République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Chadli Bendjedid El-Tarf

Faculté des Sciences et de la Technologie

Département d'Informatique

spécialité

Systèmes Informatiques intelligents

MÉMOIRE

POUR L'OBTENTION DU DIPLÔME DE MASTER

SYSTEME DE TRANSPORT BASE SUR LE DEEP LEARNING

---

par

GHOURI Fayçal

Soutenu devant la commission d'examen :

M.TOUAHRI Djamel Eddine	Président
M. BENTRAD Sassi	Examinateur
Mme. AHMED MALEK Nada	Encadrant

Année Universitaire : 2022/2023

Remerciements

À la conclusion de cette période de travail, nous souhaitons exprimer nos remerciements en premier lieu au Tout-Puissant, le Bon Dieu, pour Sa volonté, Sa santé et Sa patience qui nous ont été accordées tout au long de cette période.

Nous tenons à exprimer notre sincère gratitude à ma promotrice, Mme. **AHMED MALEK Nada**. Nous tenons à lui exprimer notre reconnaissance pour sa confiance, son dévouement, sa patience, sa disponibilité, ses conseils et son soutien constants tout au long de ce projet.

Nous souhaitons également remercier tous les membres du jury d'avoir accepté d'évaluer ce travail avec leur expertise et leur esprit critique.

Je remercie sincèrement :

monsieur **TOUAHRI DJAMEL EDDINE**, pour avoir honoré de présider les jury

monsieur **BENTRAD SASSI**, pour avoir accepté d'examiner ce travail.

Un grand merci aux PROFESSEURS de departement d'informatique en particulier à Mr. **CHEMMAM Chaouki** .

Enfin, nous tenons à exprimer notre chaleureux remerciement à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail.

Dédicaces

Au terme de cette étude, fruit de mes années de labeur, je tiens à remercier sincèrement Dieu, de m'avoir octroyé les moyens et les personnes qui m'ont aidé dans son élaboration.

Mon père (abdallâh) et ma mère(Fatima), mes frères mes Sœurs.

ainsi ma belle épouse femme qui m'ont énormément aidés à achever ce travail, que ce soit par ses conseils, ses orientations, ses disponibilité, ou même avec ses sympathie et son éternel sourire qui nous redonnent à chaque fois la volonté et la force de travailler.

Toute ma famille pour son soutien et ses sacrifices.

Tous les amis (bilel), (imad), et camarades, nombreux qu'ils sont à m'entourer et à m'encourager dont je ne peux pas citer leurs noms.

J'exprime ma reconnaissance à tout ce, qui de loin ou de prêt m'a soutenu dans la réalisation de ce mémoire.

Enfin tous ceux qui ont contribué de près ou de loin, à la réalisation de ce projet.

.

Fayçal

Table des matières

Table des figures

Liste des tableaux

Introduction générale

De nombreux constructeurs automobiles internationaux et entreprises de haute technologie se sont lancés dans le remplacement de la conduite humaine par un logiciel, passant d'une aide limitée au pilotage à l'automatisation totale. Parmi les raisons ayant poussé à l'avènement des voitures autonomes, le taux élevé d'accidents lié à l'imperfection humaine et la perte de temps majeure des déplacements en voiture.

En effet, un véhicule autonome est un véhicule automobile apte à rouler, sur une route ouverte, sans l'intervention d'un conducteur. Le concept vise à développer et produire un véhicule pouvant réellement circuler sur la voie publique dans le trafic sans l'intervention humaine en toute situation. C'est une application typique du domaine de la robotique mobile dans laquelle de nombreux acteurs sont engagés. Néanmoins des questions techniques, légales, psychologiques et juridiques restent non résolues. De ce fait, actuellement, plusieurs challenges dans ce domaine restent une tendance technologique majeure. A savoir, l'amélioration du comportement du véhicule face à des situations non prises en compte auparavant, ou encore des réponses à des questions d'ordre éthique.

Dans le cadre des véhicules autonomes, la mise en évidence et la perception de l'environnement du véhicule autonome est primordiale. En effet, c'est sur cette base que se construiront les actions sécurisées de commande des organes de conduite du véhicule telles que : accélérer, freiner, tourner etc. De ce fait, les progrès rapides des technologies de l'électronique, de l'information et des communications (menant à miniaturiser et améliorer les performances des ordinateurs, des capteurs et des réseaux) ont permis le développement de plusieurs technologies liées aux véhicules autonomes (VA). La SAE (*Society of Automotive Engineers*) qui

est l'agence de normalisation automobile, a divisé la capacité de conduite autonome d'un véhicule en six niveaux, allant des systèmes les plus élémentaires à la conduite 100% autonome. Ces niveaux permettent de mesurer le degré d'avancement de la technologie de la voiture autonome. La plateforme de conduite d'un VA est scindée en quatre (4) grandes phases : les capteurs qui représentent la source de données, la perception qui permet d'appréhender l'environnement avoisinant le véhicule, la planification de l'ensemble des comportements possibles et enfin le contrôle et la concrétisation d'une action que va entreprendre la voiture.

Dans notre travail, nous nous intéressons au volet "**perception**" dans l'architecture d'un véhicule autonome. En effet, nous allons : mettre en évidence le principe de fonctionnement des capteurs utilisés dans la voiture autonome, montrer le gain apporté par chacun et enfin effectuer une étude comparative de ces capteurs et ce, dans le but de choisir les capteurs qui nous permettent la réalisation des différents modules de perception.

Le présent rapport est scindé en quatre (4) chapitres. Dans le premier, nous allons présenter les généralités relatives au véhicule autonome à savoir : ses avantages, ses niveaux d'autonomie et la plateforme de la conduite autonome. Dans le deuxième chapitre, nous allons présenter une géneralité sur le Deep Learning. Dans le troisième chapitre, nous allons détailler le module de perception qu'on veut réaliser ainsi que notre démarche. Dans le dernier chapitre, nous exposerons les résultats des tests et la validation de module de perception sur les datasets choisis. Enfin, une conclusion générale dans laquelle nous résumons le travail réalisé et nous mettons en évidence quelques perspectives qui doivent permettre sa continuité.

Chapitre 1

Généralités sur les véhicules autonomes

1.1 Introduction

L'élaboration d'un véhicule autonome requiert l'intervention de technologies variées, allant de l'intelligence artificielle à la robotique, entre autres. La conception de tels véhicules repose principalement sur la mise en place d'une plateforme intégrant à la fois des composants matériels et logiciels, capable de contrôler le véhicule et d'effectuer des choix judicieux en temps réel. Cela soulève de nombreux défis technologiques visant à optimiser encore davantage cette plateforme pour une plus grande autonomie.

Un véhicule autonome se compose de plusieurs modules, parmi lesquels le plus pertinent pour notre étude est le module de perception. Comme son nom le suggère, ce module est celui qui permet au véhicule de percevoir et d'interpréter les éléments constitutifs de son environnement.

Ce chapitre vise à introduire les aspects essentiels relatifs à la voiture autonome, y compris ses avantages, ses différents niveaux d'autonomie, l'architecture du module de perception, ainsi que les captures nécessaires à son fonctionnement.

1.2 Motivations

La conduite autonome est définit comme la capacité d'une voiture à opérer de manière partielle ou totale, avec une intervention humaine minimale ou absente. Elle peut également être définie comme un véhicule capable de naviguer sans la nécessité d'un conducteur sur une voie publique [1].

FIGURE 1.1 – Voiture autonome (sans conducteur)

Plusieurs questions se posent sur la nécessité de l'autonomie des véhicules. Pour y répondre, il faut comprendre que l'innovation est toujours stimulée par un besoin précis. Initialement, le désir de mobilité a mené à la création de véhicules. Cependant, face à d'autres exigences, une transition vers des véhicules proposant divers niveaux d'autonomie a été observée par la suite.

Un véhicule traditionnel (avec conducteur) est un moyen de transport terrestre pour les personnes ou les biens, permettant un déplacement rapide et facile d'un endroit à un autre. Malgré les nombreux avantages que l'industrie automobile met en avant concernant la voiture traditionnelle, celle-ci présente divers problèmes liés à l'environnement, à la santé humaine et à la psychosociologie, parmi lesquels :

— La voiture est le moyen de transport individuel le plus polluant.

— La voiture en ville est dans la plupart des cas moins rapide à cause des embouteillages qui, en plus, causent la frustration et le stress chez les conducteurs.

— La voiture est l'une des causes de l'obésité grimpante dans la population et ce à cause de l'inactivité et le non exercice de la majorité des conducteurs.

— La voiture pousse à l'hyperindividualisme, c'est à dire, déconnecte le conducteur de la véritable réalité. Dans sa voiture, il est rare d'avoir des contacts sociaux spontanés avec les personnes qui l'entourent.

— La voiture est l'une des premières causes de mort accidentelle.

— La voiture fait diminuer le temps de travail utile d'un conducteur. En effet, le temps épuisé lors des déplacement pourrait être gagné en l'additionnant à celui du travail.

Ces diverses limites ont encouragé l'industrie automobile à chercher des solutions qui conservent les principaux avantages de la voiture traditionnelle tout en minimisant ses inconvénients. C'est de là qu'émerge l'idée d'une voiture capable de libérer le conducteur, offrant une vie plus facile, confortable et moins risquée. La voiture autonome offre donc plusieurs avantages, parmi lesquels :

— Réduction du stress en libérant le conducteur et les passagers des tensions liées aux embouteillages, mauvaises conditions météorologiques, mauvaise conduite d'autrui, etc.

— Limitation du nombre d'accidents de la route, puisque la plupart des accidents sont causés par des erreurs humaines comme la fatigue, la distraction, etc.

— Amélioration de la qualité de vie et du confort des personnes handicapées en leur permettant de se déplacer plus facilement et d'être plus indépendantes.

— Économie d'énergie grâce à leur système électronique intégré qui peut planifier les trajets les plus courts et les plus économiques.

— Libération du conducteur pour des tâches plus utiles et rentables, comme le travail, la lecture, etc.

Cependant, La voiture autonome peut avoir des désavantages à savoir :

— L'accentuation des pertes de certains emplois. Les emplois de conducteurs de taxis, bus et autres camions vont subir de grandes conséquences et ce à cause de leur automatisation totale.

— La vitesse incertaine de décision peut poser problème pour des événements annexes et imprévus pouvant survenir sur la route. Nous pouvons prendre comme exemple un animal qui apparaîtrait soudainement devant le véhicule. Dans ce cas de figure, comment la voiture va-t-elle réagir ? Un problème de moral que les ingénieurs informaticiens auront bien du mal à résoudre !

— Les risques de piratages informatiques sont importants. En effet, quand un pirate informatique s'attaque au système interne de la voiture, il compromettrait sa sécurité et pourrait ainsi la contrôler à distance, sans omettre le fait qu'il pourrait accéder aux informations personnelles du propriétaire tels que ses numéros de téléphone et de carte de crédit ou son lieu de résidence.

— L'incapacité provisoire de désigner un coupable en cas d'accidents. Nous remarquons ici un problème de juridiction qui apparaît lors de l'utilisation des voitures autonomes. En cas d'accident, qui serait jugé responsable ? Le constructeur ? L'éditeur du logiciel ? Impossible pour l'heure de prévoir comment le cas sera traité, car il n'y a pas encore de jurisprudence sur le sujet.

— L'aspect technique pose aussi un problème majeur. Il est à savoir que les technologies actuelles ne suffisent pas pour réaliser une voiture à 100 % autonome. De plus, la prise de décision est très délicate puisqu'elle repose sur plusieurs capteurs qui fournissent plusieurs données qui devraient être fusionnées afin de prendre la décision la plus rationnelle possible.

1.3 Classification de l'autonomie des véhicules

Lorsque l'on discute de l'automatisation de la conduite, il est courant de supposer que le véhicule est entièrement autonome, sans aucune intervention humaine. Cependant, il y a en réalité différents degrés d'autonomie dans la conduite(résumés dans la **table 1.1**). Il

Niveau d'automatisation	Nom	Exécution de la direction et accélération / décélération	Surveillance de l'environnement de conduite	Performances de repli d'une tâche de conduite dynamique	Capacité du système (modes de conduite)
Le conducteur surveille l'environnement de conduite					
0	Pas d'automatisation	Conducteur	Conducteur	Conducteur	Pas de modes de conduite
1	Assistance au conducteur	Conducteur/Système	Conducteur	Conducteur	Quelques modes de conduite
2	Automatisation partielle	Système	Conducteur	Conducteur	Quelques modes de conduite
Le système de conduite automatisé surveille l'environnement de conduite					
3	Automatisation conditionnelle	Système	Système	Conducteur	Quelques modes de conduite
4	Haute automatisation	Système	Système	Système	Quelques modes de conduite
5	Automatisation complète	Système	Système	Système	Tous les modes de conduite

TABLE 1.1 – SAE : Récapitulatif des niveaux d'autonomie du VA

est important de souligner qu'il n'y a pas de classification officielle des modèles existants et futurs. En 2014, la SAE (Society of Automotive Engineers) a établi un cadre pour distinguer les différents niveaux de conduite autonome.

Cette classification comprend six catégories, qui définissent ce que les conducteurs humains et/ou les systèmes autonomes peuvent ou ne peuvent pas faire. On peut regrouper la conduite autonome en deux grandes catégories : lorsque c'est l'humain qui surveille l'environnement, et lorsque c'est la technologie qui le fait.

Dans la première catégorie, on trouve :

Niveau 0 : Pas d'autonomie, sans assistance : À ce niveau, toutes les manœuvres sont effectuées par le conducteur humain, sans aucune assistance du véhicule. Les aides visuelles, comme les alertes de manque de carburant ou d'huile moteur, ne sont que des avertissements et ne constituent pas une aide à la conduite.

Niveau 1 : Assistance à la conduite, *eyes-on/hands-on* : Le conducteur doit toujours être présent et attentif, ne pouvant déléguer aucune tâche au véhicule. À ce niveau, le conducteur bénéficie d'une assistance pour le contrôle longitudinal (vitesse et distance avec les véhicules précédents) ou le contrôle latéral (suivi des lignes blanches), mais jamais les deux en même temps. Parmi les équipements de niveau 1, on peut citer le régulateur de vitesse, le radar de franchissement de ligne, le freinage automatique d'urgence, l'avertisseur de collision, la direction assistée, etc.

Niveau 2 : Automatisation partielle, *eyes-on/hands-off* : À ce niveau, le conduc-

teur peut confier au système le contrôle longitudinal et latéral du véhicule dans certaines circonstances. Cependant, il reste responsable de la supervision de son véhicule et doit être prêt à reprendre le contrôle total si nécessaire. On trouve à ce niveau des assistances comme le régulateur de vitesse adaptatif et le maintien de la voie.

Dans la seconde catégorie, on trouve :

Niveau 3 : Autonomie conditionnelle, *eyes-off/hands-off* : À ce niveau, le véhicule peut percevoir son environnement de conduite et agir en conséquence. Il est autonome dans certaines conditions de conduite prédéfinies, comme sur un parking. Cependant, le conducteur doit être prêt à reprendre le contrôle à tout moment.

Niveau 4 : Autonomie élevée, *eyes-off/hands-off/mind-off* : À ce niveau, aucune intervention du conducteur n'est requise, mais l'autonomie totale du véhicule est limitée à certaines conditions prédéfinies, comme une zone géographique spécifique ou certaines conditions météorologiques. Dans ces conditions, le conducteur n'est plus responsable de la conduite. Si le véhicule sort de ces conditions, le conducteur doit reprendre le contrôle, sinon le véhicule doit être capable de s'arrêter seul.

Niveau 5 : Autonomie complète, *driverless* : C'est le niveau ultime d'autonomie, où le véhicule est entièrement autonome, sans besoin d'intervention humaine et indépendamment des conditions de route ou de la météo. C'est ce niveau que tous les constructeurs visent. Si les avantages de cette technologie sont nombreux, elle présente aussi de nombreux défis.

1.4 Plateforme de conduite autonome

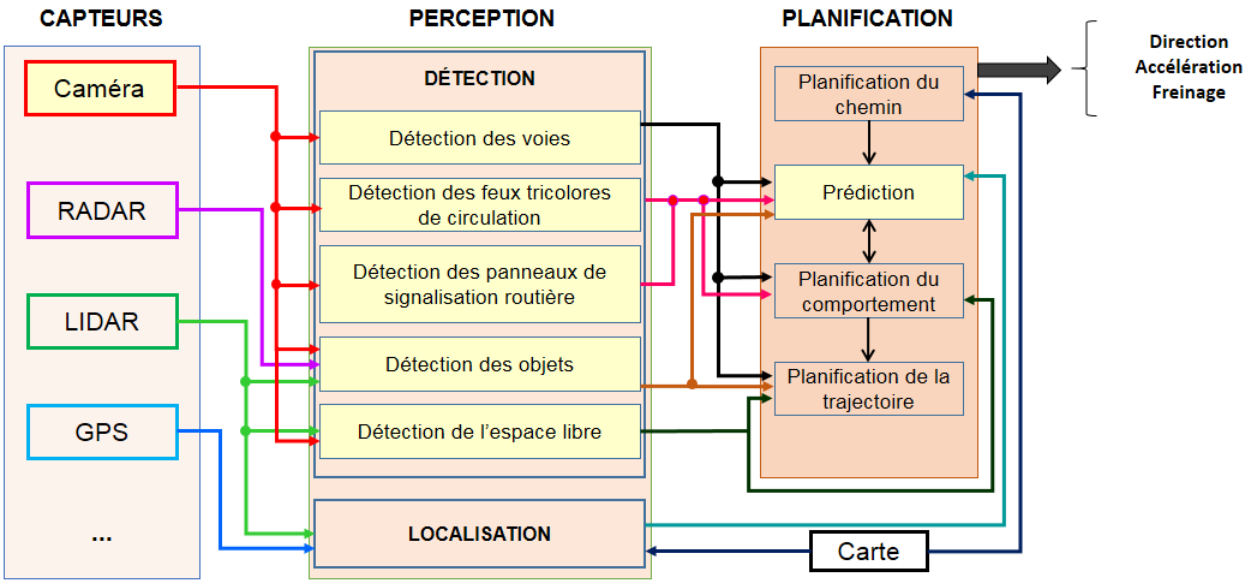


FIGURE 1.2 – Compétences de base d'un système logiciel du véhicule autonome

Les capacités fondamentales d'un système logiciel pour véhicules autonomes peuvent être se composés en trois module : la perception, la planification et le contrôle. Les interactions entre ces modules, ainsi qu'avec l'environnement sont démontrées dans la figure 1.2.

