وزارة البحث العلمي والتعليم العالي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET

DE LA RECHERCHE SCIENTIFIQUE

جامعـة عبد الحميد بن باديس مستغانم

Université Abdelhamid Ibn Badis Mostaganem

كـلـيـة الـعـلـوم و الـتـكـنـولـوجـيـا

Faculté des Sciences et de la Technologie

DEPARTEMENT DE GENIE ELECTRIQUE

MEMOIRE

Présenté pour obtenir le diplôme de

MASTER EN AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE

Par

MANSOURI Meriem *NOUR Salah Eddine*

Intitulé du sujet

Implémentation sur Raspberry Pi d'un système de détection d'obstacles par HOG-SVM et YOLOv8n.

Devant le jury composé de :

President:	MERAH Mostefa	Professeur	Université de Mostaganem
Examinateur:	ABDERRAHMANE Abdelkader	MCB	Université de Mostaganem
Encadrante:	BENDANI Djazia	MAA	Université de Mostaganem
Co encadrante:	BENCHELLAL Amel	MCB	Université de Mostaganem

Année Universitaire

2025/2026

Dédicace :

Avant toute chose, je remercie Allah, le Tout-Puissant, de m'avoir accordé la force, la patience et la persévérance nécessaires pour mener à bien ce travail.

Je dédie ce mémoire à mes très chers parents, pour leur amour inconditionnel, leurs sacrifices, leur soutien constant et leurs encouragements tout au long de mon parcours. Je leur exprime ma profonde gratitude et mon immense respect.

À mon cher frère et à mes chères sœurs, pour leur affection, leur présence et leur soutien indéfectible.

À ma chère amie, ***Douae GHERIBI****, pour son amitié sincère, sa bienveillance et les précieux moments partagés durant cette aventure.*

À mes camarades de promotion, avec qui j'ai partagé de nombreux moments d'apprentissage, d'entraide et de convivialité.

À ***Monsieur BENDANI Mohamed****, pour son accueil, sa générosité et ses conseils avisés tout au long de mon stage au sein de la société* ***SETRAM****, qui m'ont grandement aidée dans mon parcours.*

Je me remercie également moi-même pour ma détermination, ma patience et ma persévérance. Malgré les difficultés rencontrées, je n'ai jamais cessé de croire en mes capacités ni d'avancer vers mes objectifs.

Je dédie enfin ce travail à toutes les personnes qui ont contribué, de près ou de loin, à ma formation et à la réalisation de ce mémoire.

Que ce travail soit le témoignage de ma profonde gratitude envers tous ceux qui m'ont soutenue.

Meriem.

Dédicace :

Je dédie ce modeste travail à

Un homme exceptionnel, mon exemple éternel, mon soutien et ma source de joie et de bonheur, Celui qui s'est toujours sacrifié pour me voir réussir, à toi mon père. Qu'Allah le garde dans son vaste paradis.

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur, maman que j'adore.

Mes dédicaces s'adressent aussi à mes adorables Khawla et Nassima.

Ainsi qu'à mes chers frères Anes et Abed Elhamide.

A tous ceux qui me sont chers et à tous mes amis.

Salah.

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Nous adressons nos sincères remerciements à nos encadrantes, ***Madame BENCHELLAL Amel*** *et* ***Madame BENDANI Djazia****, pour leur encadrement, leurs conseils précieux, leur disponibilité et leur accompagnement tout au long de ce travail. Leur soutien nous a permis de mener à bien ce projet avec rigueur et persévérance.*

Nous remercions également l'ensemble des enseignants du département, pour la qualité de la formation dispensée tout au long de notre parcours académique, ainsi que pour les connaissances et compétences qu'ils nous ont transmises.

Nous adressons également nos sincères remerciements aux membres du jury, ***Professeur MERAH*** *et* ***Monsieur ABDERRAHMANE****, pour l'honneur qu'ils nous font en acceptant d'évaluer ce travail. Nous leur sommes reconnaissants pour le temps consacré à l'examen de ce mémoire ainsi que pour leurs remarques, suggestions et observations qui contribueront à l'amélioration de nos travaux futurs.*

Nous exprimons aussi notre reconnaissance au laboratoire de recherche ***LSS*** *pour nous avoir accueillis et offert un cadre de travail favorable à la recherche et à l'innovation.*

Nous adressons nos remerciements au responsable du laboratoire, ***Professeur ABED Mansour****, pour son soutien, ses orientations et l'intérêt qu'il a porté à notre travail.*

Enfin, nous remercions chaleureusement toutes les personnes qui ont contribué, de près ou de loin, à l'aboutissement de ce mémoire.

ملخص:

تم تطوير هذا النظام بهدف رصد ومحاكاة مختلف أنواع العوائق التي قد توثر على سلامة القيادة، مع التركيز بشكل خاص على نظام المساعدة على ركن السيارات الذاتي. يعتمد كشف وتصنيف العوائق ذات الأولوية، مثل السيارات ومخاريط المرور، على دمج تقنيات الرؤية الاصطناعية والذكاء الاصطناعي. تعتمد المقاربة الكلاسيكية الأولى على واصف HOG مدمجًا مع مصنفSVM ، في حين أن الحل الثاني، الأكثر تقدمًا وفعالية، يعتمد على نموذج التعلم العميق YOLOv8nلضمان تحديد المواقع والتعرف متعدد الكائنات بدقة وفي الوقت الفعلي. فيما يتعلق بتقدير المسافات في سياق ركن السيارات، تم اعتماد نهج يعتمد على دمج البيانات غير المتزامن أحادي وثنائي الأبعاد .(1D/2D) وبذلك يجمع النظام بين التدفق المرئي لكاميرا USB والقياسات التليمترية التكميلية الناتجة عن مستشعر الأمواج فوق الصوتية (HC-SR04)ومستشعر ميكرو-ليدار .(TF-Luna) يتيح هذا الدمج بين المستشعرات، والمدمج على منصة مضمنة Raspberry Pi 4بتصميم برمي متعدد الخيوط(Multi-thread) ، إدراكًا قويًا ومراقبة مستمرة للبيئة المباشرة للمركبة، مما يوفر دقة أعلى وموثوقية أكبر ضد إنذارات الاصطدام.

الكلمات المفتاحية :أنظمة مساعدة السائق المتقدمة(ADAS) ، الرؤية الاصطناعية، HOG-SVM، التعلم العميق، YOLOv8n، دمج البيانات D/2D1، راسبيري باي 4، مستشعر الأمواج فوق الصوتية(HC-SR04) ، ميكرو-ليدار (TF-Luna).

Résumé :

Le présent système a été développé dans le but de détecter divers types d'obstacles pouvant compromettre la sécurité lors des manœuvres de conduite, en se focalisant spécifiquement sur l'assistance au stationnement autonome. La détection et la classification des obstacles prioritaires, tels que les voitures et les cônes de signalisation, reposent sur l'intégration de techniques de vision artificielle et d'intelligence artificielle. Une première approche classique utilise le descripteur HOG associé à un classifieur SVM, tandis qu'une seconde solution, plus avancée et performante, déploie le modèle de Deep Learning YOLOv8n pour assurer une localisation et une reconnaissance multi-objets précises en temps réel. En ce qui concerne l'estimation des distances en contexte de stationnement, une approche fondée sur la fusion asynchrone de données 1D et 2D a été adoptée. Le système combine ainsi le flux vidéo d'une caméra USB avec les mesures télémétriques complémentaires issues d'un capteur à ultrasons (HC-SR04) et d'un micro-LiDAR (TF-Luna). Intégrée sur une plateforme embarquée Raspberry Pi 4 avec une architecture logicielle multi-thread, cette combinaison de capteurs permet une perception robuste et une surveillance continue de l'environnement immédiat du véhicule, offrant ainsi une précision accrue et une plus grande fiabilité face aux alertes de collision.

