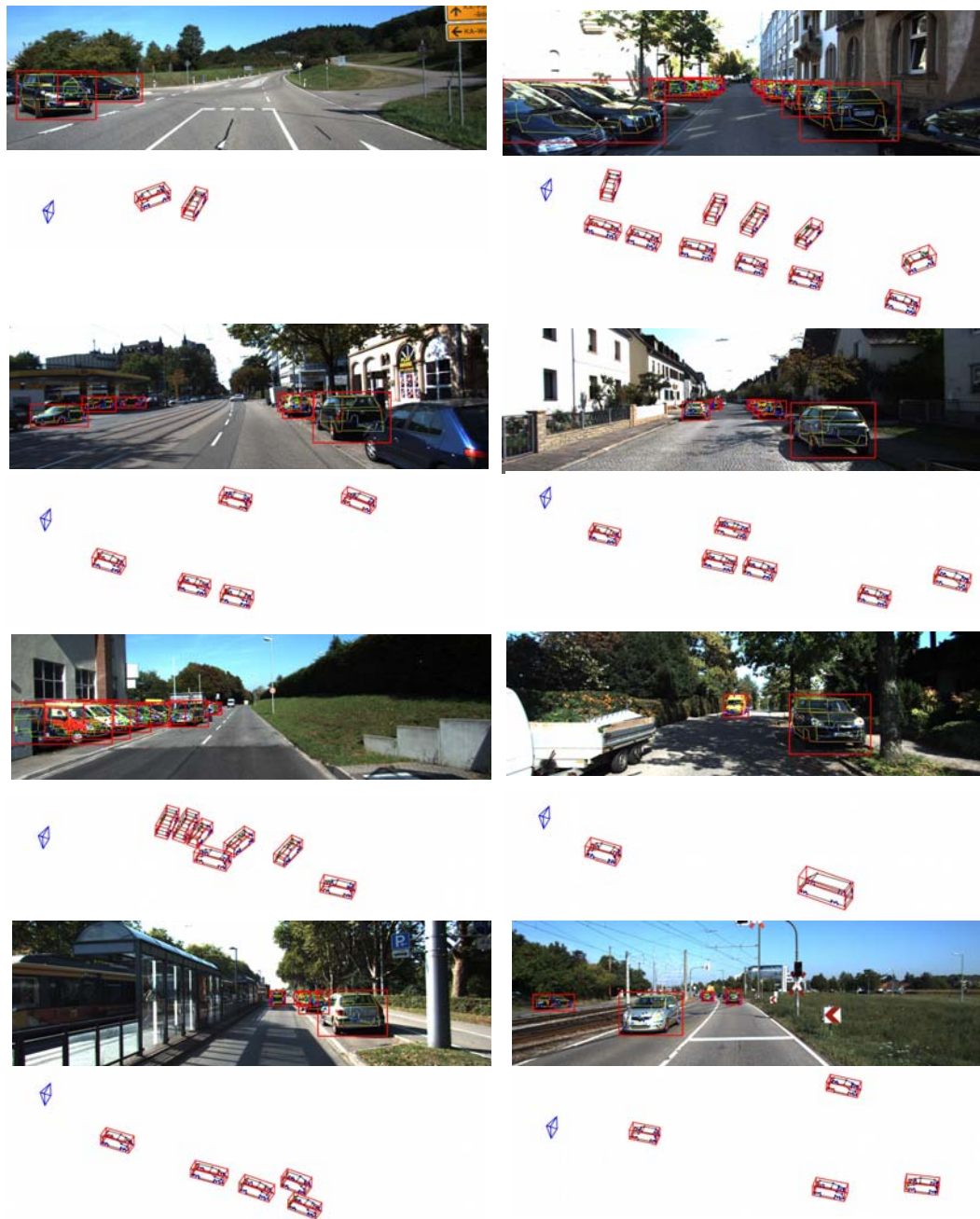


FIGURE 6.10 – Des images de résultats sur l'ensemble de test de KITTI. Pour chaque image, la détection 2D, la localisation des parties et leur visibilité sont représentées. Points rouges : les parties visibles, points verts : les parties occultées, points bleus : les parties auto-occultées. Sous chaque image, nous représentons les parties 3D et les boîtes englobantes 3D. La caméra est représentée en bleu.