1.4.1 Captures

La conception et la mise en œuvre d'un véhicule autonome requièrent une gamme de composants matériels pour garantir les diverses fonctions essentielles à son fonctionnement optimal. Les capteurs de perception jouent un rôle central dans le processus d'activation d'un véhicule autonome.

1.4.1.1 Capteur à ultrasons

Dans un véhicule autonome, les capteurs à ultrasons sont fréquemment employés pour détecter les objets à proximité, comme les trottoirs ou les véhicules stationnés, par exemple. Typiquement, ces capteurs sont utilisés durant le stationnement et pour la détection d'objets

à courte distance lors de la conduite à faible vitesse [2]. Le fonctionnement de ces capteurs est assez simple. Les capteurs à ultrasons sont principalement composés d'un émetteur et d'un récepteur. L'émetteur envoie des ondes sonores à haute fréquence qui se déplacent à la vitesse du son dans l'air. Ces ondes se propagent dans l'air jusqu'à ce qu'elles rencontrent un obstacle et se réfléchissent. Le récepteur du capteur détecte ces ondes réfléchies et mesure le temps écoulé entre l'émission initiale et le retour de l'onde, permettant ainsi de calculer la distance entre l'objet et le capteur à ultrasons en utilisant la formule présentée dans la figure 1.3.

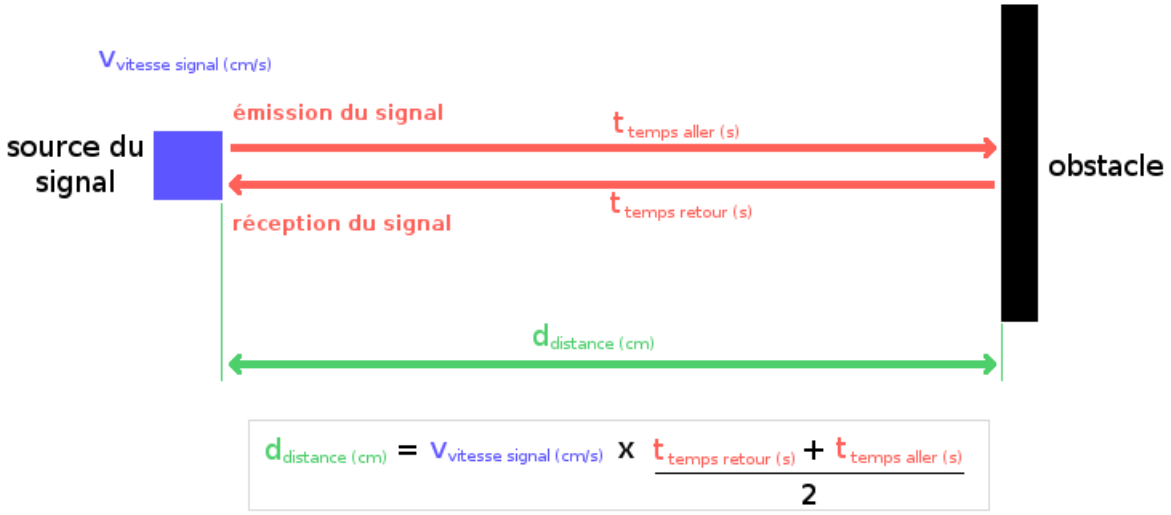


FIGURE 1.3 – Principe de fonctionnement des capteurs à ultrasons [3].

1.4.1.2 Capteurs Radar

Le RADAR (Radio Detection and Ranging) est un capteur qui présente des similitudes avec le capteur à ultrasons. Ils emploient des ondes électromagnétiques pour repérer les objets, déterminer leur distance et leur vitesse, et traquer leur mouvement. Comparativement aux capteurs à ultrasons, les capteurs RADAR peuvent détecter des objets à des distances plus éloignées, ce qui les rend idéaux pour repérer des objets en mouvement rapide, comme les autres véhicules sur l'autoroute.

Le fonctionnement de base est très similaire à celui des capteurs à ultrasons. Cependant, les RADARs offrent non seulement la possibilité de détecter la distance de plusieurs cibles

simultanément, mais ils sont également capables de déterminer avec précision la direction et la vitesse de chaque cible[2].

1.4.1.3 Capteurs LIDAR

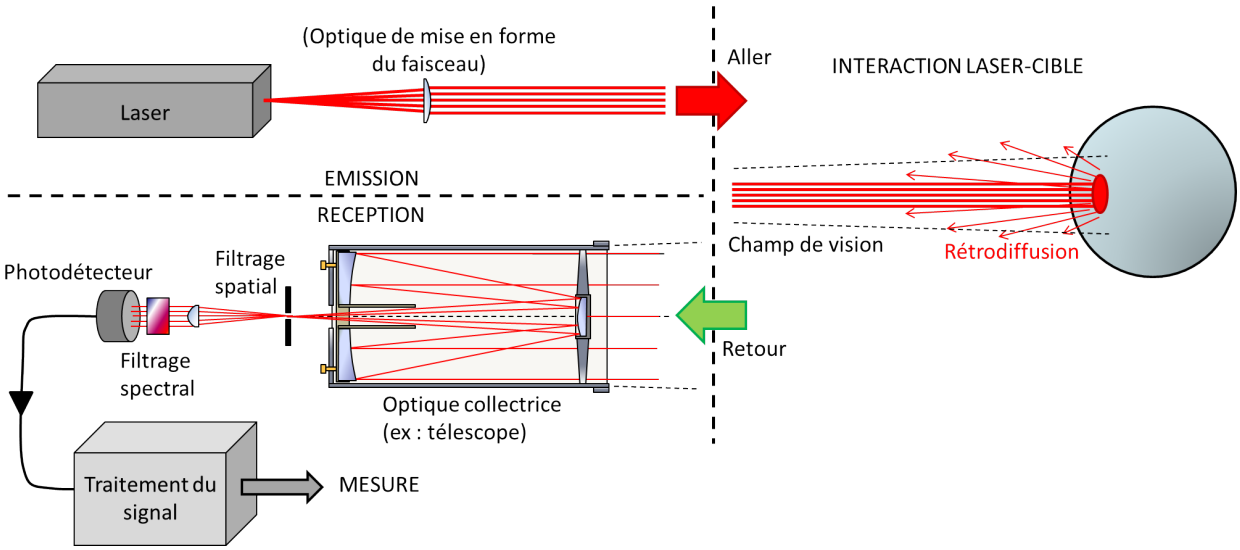


FIGURE 1.4 – Schéma d'un système LiDAR et du principe de la mesure [2]

Semblable au RADAR dans sa fonction, le LIDAR *Light Detection And Ranging*) emploie cependant de la lumière, plus spécifiquement des lasers, au lieu des ondes radio.

Le Lidar est reconnu pour sa précision exceptionnelle dans les mesures de distance, son traitement et son acquisition rapides des données, ainsi que sa capacité à fonctionner indépendamment des conditions de luminosité. Il est capable de produire une cartographie 3D en temps réel.

Malgré ses avantages, le Lidar présente quelques désavantages. Par rapport au RADAR, il est plus coûteux (le coût peut monter jusqu'à 75 000 euros), plus volumineux, consomme plus d'énergie et est plus sensible aux conditions météorologiques[2].

1.4.1.4 Capteurs caméras

Simples et discrètes, les caméras offrent une meilleure perception de l'environnement car une image permet une compréhension plus explicite. Ces caméras sont particulièrement

pertinentes pour notre étude. Les caméras qui sont largement utilisées dans les véhicules autonomes, peuvent être classées en deux types, visible (VIS) et infrarouge (IR)[2].

Les caméras VIS, capturant des longueurs d'ondes visibles entre 400 nm et 780 nm, offrent une haute résolution d'images colorées de l'environnement du véhicule à un coût réduit. Cependant, elles produisent une grande quantité de données, qui peut poser des défis pour le système de traitement. De plus, leur efficacité peut être affectée par les changements de conditions d'éclairage, ainsi que par les conditions météorologiques adverses comme la pluie, la neige ou le brouillard.

D'autre part, les caméras IR, des capteurs passifs fonctionnant dans des longueurs d'ondes infrarouges comprises entre 780 nm et 1 mm, présentent une résistance aux interférences lumineuses. Ces caméras sont souvent sollicitées dans des situations de forte luminosité, comme la conduite à la sortie d'un tunnel ou face au soleil, ainsi que pour la détection de sources de chaleur, telles que les piétons, les animaux ou d'autres véhicules.

1.4.1.5 *Localizing Ground Penetrating RADAR* (LGPR)

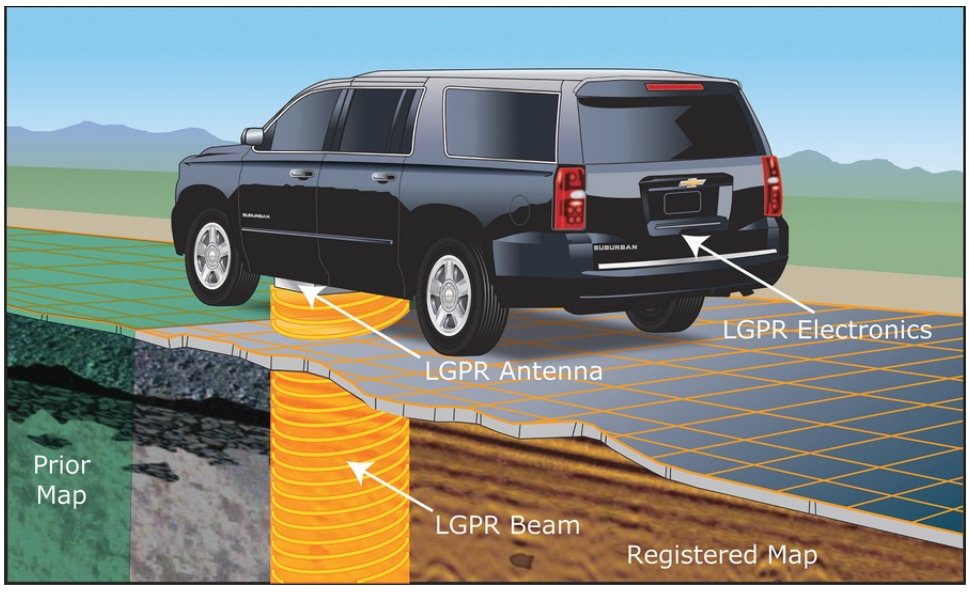


FIGURE 1.5 – Dessin conceptuel du LGPR [4]

Les systèmes GPR [4] fonctionnent en envoyant une impulsion de rayonnement dans le sol et en mesurant les réflexions qui proviennent des points de diffusion sous la surface. Les

réflexions se produisent à condition que les objets sous le sol ne soient pas significativement plus petits que la longueur d'ondes émises et qu'ils aient un contraste suffisant avec le sol environnant. Parmi les objets, nous citons : les tuyaux, les racines, les roches, etc. Les RADARs innovateurs de localisation à pénétration au sol permettent aux véhicules de garder leurs voies autonomes dans la neige, la pluie et le brouillard car les caractéristiques souterraines sont suffisamment uniques et statiques pour une certaine durée dans le temps (de six mois à une année).

Dans la **figure 1.5** qui représente le dessin conceptuel du réseau LGPR, nous voyons les signaux radio-fréquences (RF) rebondir sur les éléments souterrains pour localiser un véhicule à l'aide d'une carte préalable de la sous-surface.

Il y a deux étapes dans le fonctionnement du LGPR [4] :

***Mapping* :**

Elle consiste à élaborer une carte de l'environnement sous la route. Les données GPR des "objets" souterrains sont simplement collectées avec des balises GPS pour former la base de données initiale des caractéristiques du sous-sol. Cette base fera office de référence pour estimer l'emplacement des véhicules lors de visites ultérieures.

***Tracking* :**

La localisation en ligne s'effectue en plusieurs étapes. Lorsque le véhicule est en mouvement, les régions de données sont périodiquement extraites de la base de données pour être comparées. Une région de recherche autour de l'estimation initiale de l'emplacement est peuplée de "particules" représentant les emplacements et les orientations des candidats. Un algorithme évalue itérativement les particules pour affiner la recherche de la corrélation maximale au sein de l'échantillon. l'espace en cinq dimensions du véhicule (est, nord, hauteur, roulis et cap). Après plusieurs itérations, la corrélation la plus élevée est choisie et sa position est utilisée.

1.4.2 Perception de l'environnement

Le processus de perception, que nous voyons dans la **figure 1.6**, utilise une combinaison de capteurs de haute technologie, associés à un logiciel à la pointe de la technologie, permettant de traiter et de comprendre l'environnement autour du véhicule en temps réel.

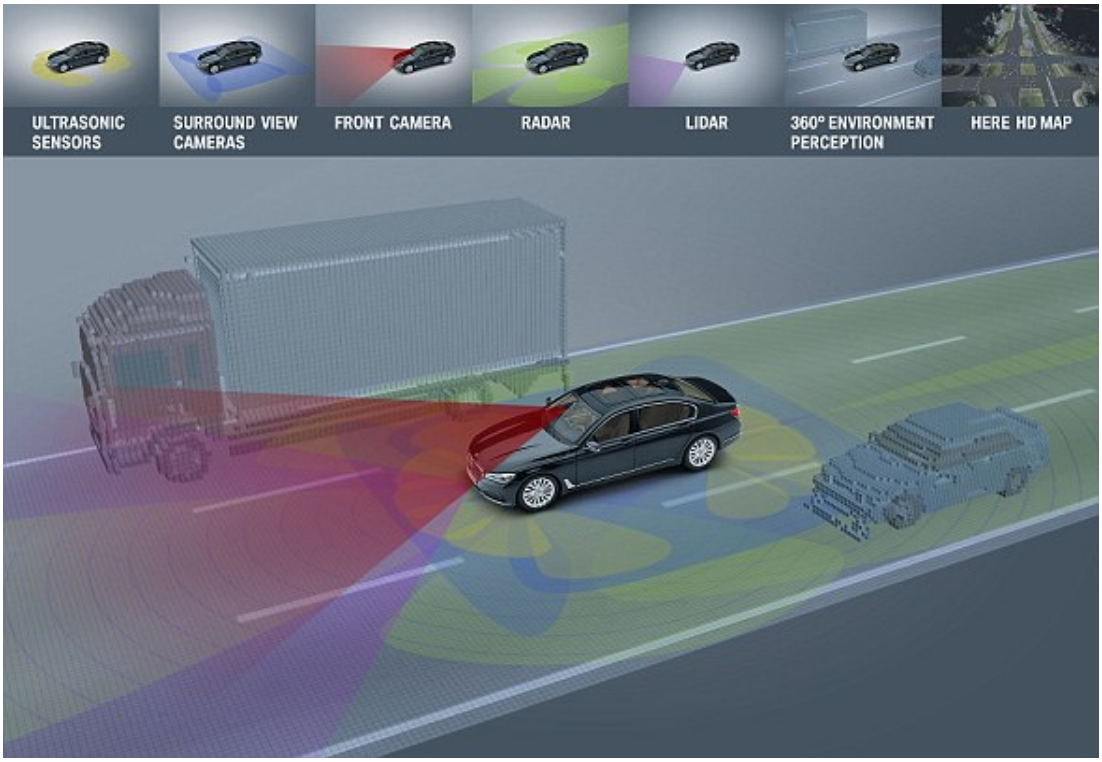


FIGURE 1.6 – La perception dans un véhicule autonome [5]

La perception dans les voitures autonomes est cruciale pour un fonctionnement sûr et fiable car les données reçues de ces processus alimentent le processus décisionnel fondamental qui détermine la manière dont le véhicule doit se déplacer ensuite ; de telle sorte que la voiture se dirige dans la bonne direction (vers sa destination) et ne mette pas en danger la vie des humains dans et autour de la voiture.

Par rapport à un conducteur humain, celui-ci doit, durant la conduite, se conformer aux voies et aux bordures de la route, identifier et interpréter les feux de signalisation et les panneaux routiers, ainsi que repérer divers objets (comme les voitures, les piétons, etc.). Pour permettre au véhicule de fonctionner de manière autonome, tous ces aspects doivent être gérés par les modules de perception avec une haute précision et en temps réel.