Mots-clés : Système d'aide à la conduite (ADAS), Vision artificielle, HOG-SVM, Deep Learning, YOLOv8n, Fusion de données 1D/2D, Raspberry Pi 4, Capteur ultrasonique (HC-SR04), Micro-LiDAR (TF-Luna).

Abstract :

The present system has been developed with the aim of detecting various types of obstacles that could compromise driving safety, specifically focusing on autonomous parking assistance. The detection and classification of priority obstacles, such as cars and traffic cones, rely on the integration of computer vision and artificial intelligence techniques. A first classical approach utilizes the HOG descriptor combined with an SVM classifier, while a second, more advanced and efficient solution deploys the YOLOv8n Deep Learning model to ensure accurate multi-object localization and recognition in real-time. Regarding distance estimation in a parking context, an approach based on the asynchronous fusion of 1D and 2D data has been adopted. The system thus combines the video stream from a USB camera with complementary telemetric measurements obtained from an ultrasonic sensor (HC-SR04) and a micro-LiDAR (TF-Luna). Integrated on a Raspberry Pi 4 embedded platform using a multi-threaded software architecture, this sensor combination enables robust perception and continuous monitoring of the vehicle's immediate environment, thereby offering increased accuracy and enhanced reliability against collision alerts.

Keywords : Advanced Driver Assistance Systems (ADAS), Computer vision, HOG-SVM, Deep Learning, YOLOv8n, 1D/2D Data fusion, Raspberry Pi 4, Ultrasonic sensor (HC-SR04), Micro-LiDAR (TF-Luna).

Table de matière

Tableau des figures

Liste des tableaux

INTRODUCTION GENERALE

Introduction générale

La mobilité urbaine représente aujourd'hui l'un des principaux défis auxquels sont confrontées les grandes agglomérations. L'augmentation constante du nombre de véhicules entraîne une saturation progressive du réseau routier, une hausse des émissions polluantes ainsi qu'une augmentation des risques d'accidents. Parmi les différentes situations de conduite, le stationnement demeure une opération particulièrement délicate, nécessitant une attention constante de la part du conducteur. Cette manœuvre est souvent source de stress, de perte de temps et de collisions mineures, notamment dans les espaces restreints et fortement fréquentés.

Face à ces problématiques, les technologies de la conduite assistée et autonome ont connu un développement remarquable ces dernières années. Les systèmes avancés d'aide à la conduite (ADAS) offrent désormais des fonctionnalités permettant non seulement d'assister le conducteur, mais également de réaliser certaines manœuvres de manière entièrement automatisée. Ces avancées reposent sur les progrès réalisés dans les domaines de la vision artificielle, du traitement d'images, de l'apprentissage automatique et des systèmes embarqués.

Le stationnement autonome nécessite qu'un véhicule soit capable de percevoir son environnement, d'identifier les obstacles présents, d'évaluer leur position et leur distance, puis de prendre les décisions appropriées afin d'assurer une manœuvre sûre et efficace. Pour répondre à ces exigences, différentes approches de détection peuvent être utilisées. Les méthodes classiques reposent notamment sur l'extraction de caractéristiques visuelles à l'aide du descripteur HOG (Histogram of Oriented Gradients) associée à un classifieur SVM (Support Vector Machine). Plus récemment, les techniques de Deep Learning, telles que l'algorithme YOLO (You Only Look Once), ont démontré des performances remarquables en matière de détection d'objets en temps réel.

Toutefois, la perception basée uniquement sur une caméra peut être affectée par plusieurs facteurs, tels que les variations d'éclairage, les conditions météorologiques ou les occultations partielles des objets. Afin d'améliorer la fiabilité du système, il est nécessaire de combiner les informations visuelles avec des données issues de capteurs de distance. La fusion de données provenant de capteurs hétérogènes, tels qu'un capteur ultrasonique et un micro-LiDAR, permet ainsi d'obtenir une perception plus robuste et plus précise de l'environnement.

Dans ce contexte, ce mémoire a pour objectif de concevoir et de développer un système de détection d'obstacles destiné à l'assistance au stationnement autonome, basé sur la fusion de

données 1D et 2D. Le système proposé combine une caméra USB, un capteur ultrasonique HC-SR04 et un micro-LiDAR TF-Luna, intégrés sur une plateforme embarquée Raspberry Pi 4.

L'objectif est d'assurer une détection fiable des obstacles et une estimation précise des distances en temps réel.

Ce mémoire est organisé en trois chapitres.

- Le premier chapitre présente les notions fondamentales relatives au stationnement autonome et aux véhicules automatisés. Il introduit les différents types de stationnement, les niveaux d'autonomie définis par la norme SAE ainsi que l'architecture générale des systèmes de stationnement autonome.
- Le deuxième chapitre est consacré aux méthodes de détection et de classification des obstacles. Il décrit les principales techniques de traitement d'images, les descripteurs HOG, les classifieurs SVM, le modèle de détection YOLO ainsi que les principes de la fusion de données multi-capteurs.
- Le troisième chapitre détaille la réalisation pratique du système proposé. Il présente la constitution de la base de données, l'évaluation de l'approche HOG-SVM, le développement de la solution basée sur YOLO, l'intégration matérielle sur Raspberry Pi 4, ainsi que la stratégie de fusion entre les données visuelles et télémétriques. Une comparaison des performances des différentes approches étudiées permet finalement de mettre en évidence les avantages de la solution retenue.

CHAPITRE 1 :

LE STATIONNEMENT AUTONOME

1.1 Introduction

L'accroissement du parc automobile en milieu urbain a rendu la gestion du stationnement particulièrement problématique, aussi bien dans les pays développés que dans les pays en développement. Cette situation engendre des difficultés croissantes pour les conducteurs, notamment en termes de recherche de places disponibles, de congestion du trafic et de sécurité. Face à ce constat, des systèmes d'aide à la conduite (ADAS – Advanced Driver Assistance Systems) ont été mis au point afin d'assister le conducteur dans la conduite et d'améliorer la sécurité routière. Le but de ces systèmes, intégrés dans le véhicule, est d'apporter de nouveaux éléments d'information sur l'environnement du véhicule de manière à aider le conducteur à prendre la bonne décision au bon moment pendant la conduite, ces systèmes utilisant divers capteurs et algorithmes pour détecter un risque de danger et interagir si besoin et ce afin de réduire la probabilité d'un accident, en avertissant le conducteur d'un risque, en évitant une collision ou en diminuant la gravité de l'accident.

Face à ces évolutions technologiques, le stationnement autonome s'impose aujourd'hui comme une solution prometteuse pour améliorer la sécurité, le confort de conduite et l'utilisation des espaces de stationnement. Ce chapitre présente les notions fondamentales liées aux véhicules autonomes et au stationnement autonome, qui constituent la base théorique nécessaire à la compréhension du système développé dans ce mémoire. [1]

1.2 Définition et concepts fondamentaux

1.2.1 Définition du stationnement dans un parking

Le stationnement correspond au fait qu'un véhicule reste immobile pendant une certaine durée, qu'elle soit courte ou longue. C'est un aspect crucial dans la planification des déplacements, particulièrement en milieu citadin, où la gestion des espaces s'avère indispensable. Effectivement, le stationnement se doit de suivre des normes spécifiques et de considérer la distribution de l'espace entre les utilisateurs et les collectivités.