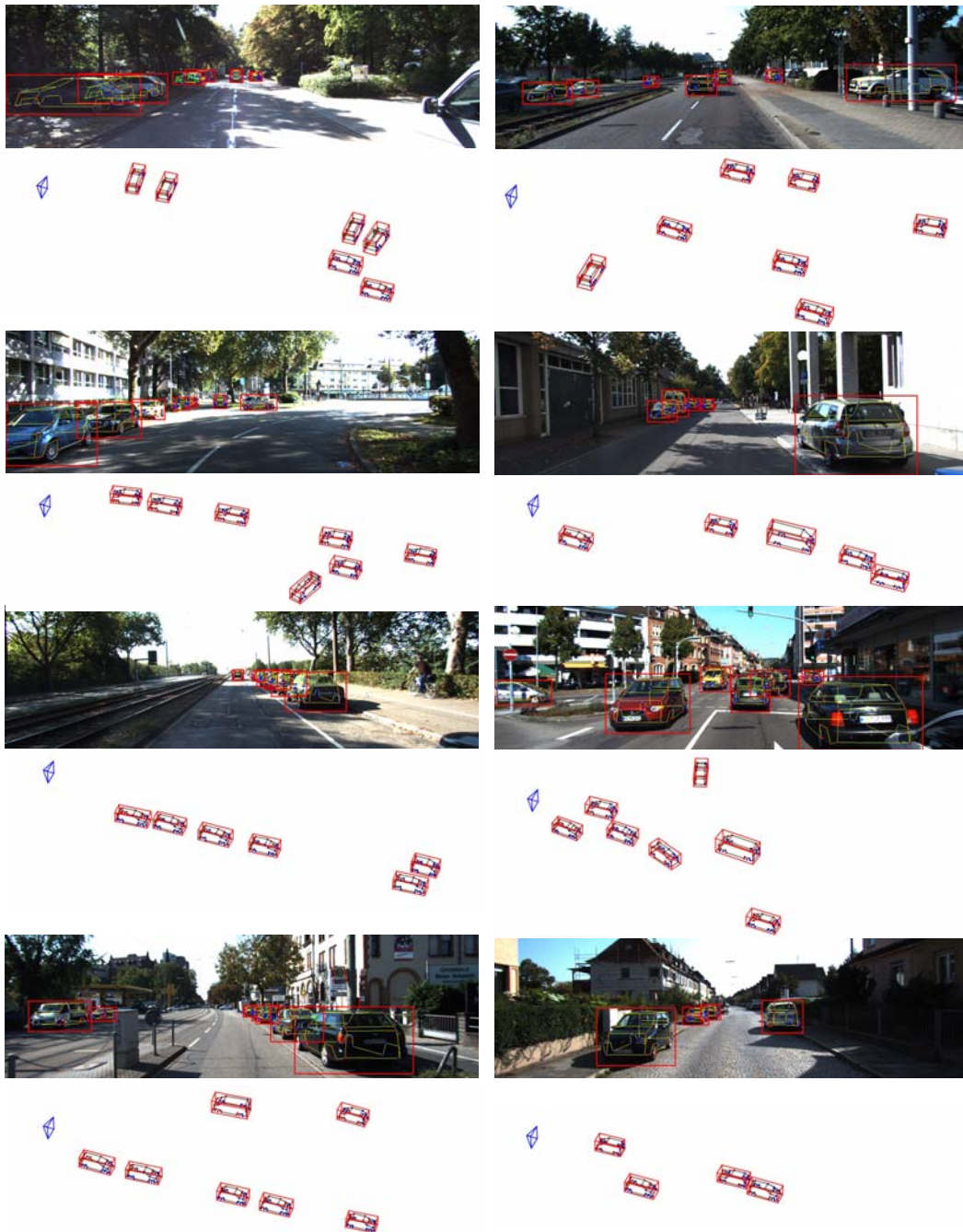


FIGURE 6.11 – Des images de résultats sur l'ensemble de test de KITTI. Pour chaque image, la détection 2D, la localisation des parties et leur visibilité sont représentées. Points rouges : les parties visibles, points verts : les parties occultées, points bleus : les parties auto-ocultées. Sous chaque image, nous représentons les parties 3D et les boîtes englobantes 3D. La caméra est représentée en bleu.



FIGURE 6.12 – Exemples de résultats pour l’analyse 3D d’objets multi-classes (véhicules, piétons, cyclistes) par extension du Deep MANTA. Les boîtes 3D des véhicules, des piétons et des cyclistes sont représentées en rouge, vert, et jaune respectivement.

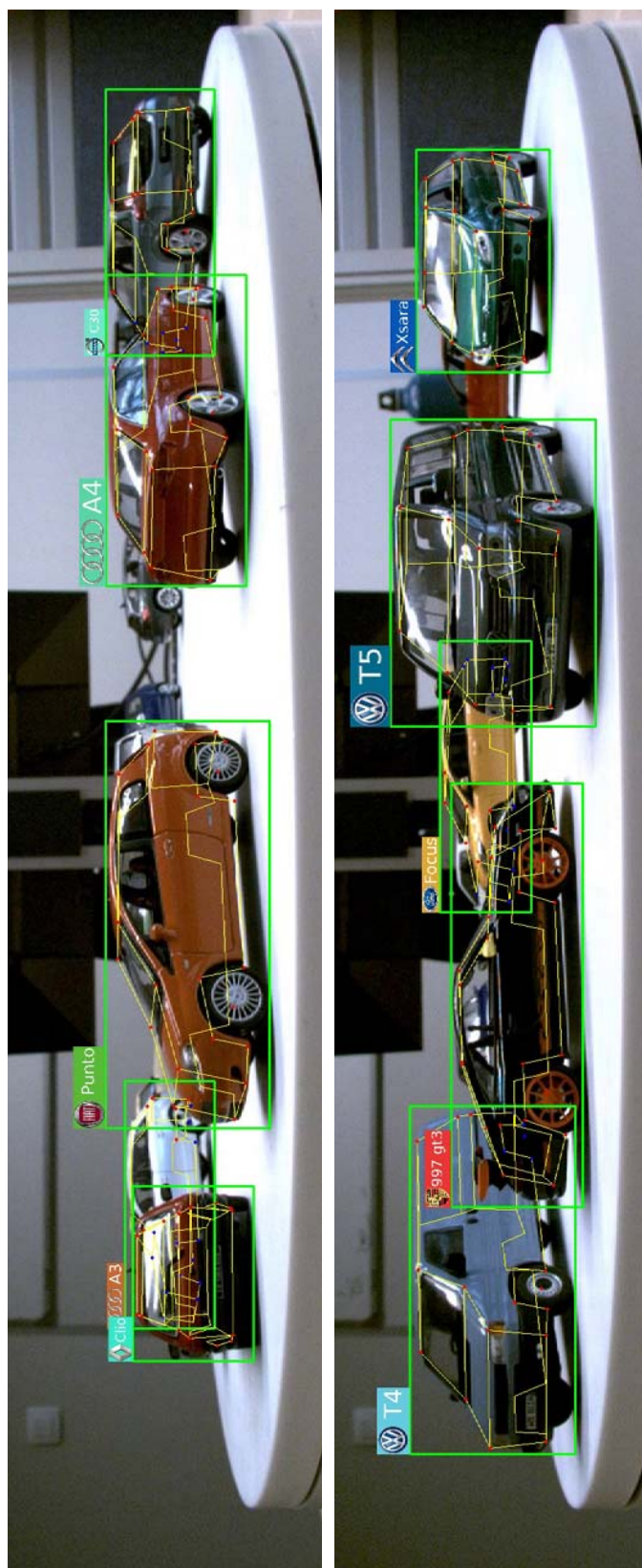


FIGURE 6.13 – Exemple de résultats pour l'analyse 3D de véhicules et la reconnaissance de marques et modèles par extension du Deep MANTA.

Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés à l'analyse fine 2D et 3D de véhicules par réseaux de neurones profonds. Plusieurs aspects de l'analyse fine de véhicules à partir d'une image ont été étudiés : la détection 2D, l'estimation du point de vue, la localisation 3D et la reconnaissance de marques et modèles. Généralement, la littérature existante sur ces sujets propose de traiter ces différents problèmes séparément. En partant de cette observation, notre objectif final était d'aboutir à un système complet d'analyse fine. Pour y parvenir, nous avons proposé de nouvelles méthodes.