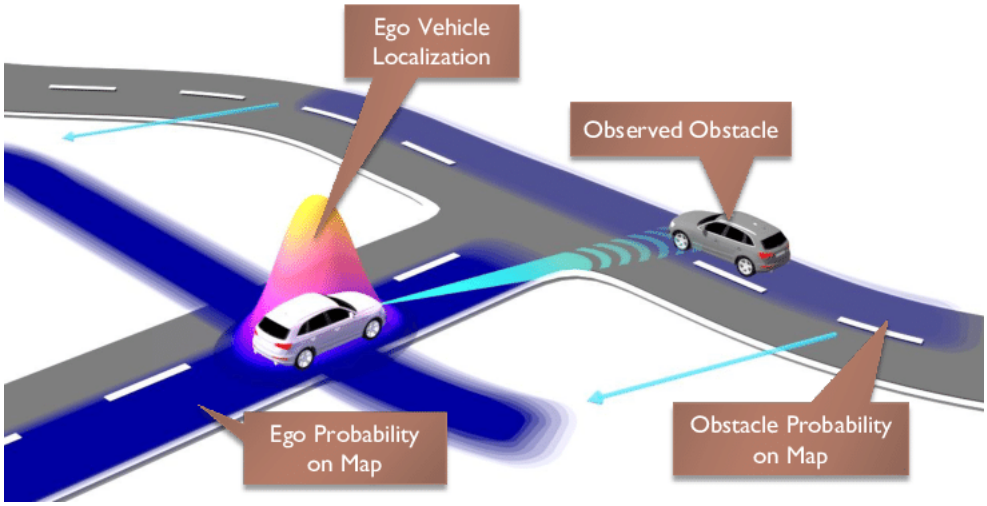


FIGURE 1.7 – La localisation dans un véhicule autonome

Le module de perception doit intégrer des systèmes informatiques, tels que les réseaux neuronaux convolutionnels, permettant de traiter et de fusionner les informations provenant des dispositifs de détection comme les RADARs, les LiDARs et les caméras. Dans le contexte de notre étude, nous nous concentrerons sur cet aspect, car c'est celui qui définira le degré d'autonomie de la voiture. En effet, cette autonomie dépend de la sécurité et de la fiabilité des actions que la voiture va entreprendre en se basant sur sa perception de l'environnement. C'est une tâche hautement complexe car le véhicule doit l'accomplir à deux niveaux : le premier concerne la perception d'un environnement en constante évolution durant la conduite, par exemple : les piétons, les panneaux de signalisation, les barrières, etc. ; et le second concerne la perception des autres véhicules. Ainsi, à partir de plusieurs capteurs, le véhicule doit extraire les données les plus pertinentes et significatives pour comprendre son environnement. Dans le processus de perception nous trouvons une autre phase nommée processus de localisation (**figure 1.7**). C'est une étape mise en œuvre dans la majorité des robots et des véhicules pour se localiser avec une très faible marge d'erreur. Si nous voulons prendre des décisions telles que : dépasser un véhicule ou simplement définir un itinéraire, nous devons savoir ce qui nous entoure et où nous en sommes. Seulement avec cette information, nous pouvons définir cet itinéraire.

1.4.3 Planification de trajectoires

Cette étape détermine chaque manœuvre qu'un véhicule autonome doit exécuter pour satisfaire un choix comportemental. La composante de planification et de surveillance génère une trajectoire sans obstacles et compose le plan de sa mise en œuvre à partir des fonctions de composition déployées sur le véhicule. Elle agit comme un superviseur qui décompose les tâches, choisit d'autres méthodes pour les atteindre et surveille l'exécution.

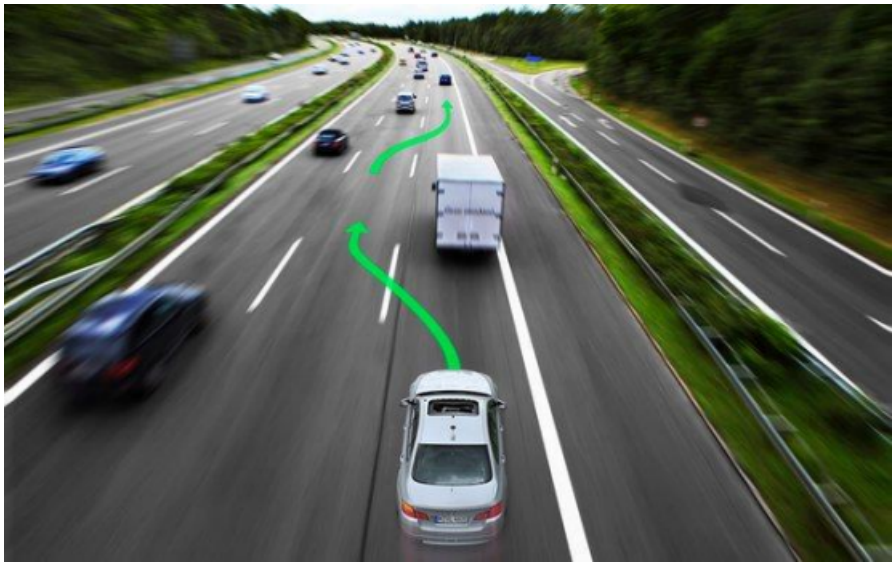


FIGURE 1.8 – Planification de trajectoires

1.4.4 Contrôle du véhicule

Le contrôle du véhicule est essentiel pour guider une voiture le long de la trajectoire prévue. La tâche du contrôle général est divisée en contrôle latéral et longitudinal. Cela permet au système de commande de gérer indépendamment les caractéristiques du véhicule (telles que les forces maximales autorisées, les angles de braquage maximaux, etc.) et l'analyse du compromis sécurité-confort.

Le bloc de contrôle de la trajectoire prend en entrée une trajectoire (générée au niveau de la planification et de la surveillance) et contrôle les modules latéraux et longitudinaux. La trajectoire qui représente un état futuriste, est donné par l'une des fonctions de composition de planification des chemins. Par exemple, si un changement de voie est nécessaire,

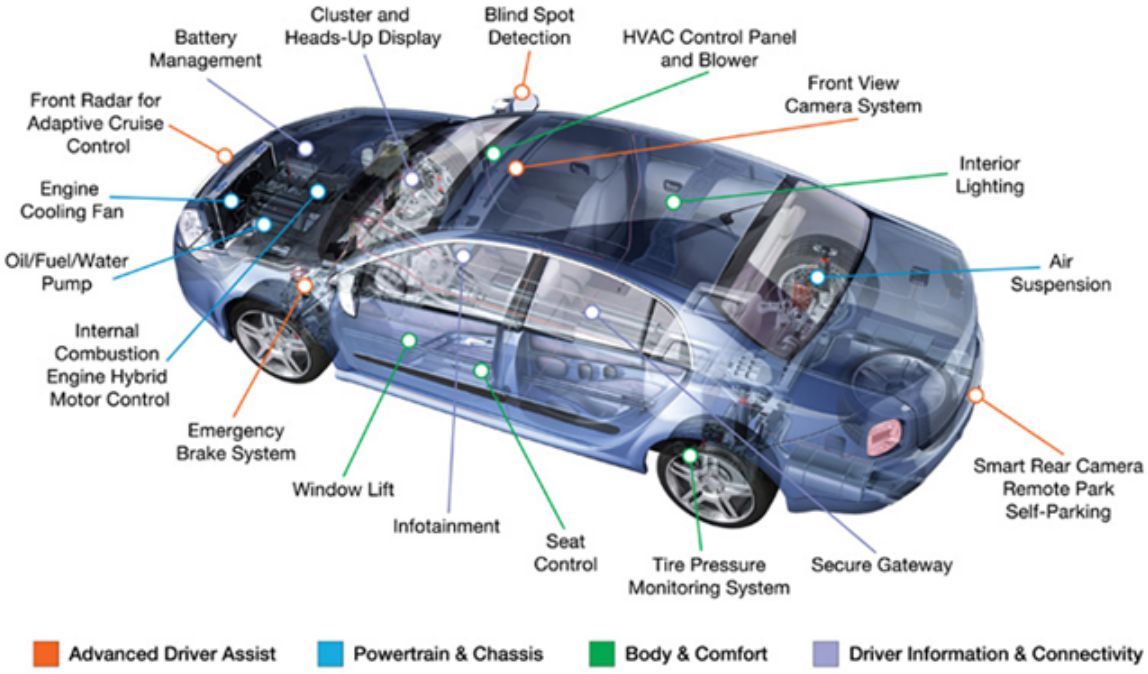


FIGURE 1.9 – Contrôle d'un véhicule autonome

la trajectoire représentera la position souhaitée en termes de coordonnées et d'orientations, sans aucune information sur la manière dont l'accélération, la direction ou le freinage seront effectués.

L'algorithme de contrôle longitudinal reçoit l'état longitudinal souhaité (tel que le freinage jusqu'à 40 km/h) et décide si l'action sera effectuée en accélérant, en freinant, en réduisant les gaz ou en utilisant le module de transmission (c'est-à-dire le freinage moteur). L'algorithme de contrôle latéral calcule l'angle de braquage de la cible en fonction des propriétés dynamiques du véhicule et de la trajectoire de la cible.

1.5 Conclusion

Dans ce chapitre, nous avons introduit les notions fondamentales liées aux véhicules autonomes. Nous avons exposé les bénéfices d'un véhicule autonome, ses degrés d'autonomie et la structure de la plateforme de conduite autonome. En outre, nous avons souligné la complexité de la phase de perception qui est notre centre d'intérêt. Nous avons également

décrit en détail les capteurs qui facilitent la collecte des données indispensables au processus de perception.

Chapitre 2

Géneralités sur Deep Learning

2.1 Introduction

Le module de perception représente le cerveau du système d'un véhicule autonome. La mise en œuvre de ce système nécessite l'adoption de solutions efficaces qui permettent le traitement et l'extraction des données provenant des capteurs installés sur le véhicule autonome.

En effet, l'intelligence artificielle, en particulier les réseaux de neurones convolutionnels, a marqué une avancée significative en termes de technologies utilisées dans les modules de perception intégrés dans les véhicules autonomes.

2.2 Catégorisation des algorithmes d'apprentissage

En règle générale, les algorithmes d'apprentissage se divisent en deux catégories distinctes : l'apprentissage automatique, également désigné par "*Machine Learning*", et l'apprentissage profond, ou "*Deep Learning*". Ce dernier a principalement vu le jour grâce à l'usage des réseaux de neurones artificiels (ANN), qui ont constitué le fondement pour l'évolution des réseaux de neurones convolutifs, ou "CNN" (Convolutional Neural Network). Depuis son émergence au début des années 90, cette technologie, aussi appelée "Convnet"

(Convolutional Network), a obtenu des résultats remarquables dans le domaine de la vision par ordinateur, ce qui est particulièrement pertinent pour notre champ de recherche.

2.3 Apprentissage en profondeur

L'apprentissage profond est une sous partie de l'apprentissage automatique. Il repose sur la notion de réseaux de neurones artificiels qui représentent un outil pour faire de l'apprentissage supervisé. Ils s'appuient sur un modèle mathématique inspiré de l'architecture biologique du cerveau humain[6]. En effet, ce dernier fait preuve de l'existence de réseaux neuronaux massifs qui peuvent réussir aux tâches cognitives, perceptuelles (reconnaissance des visages, parole, etc.) et de contrôle (mouvements et fonctionnement du corps). Cette machine biologique est constituée plus de 10 milliards de neurones interconnectés entre eux [7].

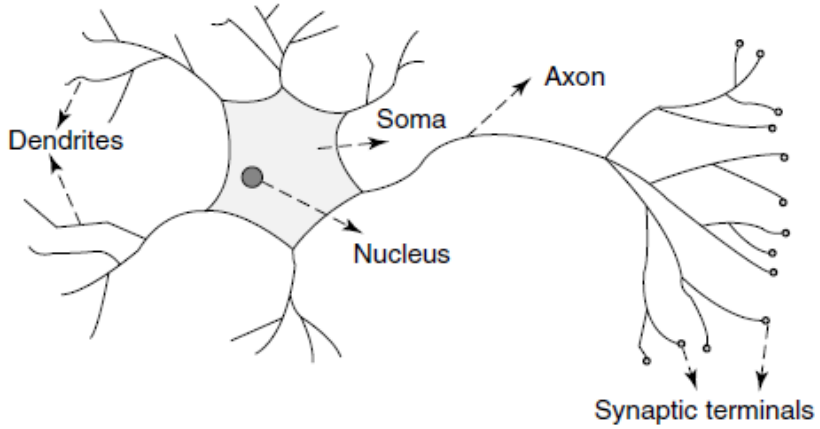


FIGURE 2.1 – Neurone biologique [7].

Le fonctionnement d'un neurone (figure 2.1) se base sur des réactions biochimiques pour recevoir, traiter et transmettre de l'information. Ces réactions sont trop complexes et difficiles à expliquer dans un rapport scientifique. Cependant, on peut le résume dans le principe suivant :

- Les données sont intervenues de la part des autres neurones, à travers *dendrites*, vers la partie principale qu'on appelle *Soma* ;

- Ces données sont traitées au niveau du *nucleus* qui représente le corps du *Soma* ;
- Les résultats de ces traitements sont transmis vers les autres neurones par une sortie principale qui s'appelle *Axon*. Cette dernière est subdivisée en plusieurs liens qui se terminent par des *Synaptic terminals*.

En réalité, les données transmises d'un neurone à un autre sont des petites charges électriques. Le principe est d'augmenter ou d'abaisser le potentiel électrique à l'intérieur du corps de la cellule réceptrice. Si le potentiel atteint un seuil, une impulsion est envoyée vers le bas de l'axone et la cellule est « déclenchée »[7].

Ce fonctionnement biologique a inspiré les informaticiens et bien avant les mathématiciens afin de créer les réseaux de neurones artificiels (*Artificial neural networks* (**ANN**)) dont la première proposition été de la part de McCulloch et Pitts en 1943[8]. Ces réseaux peuvent être définis comme interconnexion entre neurones artificiels, ou simplement neurones ou nœuds.

Dans un modèle mathématique simplifié du neurone, les effets des Synaptic sont représentés par des poids de connexion qui modulent l'effet des signaux d'entrée associés, et le rôle de nucleus est représenté par une fonction de transfert ou d'activation. L'impulsion neuronale est alors calculée comme la somme pondérée des signaux d'entrée, transformés par la fonction de transfert.

Un neurone artificiel typique et la modélisation d'un réseau neuronal à couches multiples sont illustrés à la figure 2.2. Le neurone (O) qui reçoit un flux de signal des entrées x_1, x_2,... , x_n, produit un signal de sortie donné par la relation suivante :

$$O = f\left(\sum_{i=1}^{n} w_{\mathrm{i}} x_{\mathrm{i}}\right)$$

où w_j est le vecteur de poids et la fonction f est appelée fonction d'activation.

Il mérite de signaler que la capacité d'apprentissage d'un neurone artificiel est obtenue en ajustant les poids en fonction de l'algorithme d'apprentissage choisi[7].

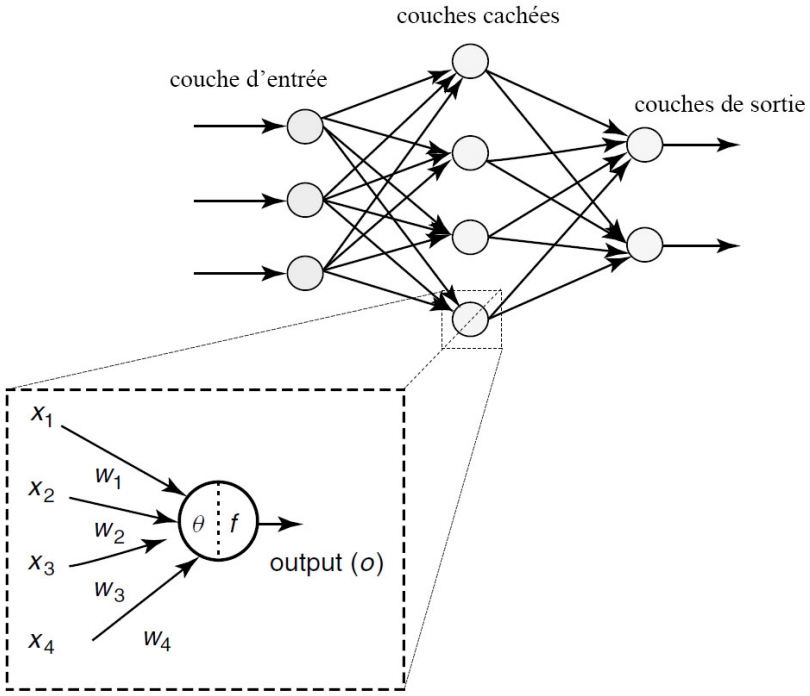


FIGURE 2.2 – Composition de réseau de neurones artificiels (ANN) [7].

2.3.1 Architecture d'un réseau CNN

Une architecture de réseau de neurones convolutifs est formée typiquement par un empilement de différents types de couches de traitement, à savoir : couche de convolution ; couche d'activation ; couche de *pooling* ; couche entièrement connectée et couche de sur-échantillonnage.

2.3.1.1 Couche de convolution

Une couche de convolution est la composante la plus importante d'une CNN, et constitue toujours au moins sa première couche. Elle comprend des matrices carrées de nombres discrets appelés filtres (également appelés *kernels*) de taille $f \times f$. Le but de cette couche est de repérer la présence d'un ensemble de caractéristiques (*features*) dans les images reçues en entrée et les extraires sous forme d'une carte de caractéristiques (*features map*).