D'un côté, chaque conducteur se doit de dénicher un lieu approprié pour garer son véhicule dans des conditions d'accueil satisfaisantes (accès, sécurité, etc.). Pour satisfaire cette exigence, des zones dédiées sont configurées, que l'on nomme parkings ou aires de stationnement. Ces espaces peuvent être localisés dans divers lieux, tels que les espaces publics, les sites de travail ou les zones résidentielles, et ils peuvent être de nature publique ou privée.

1.2.2 Différents types des parkings

a) Parking de surface :

Elle se situe de plain-pied, à l'extérieur, sur l'espace public ou privé. Ce type de stationnement comprend le stationnement en voie publique (places le long d'une rue, d'un quai…), et les espaces dégagés à cet effet entre bâtiments, ou établisseur d'anciens champs, d'anciens terrains vagues, etc.

Figure 1. 1 . Exemple de parking de surface. *[2].*

b) Parking aérien :

Un parking aérien est un dispositif de stationnement qui constitue une solution adaptée, permettant d'optimiser l'espace à disposition dans les zones urbaines denses. Ces parkings, également appelés en silo ou en plein air, sont constitués de plusieurs niveaux superposés

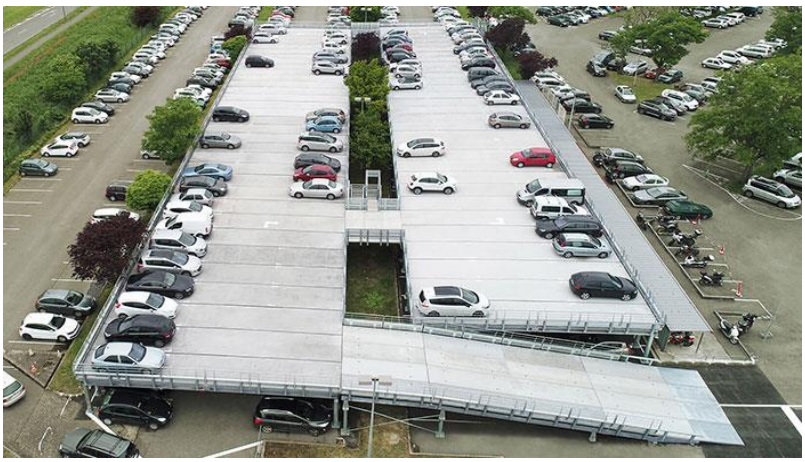

Figure 1. 2. Exemple de parking aérien. [2].

permettant d'accueillir un nombre important de voitures, tout en ayant une faible emprise au sol.

c) Parking souterrain :

Un parking souterrain est un type de stationnement couramment déployée dans les milieux urbains pour augmenter l'espace utilisé. Il constitue un espace de stationnement fermé et plus ou moins protégé pour l'automobiliste.

Figure 1. 3. Exemple de parking souterrain. [2].

d) Les parkings automatiques :

Ce sont généralement des ouvrages souterrains ou en élévation dont les rampes sont remplacées par des systèmes de levage et de translation automatiques, et qui permettent de gérer automatiquement le stationnement des véhicules à l'intérieur sans avoir besoin de l'interventions du conducteur. [3]

Figure 1. 4. Exemple de parking automatique. [4].

1.2.3 Définition du concept d'autonomie

L'autonomie dans le domaine des systèmes automobiles intelligents signifie que les véhicules sont capables d'effectuer des tâches sans intervention de l'homme. Cela veut dire que les véhicules peuvent voir ce qui les entoure, prendre des décisions et faire des actions tout seuls. Les véhicules utilisent des capteurs pour voir ce qui se passe autour d'eux, de systèmes pour traiter toutes les informations et des algorithmes pour déterminer la conduite à tenir. Tout cela permet aux véhicules de conduire sans l'aide des personnes.

Donc l'autonomie correspond à la capacité d'un système de conduite automatisée à réaliser tout ou partie de la tâche de conduite dynamique sans intervention du conducteur. [5]

1.2.4 Les niveaux d'autonomie

La classification des niveaux d'autonomie des véhicules se base sur une échelle qui décrit le degré d'intervention humaine nécessaire lors de la conduite. Cette classification permet de distinguer différents niveaux d'automatisation, allant d'aucune autonomie à une autonomie complète.

• Niveau 0 : Conduite entièrement manuelle :

Le conducteur est responsable de toutes les tâches de conduite, comme la direction, l'accélération et le freinage. Le système peut simplement émettre des alertes sans intervenir.

• Niveau 1 : Assistance au conducteur :

Le système peut aider le conducteur pour une tâche spécifique, comme le régulateur de vitesse ou l'aide au maintien de voie. Cependant, le conducteur doit rester vigilant et prêt à intervenir à tout moment.

• Niveau 2 : Automatisation partielle :

Le système peut gérer à la fois la direction et la vitesse du véhicule. Néanmoins, le conducteur doit constamment surveiller la route et être prêt à reprendre le contrôle si nécessaire.

• Niveau 3 : Automatisation conditionnelle :

Le véhicule peut prendre le contrôle dans certaines conditions précises. Le conducteur doit néanmoins être capable de reprendre le contrôle rapidement si la situation l'exige.

• Niveau 4 : Automatisation élevée :

Le véhicule peut effectuer toutes les tâches de conduite de manière autonome dans des conditions spécifiques. Dans ce cas, le conducteur n'est pas obligé de surveiller constamment la route.

• Niveau 5 : Automatisation complète :

Le véhicule est totalement autonome et peut gérer toutes les situations sans l'intervention d'un conducteur. [5]

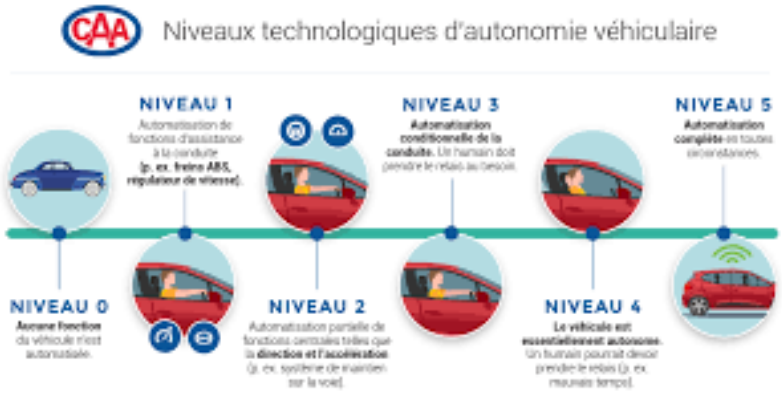

Figure 1. 5. Les niveaux technologiques d'autonomie véhiculaire. [6].

1.3 Généralités sur le stationnement autonome

1.3.1 Définition du stationnement autonome

Le stationnement autonome est une fonctionnalité avancée des véhicules intelligents permettant à un véhicule de rechercher une place de stationnement, puis d'effectuer automatiquement les manœuvres nécessaires pour se garer sans intervention humaine. Cette

fonctionnalité repose sur l'utilisation de capteurs et d'actionneurs embarqués (caméras, radars, lidars, etc.), ainsi que sur des algorithmes de perception, de planification et de contrôle.