Les solutions proposées se basent sur deux partis pris. Le premier est l'utilisation des réseaux de neurones convolutifs profonds comme brique de base d'apprentissage automatique. Ce choix est motivé par leur capacité à résoudre des problèmes de vision très complexes grâce à l'extraction de caractéristiques visuelles discriminantes de haut niveau. Cependant, ces réseaux ont besoin d'un grand nombre de données d'apprentissage annotées pour être performants. De plus, les annotations nécessaires pour l'analyse fine ne sont pas toujours faciles à obtenir. Ainsi, le second parti pris de nos travaux réside dans l'utilisation de modèles 3D de véhicules pour générer automatiquement un grand nombre d'exemples d'apprentissage et leurs annotations précises, sans aucun effort d'étiquetage manuel. Toutes les annotations nécessaires à l'analyse fine de véhicules (visibilité, point de vue, marques et modèles...) peuvent être générées automatiquement grâce aux modèles 3D. De plus, dans certains cas, les modèles 3D permettent de fournir des annotations qu'un homme aurait du mal à réaliser (annotations géométriques, marques et modèles...). Dans nos solutions, les modèles 3D sont également activement utilisés pour retrouver les informations 3D des véhicules dans une image (point de vue précis et localisation 3D).

Nous avons proposé un classifieur de marques et modèles de véhicules appris sur des données synthétiques générées avec des rendus 3D non-photoréalistes

de type contours. Ce classifieur est ensuite utilisé pour retrouver la marque et le modèle des véhicules dans des images réelles couleurs. L'originalité de ce travail réside dans l'apprentissage de caractéristiques visuelles discriminantes et communes à des cartes de contours et des images couleurs. Ce travail met en avant la capacité des réseaux de neurones à extraire des caractéristiques visuelles invariantes au domaine (contours et couleurs). Ce travail a fait l'objet de deux publications [2, 5].

Les tâches de détection de véhicules et d'estimation du point de vue précis ont également été abordées. Une nouvelle méthode basée sur des modèles d'apparence de parties et de cohérence géométrique a été introduite. L'originalité de cette approche réside dans l'utilisation exclusive de données synthétiques pour apprendre l'apparence et la géométrie des véhicules. Elle donne de bons résultats sur des images réelles mais reste assez sensible à la résolution des véhicules dans celles-ci. Ce travail a fait l'objet d'une publication [3].

Enfin, nous avons proposé une nouvelle approche multi-tâches permettant d'analyser les véhicules à partir d'une image de scène routière complexe. La première originalité de ce travail est l'introduction d'une architecture neuronale à raffinement apportant une grande précision de détection. La seconde concerne l'aspect multi-tâches de la méthode. Pour chaque détection, notre approche fournit la localisation des parties, la caractérisation de la visibilité de celles-ci ainsi que le gabarit 3D, l'orientation et la localisation 3D du véhicule dans la scène. La troisième originalité est en lien avec la génération des données pour l'apprentissage du réseau multi-tâches. Nous avons proposé un processus d'annotation intelligent, basé sur des modèles 3D de véhicules, pour annoter semi-automatiquement des images réelles faiblement annotées. Les résultats de cette méthode la placent à l'état de l'art en détection de véhicules, en estimation de l'orientation et en localisation 3D sur le *benchmark* KITTI dédié au véhicule autonome. Ce travail a fait l'objet d'une publication [1].

Perspectives

Les travaux réalisés dans cette thèse, laissent place à de nombreuses perspectives et certains questionnements.

Réseaux Multi-tâches et Multi-classes. Une première perspective intéressante serait d'étendre nos travaux à d'autres types d'objets. De premiers résultats qualitatifs ont été apportés sur ce point, notamment dans les perspectives du chapitre 6, mais nous pourrions évaluer cela quantitativement sur certaines bases publiques (Kitti [Geiger 2012], Pascal3D+ [Xiang 2014] et ObjectNet3D [Xiang 2016]).

Réseau de propositions d'objets 3D. Une extension en cours d'implémentation pour l'analyse 3D d'objets est celle d'un réseau de proposition d'objets 3D. Dérivé du réseau de proposition 2D de [Ren 2015], ce RPN pourrait directement proposer un ensemble de boîtes 3D contenant des objets à partir d'une image. En d'autres termes, les ancres 2D proposées dans le RPN seraient remplacées par des ancres 3D en discrétisant angulairement et spatialement l'espace 3D.

Segmentation sémantique et d'instances. Une troisième perspective consisterait à exploiter au maximum le concept multi-tâches. Il serait possible d'imaginer un seul réseau de neurones permettant de détecter et d'analyser finement les objets tout en fournissant additionnellement une carte de segmentation sémantique ainsi que la segmentation de chaque objet détecté.

Les jeux vidéos pour les données. Des approches récentes utilisent des jeux vidéo très réalistes (notamment *Grand Theft Auto* GTA V) pour générer automatiquement ou semi-automatiquement des données d'apprentissage [Richter 2016, Johnson-Roberson 2016]. Ce jeu présente une grande variabilité de mondes virtuels et les informations 3D des objets présents dans les scènes peuvent être récupérées. De cette manière, des bases de données annotées (en 2D et en 3D) peuvent automatiquement être générées. Une perspective intéressante de cette thèse serait de tester les algorithmes développés par apprentissage sur ce type de données.