Le principe de fonctionnement est de faire positionner le filtre tout en haut à gauche de l'image (de taille $h \times w$) puis faire un produit de chaque élément de ce filtre avec l'élément

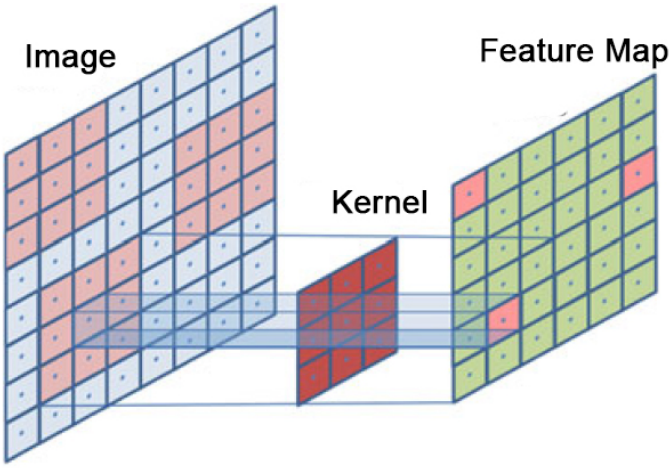


FIGURE 2.3 – Opération de convolution [9].

de l'image et on somme le tout. Le résultat est stocké dans une nouvelle case. Ensuite, on fait décaler d'un certain pas (appelé *stride* (s)) le filtre vers la droite et on répète l'opération précédente. Lorsqu'il arrivera au bout de l'image, il se décalera d'un pas vers le bas ainsi de suite jusqu'à ce que le filtre parcoure la totalité de l'image. La figure 2.3 montre un exemple de l'opération de convolution où les cases roses sont le résultat de l'application du filtre sous les sous-espaces colorés en rose dans l'image. De plus, la taille de la carte des caractéristiques ($h' \times w'$) dépend de trois facteurs : la taille de l'image en entrée, le pas et le Z *zero padding* (p). Ce dernier consiste à ajouter des zéros autour de l'image avant de commencer la convolution [9]. Cette taille est définie par les formules :

$$h' = \frac{h - f + s + p}{s},\ w' = \frac{w - f + s + p}{s}.$$

2.3.1.2 Couche d'activation

En général, les couches de convolution sont suivies par une couche d'activation non linéaire. Cette dernière prend toutes les valeurs des cartes des caractéristiques réelles pour recalculer d'autres valeurs qui sont généralement inclues dans des intervalles tels que [0,1] et [-1,1] (elles peuvent dépasser ces intervalles tout dépend de la fonction utilisée). Les fonctions de non-linéarité sont utilisées pour modéliser la mise en feu (ou les activations) de neurones spécifiques, ce principe est inspiré de la théorie biologique sur la façon dont les neurones du

cerveau sont connectés et permettent le traitement de l'information.

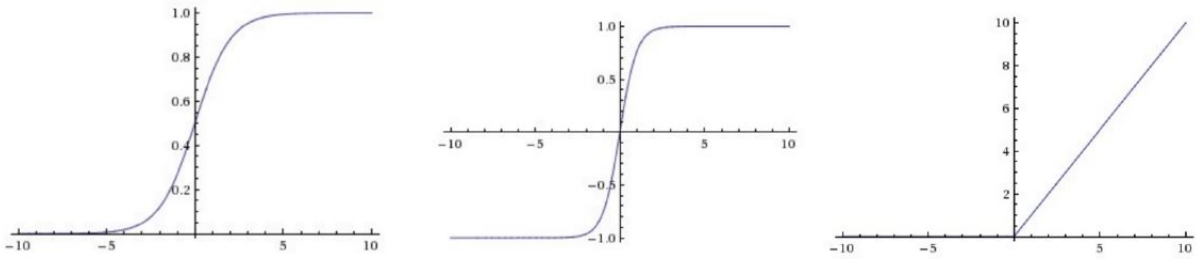


FIGURE 2.4 – Fonctions d'activation. A gauche, la fonction sigmoïde. Au milieu, la non-linéarité Tanh et à droite la fonction ReLU [10].

En effet, il existe une variété de fonctions d'activation telles que Sigmoïd, Relu, Tanh, etc. (la figure 2.4 montre l'allure de quelques fonctions d'activation). La fonction d'activation du Sigmoïde prend en compte un nombre réel comme entrée, et produit un nombre compris entre [0,1]. Il est défini comme suit :

$$F_{\text{sigm}}(z) = \frac{1}{1 + e^{-}z}.$$

La fonction d'activation Tanh met en œuvre la fonction tangente hyperbolique pour écraser les valeurs d'entrée dans la plage de [-1,1]. Elle est représentée comme suit :

$$F_{\tanh}(z) = \frac{e^{z} - e^{-}z}{e^{z} + e^{-}z}.$$

La fonction d'activation ReLU est une simple fonction d'activation qui est d'une importance pratique en raison de sa rapidité de calcul. Elle fait correspondre l'entrée à 0 si elle est négative et garde sa valeur inchangée si elle est positive. Cela peut être représenté comme suit :

$$F_{\text{ReLU}}(z) = max(0, z).$$

2.3.1.3 Couche de pooling

Cette couche est souvent placée entre deux couches de convolution. Elle reçoit en entrée plusieurs cartes des caractéristiques, et applique à chacune parmi elles l'opération de *pooling*

(appelé aussi *subsampling*). Elle consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Pour cela, on découpe l'image en cellules régulières de taille $f \times f$, puis on calcule soit la moyenne soit le maximum des valeurs de chaque cellule. En pratique, on utilise souvent des cellules carrées de petite taille pour ne pas perdre trop d'informations. Les choix les plus communs sont des cellules adjacentes pour f=2 qui ne se chevauchent pas (stride s=2), la figure 2.5 représente l'opération de *pooling* avec ces paramètres. En effet, cette couche permet de réduire les calculs et le nombre de paramètres dans le réseau. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage. On obtient en sortie le même nombre de cartes qu'en entrée, mais celles-ci sont bien plus petites [9]. Si on met en entrée de cette couche une carte de caractéristiques de taille $h \times w$, alors la taille de sortie sera $h' \times w'$ tels que :

$$h' = \frac{h - f + s}{s},\ w' = \frac{w - f + s}{s}.$$

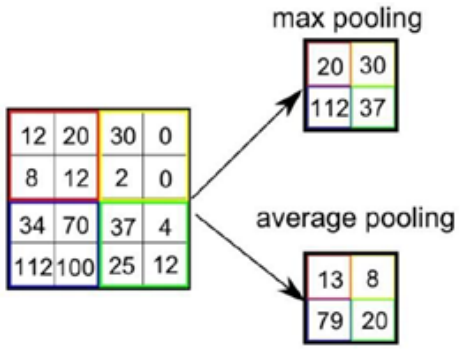


FIGURE 2.5 – Opération de *pooling* [9].

2.3.1.4 Couche entièrement connectée

La couche entièrement connectée (Fully Connected FC) peut être considérée comme une couche de convolution avec des filtres de taille 1×1. Le nombre d'unités de cette couche est N qui est le nombre total de classes. Chaque unité est reliée de manière dense à toutes les unités de la couche précédente. Dans un CNN typique, Ces couches sont généralement placées vers la fin de l'architecture. Toutefois, la littérature fait état de quelques architectures réussies qui utilisent ce type de couches à un emplacement intermédiaire dans un réseau CNN [9].

2.3.1.5 Couche de sur-échantillonnage

Tout ce qu'on a montré comme couches jusqu'à maintenant c'est propre aux architectures dédiées au problème de classification. En effet, dans telle architecture, on continue de produire des cartes des caractéristiques de taille réduite dans chaque couche de *pooling* ou de convolution (avec un pas supérieure ou égale à 2) jusqu'à arriver à la couche de classification où le résultat sera juste une classe (étiquette). Par contre, pour la segmentation, le résultat doit être une prédiction à pleine résolution. Cela signifie que si on met en entrée une image de taille $h \times w$, le résultat sera une image de taille $h \times w$ dont chaque pixel est coloré d'une couleur spécifique selon la classe à laquelle il appartient, et cette fonctionnalité est assurée grâce à la couche de sur-échantillonnage (*upsampling*).

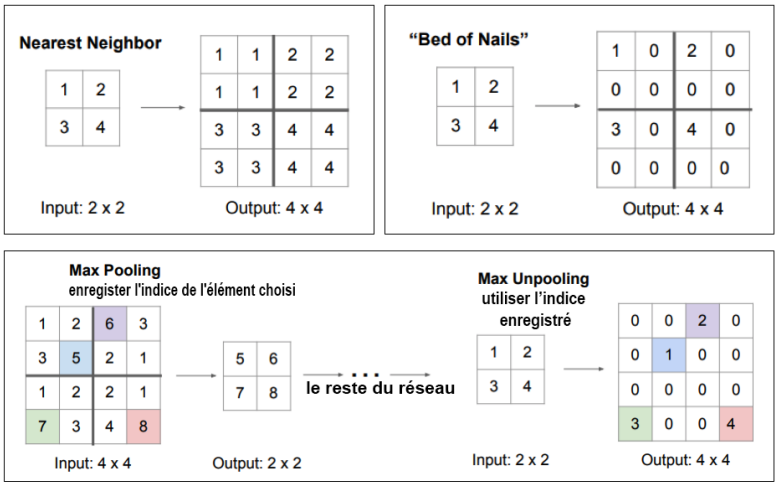


FIGURE 2.6 – Exemples de l'opération de *Unpooling* [11].

En outre, Le rôle principal de la couche *upsampling* est de produire une carte de caractéristiques de taille plus élevée que celle en entrée en utilisant plusieurs méthodes. *unpooling* (l'inverse de *pooling*) est une opération qui peut être utilisée dans la couche de sur-échantillonnage. Il existe plusieurs approches pour assurer cette opération telles que « *Nearest Neighbor* », « *Bed of Nail* », « *Max Unpooling* », etc. La figure 2.6 montre quelques applications numériques de cette opération ainsi que les convolutions transposées. En effet, Cette dernière n'est pas l'approche la plus populaire dans la couche de sur-échantillonnage.

Alors qu'une opération de convolution typique prend le produit des valeurs actuellement

dans la vue du filtre et produit une seule valeur pour la position de sortie correspondante, une convolution de transposition fait essentiellement le contraire. Pour une convolution de transposition, nous prenons une valeur unique de la carte des caractéristiques à basse résolution et nous multiplions tous les poids de notre filtre par cette valeur, et on projette ces valeurs pondérées dans la carte des caractéristiques de sortie [11]. La figure 2.7 montre un exemple de cette opération.

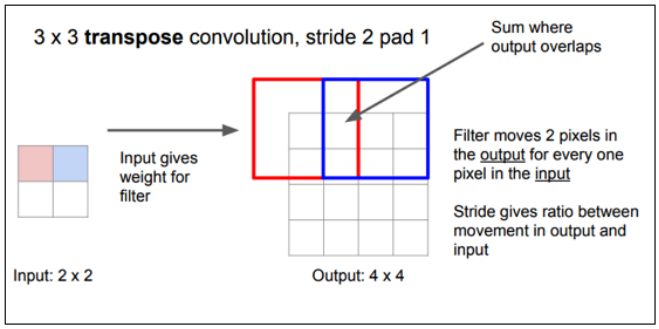


FIGURE 2.7 – Opération de convolutions transposées [11].

2.3.2 Segmentation des scènes

La segmentation d'images est une tâche importante en vision par ordinateur, qui consiste à diviser une image en plusieurs segments ou régions, chaque région représentant un objet ou une partie d'un objet.

La segmentation sémantique consiste à diviser une image en plusieurs segments selon les catégories d'objets ou de classes auxquels ils appartiennent. En d'autres termes, la segmentation sémantique fournit une étiquette à chaque pixel de l'image en fonction de la catégorie d'objet à laquelle il appartient.

La segmentation d'instance, quant à elle, consiste également à diviser une image en plusieurs segments, mais cette fois-ci, chaque segment correspond à une instance spécifique d'un objet plutôt qu'à sa catégorie. Cela signifie que chaque instance d'un objet est étiquetée différemment dans l'image, même si elle appartient à la même catégorie.

Pour mieux comprendre la différence, prenons l'exemple d'une image contenant plusieurs

personnes (figure 2.8). En segmentation sémantique, tous les pixels qui représentent des personnes seraient étiquetés avec la même catégorie "personne". En revanche, en segmentation d'instance, chaque personne serait étiqueté individuellement, avec un identifiant unique pour chaque instance.

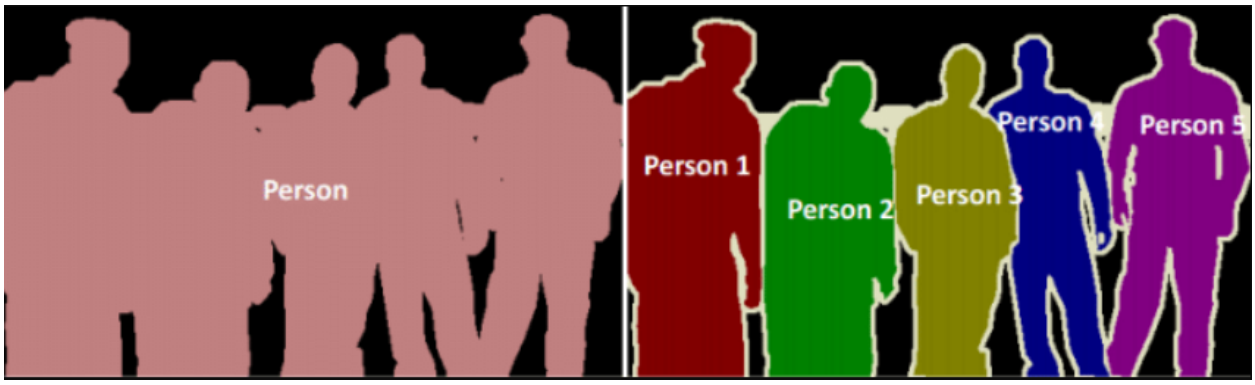


FIGURE 2.8 – Segmentation sémantique vs segmentation d'instance.

2.3.3 Métriques d'évaluation des modèles de la segmentation

Idéalement, un modèle devrait être évalué sous multiples aspects, tels que la précision quantitative, la vitesse de traitement et l'espace de stockage nécessaire (empreinte mémoire). L'évaluation des vitesses peut être délicate puisqu'elle dépend du matériel et de l'expérimentation, mais c'est un facteur important dans les applications en temps réel. De plus, l'empreinte mémoire est aussi importante, spécialement si le modèle est prévu pour les petits appareils ayant une capacité de mémoire limitée. Cependant, la plupart des travaux de recherche réalisés jusqu'à présent se concentrent sur les mesures pour évaluer la précision du modèle. Nous résumons ci-dessous les mesures les plus célèbres pour évaluer l'exactitude des algorithmes de segmentation. Bien que les mesures quantitatives soient utilisées pour comparer différents modèles sur des points de référence, la qualité visuelle des résultats des modèles est également importante pour décider quel est le meilleur modèle (car l'homme est le consommateur final d'un grand nombre de modèles développés pour les applications de vision par ordinateur) [12].

— **La précision des pixels** (Pixel accuracy (PA)) trouve simplement le rapport entre les pixels correctement classés et le nombre total de pixels. Pour les classes K + 1 (K

classes et une comme arrière-plan), la précision des pixels est définie comme suit :

$$PA = \frac{\sum_{i=0}^{K} p_{\text{ii}}}{\sum_{i=0}^{K} \sum_{j=0}^{K} p_{\text{ij}}},$$

où p_{ij} est le nombre de pixels de la classe i prédite comme classe j.

— **La précision moyenne des pixels** (Mean Pixel Accuracy (MPA)) est la version étendue de PA, dans laquelle le rapport des pixels corrects est calculé par classe et ensuite moyenné sur le nombre total de classes, comme suit :

$$MPA = \frac{1}{K+1} \sum_{i=0}^{K} \frac{p_{\text{ii}}}{\sum_{j=0}^{K} p_{\text{ij}}}.$$

— **L'intersection sur Union** (Intersection over Union (IoU) ou l'indice Jaccard) est l'une des mesures les plus couramment utilisées dans la segmentation sémantique. Il est défini comme la zone d'intersection entre l'image prédite et la vraie image (ground truth), divisée par la zone d'union entre elles :

$$IoU = J(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

où A et B désignent la vraie image et les prévisions respectivement. IoU est compris entre 0 et 1.

— **Mean-IoU** est une autre mesure, définie comme la moyenne IoU de toutes les classes. Elle est largement utilisée pour rendre compte des performances des algorithmes de segmentation modernes.

— **La précision** (precision), **le rappel** (recall) et **le score F1** sont des mesures connues pour rendre compte de la précision de nombreux modèles classiques de segmentation d'images. La précision et le rappel peuvent être définis pour chaque classe comme suit :

$$precision = \frac{TP}{TP + FP} \quad , \quad recall = \frac{TP}{TP + FN},$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

où TP, FP et FN sont définie comme suit :

- Les vrais positifs (True Positives TP) : le pixel appartient à la classe et il est classifié correctement ;
- Les faux positifs (False Positives FP) : le pixel n'appartient pas à la classe et il n'est pas classifié correctement ;
- Les faux négatifs (False Negatives FN) : le pixel appartient à la classe et il n'est pas classifié correctement.

— **Le coefficient Dice** est une autre mesure répandue pour la segmentation d'images, qui peut être défini comme le double de la zone de chevauchement des masques prédits et ceux originaux, divisé par le nombre total de pixels dans les deux images. Le coefficient Dice est très similaire à l'IoU :

$$Dice = \frac{2|A \cap B|}{|A|+|B|}.$$

2.4 Modules de perception

Comme nous l'avons vu dans le chapitre précédent, les modules de perception dans une voiture autonome sont les éléments les plus critiques pour assurer une conduite sûre et efficace. Basé sur des architectures d'apprentissage profond déstinés à la segmentation, ces module permetent à la voiture de comprendre son environnement comme un être humain. Cela signifie qu'elle doit être capable de percevoir les objets et les obstacles dans son champ de vision, ainsi que de comprendre leur nature et leur comportement.