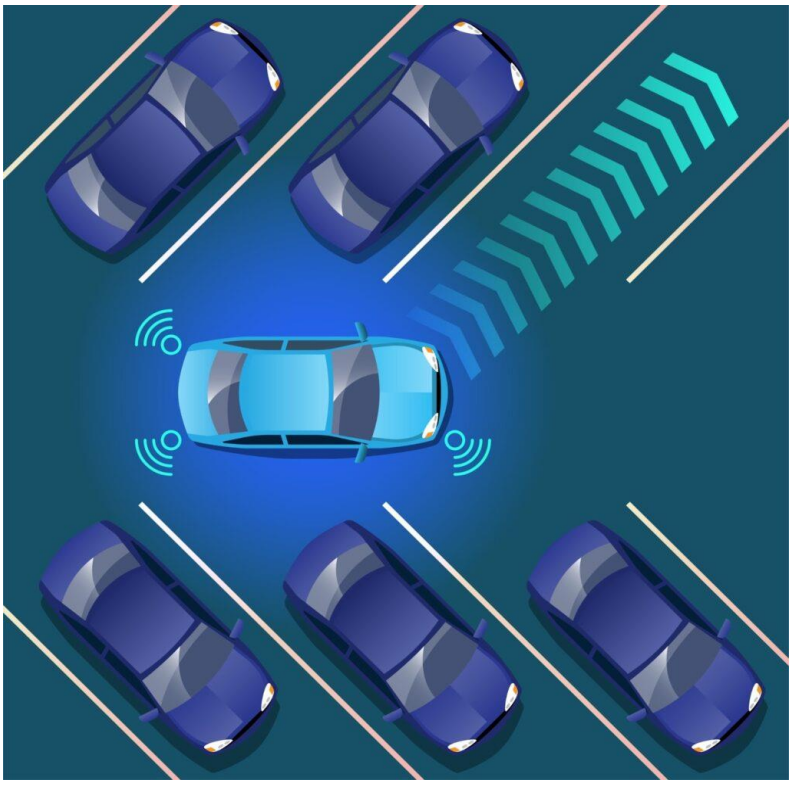

Figure 1. 6. Système de stationnement autonome. [7].

1.3.2 L'évolution du stationnement autonome

Le stationnement autonome fait partie de l'évolution des systèmes d'aide à la conduite et des technologies de conduite automatisée. Ces systèmes ont beaucoup progressé au fil du temps, passant de simples aides à la conduite à des solutions entièrement autonomes.

• Premières aides au stationnement :

Les premiers systèmes de stationnement étaient basés sur des capteurs ultrasoniques. Ils permettaient de détecter les obstacles et d'alerter le conducteur avec des signaux sonores et le conducteur était toujours responsable de la manœuvre.

• Apparition du stationnement automatique :

Avec les avancées technologiques, des systèmes capables de contrôler certaines fonctions du véhicule ont été développés. Le stationnement automatique permettait de prendre en charge la direction et le conducteur contrôlait l'accélération et le freinage.

• Évolution vers le stationnement autonome :

Aujourd'hui, les systèmes modernes utilisent des capteurs multiples (ultrasoniques, radars, caméras) et des algorithmes avancés de traitement de données et d'intelligence

artificielle. Ces technologies permettent au véhicule de détecter son environnement, de planifier une trajectoire et d'exécuter la manœuvre de stationnement sans intervention humaine.

• Tendances actuelles :

Le stationnement autonome s'inscrit dans le développement des véhicules intelligents et connectés. Les recherches actuelles visent à améliorer la précision, la sécurité et la fiabilité des systèmes en combinant les données de différents capteurs. [5]

1.3.3 Fonctionnement du stationnement autonome

Le stationnement autonome repose sur plusieurs étapes clés. Ces étapes permettent au véhicule de voir son environnement, de prendre des décisions et de stationner en toute sécurité. Voici comment cela fonctionne :

• Détection de place :

Le véhicule utilise différents capteurs comme des capteurs à ultrasons, des radars ou des caméras pour trouver une place de stationnement disponible. Il mesure alors les dimensions de la place pour voir si elle convient à la taille du véhicule.

• Analyse de l'environnement :

Une fois la place trouvée, le véhicule regarde autour de lui. Il détecte les obstacles comme d'autres véhicules, des piétons ou des bordures, et il comprend la scène qui l'entoure en utilisant des algorithmes de vision ou en combinant les données de plusieurs capteurs.

• Planification de trajectoire :

Le véhicule calcule ensuite le meilleur chemin pour atteindre la place de stationnement. Il tient compte de la façon dont le véhicule peut bouger et des obstacles qu'il a détectés, pour s'assurer que la manœuvre se déroule sans problème et sans collision.

• Exécution du stationnement :

Enfin, le véhicule exécute la manœuvre de stationnement en contrôlant la direction, l'accélération et le freinage. Il ajuste son chemin en temps réel pour garantir la précision et la sécurité. [5]

Figure 1.6: Automated Parallel Parking System and Driver Interface Status

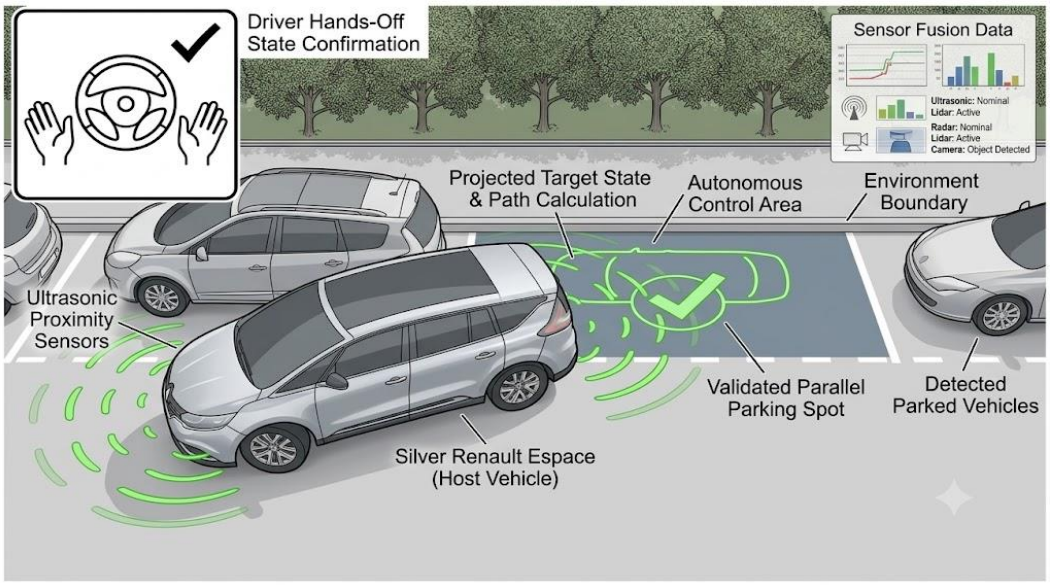

Figure 1. 7. État du système de stationnement parallèle automatisé et de l'interface conducteur.

1.3.4 Architecture d'un système de stationnement autonome

L'architecture d'un système de stationnement autonome repose sur une organisation fonctionnelle composée de plusieurs modules interconnectés. Ces modules permettent au véhicule de percevoir son environnement, de prendre des décisions et d'exécuter des actions.