Bibliographie

- [Abadi 2016] M. Abadi, A. Agarwal et P. Barham. *TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems*, 2016. (Cité en page 8.)
- [Abu-El-Haija 2016] S. Abu-El-Haija, N.g Kothari, J. Lee, A. Natsev, G. Toderici, B. Varadarajan et S. Vijayanarasimhan. *YouTube-8M : A Large-Scale Video Classification Benchmark*. arXiv :1609.08675, 2016. (Cité en page 8.)
- [Ahonen 2006] T. Ahonen, A. Hadid et M. Pietikainen. *Face Description with Local Binary Patterns : Application to Face Recognition*. PAMI, 2006. (Cité en page 42.)
- [Andriluka 2009] M. Andriluka, S. Roth et B. Schiele. *Pictorial Structures Revisited : People Detection and Articulated Pose Estimation*. CVPR, 2009. (Cité en page 40.)
- [Arbelaez 2011] P. Arbelaez, M. Maire, C. Fowlkes et J. Malik. *Contour Detection and Hierarchical Image Segmentation*. PAMI, 2011. (Cité en page 60.)
- [Aubry 2014] M. Aubry, D. Maturana, A. Efros, B. Russell et J. Sivic. *Seeing 3D chairs : exemplar part-based 2D-3D alignment using a large dataset of CAD models*. CVPR, 2014. (Cité en page 76.)
- [Belongie 2000] S. Belongie, J. Malik et J. Puzicha. *Shape Context : A new descriptor for shape matching and object recognition*. In NIPS, 2000. (Cité en page 44.)
- [Bourdev 2009] L. Bourdev et J. Malik. *Poselets : Body Part Detectors Trained Using 3D Human Pose Annotations*. ICCV, 2009. (Cité en page 43.)
- [Burl 1996] M. C. Burl et P. Perona. *Recognition of Planar Object Classes*. CVPR, 1996. (Cité en page 39.)
- [Cadena 2015] C. Cadena, A. Dick et Ian Reid. *A Fast, Modular Scene Understanding System using Context-Aware Object Detection*. ICRA, 2015. (Cité en page 120.)
- [Cai 2016] Z. Cai, Q. Fan, R. Feris et N. Vasconcelos. *A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection*. ECCV, 2016. (Cité en page 120.)
- [Canny 1986] J. Canny. *A Computational Approach To Edge Detection*. PAMI, 1986. (Cité en pages x, 60 et 62.)

- [Carreira 2010] J. Carreira et C. Sminchisescu. *Constrained Parametric Min-Cuts for Automatic Object Segmentation*. CVPR, 2010. (Cité en page 35.)
- [Chen 2014] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun et A. Yuille. *Detect What You Can : Detecting and Representing Objects using Holistic Models and Body Parts*. CVPR, 2014. (Cité en page 40.)
- [Chen 2015] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler et R. Urtasun. *3D Object Proposals for Accurate Object Class Detection*. NIPS, 2015. (Cité en pages 36, 118, 119, 120, 122 et 123.)
- [Chen 2016] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler et R. Urtasun. *Monocular 3D Object Detection for Autonomous Driving*. CVPR, 2016. (Cité en pages ix, 36, 37, 118, 119, 120, 122, 123 et 126.)
- [Choi 2010] M. J. Choi, J. J. Lim, A. Torralba et A. S. Willsky. *Exploiting hierarchical context on a large database of object categories*. CVPR, 2010. (Cité en page 30.)
- [Chopra 2005] S. Chopra, R. Hadsell et Y. LeCun. *Learning a similarity metric discriminatively, with application to face verification*. CVPR, 2005. (Cité en page 65.)
- [Clevert 2016] D. Clevert, T. et S. Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. ICLR, 2016. (Cité en page 21.)
- [Collobert 2002] R. Collobert, S. Bengio et J. Marithoz. *Torch : A Modular Machine Learning Software Library*, 2002. (Cité en page 8.)
- [Cortes 1995] C. Cortes et V. Vapnik. *Support-Vector Networks*. ML, 1995. (Cité en page 10.)
- [Dai 2016] J. Dai, Y. Li, K. He et J. Sun. *R-FCN : Object Detection via Region-based Fully Convolutional Networks*. NIPS, 2016. (Cité en page 36.)
- [Dalal 2005] N. Dalal et B. Triggs. *Histograms of Oriented Gradients for Human Detection*. CVPR, 2005. (Cité en pages 11, 32, 34, 38, 40 et 47.)
- [Deng 2009] J. Deng, W. Dong, R. Socher, L. Li, K. Li et L. Fei-fei. *Imagenet : A large-scale hierarchical image database*. CVPR, 2009. (Cité en pages 8 et 30.)
- [DiCarlo 2007] J. J. DiCarlo et D. D. Cox. *Untangling invariant object recognition*. Trends in Cognitive Sciences, 2007. (Cité en page 9.)

- [Duchi 2011] J Duchi, E Hazan et Y Singer. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. JMLR, 2011. (Cité en page 17.)
- [Endres 2010] I. Endres et D. Hoiem. *Category Independent Object Proposals*. ECCV, 2010. (Cité en page 35.)
- [Everingham 2010] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn et A. Zisserman. *The Pascal Visual Object Classes (VOC) Challenge*. IJCV, 2010. (Cité en pages 30, 48 et 66.)
- [Felzenszwalb 2004] P. Felzenszwalb et D. Huttenlocher. *Efficient Graph-Based Image Segmentation*. IJCV, 2004. (Cité en page 35.)
- [Felzenszwalb 2005] P.F. Felzenszwalb et D.P. Huttenlocher. *Pictorial Structures for Object Recognition*. IJCV, 2005. (Cité en page 40.)
- [Felzenszwalb 2010] P.F. Felzenszwalb, R.B. Girshick, D. McAllester et D.Ramanan. *Object detection with discriminatively trained part based models*. PAMI, 2010. (Cité en pages 32, 34, 40, 41, 44, 82, 86 et 120.)
- [Fergus 2003] R. Fergus, P. Perona et A. Zisserman. *Object class recognition by unsupervised scale-invariant learning*. CVPR, 2003. (Cité en page 40.)
- [Ferrari 2004] V. Ferrari, T. Tuytelaars et L. J. V. Gool. *Integrating Multiple Model Views for Object Recognition*. CVPR, 2004. (Cité en page 41.)
- [Ferrari 2007] V. Ferrari, F. Jurie et C. Schmid. *Accurate Object Detection with Deformable Shape Models Learnt from Images*. CVPR, 2007. (Cité en page 32.)
- [Ferrari 2010] V. Ferrari, F. Jurie et C. Schmid. *From images to shape models for object detection*. IJCV, 2010. (Cité en page 32.)
- [Fidler 2012] S. Fidler, S. Dickinson et R. Urtasun. *3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model*. NIPS, 2012. (Cité en pages 44 et 117.)
- [Fischler 1973] M. A. Fischler et R. A. Elschlager. *The Representation and Matching of Pictorial Structures*. IEEE Trans. Computer., 1973. (Cité en page 40.)
- [Fischler 1981] M.A. Fischler et R.C. Bolles. *Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Communications of the ACM, 1981. (Cité en page 80.)