La précision et **la rapidité** sont des aspects clés de ces modules de perception. **En temps réel**, la voiture doit être en mesure de traiter les données des capteurs pour prendre des décisions rapides et précises. Elle doit être capable de détecter et d'identifier rapidement les piétons, les cyclistes, les autres véhicules et les objets fixes. Elle doit également être capable de prédire leur trajectoire et leur comportement futur pour anticiper les actions nécessaires pour éviter une collision.

Discussion

Nous avons remarqué que la plupart des architectures que nous avons étudiées pour les véhicules autonomes privilégient la précision du modèle entraîné, souvent au détriment du temps de réponse. Pourtant, le temps de réponse est aussi crucial que la précision dans ce domaine. En effet, un véhicule autonome qui a une bonne compréhension de son environnement mais qui ne peut pas réagir en temps réel risque de prendre des décisions tardives, pouvant causer des conséquences catastrophiques.

En effet, les architectures idéales pour les véhicules autonomes sont celles qui parviennent à trouver un compromis optimal entre le temps de réponse et la précision, car il est crucial d'avoir une bonne compréhension de l'environnement tout en étant capable de prendre des décisions en temps réel. Notre travail a pour ambition de contribuer à cette recherche d'équilibre entre ces deux éléments clés.

Afin de trouver cet équilibre, nous avons étudié les architectures qui ont réussi à l'atteindre dans le domaine de la segmentation en général. Cette recherche, qui n'a pas été facile du fait de la présence de nombreuses solutions qui ont donné de bons et même d'excellents résultats dans ce domaine, nous a conduit à choisir deux architectures à savoir Yolact [13] et solo [14].

2.5 Conclusion

Ce chappitre a fourni une introduction essentielle à cette approche révolutionnaire de l'apprentissage automatique. Nous avons exploré divers aspects, notamment la catégorisation des algorithmes d'apprentissage, en mettant l'accent sur les réseaux de neurones profonds, tels que les réseaux de neurones convolutifs (CNN). En examinant l'architecture d'un CNN, nous avons passé en revue les différentes couches, telles que les couches de convolution, d'activation, de pooling, entièrement connectées et de sur-échantillonnage. Nous avons également abordé des applications spécifiques du Deep Learning, comme la segmentation des scènes, en soulignant l'importance des métriques d'évaluation pour mesurer les performances des

modèles de segmentation.

Chapitre 3

Conception et modélisation

3.1 Introduction

Revenons à l'architecture que nous avons abordée dans le chapitre 1. En effet, notre recherche se concentre sur la détection des objets (figure 3.1). Dans la suite de ce chapitre, nous examinerons la conception de notre système.

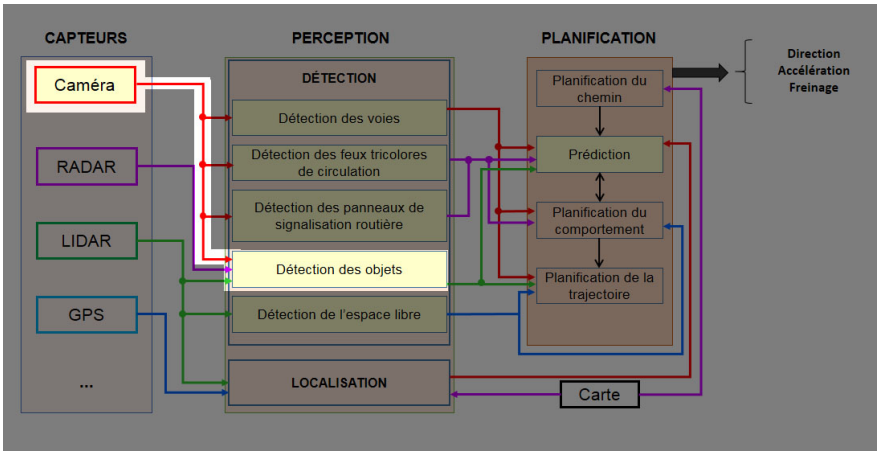


FIGURE 3.1 – Système de Detection

3.2 Conception

Le fonctionnement de notre système est représenté dans la figure 3.2. La conception est simple : notre système doit recevoir en temps réel une image provenant des caméras installées

sur le véhicule. Son rôle principal est de détecter différents objets tels que les voitures, les camions, etc. De plus, notre système doit être capable de localiser avec précision la position, l'environnement (circonférence) et la nature de chaque objet détecté. Les informations ainsi obtenues doivent être transmises aux autres modules dans un format bien défini. Pour faciliter l'évaluation de notre système, celui-ci doit également fournir l'image d'origine avec le masque des objets colorés (comme le montre la figure).

Il est essentiel que tout ce processus se déroule en temps réel. C'est pourquoi nous avons choisi deux architectures, YOLACT et SOLOv2, qui ont déjà démontré de bons résultats dans d'autres domaines d'application.

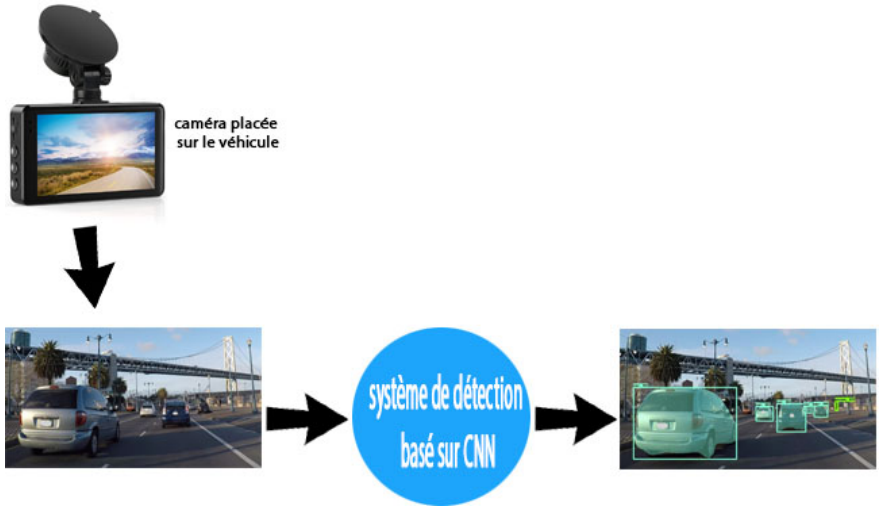


FIGURE 3.2 – Conception du Système de Detection.

3.3 Architecture YOLACT

YOLACT (figure 3.3) est une architecture de segmentation d'instance en temps réel développée en 2019 par Daniel Bolya et al. . Cette architecture a remporté le prix du meilleur papier lors de la Conférence sur les technologies de reconnaissance de formes et d'analyse d'images (CVPR) en 2019.

En effet, les archirectures ordinaires, comme Mask R-CNN, doivent effectuer la segmentation pour chaque instance d'objet dans une image séparément. Cela signifie que si une image contient plusieurs instances d'objets, le réseau doit effectuer la segmentation pour chaque

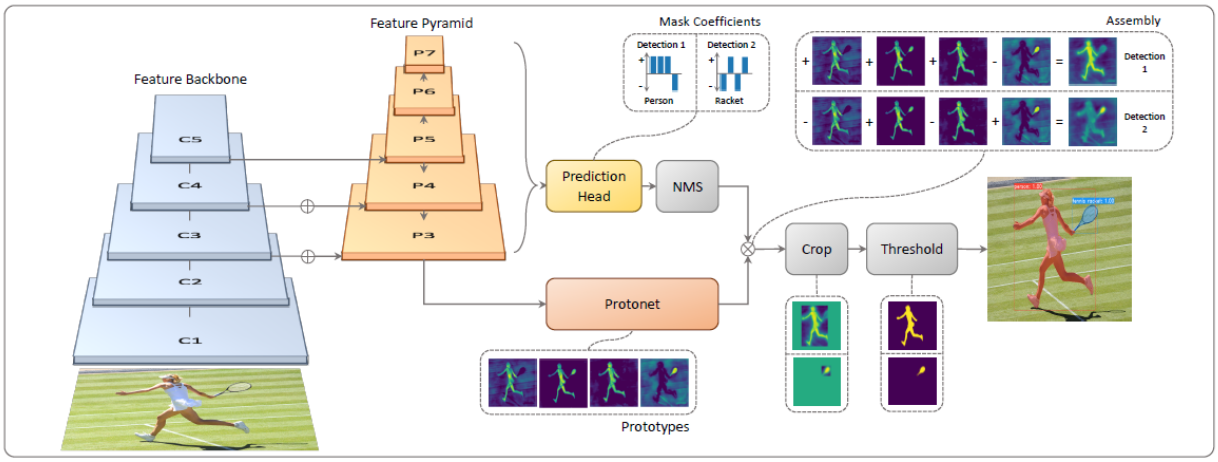


FIGURE 3.3 – Architecture YOLACT [13]

instance individuelle, ce qui peut être très coûteux en termes de temps de calcul.

Cependant, YOLACT est une méthode de segmentation à une seule étape. Le réseau ne nécessite qu'une seule passe pour segmenter toutes les instances d'objets dans une image. Cela est possible en générant d'abord un ensemble de masques prototypes globaux sur toute l'image, qui sont des représentations de masques réutilisables et indépendantes des catégories d'objets. Ensuite, des coefficients de combinaison linéaires sont prédits pour chaque instance d'objet, qui sont utilisés pour assembler les masques prototypes en masques d'instance individuels.

on peut résumé son fonctionnement comme suit :

1. **Extraction de caractéristiques** : elle utilise une version modifiée de ResNet-101 pour extraire des caractéristiques de l'image d'entrée. Cela se fait en passant l'image à travers plusieurs couches de convolution et de pooling.

2. **Génération de masques prototypes** : elle génère un ensemble de masques prototypes à partir des caractéristiques extraites de l'image d'entrée. Ces masques prototypes sont générés à l'aide d'un réseau de neurones entièrement convolutif (Fully Convolutional Network, FCN) qui est entraîné pour prédire des masques pour chaque instance possible.

3. **Prédiction des coefficients de masque** : Pour chaque instance, elle prédit un ensemble de coefficients de masque qui indiquent comment combiner les masques proto-

types pour générer un masque précis pour cette instance.

4. **Prédiction des caractéristiques de boîte englobante** : elle prédit également les caractéristiques de boîte englobante pour chaque instance, y compris sa position, sa taille et sa forme.
5. **Calcul de masques d'instance** : Les masques d'instance finaux sont calculés en combinant les masques prototypes avec les coefficients de masque prédits pour chaque instance. Cette étape permet de générer des masques d'instance précis pour chaque objet détecté.
6. **Post-traitement** : Enfin, des étapes de post-traitement sont appliquées pour améliorer les résultats, notamment en éliminant les doublons et en affinant les contours des masques d'instance.

Evaluation

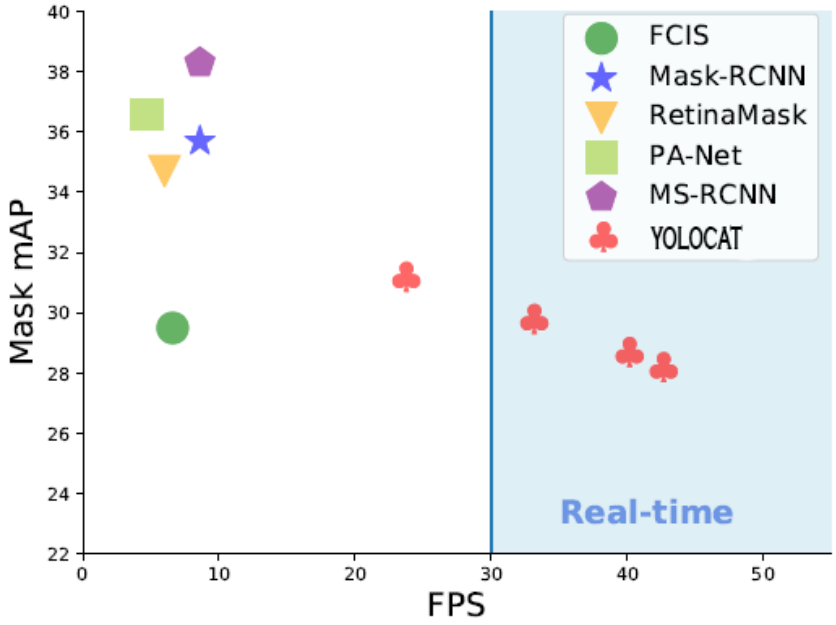


FIGURE 3.4 – comparaison entre Équilibre entre vitesse et performance pour différentes méthodes de segmentation d'instances par rapport YOLACT[13]. (Il est important de noter que les valeurs de la colonne représentent les mesures de précision, tandis que les lignes correspondent au nombre d'images (frame) traitées par seconde (FPS).

L'évaluation de YOLACT a été effectuée en utilisant le jeu de données COCO (*Common Objects in Context*)[15] qui est un ensemble de données largement utilisé pour l'entraînement

et l'évaluation des modèles de détection d'objets, de segmentation sémantique et d'autres tâches liées à la compréhension et à l'analyse d'images. Il comprend plus de 200 000 images et contient plus d'un million d'annotations d'objets répartis sur 80 catégories différentes, ce qui en fait une ressource précieuse pour le développement et l'évaluation des algorithmes de vision par ordinateur.

YOLACT a été entraîné et évalué sur le dataset COCO, ce qui a conduit aux résultats présentés dans la figure 3.4. Cette figure illustre la relation entre la précision du modèle et le nombre de frames traitées par seconde (FPS). Les résultats démontrent clairement que YOLACT parvient à atteindre une précision élevée en temps réel, ce qui représente un accomplissement sans précédent par rapport à toutes les autres architectures existantes antérieures à YOLACT. Ces performances exceptionnelles soulignent l'efficacité et la puissance de YOLACT en matière de détection et de segmentation d'objets dans des scènes en temps réel, ce qui justifie son choix en tant qu'objet d'étude dans notre domaine de recherche.

3.4 Architecture SOLO

SOLO [14] (Segmenting Objects by Locations) est une architecture de segmentation d'instances proposée en 2020 par Xinlong Wang et al. qui se distingue par son approche de détection par points. Elle vise à prédire les masques d'instances et les catégories d'objets en utilisant des têtes de détection spécifiques. En effet, SOLO divise l'image en une grille régulière de points appelée "grid". Chaque point de la grille est responsable de la prédiction d'un ensemble de masques d'instances, généralement associés à différentes tailles ou formes. Les points de la grille servent de référence spatiale pour prédire les masques d'instances correspondants (figure 3.5).

Effectivement, bien que cette architecture ait obtenu de bons résultats, elle présente quelques limitations, notamment en termes de temps d'exécution et de consommation d'espace mémoire considérables nécessaires pour l'entraînement ou la production de résultats.

En réponse à cela, les chercheurs ont proposé, au cours de la même année, une deuxième version de SOLO appelée SOLOv2 [16]. Cette nouvelle version a réussi à surpasser toutes les

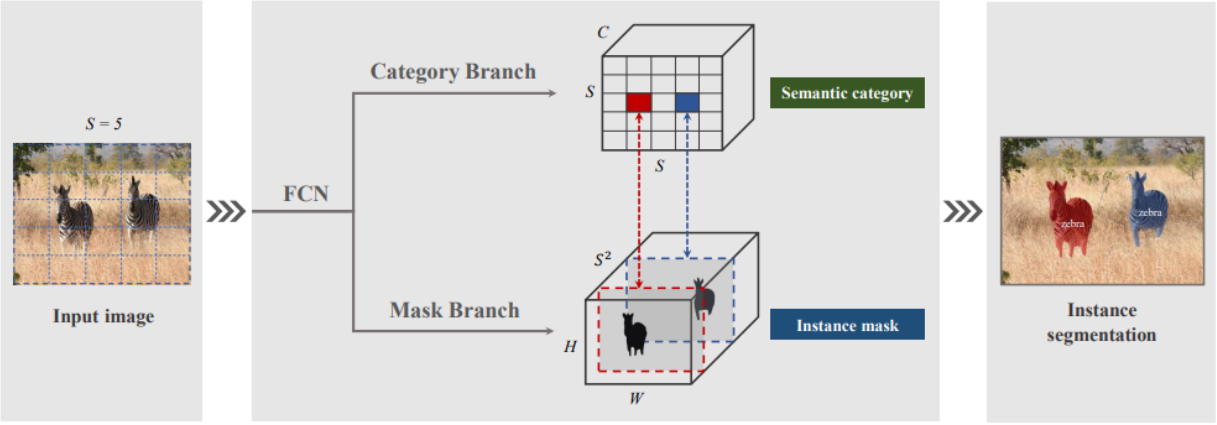


FIGURE 3.5 – représentation simplifié de l'architecture SOLO[14].

limitations de la première version et a démontré une amélioration significative de la précision en temps réel par rapport aux architectures contemporaines, notamment YOLACT.