De manière générale, cette architecture est structurée en trois niveaux principaux : la perception, la décision et l'action.

• Module de perception :

Le module de perception permet au véhicule de collecter des informations sur son environnement. Il utilise différents capteurs tels que des capteurs ultrasoniques, des radars et des caméras. Ces capteurs fournissent des données sur la présence d'obstacles, la détection des places de stationnement et les distances environnantes.

• Module de traitement et de décision :

Ce module constitue le cerveau du système de stationnement autonome. Il comprend des algorithmes de perception, comme la vision et la détection d'objets, ainsi que la planification de trajectoire et la prise de décision. Le module de traitement et de décision traite les données issues des capteurs afin d'identifier une place de stationnement, de calculer une trajectoire optimale et d'éviter les obstacles.

• Module de contrôle :

Le module de contrôle permet d'exécuter physiquement la manœuvre de stationnement. Il contrôle la direction, l'accélération et le freinage. Le système utilise des contrôleurs pour suivre la trajectoire planifiée avec précision et stabilité. Cela permet au véhicule de stationner de manière autonome et sécurisée. [9]

1.3.5 Différents types de stationnement

Il existe différents types de stationnement. Les systèmes de stationnement autonome doivent être capables de gérer différentes configurations de stationnement. Ces configurations sont définies en fonction de l'orientation du véhicule par rapport à la place disponible.

Les principaux types de stationnement sont les suivants :

• Le stationnement parallèle : Également appelé créneau, consiste à garer le véhicule parallèlement au trottoir. Cela se fait généralement entre deux véhicules. Ce type de stationnement est considéré comme le plus complexe. En effet, il y a des contraintes d'espace et il faut planifier précisément la trajectoire du véhicule.

• Le stationnement perpendiculaire : Également appelé stationnement en bataille, consiste à garer le véhicule perpendiculairement à la route. Cela signifie que le véhicule forme un angle de 90° avec la route. Ce type de stationnement est plus simple que le créneau et il est très courant dans les parkings publics et privés.

• Le stationnement en épi : Également appelé stationnement incliné, correspond à un angle incliné par rapport à la voie de circulation. Cet angle est généralement compris entre 30° et 60°. Ce type de stationnement facilite l'entrée et la sortie du véhicule. Il est souvent utilisé dans les parkings qui ont une forte capacité**.** [10]

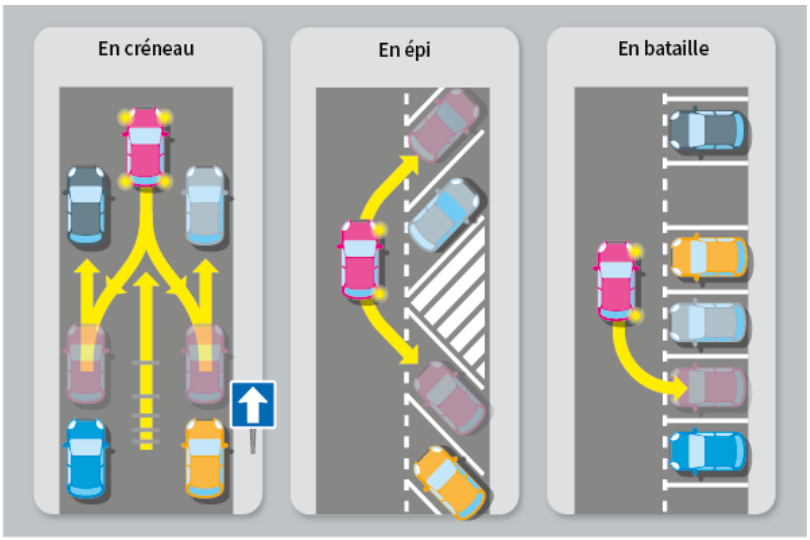

Figure 1. 8. Les différents types de stationnement. [10].

1.4 Conclusion

Dans ce chapitre, nous avons expliqué les concepts de base liés au stationnement et aux véhicules autonomes. Nous avons commencé par présenter les différents types de stationnement et leurs caractéristiques spécifiques. Ensuite, nous avons examiné la notion d'autonomie et ses différents niveaux, ce qui nous a permis de mieux comprendre le rôle de l'intervention humaine dans les systèmes automatisés. Enfin, nous avons analysé le stationnement autonome, en soulignant son évolution, ses principes de fonctionnement et les technologies qui le rendent possible.

Ces informations constituent une base théorique importante pour comprendre les systèmes de stationnement intelligents et leur fonctionnement.

CHAPITRE 2 : DETECTION ET CLASSIFICATION DES OBSTACLES.

2.1 Introduction

Le développement des systèmes intelligents de stationnement autonome nécessite l'utilisation de techniques avancées capables d'assurer une perception fiable de l'environnement. Dans ce contexte, la détection et l'identification des obstacles représentent des fonctions essentielles pour garantir la sécurité et l'efficacité des manœuvres de stationnement.

Ce chapitre est consacré à l'étude des méthodes utilisées pour l'analyse des données issues des capteurs et des systèmes de vision artificielle. Il présente les différentes étapes du traitement d'image, depuis l'acquisition des données jusqu'à l'identification des obstacles à l'aide de techniques de classification et de détection.

Les méthodes classiques basées sur l'extraction de caractéristiques ainsi que les approches récentes fondées sur l'apprentissage profond seront également abordées. Une attention particulière sera portée à l'utilisation conjointe des données 1D et 2D afin d'améliorer les performances du système proposé.

2.2 Acquisition d'images

La première étape du processus de détection des obstacles dans les systèmes de vision artificielle est l'acquisition d'image qui consiste à capter des images de l'environnement à l'aide d'un capteur optique de type caméra afin de récupérer les données nécessaires au traitement et à l'analyse.

Dans les systèmes embarqués, la caméra joue un rôle essentiel en tant que capteur 2D et elle fournit une représentation visuelle riche de la scène. Les images sont prises de manière numérique et codées dans l'espace de couleur RGB (Red, Green, Blue), où chaque pixel est représenté par une combinaison de trois composants de couleur. L'image RGB apporte de nombreuses informations mais entraîne un accroissement des complexités des traitements.

Du fait de ces complexités, il est souvent préférable de convertir les images RGB en niveaux de gris pour en réduire la quantification informationnelle colorimétrique à une seule information colorimétrique d'intensité lumineuse. Grâce à cette contrainte d'information, les quantités de données à traiter se réduisent tout en maintenant les informations essentielles pour le traitement, à savoir la structure et les contours des objets présents dans l'image. Pour les systèmes de détection d'obstacles, il est important d'effectuer cette conversion en niveaux de gris, car elle facilite des traitements ultérieurs de binarisation, d'opérations morphologiques et d'extraction de caractéristiques. Par ailleurs, poursuivre avec la caméra dans le cadre des systèmes embarqués, notamment ceux embarqués sur des plateformes de type Raspberry Pi,

permet une acquisition en temps réel, aussi adaptée aux applications de stationnement autonome.*[11]*

2.3 Extraction de caractéristiques par HOG

L'extraction de caractéristiques constitue une étape essentielle dans le processus de détection et de classification des obstacles. Elle permet de transformer l'image en un vecteur numérique exploitable par les algorithmes de classification.