- [Freund 1997] Y. Freund et R.E. Schapire. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 1997. (Cité en page 33.)
- [Furukawa 2010] Y. Furukawa et J. Ponce. *Accurate, Dense, and Robust Multiview Stereopsis*. PAMI, 2010. (Cité en page 45.)
- [Ganin 2015] Y. Ganin et V. Lempitsky. *Unsupervised Domain Adaptation by Backpropagation*. ICML, 2015. (Cité en pages 59 et 65.)
- [Gao 2003] X. Gao, X. Hou, J. Tang et H. Cheng. *Complete Solution Classification for the Perspective-Three-Point Problem*. PAMI, 2003. (Cité en page 80.)
- [Gao 2011] T. Gao, B. Packer et D. Koller. *A segmentation-aware object detection model with occlusion handling*. CVPR, 2011. (Cité en page 42.)
- [Garcia 2002] C. Garcia et M. Delakis. *A Neural Architecture for Fast and Robust Face Detection*. ICPR, 2002. (Cité en page 34.)
- [Geiger 2011] A. Geiger, C. Wojek et R. Urtasun. *Joint 3D Estimation of Objects and Scene Layout*. NIPS, 2011. (Cité en page 41.)
- [Geiger 2012] A. Geiger, P. Lenz et R. Urtasun. *Are we ready for Autonomous Driving ? The KITTI Vision Benchmark Suite*. CVPR, 2012. (Cité en pages xi, 30, 43, 50, 105, 117, 118, 119 et 135.)
- [Ghodrati 2014] A. Ghodrati, M. Pedersoli et T. Tuytelaars. *Is 2D Information Enough For Viewpoint Estimation ?* BMVC, 2014. (Cité en page 41.)
- [Girshick 2011] R.B. Girshick, P.F. Felzenszwalb et D.A. McAllester. *Object Detection with Grammar Models*. NIPS, 2011. (Cité en page 43.)
- [Girshick 2014] R. Girshick, J. Donahue, T. Darrell et J. Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. CVPR, 2014. (Cité en page 35.)
- [Girshick 2015] R. Girshick. *Fast R-CNN*. ICCV, 2015. (Cité en pages 23, 35, 37, 105, 109 et 113.)
- [Glasner 2011] D. Glasner, M. Galun, S. Alpert, R. Basri et G. Shakhnarovich. *Viewpoint-aware Object Detection and Pose Estimation*. ICCV, 2011. (Cité en pages 45 et 94.)
- [Gu 2010] C. Gu et X. Ren. *Discriminative Mixture-of-templates for Viewpoint Classification*. ECCV, 2010. (Cité en page 41.)

- [Harris 1988] C. Harris et M. Stephens. *A combined corner and edge detector*. In Proc. of Fourth Alvey Vision Conference, 1988. (Cité en page 39.)
- [He 2015] K. He, X. Zhang, S. Ren et J. Sun. *Delving deep into rectifiers : Surpassing human-level performance on imagenet classification*. ICCV, 2015. (Cité en page 21.)
- [He 2016] K. He, X. Zhang, S. Ren et J. Sun. *Deep Residual Learning for Image Recognition*. CVPR, 2016. (Cité en pages 23, 26 et 27.)
- [He 2017] K. He, G. Gkioxari, P. Dollár et R. Girshick. *Mask R-CNN*. arXiv :1703.06870, 2017. (Cité en page 36.)
- [Hejrati 2012] M. Hejrati et D. Ramanan. *Analyzing 3D Objects in Cluttered Images*. NIPS, 2012. (Cité en page 45.)
- [Huang 2016] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama et K. Murphy. *Speed/accuracy trade-offs for modern convolutional object detectors*. arXiv :1611.10012, 2016. (Cité en page 36.)
- [Ioffe 2015] S. Ioffe et C. Szegedy. *Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift*. ICML, 2015. (Cité en page 22.)
- [Jia 2014] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama et T. Darrell. *Caffe : Convolutional Architecture for Fast Feature Embedding*. arXiv preprint arXiv :1408.5093, 2014. (Cité en page 8.)
- [J.J.Lim 2014] J.J.Lim, A.Khosla et A.Torralba. *FPM : Fine pose Parts-based Model with 3D CAD models*. ECCV, 2014. (Cité en page 76.)
- [Johnson-Roberson 2016] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar et R. Vasudevan. *Driving in the Matrix : Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks ?* <http://arxiv.org/abs/1610.01983>, 2016. (Cité en page 135.)
- [Kiefel 2014] M. Kiefel et P. Gehler. *Human Pose Estimation with Fields of Parts*. ECCV, 2014. (Cité en page 40.)
- [Kingma 2015] D. Kingma et J. Ba. *Adam : A Method for Stochastic Optimization*. ICLR, 2015. (Cité en page 17.)

- [Kong 2016] T. Kong, A. Yao, Y. Chen et F. Sun. *HyperNet : Towards Accurate Region Proposal Generation and Joint Object Detection*. CVPR, 2016. (Cité en pages 36 et 105.)
- [Krause 2013] J. Krause, M. Stark, J. Deng et L. Fei-Fei. *3D object representations for fine-grained categorization*. ECCV, 2013. (Cité en pages 47 et 56.)
- [Krause 2015] J. Krause, H. Jin, J. Yang et L. Fei-Fei. *Fine-Grained Recognition without Part Annotations*. CVPR, 2015. (Cité en page 45.)
- [Krizhevsky 2012] A. Krizhevsky, I. Sutskever et G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. NIPS, 2012. (Cité en pages ix, 8, 24, 25 et 38.)
- [Kullback 1951] S. Kullback et R. A. Leibler. *On Information and Sufficiency*. Annals of Mathematical Statistics, 1951. (Cité en page 68.)
- [Laurentini 1994] A. Laurentini. *The visual hull concept for silhouette-based image understanding*. PAMI, 1994. (Cité en page 46.)
- [Lecun 1986] Y. Lecun. *Learning Processes in an Asymmetric Threshold Network*. Disordered systems and biological organization, 1986. (Cité en page 14.)
- [LeCun 1989] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard et L. D. Jackel. *Handwritten digit recognition with a back-propagation network*. NIPS, 1989. (Cité en page 17.)
- [Lecun 1998] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner. *Gradient-based learning applied to document recognition*. ISP, 1998. (Cité en pages 18 et 82.)
- [Leibe 2004] B. Leibe, A. Leonardis et B. Schiele. *Combined Object Categorization and Segmentation With An Implicit Shape Model*. ECCV workshop, 2004. (Cité en pages 38, 39, 41 et 45.)
- [Lepetit 2009] V. Lepetit, F. Moreno-Noguer et P. Fua. *EPnP : An Accurate $O(n)$ Solution to the PnP Problem*. IJCV, 2009. (Cité en pages 80 et 112.)
- [Levenberg 1944] K. Levenberg. *A method for the solution of certain non-linear problems in least squares*. Quarterly of Applied Mathematics 2, 1944. (Cité en pages 77, 80 et 92.)
- [Li 2014] B. Li, T. Wu et S. Zhu. *Integrating Context and Occlusion for Car Detection by Hierarchical And-Or Model*. ECCV, 2014. (Cité en page 120.)
- [Liebelt 2008] J. Liebelt, C. Schmid et K. Schertler. *Viewpoint-Independent Object Class Detection using 3D Feature Maps*. CVPR, 2008. (Cité en page 44.)