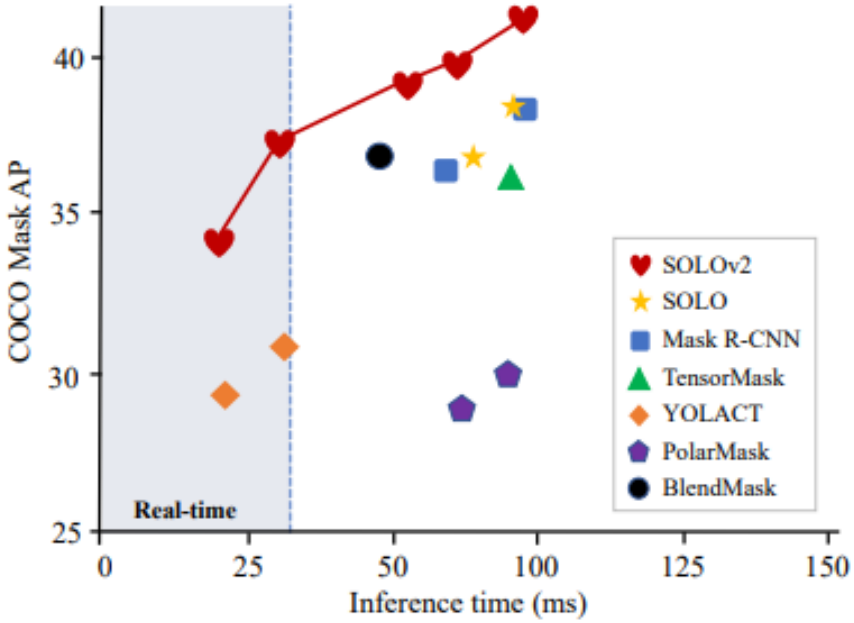


FIGURE 3.6 – comparaison entre Équilibre entre vitesse et performance pour différentes méthodes de segmentation d'instances par rapport SOLOv2[16].(Dans la figure, la colonne représente également la précision, tandis que la ligne représente le temps nécessaire pour traiter une seule image en temps réel, c'est-à-dire moins de 30 ms par image.)

Evaluation

Dans la figure 3.6, la précision obtenue par les différentes architectures est représentée en fonction du temps nécessaire pour traiter une seule image. Pour illustrer la lecture de ce

diagramme, prenons l'exemple de 25 ms. On peut constater que l'architecture SOLOv2 a atteint une précision d'environ 35 lorsqu'elle traite chaque image en 25 ms. Il est également remarquable que seules les architectures SOLOv2 et YOLACTont pu atteindre une précision satisfaisante en temps réel, contrairement aux autres architectures, y compris la première version de SOLO.

3.5 Méthodologie de recherche et plan d'action

En se basant sur les informations précédentes et compte tenu des résultats impressionnants obtenus par ces deux architectures récentes, il est important de noter que leur utilisation dans le domaine des véhicules autonomes est encore limitée. C'est pourquoi nous avons décidé d'exploiter ces architectures pour concevoir notre propre module de perception.

3.5.1 Présentation de Dataset BDD100k

Nous ne pouvons pas présenter notre approche sans évoquer le dataset BDD100k[17]. Il s'agit d'un ensemble de données utilisé dans la recherche en vision par ordinateur et en intelligence artificielle, notamment dans le domaine de la conduite autonome. BDD100k contient 100 000 images de haute résolution annotées provenant de caméras embarquées dans des véhicules, fournissant des informations sur les objets présents dans les images tels que :

1. Piéton
2. Cycliste
3. Voiture
4. Camion
5. Bus
6. Train
7. Motocyclette
8. Vélo

La détection et la segmentation de ces huit classes sont des fonctionnalités clés de notre travail. Ces informations jouent un rôle crucial dans l'entraînement et l'évaluation des algorithmes de détection et de segmentation d'objets dans les environnements routiers. Le dataset BDD100k représente une ressource inestimable pour le développement de modèles performants en matière de perception visuelle destinés aux systèmes de conduite autonome.

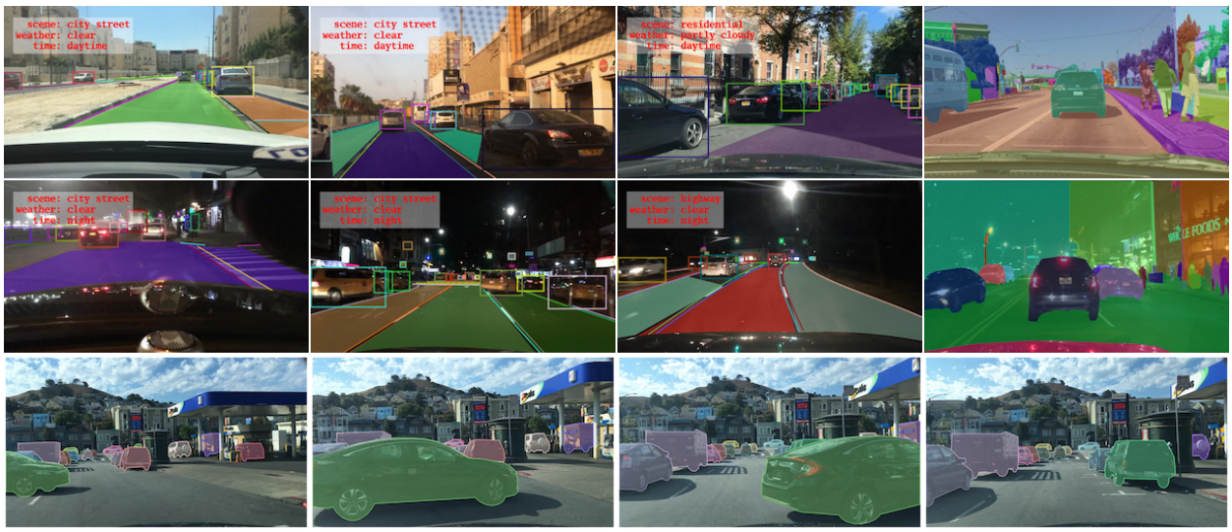


FIGURE 3.7 – Illustration d'images extraites du dataset BDD100k[17].

Transfert d'apprentissage

Le transfert d'apprentissage [18] (ou transfer learning en anglais) est une technique d'apprentissage automatique qui permet de transférer les connaissances acquises lors de l'apprentissage d'une tâche à une autre tâche similaire. Au lieu de former un modèle à partir de zéro pour chaque tâche, le transfert d'apprentissage consiste à utiliser un modèle pré-entraîné sur une tâche pour initialiser le réseau de neurones pour une autre tâche. Cela permet de gagner du temps et des ressources en utilisant des modèles déjà entraînés sur des données volumineuses et complexes, tout en obtenant de meilleures performances sur la tâche cible avec moins de données d'apprentissage.

3.5.2 Plan d'action

Notre approche consiste à entraîner et évaluer les deux architectures SOLOv2 et YOLACT sur le dataset BDD100K. Pour ce faire, nous avons élaboré un plan de travail en plusieurs étapes :

1. Nous allons utiliser un modèle YOLACT pré-entraîné [1] sur le dataset COCO et appliquer un transfert d'apprentissage en utilisant le dataset BDD100K. Cette approche est justifiée par le fait que le dataset COCO contient plusieurs catégories d'objets, y compris ceux qui sont nécessaires pour les véhicules autonomes (voitures, personnes, etc.). En conséquence, cela devrait permettre d'obtenir un modèle plus précis.
2. Pour SOLOv2, nous allons procéder de la même manière en utilisant un modèle pré-entraîné disponible sur GitHub [2].

3.6 Conclusion

Dans ce chapitre, nous avons exploré les architectures YOLACT et SOLOv2 pour la segmentation d'instances dans le contexte des véhicules autonomes. Nous avons examiné en détail le fonctionnement de ces architectures, leur approche novatrice et leurs performances impressionnantes en termes de détection et de segmentation d'objets.

Nous avons également souligné l'importance de trouver un équilibre entre la précision et le temps de réponse dans la conception des modules de perception des véhicules autonomes. Les architectures SOLO et SOLOv2 ont démontré leur capacité à atteindre cet équilibre en offrant à la fois des performances élevées et une exécution en temps réel.

Nous avons formulé notre plan d'action en utilisant SOLOv2 et YOLACT comme base pour développer notre propre module de perception. Nous avons proposé de collecter et de préparer les données à partir du dataset BDD100K, d'explorer les architectures, de les adapter aux spécificités des véhicules autonomes, d'appliquer le transfert d'apprentissage et d'évaluer les modèles résultants.

1. https ://github.com/open-mmlab/mmdetection/tree/main/configs/yolact
2. https ://github.com/open-mmlab/mmdetection/tree/main/configs/solov2

Le dataset BDD100K a été présenté comme une ressource précieuse pour l'entraînement et l'évaluation des algorithmes de vision par ordinateur dans le domaine de la conduite autonome. Nous avons également souligné l'importance du transfert d'apprentissage, qui permet d'utiliser les connaissances acquises à partir d'une tâche pour améliorer les performances dans une autre tâche similaire.

Enfin, nous avons évoqué notre méthodologie d'évaluation, qui comprend la comparaison des modèles obtenus avec les modèles initiaux, la comparaison entre SOLOv2 et YOLACT, ainsi que l'évaluation qualitative sur des vidéos réelles de dashcam.

Chapitre 4

Évaluation et Interprétation des Résultats

4.1 Introduction

Dans le chapitre antérieur, nous avons exposé l'objectif de notre recherche ainsi que le plan d'action qui décrit l'approche adoptée pour illustrer notre travail .Ce chapitre présente les résultats obtenus dans le cadre de notre étude, qui vise à évaluer l'efficacité des modèles pré-entraînés YOLACT et SOLOV2 dans un contexte des véhicules autonomes.

En premier lieu, nous décrirons les modalités pratiques que nous avons suivies pour entraîner ces modèles, en mettant en évidence les paramètres et les techniques spécifiques utilisées. Nous détaillerons également les jeux de données utilisés pour l'entraînement, en insistant sur leur composition et leur pertinence par rapport à notre objectif de recherche.

Ensuite, nous présenterons les résultats quantitatifs obtenus, en analysant les mesures de performance. Nous comparerons également ces résultats avec ceux d'autres approches existantes, afin d'évaluer l'amélioration apportée par nos modèles.

En parallèle, nous fournirons une évaluation qualitative des résultats, en mettant l'accent sur la qualité de la segmentation des objets, la précision des contours et la capacité des modèles à identifier et à suivre les objets dans des scénarios complexes. Des exemples visuels

seront également présentés pour illustrer ces observations.

Enfin, ce chapitre se clôturera par une discussion des résultats obtenus, mettant en évidence les forces et les limites de nos modèles pré-entraînés. Nous examinerons les facteurs influençant les performances, tels que la taille de l'échantillon d'entraînement, les paramètres d'apprentissage et les choix architecturaux. En nous appuyant sur ces résultats, nous formulerons également quelques perspectives pour de futures recherches et améliorations potentielles.

4.2 Présentation des outils

FIGURE 4.1 – Logo MMDetection[19].

Avant de présenter nos résultats, il est essentiel de discuter des outils utilisés dans notre travail. Notre recherche s'appuie principalement sur MMDetection [19], une boîte à outils de détection d'objets qui offre une large gamme de méthodes pour la détection d'objets et **la segmentation d'instances**. Au fil du temps, MMDetection est devenue une plateforme unifiée qui fournit non seulement des codes d'entraînement et d'inférence, mais également des poids pré-entraînés pour plus de 200 modèles de réseaux de neurones, y compris les architectures clés que nous avons utilisées, telles que YOLACT et SOLOv2.

4.3 Préparation des données

Avant de commencer le processus d'entraînement, il convient de noter que les modèles dédiés pour la segmentation d'instances disponibles dans le framework MMDetection ont été préalablement entraînés. Afin de ré-entraîner l'un de ces modèles avec le Dataset BDD100K en utilisant les scripts d'entraînement et de validation de MMDetection, il est impératif que ce jeu de données soit formaté selon une format bien définié.

Le format d'annotation que MMDetection exigé est largement utilisé dans le domaine de la vision par ordinateur pour annoter des objets dans le cadre de tâches de détection et de segmentation d'objets. Sa popularité résulte de sa structure bien définie et de sa compatibilité avec de nombreux frameworks d'apprentissage automatique.

```
{
    "images": [
        {
            "id": 1,
            "width": 800,
            "height": 600,
            "file_name": "image1.jpg"
        }
    ],
    "categories": [
        {
            "id": 1,
            "name": "person"
        }
    ],
    "annotations": [
        {
            "id": 1,
            "image_id": 1,
            "category_id": 1,
            "bbox": [10, 20, 100, 200],
            "segmentation": [...],
            "area": 20000
        }
    ]
}
```

Listing 4.1 – Format des annotations du Dataset COCO

Le format est définit comme suit :

— Le champ ***images*** contient une liste d'images annotées, où chaque image est identifiée

par un identifiant unique (*id*), sa largeur (*width*), sa hauteur (*height*) et son nom de fichier (*file_name*).

— Le champ ***categories*** représente les catégories d'objets annotées, avec pour chaque catégorie un identifiant unique (*id*), un nom (*name*) et éventuellement une super-catégorie à laquelle elle appartient (*supercategory*).

— Le champ ***annotations*** regroupe les annotations d'objets, définies par un identifiant unique (*id*), l'identifiant de l'image à laquelle elles sont associées (*image_id*), l'identifiant de la catégorie d'objet (*category_id*), les coordonnées du rectangle englobant (*bbox*) spécifiant la position et la taille de l'objet, les informations de segmentation décrivant la forme de l'objet (*segmentation*), ainsi que la superficie de l'objet (*area*).

Il est important de souligner que le champ ***segmentation*** permet de représenter les masques binaires des objets. Ces masques sont définis comme une liste de coordonnées qui spécifient les contours de l'objet. Selon la forme de l'objet, ces coordonnées peuvent prendre différentes représentations, telles que des contours fermés ou des polygones définis par une séquence de points.

En raison des différences majeures entre le format du dataset BDD100K et cette format, nous avons utilisé les scripts de transformation fournis sur le dépôt officiel de BDD100K sur GitHub (avec certaines modifications) pour préparer deux fichiers d'annotations distincts : l'un dédié à l'entraînement et l'autre à la validation.

En exploitant ces scripts adaptés, nous avons pu convertir les annotations originales du dataset BDD100K dans un format compatible avec notre modèle et les outils d'entraînement que nous utilisons. Cette étape de préparation des données était essentielle pour garantir que notre modèle puisse être entraîné correctement sur le jeu de données BDD100K.

FIGURE 4.2 – Logo Python.

4.4 Outils de travail

4.4.1 Langage Python

Python est un langage de programmation largement utilisé dans le domaine de l'intelligence artificielle (IA) en raison de sa simplicité, de sa flexibilité et de sa vaste bibliothèque d'outils spécialisés. Grâce à des bibliothèques populaires comme TensorFlow, Keras et PyTorch, Python permet aux développeurs de créer et de mettre en œuvre des modèles d'apprentissage automatique et de deep learning, de développer des systèmes de vision par ordinateur, de traiter des données massives et de réaliser des tâches de traitement du langage naturel. Son écosystème riche et sa communauté active en font un choix privilégié pour explorer et innover dans le domaine de l'IA.

4.4.2 PyTorch

FIGURE 4.3 – Logo PyTorch.

PyTorch est un puissant framework open-source d'apprentissage automatique et de deep learning développé par Facebook. Il est devenu extrêmement populaire en raison de sa flexi-

bilité, de sa facilité d'utilisation et de sa performance. PyTorch offre une interface intuitive et dynamique qui facilite la construction et l'entraînement de modèles de réseaux neuronaux. Il permet également aux développeurs de tirer parti du calcul sur GPU pour accélérer les opérations de calcul intensives. PyTorch prend en charge des tâches variées telles que la classification, la détection d'objets, la segmentation, la génération de texte, et bien d'autres. Grâce à une communauté active et à des ressources en ligne abondantes, PyTorch est devenu l'un des choix les plus populaires pour la recherche et l'application pratique dans le domaine du deep learning.

4.4.3 PyCharm

FIGURE 4.4 – Logo PyCharm.

PyCharm est un environnement de développement intégré (IDE) spécialement conçu pour les développeurs Python. Développé par JetBrains, PyCharm offre une gamme complète d'outils et de fonctionnalités pour faciliter la création, le débogage et le déploiement d'applications Python. Il propose un éditeur de code avancé avec des fonctionnalités telles que la coloration syntaxique, l'achèvement automatique du code, la refactorisation, la navigation intelligente et le débogage intégré. PyCharm prend également en charge la gestion de projets, la gestion des packages, l'intégration avec des outils de versionnement, et offre des fonctionnalités de test et de profilage pour aider les développeurs à améliorer la qualité de leur code. Avec son interface conviviale et sa compatibilité avec les différentes plateformes, PyCharm est un choix populaire parmi les développeurs Python pour accroître leur productivité et

leur efficacité dans le développement d'applications.