Pour effectuer cette tâche, le descripteur HOG (*Histogram of Oriented Gradients*) est largement utilisé en vision par ordinateur pour la détection d'objets. Cette méthode repose sur l'analyse des gradients présents dans l'image afin d'extraire des informations relatives aux formes et aux contours des objets.

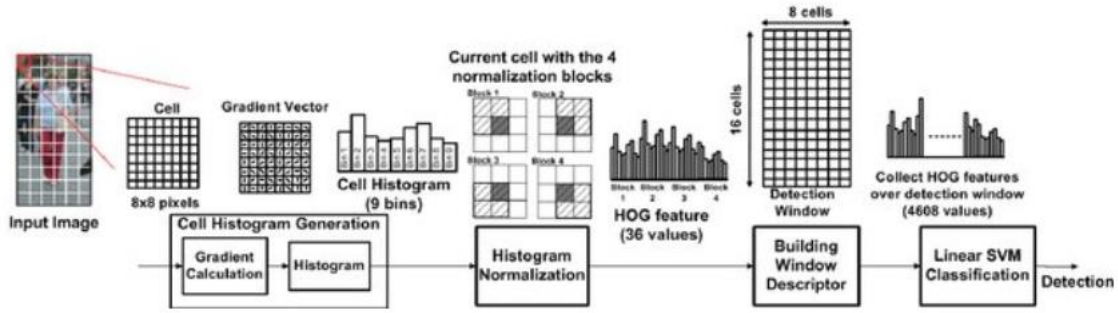

Figure 2. 1. Architecture de descripteur HOG-SVM [12]

Le principe de fonctionnement du descripteur HOG repose sur plusieurs étapes :

- Conversion de l'image en niveaux de gris afin de simplifier le traitement et de se concentrer sur les variations d'intensité.
- Calcul du gradient de l'image, qui permet d'extraire les variations d'intensité.
- Découpage de l'image en petites cellules.
- Calcul des histogrammes des orientations des gradients dans chaque cellule.
- Normalisation des blocs pour rendre l'algorithme robuste aux variations d'éclairage.

Le vecteur de caractéristiques ainsi obtenu traduit la distribution des contours de l'image, ce qui en fait un descripteur efficace pour la détection des obstacles.

2.3.1 Avantages du HOG

Le descripteur HOG présente plusieurs avantages :

- Bonne représentation des formes et des contours.
- Robustesse aux variations d'illumination.
- Compatibilité avec les algorithmes de classification comme le SVM.

2.3.2 Rôle dans le système

Au sein de ce système, le descripteur HOG est qu'il va permettre de convertir les images en vecteurs caractéristiques qui seront ensuite utilisés comme entrées à l'algorithme de classification ; il s'agit donc d'un lien fondamental entre le processus de traitement d'images et celui de l'apprentissage. *[13]*

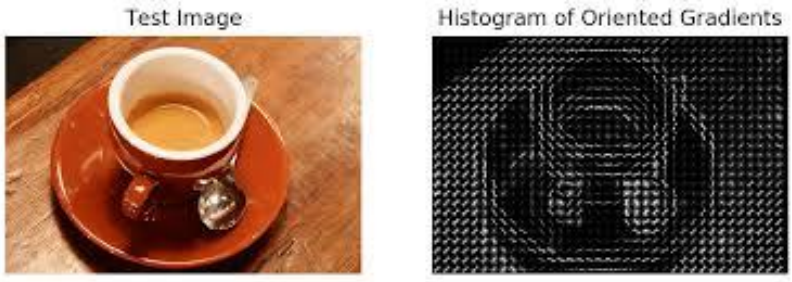

Figure 2. 2. Exemple de descripteur HOG [14]

2.4 Méthodologie générale de classification par vision artificielle

La détection et la classification des obstacles reposent sur une approche combinant la vision artificielle et l'apprentissage supervisé. Dans ce cadre, les images sont généralement prétraitées afin d'uniformiser les données. Elles sont converties en niveaux de gris, puis redimensionnées à une taille fixe (par exemple 128 × 128 pixels) afin d'assurer une cohérence entre les échantillons.

Par la suite, des caractéristiques discriminantes sont extraites à partir des images à l'aide du descripteur HOG (Histogram of Oriented Gradients). Cette méthode permet de représenter efficacement les contours, les formes et les structures locales des objets présents dans l'image.

Chaque image est ainsi transformée en un vecteur de caractéristiques exploitable par les algorithmes de classification.

Enfin, la classification est effectuée à l'aide d'un classifieur de type SVM (Support Vector Machine), qui cherche à séparer les différentes classes en maximisant la marge entre elles. Une fois le modèle entraîné sur une base de données annotée, il peut être utilisé pour la prédiction de nouvelles observations.

2.4.1 Classification par SVM

La classification des obstacles constitue une étape essentielle dans le système proposé. Elle permet d'identifier la nature des objets détectés à partir des caractéristiques extraites. Dans ce travail, la classification est réalisée à l'aide de la méthode SVM (Support Vector Machine).

Les SVM, ou *Support Vector Machines*, sont des algorithmes d'apprentissage automatique utilisés surtout pour la classification. Leur but est de séparer des données en deux catégories en trouvant la meilleure frontière possible entre elles. Cette frontière s'appelle un hyperplan, et l'idée centrale du SVM est de choisir celui qui généralise le mieux sur de nouvelles données.

Les SVM sont particulièrement efficaces quand on dispose de peu de données d'entraînement. Ils font partie des classificateurs linéaires, mais peuvent aussi traiter des cas non linéaires grâce à la technique du noyau, appelée kernel trick.

2.4.1.1 Classificateur SVM linéaire

On utilise le SVM linéaire lorsque les données sont linéairement séparables. Cela signifie qu'il est possible de tracer une ligne droite pour séparer parfaitement les différentes classes d'obstacles.

2.4.1.2 Formulation mathématique

L'hyperplan de séparation optimal est défini mathématiquement par l'équation suivante :

$$h(x) = \omega . x + b \qquad \textbf{Equation 2.1}$$

Chaque composante de l'équation de l'hyperplan joue un rôle essentiel dans le processus de classification :

- **(x) : vecteur d'entrée**

Le vecteur (x) représente l'ensemble des caractéristiques extraites de l'objet à classifier.

- **(w) : vecteur des poids**

Le vecteur (w) est appelé vecteur normal à l'hyperplan de séparation. Il est perpendiculaire à la frontière de décision et détermine son orientation dans l'espace des caractéristiques. Les valeurs

de ses composantes sont ajustées lors de la phase d'apprentissage afin d'attribuer une importance différente à chaque caractéristique et d'obtenir la meilleure séparation possible entre les classes.

- **(b) : biais**

Le paramètre (b) est un scalaire réel qui détermine la position de l'hyperplan dans l'espace des caractéristiques. Il permet de déplacer la frontière de décision sans modifier son orientation, assurant ainsi une séparation optimale des données. Grâce au biais, l'hyperplan n'est pas contraint de passer par l'origine du repère.

La règle de décision est la suivante :

- si $h(x) \geq 0$, l'entrée appartient à la première classe ;
- si $h(x) < 0$, elle appartient à la seconde classe.