- [Liebelt 2010] J. Liebelt et C. Schmid. *Multi-View Object class Detection with a 3D Geometrique Model*. CVPR, 2010. (Cit  en pages 43, 44 et 76.)
- [Lim 2013] J.J. Lim, H. Pirsiavash et A. Torralba. *Parsing IKEA Objects : Fine Pose estimation*. ICCV, 2013. (Cit  en pages 43 et 76.)
- [Lin 2014a] T-Y Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll r et C. L. Zitnick. *Microsoft COCO : Common Objects in Context*. ECCV, 2014. (Cit  en pages 8 et 30.)
- [Lin 2014b] Y-L. Lin, V.I. Morariu, W. Hsu et L.S. Davis. *Jointly Optimizing 3D Model Fitting and Fine-Grained Classification*. ECCV, 2014. (Cit  en pages ix, 44, 45, 47, 56, 76 et 85.)
- [Liu 2012] J. Liu, A. Kanazawa, D. Jacobs et P. Belhumeur. *Dog breed classification using part localization*. ECCV, 2012. (Cit  en page 45.)
- [Liu 2016] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C-Y Fu et A. C. Berg. *SSD : Single Shot MultiBox Detector*. ECCV, 2016. (Cit  en page 36.)
- [Long 2014] C. Long, X. Wang, G. Hua, M. Yang et Y. Lin. *Accurate Object Detection with Location Relaxation and Regionlets Relocalization*. ACCV, 2014. (Cit  en page 120.)
- [Long 2015a] J. Long, E. Shelhamer et T. Darrell. *Fully Convolutional Networks for Semantic Segmentation*. CVPR, 2015. (Cit  en page 88.)
- [Long 2015b] M. Long et J. Wang. *Learning Transferable Features with Deep Adaptation Networks*. ICML, 2015. (Cit  en pages 59 et 65.)
- [Lopez-Sastre 2011] R.J. Lopez-Sastre, T. Tuytelaars et S. Savarase. *Deformable Part Models Revisited : A Performance Evaluation for Object Category Pose Estimation*. ICCV, 2011. (Cit  en pages 41, 42 et 76.)
- [Lowe 1999] D.G. Lowe. *Object Recognition from Local Scale-Invariant Features*. ICCV, 1999. (Cit  en pages 11 et 46.)
- [Maas 2013] A. L. Maas, A. Y. Hannun et A. Y. Ng. *Rectifier nonlinearities improve neural network acoustic models*. ICML, 2013. (Cit  en page 21.)
- [Marquardt 1963] D. Marquardt. *An algorithm for least-squares estimation of non linear parameters*. Journal of the Society for Industrial and Applied Mathematics, 1963. (Cit  en pages 77, 80 et 92.)

- [Massa 2015] F. Massa, B. Russell et M. Aubry. *Deep Exemplar 2D-3D Detection by Adapting from Real to Rendered Views*. ArXiv e-prints, vol. abs/1512.02497, 2015. (Cité en pages 59 et 65.)
- [McCulloch 1943] W.S. McCulloch et W. Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. Bulletin of mathematical biophysics, 1943. (Cité en pages 7, 8 et 12.)
- [Minsky 1969] M. Minsky et S. Papert. *Perceptrons : an introduction to computational geometry*. M.I.T. Press, 1969. (Cité en pages 8 et 13.)
- [Mottaghi 2015] R. Mottaghi, Y. Xiang et S. Savarese. *A Coarse-to-Fine Model for 3D Pose Estimation and Sub-Category Recognition*. CVPR, 2015. (Cité en page 47.)
- [Nilsback 2008] M-E. Nilsback et A. Zisserman. *Automated Flower Classification over a Large Number of Classes*. ICVGIP, 2008. (Cité en page 45.)
- [Ohn-Bar 2015] E. Ohn-Bar et M.M. Trivedi. *Learning to Detect Vehicles by Clustering Appearance Patterns*. T-ITS, 2015. (Cité en page 120.)
- [Ojala 1994] T. Ojala, M. Pietikainen et D. Harwood. *A Comparative Study of Texture Measures with Classification Based on Feature Distributions*. IAPR, 1994. (Cité en page 11.)
- [Ozuysal 2009] M. Ozuysal, V. Lepetit et P. Fua. *Pose Estimation for Category Specific Multiview Object Localization*. CVPR, 2009. (Cité en pages 41, 46 et 76.)
- [P. Dollar 2013] L. Zitnick P. Dollar. *Structured Forests for Fast Edge Detection*. ICCV, 2013. (Cité en page 60.)
- [Payet 2011] N. Payet et S. Todorovic. *From Contours to 3D Object Detection and Pose Estimation*. ICCV, 2011. (Cité en page 41.)
- [Peng 2015] X. Peng, B. Sun, K. Ali et K. Saenko. *Learning Deep Object Detectors from 3D Models*. ICCV, 2015. (Cité en page 43.)
- [Pepik 2012] B. Pepik, M. Stark, P. Gehler et B. Schielen. *Teaching 3D Geometry to Deformable Part Models*. CVPR, 2012. (Cité en pages 41, 43, 76, 94, 96 et 97.)
- [Pepik 2013] B. Pepik, M. Stark, P. Gehler et B. Schiele. *Occlusion Patterns for Object Class Detection*. CVPR, 2013. (Cité en pages 43 et 120.)
- [Pepik 2015a] B. Pepik, M. Stark, P. Gehler et B. Schiele. *Multi-view and 3D Deformable Part Models*. PAMI, 2015. (Cité en page 120.)