4.5 Entrainement

Durant notre travail, nous avons utilisé une machine avec les caractéristiques suivantes :

— Système d'exploitation : Windows 11 ;

— Processeur : Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz ;

— Mémoire RAM : 24,0 Go ;

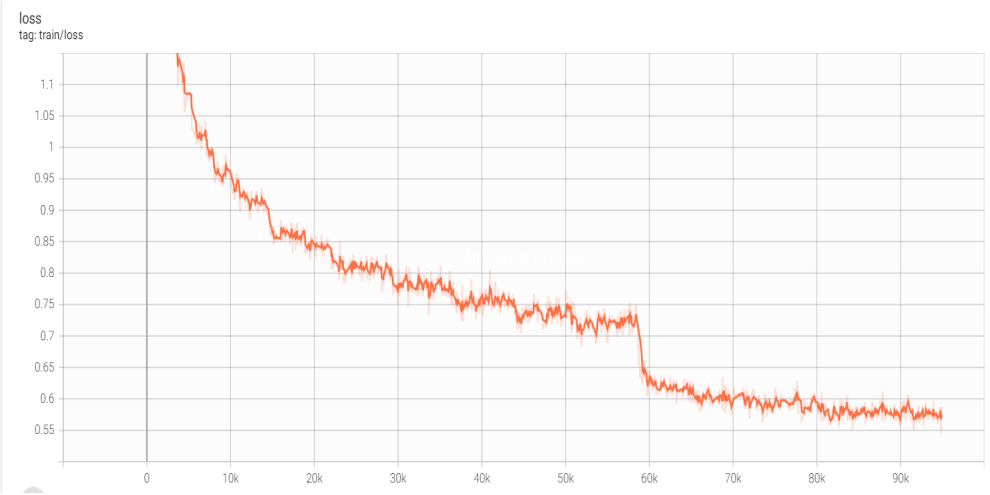
— Carte graphique : NVIDIA GeForce GTX 1050 Ti ;

— Version de Python : 3.8.16 ;

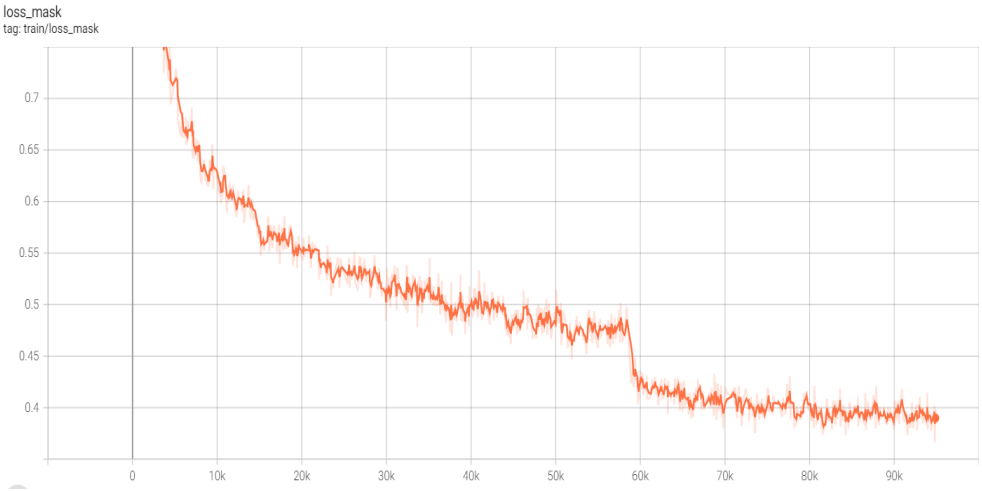
— Version de PyTorch : 2.0.0 ;

L'entraînement de YOLACT a duré 9 heures et 39 minutes pour 13 époques, tandis que pour SOLOV2, cela a pris 25 heures et 6 minutes pour 37 époques. La Figure 4.5 et la Figure 4.8 illustrent la progression des valeurs de perte (loss) et de perte de masque (loss_mask) au fil des itérations.

Il est important de noter que le terme 'Loss', ou 'Perte', désigne la valeur cumulée de l'écart calculé durant le processus d'entraînement. Cet écart représente la différence entre la prédiction du modèle et les véritables valeurs de référence. L'objectif de l'entraînement est de minimiser cette perte. Par ailleurs, la 'Loss_mask' ou 'Perte de segmentation' est un élément de la fonction de perte globale qui évalue l'écart entre les masques de segmentation prédits et ceux de référence. Cette valeur joue un rôle crucial pour évaluer les progrès dans l'entraînement des architectures conçues pour la segmentation d'instances, comme c'est le cas dans notre situation.



(a) Diagramme Loss



(b) Diagramme LossMask

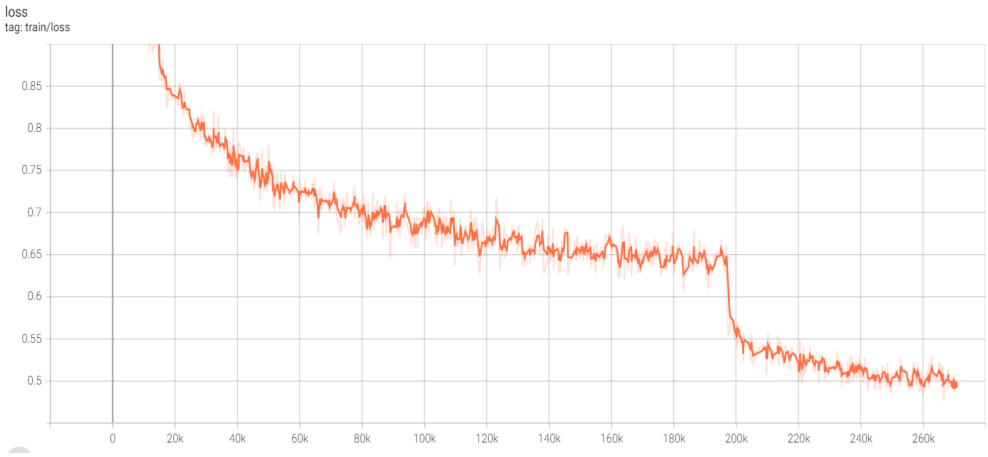
FIGURE 4.5 – Diagrames de perte de Yolact.

4.6 Evaluation

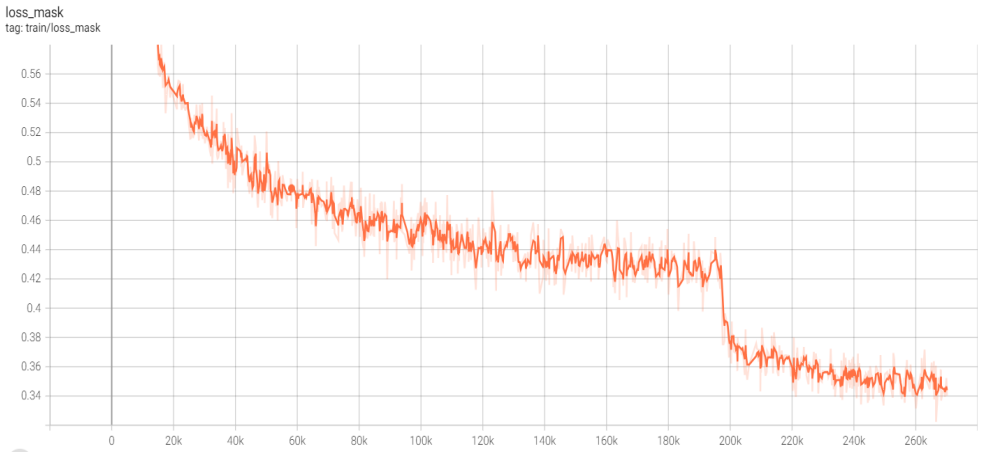
4.6.1 Evaluation quantitative

En effet, pendant le processus d'entraînement, nous avons configuré les fichiers de manière à effectuer un test de validation après chaque époque pour évaluer la précision moyenne, également connue sous le nom de Mean Average Precision (mAP) de segmentation. Cette métrique est utilisée pour évaluer la précision des modèles destinés à la segmentation. Concrètement, une valeur AP est calculée pour chaque classe d'objet, et la moyenne de ces valeurs AP pour toutes les classes donne la mAP.

Les Figures 4.7 et 4.8 illustrent cette mAP de segmentation. On peut observer une aug-



(a) Diagramme Loss



(b) Diagramme Loss

FIGURE 4.6 – Diagrames de perte de SOLOV2.

mentation quasi uniforme de la précision pour YOLACT jusqu'à l'époque 12 où elle atteint sa valeur maximale (30,4%), puis commence à diminuer légèrement lors de l'époque suivante (époque n°13). En comparaison, pour l'architecture SOLO, la précision a connu une augmentation significative pendant les 10 premières époques. Puis, une autre augmentation notable a été observée à partir de l'époque 27, atteignant sa valeur maximale de 33.9% lors de l'époque 36. Enfin, une légère dégradation a été notée lors de l'époque 37.

Les performances obtenues par l'architecture SOLOv2 se révèlent prometteuses comparativement à d'autres architectures telles que Mask R-CNN [20] et Cascade Mask R-CNN [21], dont les résultats sont illustrés dans le tableau 4.1 (ces valeurs ont été extraites du référentiel officiel de BDD100K [1]). Ces résultats démontrent également que la précision ob-

1. https://github.com/SysCV/bdd100k-models/tree/main/ins_seg

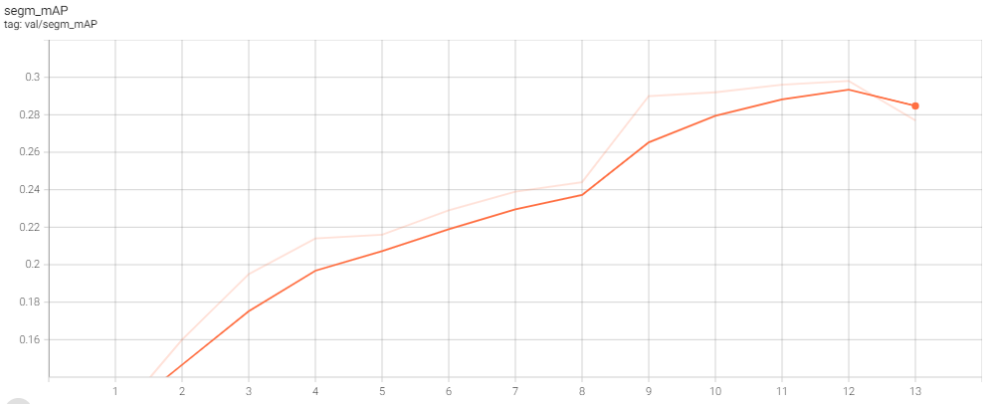


FIGURE 4.7 – Segmentation Mean Average Precision (mAP) pour Yolact.

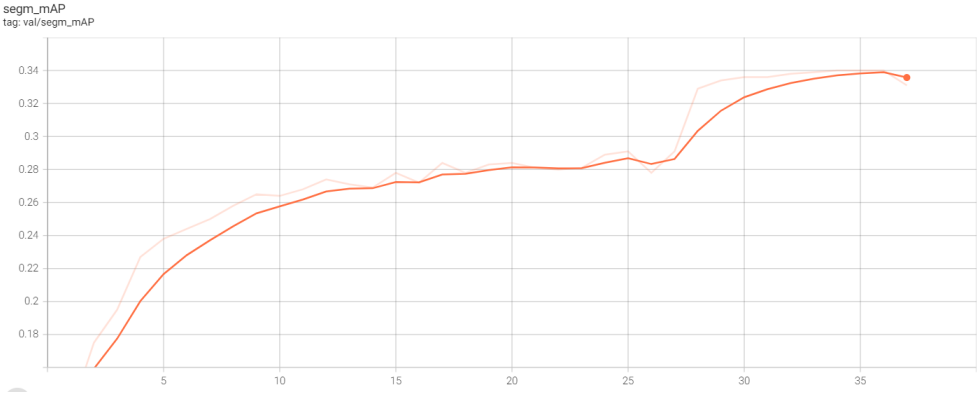


FIGURE 4.8 – Segmentation Mean Average Precision (mAP) pour SOLO.

tenue par SOLOv2 est supérieure à celle de YOLACT.

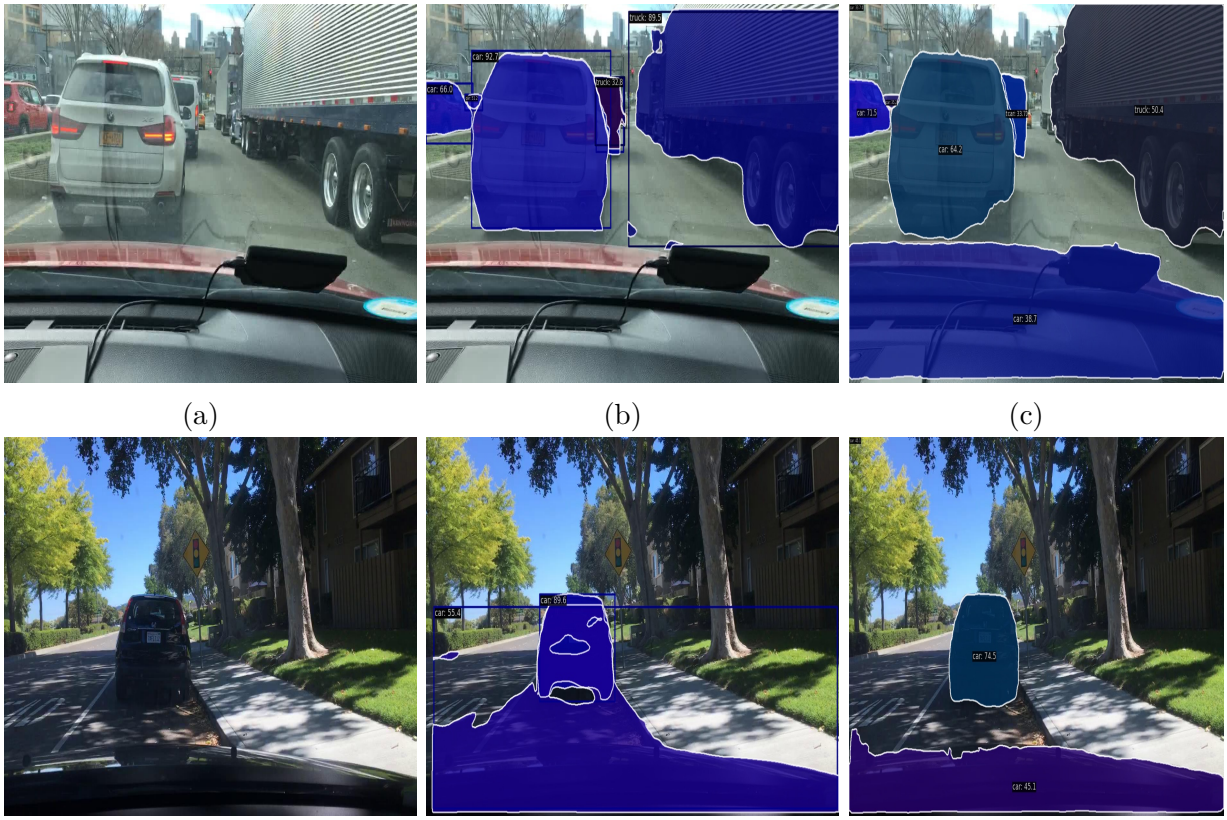
TABLE 4.1 – Comparaison des scores mAP pour différentes architectures

Architecture	mAP (%)
Mask R-CNN [20]	20.48
Cascade Mask R-CNN [21]	18.58
YOLACT	29.34
SOLOv2	33.9

4.6.2 Evaluation qualitative

Puisque l'aspect visuel est au cœur de notre projet, nous avons choisi de tester nos modèles sur l'ensemble d'images de la partie validation du jeu de données BDD100K. Ces résultats ont ensuite été comparés avec ceux obtenus à partir des modèles pré-entraînés que nous avions initialement sélectionnés.

En plus des résultats de précision numérique qui démontrent que la précision de SOLOv2 est supérieure à celle de YOLACT, les résultats visuels confirment également cette tendance. La comparaison visuelle met en évidence des performances supérieures de SOLOv2 par rapport à YOLACT.



(a) (b) (c)

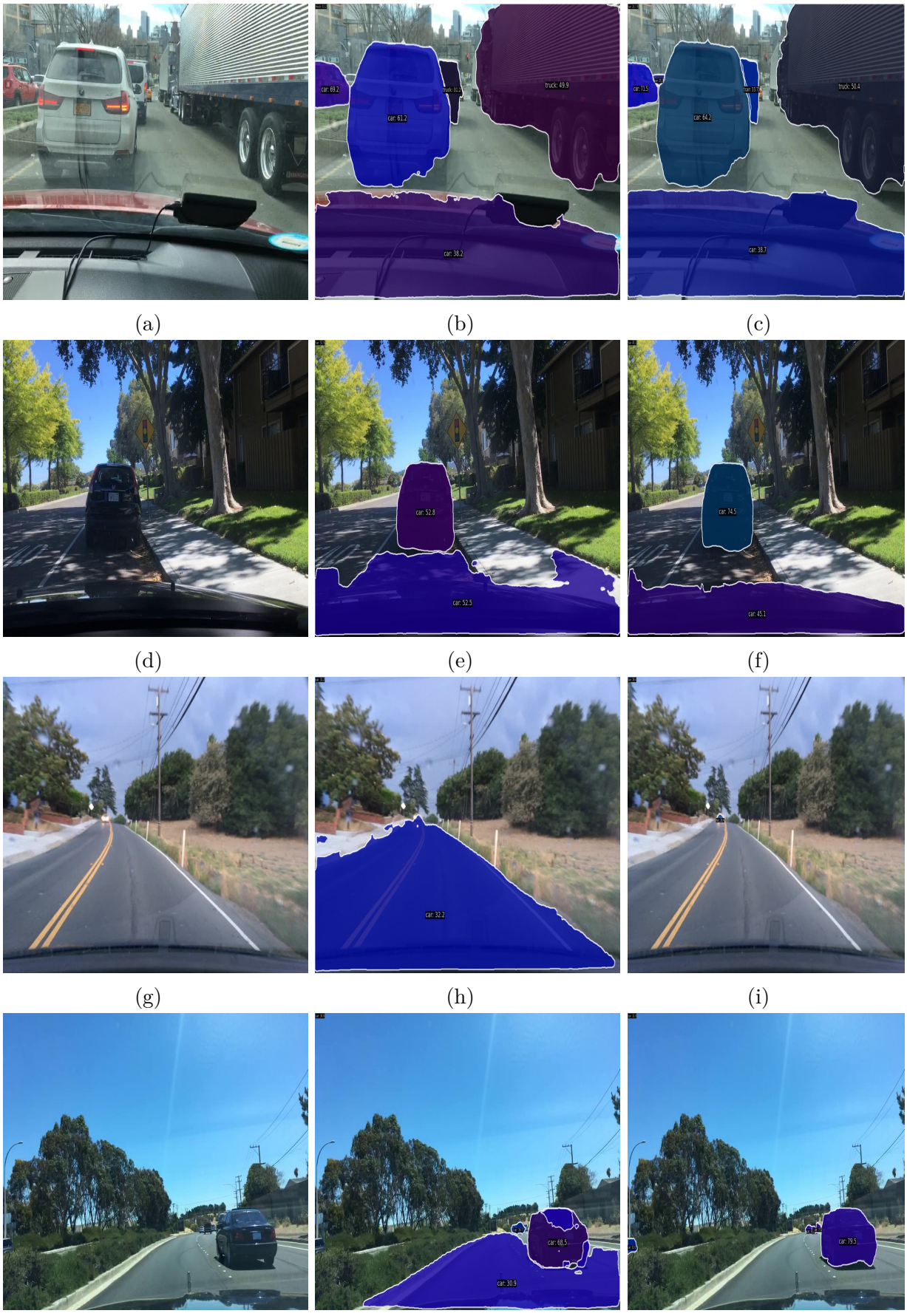
(d) image originale (e) YOLACT (f) SOLOv2

FIGURE 4.9 – Demontration de quelques erreurs de segmentation par YOLACT par rapport à SOLOv2.