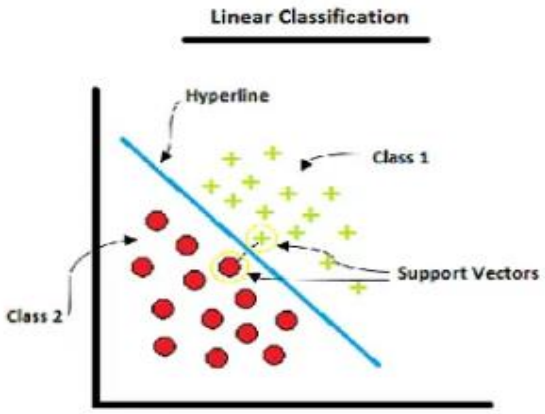

Figure 2. 3. Classification SVM linéaire. [15]

2.4.1.3 Classificateur SVM non linéaire

Un classificateur non linéaire est un modèle de classification qui ne sépare pas les classes avec une simple droite ou un hyperplan, mais avec une frontière plus complexe et courbée.

Dans le cas des SVM, cela se fait souvent grâce à l'astuce du noyau : on transforme les données dans un espace de dimension plus élevée, où elles deviennent séparables linéairement. En pratique, cela permet de traiter des données qui ne peuvent pas être séparées correctement dans l'espace d'origine.

2.4.1.4 Principe de Kernel

Le kernel se révèle par exemple très utile pour séparer deux classes au sein d'un ensemble de données [17] à deux dimensions, où les données ne sont pas séparables linéairement.

L'utilisation d'une fonction polynomiale compliquerait le problème de classification. Il est donc préférable de transformer les données dans un espace 3D où les données deviennent séparables par un classificateur linéaire.

Il devient alors possible de transformer les données de la 2D vers la 3D, de trouver une frontière de décision linéaire en adaptant le classificateur linéaire dans l'espace 3D, et de cartographier la frontière de décision linéaire à nouveau dans l'espace 2D pour créer une frontière de décision non-linéaire en 2D.

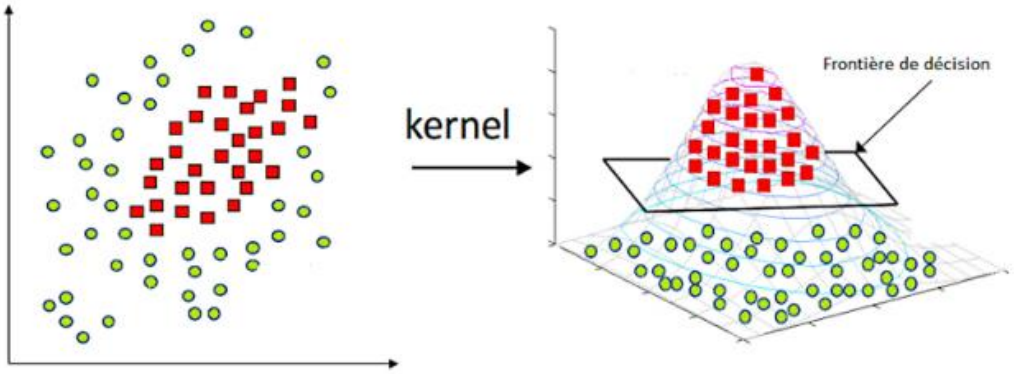

Figure 2. 4 .SVM Classification non linéaire, méthode de Kernel [16]

2.5 Détection d'obstacles par l'algorithme YOLO

Au cours des dernières années, les modèles Deep Learning ont gagné en renommée pour leur capacité à traiter l'information visuelle et ils sont devenus un élément clé de nombreuses applications de vision par ordinateur. Parmi les principaux problèmes que ces modèles peuvent résoudre est la détection et la localisation des objets dans les images. La détection d'objets est utilisée dans de divers domaines, y compris la conduite autonome, la vidéosurveillance et les soins de santé.

Parmi ces modèles, YOLO est l'un des plus connus et plus utilisés. Il permet de réaliser la détection d'objets en une seule étape et d'atteindre un très bon niveau de précision et une vitesse de détection élevée et il est particulièrement bien adapté pour les applications en temps réel. [18]

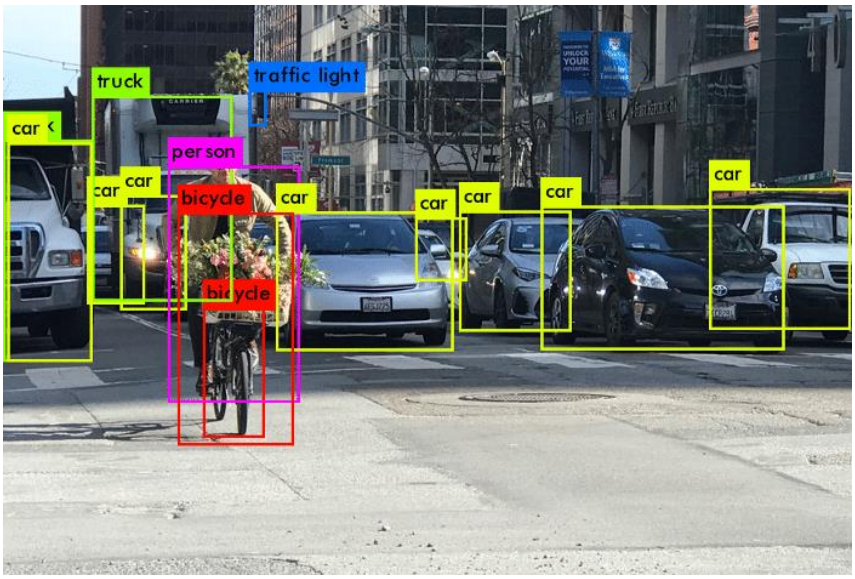

Figure 2. 5 . Exemple de détection par YOLO. [19]

2.5.1 Définition de YOLO

You Only Look Once, plus connu sous l'acronyme YOLO, est un framework de détection d'objets en temps réel qui a transformé le domaine de la vision par ordinateur depuis son introduction en 2015.

Contrairement aux méthodes classiques de l'époque, qui nécessitaient de multiples analyses par image, YOLO ne "regarde" qu'une seule fois les pixels de chaque image pour prédire la présence et la position des objets, d'où son nom évocateur.

En quelques années, YOLO est devenu l'un des algorithmes de détection d'objets les plus populaires, plébiscité pour sa vitesse et son efficacité.

2.5.2 Principe de fonctionnement

Le principe de YOLO consiste à diviser l'image en plusieurs cellules formant une grille. Chaque cellule est responsable de la détection des objets présents dans sa région.

Pour chaque objet détecté, l'algorithme prédit :

- La position de l'objet dans l'image.
- La classe de l'objet.
- Un score de confiance indiquant la probabilité de détection.

La localisation des objets est généralement représentée par une boîte englobante appelée *bounding box*. [18].