- [Pepik 2015b] B. Pepik, M. Stark, P. V. Gehler, T. Ritschel et B. Schiele. *3D Object Class Detection in the Wild*. CVPR, 2015. (Cité en page 76.)
- [Pishchulin 2013] L. Pishchulin, M. Andriluka, P. Gehler et B. Schiele. *Poselet Conditioned Pictorial Structures*. CVPR, 2013. (Cité en page 40.)
- [Prokaj 2009] J. Prokaj et G. Medioni. *3-D Model Based Vehicle Recognition*. WACV, 2009. (Cité en pages 46 et 56.)
- [Ramnah 2014] K. Ramnah, S.N. Sinha et R. Szeliski. *Car Make and Model Recognition using 3D Curve Alignment*. WACV, 2014. (Cité en pages 46 et 56.)
- [Ranjan 2016] R. Ranjan, V. M. Patel et R. Chellappa. *HyperFace : A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition*. arXiv :1603.01249, 2016. (Cité en page 43.)
- [Redmon 2016] J. Redmon, S. Divvala, R. Girshick et A. Farhadi. *You Only Look Once : Unified, Real-Time Object Detection*. CVPR, 2016. (Cité en page 36.)
- [Ren 2015] S. Ren, K. He, R.B. Girshick et J. Sun. *Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks*. NIPS, 2015. (Cité en pages 30, 35, 36, 104, 105, 106, 109, 112, 113, 119, 120 et 135.)
- [Richter 2016] S. R. Richter, V. Vineet, S. Roth et V. Koltun. *Playing for Data : Ground Truth from Computer Games*. ECCV, 2016. (Cité en page 135.)
- [Rosenblatt 1958] F. Rosenblatt. *The perceptron : A probabilistic model for information storage and organization in the brain*. Psychological Review, 1958. (Cité en pages 8 et 13.)
- [Rumelhart 1986] D.E. Rumelhart et J.L. McClelland. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Vol. 1 : Foundations*. MIT Press, 1986. (Cité en page 14.)
- [Savarese 2007] S. Savarese et L. Fei-Fei. *3D generic object categorization, localization and pose estimation*. ICCV, 2007. (Cité en pages 41, 78, 93 et 98.)
- [Seemann 2007] E. Seemann, M. Fritz et B. Schiele. *Towards Robust Pedestrian Detection in Crowded Image Sequences*. CVPR, 2007. (Cité en page 39.)

- [Sermanet 2014] P. Sermanet, X. Zhang D. Eigen, M. Mathieu, R. Fergus et Y. LeCun. *OverFeat : Integrated Recognition, Localization and Detection using Convolutional Networks*. ICLR, 2014. (Cité en pages 32, 34 et 88.)
- [Simonyan 2014] K. Simonyan et A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. ICLR, 2014. (Cité en pages ix, 24, 25, 38 et 118.)
- [Snavely 2006] N. Snavely, S. M. Seitz et R. Szeliski. *Photo Tourism : Exploring Photo Collections in 3D*. SIGGRAPH, 2006. (Cité en page 45.)
- [Sobel 1968] I. Sobel et G. Feldman. *A 3x3 isotropic gradient operator*. PCSA, 1968. (Cité en page 62.)
- [Sohn 2016] K. Sohn. *Improved Deep Metric Learning with Multi-class N-pair Loss Objective*. NIPS, 2016. (Cité en page 48.)
- [Song 2016] H. O. Song, Y. Xiang, S. Jegelka et S. Savarese. *Deep Metric Learning via Lifted Structured Feature Embedding*. CVPR, 2016. (Cité en page 48.)
- [Srivastava 2014] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov. *Dropout : A Simple Way to Prevent Neural Networks from Overfitting*. JMLR, 2014. (Cité en page 21.)
- [Stark 2010] M. Stark, M. Goesele et B. Schiele. *Back to the Future : Learning Shape Models from 3D CAD Data*. BMVC, 2010. (Cité en pages 41, 43, 76 et 94.)
- [Stark 2012] M. Stark, J. Krause, B. Pepik, D. Meger, J.J. Little, B. Schiele et D. Koller. *Fine-Grained Categorization for 3D Scene Understanding*. BMVC, 2012. (Cité en page 47.)
- [Su 2009] H. Su, M. Sun, L. Fei-Fei et S. Savarese. *Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories*. ICCV, 2009. (Cité en page 41.)
- [Su 2015] H. Su, Y. Li et L.J. Guibas. *Render for CNN : Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views*. ICCV, 2015. (Cité en pages 41, 42 et 43.)
- [Sun 2012] M. Sun, M. Telaprolu, H. Lee et S. Savarese. *An Efficient Branch-and-Bound Algorithm for Optimal Human Pose Estimation*. CVPR, 2012. (Cité en page 40.)

- [Szegedy 2013] C. Szegedy, A. Toshev et D. Erhan. *Deep Neural Networks for Object Detection*. NIPS, 2013. (Cité en pages 32 et 34.)
- [Szegedy 2015] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke et A. Rabinovich. *Going Deeper with Convolutions*. CVPR, 2015. (Cité en pages 26, 38, 67 et 118.)
- [Szegedy 2016] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens et Z. Wojna. *Rethinking the Inception Architecture for Computer Vision*. CVPR, 2016. (Cité en page 26.)
- [Taigman 2014] Y. Taigman, M. Yang, M.A. Ranzato et L. Wolf. *DeepFace : Closing the Gap to Human-Level Performance in Face Verification*. CVPR, 2014. (Cité en page 45.)
- [Thomas 2006] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele et L. Van Gool. *Towards Multi-View Object Class Detection*. CVPR, 2006. (Cité en page 41.)
- [Tieleman 2012] T. Tieleman et G. Hinton. *RMSProp : Divide the gradient by a running average of its recent magnitude*. NNML, 2012. (Cité en page 17.)
- [Torralba 2004] A. Torralba, K. P. Murphy et W. T. Freeman. *Sharing Features : Efficient Boosting Procedures for Multiclass Object Detection*. CVPR, 2004. (Cité en page 34.)
- [Torralba 2007] A. Torralba, K. P. Murphy et W. T. Freeman. *Sharing visual features for multiclass and multiview object detection*. PAMI, 2007. (Cité en page 34.)
- [Tulsiani 2015] S. Tulsiani et J. Malik. *Viewpoints and Keypoints*. CVPR, 2015. (Cité en pages 41 et 76.)
- [Tzeng 2014] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko et T. Darrell. *Deep Domain Confusion : Maximizing for Domain Invariance*. CoRR, vol. abs/1412.3474, 2014. (Cité en pages 59 et 65.)
- [Uijlings 2013] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers et A.W.M. Smeulders. *Selective Search for Object Recognition*. IJCV, 2013. (Cité en pages 35 et 36.)
- [Van der Maaten 2008] L. Van der Maaten et G.E. Hinton. *Visualizing High-Dimensional Data Using t-SNE*. JMLR, 2008. (Cité en page 68.)
- [Viola 2001] P.A. Viola et M.J. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR, 2001. (Cité en pages 11 et 33.)