Lors de notre étude sur un ensemble d'images sélectionnées pour nos tests, nous avons observé un manque de précision dans certaines images traitées par l'ancien modèle pré-entraîné SOLOv2. Cette constatation est clairement illustrée dans la figure 4.10 (première ligne), où l'on peut remarquer une légère déformation du masque généré par l'ancien modèle par rapport à celui généré par notre modèle. Bien que ce cas ne soit pas critique, il est important de souligner les autres résultats obtenus montrent les erreurs fatales produites par l'ancien modèle sont clairement visibles.

Dans la deuxième ligne, nous pouvons observer que l'ancien modèle a commis une erreur dans la segmentation de la route en la confondant avec la voiture, contrairement à notre

modèle qui a correctement identifié les limites du capot de la voiture, en tenant compte de la perspective de la caméra. Nous rencontrons un problème similaire dans la troisième ligne, où l'ancien modèle a identifié la route comme étant une voiture, tandis que notre modèle a correctement identifié la voiture malgré sa distance.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) image originale (k) ancien SOLO (l) notre modèle

FIGURE 4.10 – Demontration de quelques erreurs de segmentation par ancien modèle par rapport à notre modèle.

FIGURE 4.11 – Résultats du test sur une vidéo prise sur l'autoroute vers Alger centre

FIGURE 4.12 – Résultats du test sur une vidéo prise sur l'autoroute vers Alger centre

Dans la quatrième ligne, l'ancien modèle a complètement confondu la route et la voiture, contrairement à notre modèle qui a réussi à les distinguer correctement.

Pour évaluer l'applicabilité pratique de nos modèles, nous avons utilisé aussi des vidéos librement accessibles sur internet, prises sur les autoroutes algériennes à l'aide de caméras montées sur des véhicules. Nous avons opté pour cette méthode de validation car elle est la plus proche de la réalité, reproduisant les conditions réelles qu'un véhicule autonome peut rencontrer lors de ses déplacements, comme les vibrations du véhicule, la luminosité de l'environnement et la qualité des images capturées par la caméra.

De plus, le choix des autoroutes algériennes s'inscrit dans notre ambition de développer un module de perception pour un véhicule autonome algérien. Cela nécessite de vérifier si nos modèles sont capables de détecter les marques de véhicules présentes dans la société

algérienne, bien qu'ils aient été entraînés sur des jeux de données comportant d'autres marques. Les figures 4.11 et 4.12 présentent quelques résultats que nous avons obtenus.

4.6.3 Evaluation du temps de réponse

Rappelons que notre objectif initial était de concevoir un modèle capable de fournir des prédictions précises en temps réel. À cet effet, l'évaluation du temps de réponse représente un élément crucial de notre travail. Nous avons obtenu ces résultats en introduisant un ensemble d'images de taille prédéfinie, puis en mesurant le temps nécessaire pour que notre modèle effectue les calculs correspondants. Le tableau ci-dessus présente les résultats obtenus.

TABLE 4.2 – Comparaison des temps de réponse

resolution	1080p (1920x1080)	720p (1280x720)	480p (640x480)	360p (480x360)	240p (320x240)
temps(ms/f)	126.80	86.80	45.14	43.27	41.35

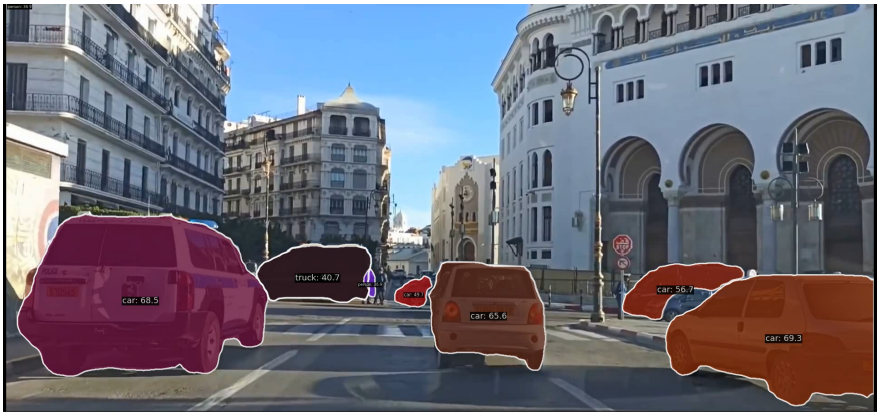


FIGURE 4.13 – Affichage des résultats sur une image de résolution 1920x1080 pixels (1080p).

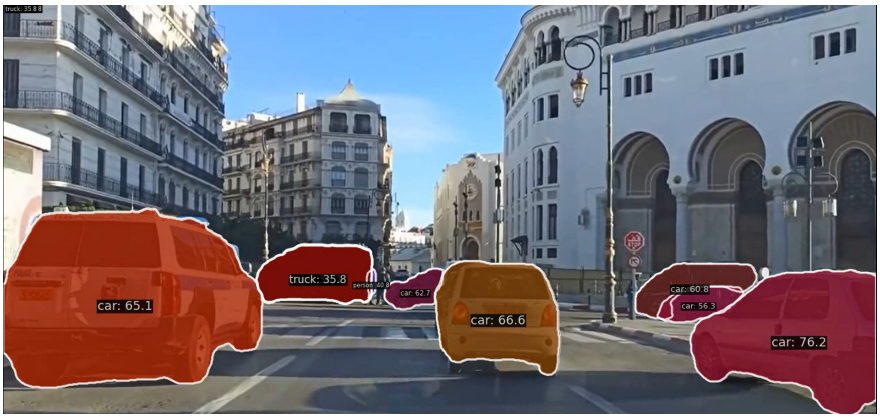


FIGURE 4.14 – Affichage des résultats sur une image de résolution 1280x720 pixels (720p).

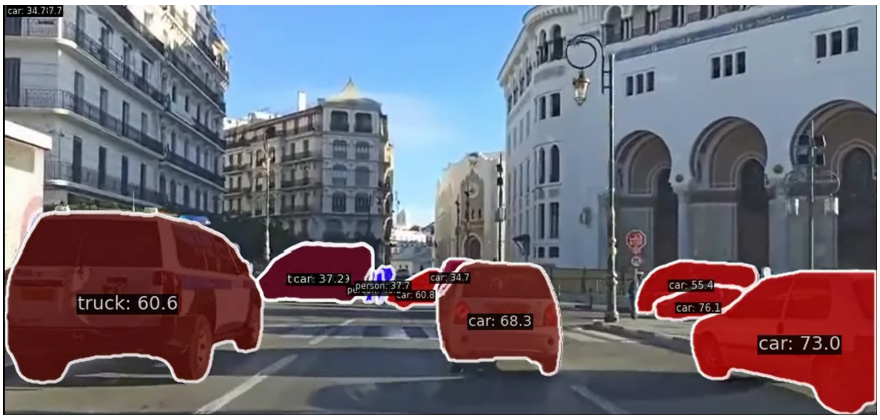


FIGURE 4.15 – Affichage des résultats sur une image de résolution 640x480 pixels (480p).

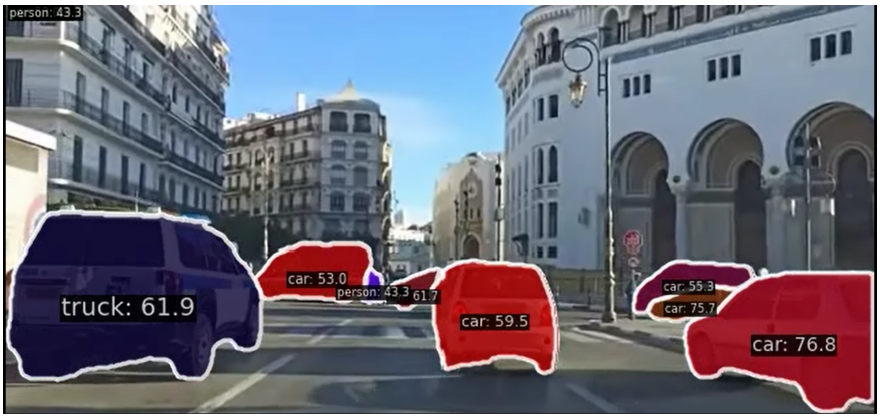


FIGURE 4.16 – Affichage des résultats sur une image de résolution 480x360 pixels (360p).

FIGURE 4.17 – Affichage des résultats sur une image de résolution 320x240 pixels (240p).

4.7 Discussion

Au vu des observations précédentes, nous pouvons affirmer que les résultats obtenus sont satisfaisants. Nos modèles ont atteint une précision appréciable tout en offrant un temps

de réponse compatible avec les exigences du temps réel. Cependant, notre solution présente quelques limites que nous souhaitons dépasser dans de futurs travaux.

D'abord, notre modèle a montré certaines lacunes de détection lors des tests que nous avons réalisés. De plus, un module de perception pratique doit être fonctionnel dans toutes les conditions qu'un véhicule autonome peut rencontrer, telles que la perception de nuit et dans des conditions météorologiques défavorables. Nous aspirons à dépasser toutes ces limites dans nos futurs travaux.

Alors que le chemin vers un module de perception pratique pour un véhicule autonome reste long, nous considérons notre réalisation comme un premier pas dans cette direction.

4.8 Conclusion

Dans ce chapitre, nous avons exposé les résultats découlant de l'entraînement et de l'évaluation de nos modèles. L'évaluation a été menée selon trois axes principaux : quantitatif, qualitatif et le temps de réponse. Pour les tests qualitatifs, nous avons opté pour l'utilisation d'images aléatoires extraites du jeu de données BDD100K et de vidéos capturées sur les autoroutes algériennes, ce qui nous a permis de tester l'efficacité des modèles dans des conditions proches de la réalité.

En matière de performance, nos modèles ont démontré une précision convenable et un temps de réponse adapté pour une utilisation en temps réel. Toutefois, nous avons identifié certains défis à relever dans le futur, en particulier ceux liés aux limites inhérentes aux modèles, comme leur performance dans des conditions de faible luminosité et dans des conditions météorologiques défavorables, qui demeurent des enjeux majeurs à résoudre.

Conclusion générale

Aujourd'hui, le véhicule autonome constitue un axe de recherche de pointe dans tout les domaines. En effet, la complexité de sa réalisation puise des attentes liées au déploiement de celui-ci. De ce fait, nous rencontrons plusieurs challenges dans le domaine de la robotique et de l'intelligence artificielle, allant des outils physiques aux outils logiciels.

A travers notre rapport, nous avons donné une vision globale des outils liés à la voiture intelligente. En effet, ils constituent la clef pour comprendre et anticiper le monde de demain, dans le sens où la réalisation de la voiture autonome du futur devrait offrir, en premier lieu, plus de sécurité et de confort c'est à dire, l'amélioration de la sécurité routière et la réduction du nombre et de la gravité des accidents, le gain de temps pour d'autres tâches, etc. En second lieu, cette voiture devrait garantir une mobilité intelligente, voire une optimisation des temps de trajet, une meilleure exploitation de l'infrastructure routière et une plus grande fluidité du trafic.

Afin de répondre aux charges de conduite, le véhicule est muni de capteurs qui remplacent les différents sens d'un conducteur. Ils permettent au véhicule, par les diverses données qu'ils fournissent, de collecter les informations qu'il devrait traiter pour comprendre l'environnement de conduite et donc prendre la bonne décision. Ainsi, nous remarquons l'importance de la perception. Cette dernière est le volet principal sur lequel s'appuient les autres volets de la plateforme de conduite (la planification et le contrôle). Nous nous sommes intéressé, dans ce travail, aux différents capteurs et modules de perception.

Nous fournissons dans ce rapport un cahier de charge des différents capteurs de perception avec une comparaison qualitative de ces derniers. Ce cahier permettra de choisir le(s)

capteur(s) adéquat(s) à une situation de conduite donnée.

Ce travail nous a aidé à visualiser les différents problèmes que rencontrent les chercheurs de ce domaine. En effet, basé sur l'apprentissage, les datasets utilisés pour entraîner un modèle donné pour une tâche donnée peuvent influencer considérablement les résultats de la perception. Pour notre cas, afin d'améliorer les résultats de détection.

De plus, le compromis du temps réel et de la précision de détection est très délicat à réaliser. Le but, pour le véhicule autonome, est d'avoir le meilleur résultat et à moindre coût d'énergie et de temps. D'où la perspective d'optimiser tous nos modules réalisés pour les intégrer dans un véhicule autonome réel.

Ainsi, ce travail nous a été très fructueux d'une part, à travers les objectifs que nous estimons avoir réalisés, puis de par les différents résultats qui, après recherche et interprétation, ont suscité notre curiosité pour des améliorations et des réalisations futures.

Bibliographie

[1] A. et Digital Aquitaine, "Le véhicule autonome en milieu urbain : Définition, enjeux et perspectives," 2017.

[2] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," 2019.

[3] "Gpio – télémètre à ultrason srf02."

[4] M. Cornick, J. Koechling, B. Stanley, and B. Zhang, "Localizing ground penetrating radar : A step toward robust autonomous ground vehicle localization," *Journal of field robotics*, vol. 33, no. 1, pp. 82–102, 2016.

[5] "Automatisiertes fahren bei der bmw group."

[6] D. Graupe, *Principles of artificial neural networks*. 2013.

[7] A. A. Oklahoma and A. Abraham, "129 artificial neural networks," 2005.

[8] W. S. Mcculloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*. Univ. Press, 1943.

[9] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," 2018.

[10] S. Muruganandham, "Semantic segmentation of satellite images using deep learning," Master's thesis, Czech Technical University in Prague ,Luleå University of Technology, 2016.

[11] J. Jordan, "An overview of semantic image segmentation.." https://www.jeremyjordan.me/semantic-segmentation/, 2018. Consulté le 03/05/2020.

[12] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning : A Survey," 2020.

[13] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact : Real-time instance segmentation," 2019.

[14] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "Solo : Segmenting objects by locations," 2020.

[15] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco : Common objects in context," 2015.

[16] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2 : Dynamic and fast instance segmentation," 2020.

[17] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k : A diverse driving dataset for heterogeneous multitask learning," 2020.

[18] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning : a friendly introduction," *Journal of Big Data*, vol. 9, p. 102, Oct. 2022.

[19] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection : Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv :1906.07155*, 2019.

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[21] Z. Cai and N. Vasconcelos, "Cascade r-cnn : High quality object detection and instance segmentation," 2019.

Résumé: De nos jours, la conduite autonome est un domaine de recherche crucial. Ce rapport met en évidence le problème de la perception des véhicules autonomes grâce à divers capteurs de perception. Nous proposons un module de perception pour la segmentation des objets sur la route, tels que les voitures, les camions, les bus, etc. Pour implémenter ce module, nous avons travaillé avec deux architectures, à savoir YOLACT et SOLOv2. Nous avons comparé ces deux architectures neuronales afin d'optimiser la détection et donc la perception lors d'une scène de conduite sur une autoroute algérienne.

LES MOTS CLES: conduite autonome, perception, capteurs de perception, modules de perception, scène de conduite.

Abstract: Nowadays, autonomous driving is a crucial research field. This report highlights the issue of perception in autonomous vehicles using various perception sensors. We propose a perception module for object segmentation on the road, such as cars, trucks, buses, etc. To implement this module, we worked with two architectures, namely YOLACT and SOLOv2. We compared these two neural architectures to optimize detection and perception during a driving scene on an Algerian highway.

KEYWORDS: autonomous driving, perception, perception sensors, perception modules, driving scene.

الملخص:

القيادة الذاتية هي مجال بحث حاسم في وقتنا الحالي. يسلط هذا التقرير الضوء على مشكلة تصور المركبات الذاتية باستخدام مجموعة متنوعة من أجهزة الاستشعار للتصور. نقترح وحدة تصور لتقسيم الكائنات على الطريق مثل السيارات والشاحنات والحافلات وغيرها .لتنفيذ هذه الوحدة، عملنا مع نوعين من الهياكل وهما YOLACT و SOLOv2 قمنا بمقارنة هاتين الهيكلين العصبيين لتحسين الكشف وبالتالي التصور خلال مشهد قيادة على طريق سريع جزائري.

الكلمات الدالة: القيادة الذاتية، الإدراك، أجهزة استشعار الإدراك، وحدات الإدراك ، مشهد القيادة.