- [Wang 2008] L. Wang, S. You et U. Neumann. *Supporting Range and Segment-based hysteresis thresholding in edge detection*. ICIP, 2008. (Cit  en pages x, 60, 62, 63, 66, 69 et 85.)
- [Wang 2009] X. Wang, T. X. Han et S. Yan. *An HOG-LBP human detector with partial occlusion handling*. ICCV, 2009. (Cit  en page 42.)
- [Weber 2000] M. Weber, M. Welling et P. Perona. *Unsupervised Learning of Models for Recognition*. ECCV, 2000. (Cit  en page 39.)
- [Wohlhart 2012] P. Wohlhart, M. Donoser, P. M. Roth et H. Bischof. *Detecting Partially Occluded Objects with an Implicit Shape Model Random Field*. ACCV, 2012. (Cit  en page 39.)
- [Wojek 2011] C. Wojek, S. Walk, S. Roth et B. Schiele. *Monocular 3D Scene Understanding with Explicit Occlusion Reasoning*. CVPR, 2011. (Cit  en page 42.)
- [Wu 2005] B. Wu et R. Nevatia. *Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors*. ICCV, 2005. (Cit  en page 42.)
- [Xiang 2012] Y. Xiang et S. Savarese. *Estimating the Aspect Layout of Object Categories*. CVPR, 2012. (Cit  en pages 45, 46, 76 et 85.)
- [Xiang 2014] Y. Xiang, R. Mottaghi et S. Savarese. *Beyond PASCAL : A Benchmark for 3D Object Detection in the Wild*. WACV, 2014. (Cit  en pages 41, 43, 76 et 135.)
- [Xiang 2015] Y. Xiang, W. Choi, Y. Lin et S. Savarese. *Data-Driven 3D Voxel Patterns for Object Category Recognition*. CVPR, 2015. (Cit  en pages 43, 44, 51, 118, 119, 120, 121, 122 et 123.)
- [Xiang 2016] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas et S. Savarese. *ObjectNet3D : A Large Scale Database for 3D Object Recognition*. ECCV, 2016. (Cit  en pages 43 et 135.)
- [Xiang 2017] Y. Xiang, W. Choi, Y. Lin et S. Savarese. *Subcategory-aware Convolutional Neural Networks for Object Proposals and Detection*. WACV, 2017. (Cit  en pages 36, 105, 118, 119 et 120.)
- [Xie 2015] S. Xie et Z. Tu. *Holistically-Nested Edge Detection*. ICCV, 2015. (Cit  en page 60.)
- [Yang 2013] Y. Yang et D. Ramanan. *Articulated Pose Estimation with Flexible Mixtures-of-parts*. PAMI, 2013. (Cit  en page 40.)

- [Yang 2015] L. Yang, P. Luo, C.C. Loy et X. Tang. *A Large-Scale Car Dataset for Fine-Grained Categorization and Verification*. CVPR, 2015. (Cité en pages ix, 47, 56, 59 et 63.)
- [Yang 2016] F. Yang, W. Choi et Y. Lin. *Exploit All the Layers : Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers*. CVPR, 2016. (Cité en pages 36, 105 et 120.)
- [Yoruk 2013] E. Yoruk et R. Vidal. *Efficient Object Localization and Pose Estimation with 3D Wireframe Models*. ICCV Workshops, 2013. (Cité en pages 44 et 45.)
- [Zeiler 2012] M. D. Zeiler. *ADADELTA : An Adaptive Learning Rate Method*. arXiv preprint :1212.5701, 2012. (Cité en page 17.)
- [Zhang 2013a] Y Zhang, Y. H Benhamza, K. Idrissi et C Garcia. *Incremental Principal Component Analysis-based Sparse Representation for Face Pose Classification*. International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS 2013), 2013. (Cité en page 41.)
- [Zhang 2013b] Y Zhang, K Idrissi et C Garcia. *A Dictionary-Learning Sparse Representation Framework for Pose Classification*. MLSP, 2013. (Cité en page 41.)
- [Zhang 2014] N. Zhang, J. Donahue, R. Girshick et T. Darre. *Part-based R-CNNs for Fine-grained Category Detection*. ECCV, 2014. (Cité en page 45.)
- [Zheng 2015] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang et P. Torr. *Conditional Random Fields as Recurrent Neural Networks*. ICCV, 2015. (Cité en page 37.)
- [Zhu 2014] M. Zhu, K. Derpanis, Y. Yang, S. Brahmabhatt, M. Zhang, C. Phillips, M. Lecce et K. Daniilidis. *Single Image 3D Object Detection and Pose Estimation for Grasping*. ICRA, 2014. (Cité en page 45.)
- [Zhu 2015] M. Zhu, X. Zhou et K. Daniilidis. *Single Image Pop-Up from Discriminatively Learned Parts*. ICCV, 2015. (Cité en page 44.)
- [Zia 2011] M. Zeeshan Zia, M. Stark, B. Schiele et K. Schindler. *Revisiting 3D Geometric Models for Accurate Object Shape and Pose*. ICCV-WS, 2011. (Cité en pages 43, 44, 76, 82, 85, 94, 95 et 96.)
- [Zia 2013a] M. Zeeshan Zia, M. Stark, B. Schiele et K. Schindler. *Detailed 3D Representations for Object Modeling and Recognition*. PAMI, 2013. (Cité en pages 44, 63, 76, 77, 85, 94, 95, 97 et 123.)

- [Zia 2013b] M. Zeeshan Zia, M. Stark et K. Schindler. *Explicit Occlusion Modeling for 3D Object Class Representations*. CVPR, 2013. (Cité en pages 42, 44, 76 et 85.)