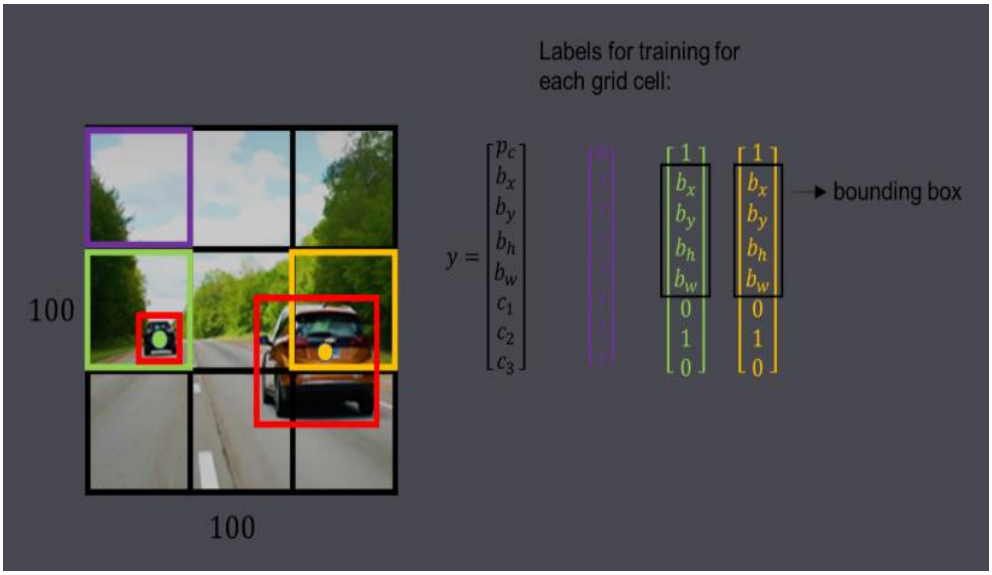

Figure 2. 6. Principe de bounding box [18].

2.5.3 Architecture de YOLO

Dans ce travail, on a utilisé YOLOv8 nano, qui est une architecture de vision par ordinateur pour la détection d'objets, développée par Ultralytics, qui repose sur une structure en trois parties principales : Backbone, Neck et Head.

Backbone : Extraction des caractéristiques.

Le Backbone est la première partie du réseau. Son rôle est d'analyser l'image et d'en extraire les informations importantes.

Par exemple, si l'image contient une voiture :

- Les premières couches détectent les contours et les lignes.

- Les couches suivantes détectent les formes (roues, vitres, carrosserie).
- Les couches profondes reconnaissent des objets plus complexes comme une voiture entière ou un cône de signalisation.

Le Backbone transforme donc l'image brute en un ensemble de caractéristiques (features) qui décrivent le contenu de l'image.

On peut le comparer à l'œil humain qui observe une scène et identifie progressivement les formes présentes.

Neck : Fusion des informations.

Après l'extraction des caractéristiques, ces informations sont envoyées vers le Neck. Son rôle est de combiner les informations provenant de différentes échelles de l'image :

- Les couches profondes contiennent beaucoup d'informations sur les grands objets.
- Les couches plus superficielles contiennent davantage de détails sur les petits objets.

Cette étape améliore considérablement la détection des objets de tailles différentes.

Head : Détection et classification.

Le Head est la dernière partie du réseau. À partir des caractéristiques fournies par le Neck, il réalise trois tâches :

- Localiser les objets dans l'image.
- Déterminer leur catégorie (voiture, cône, piéton, etc.).
- Calculer un score de confiance indiquant la probabilité que la détection soit correcte.

Le résultat est représenté sous forme de boîtes englobantes (Bounding Boxes) accompagnées du nom de la classe détectée et du niveau de confiance.

[20]

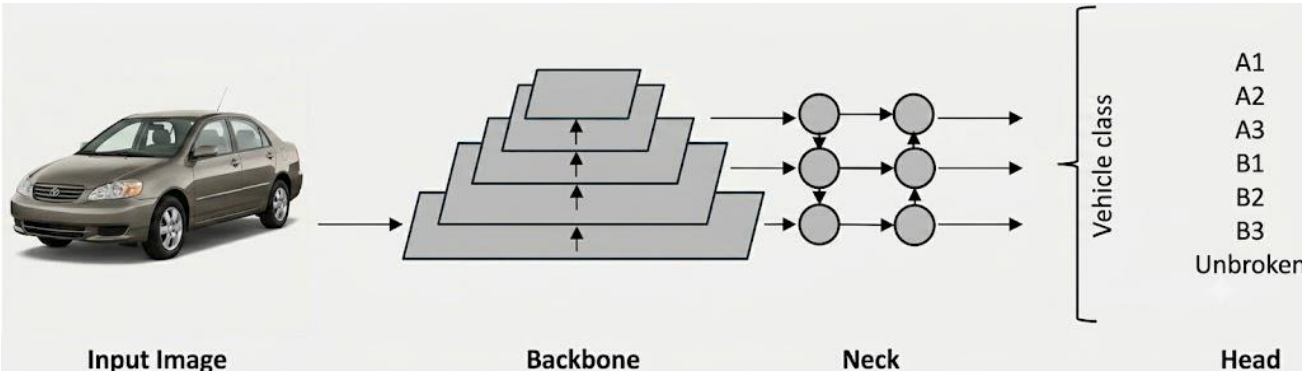

Figure 2. 7. Architecture de modèle YOLOv8.

2.6 Fusion de données 1D et 2D :

La fusion de données est une approche qui consiste à combiner des informations provenant de différentes sources et types de capteurs afin d'améliorer la qualité, la fiabilité et la complétude de la perception d'un système. En intégrant des données complémentaires issues de technologies variées telles que les caméras, les radars, les lidars ou encore les capteurs ultrasoniques, il devient possible de compenser les limites propres à chaque dispositif pris individuellement. Cette combinaison permet ainsi d'obtenir une représentation plus robuste et plus précise de l'environnement, particulièrement dans des contextes complexes où certaines mesures peuvent être bruitées, incomplètes ou perturbées par des conditions extérieures.

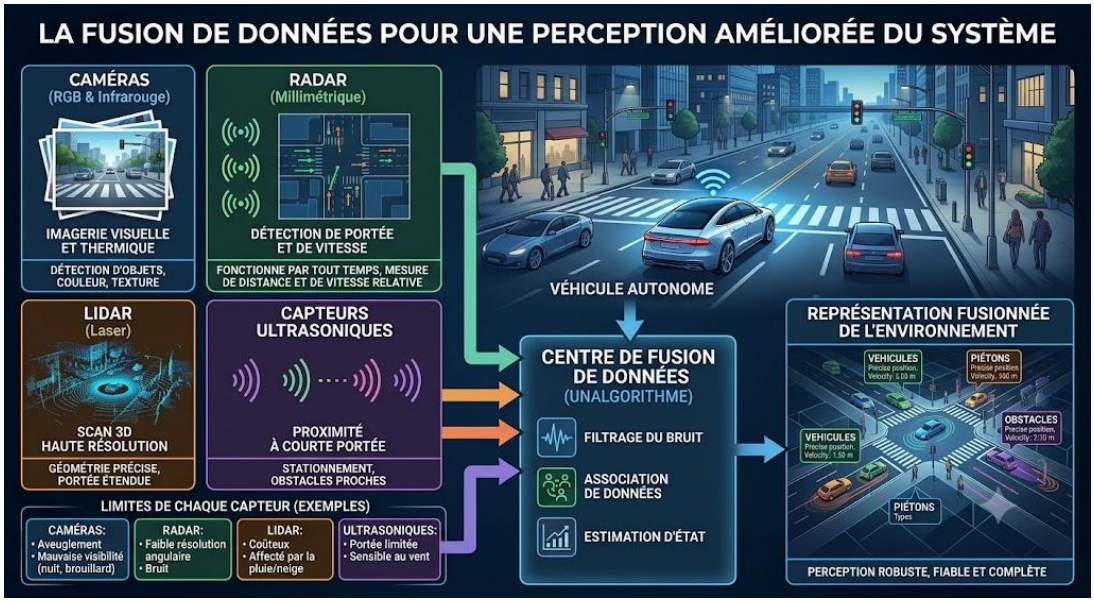

Figure 2. 8 . Exemple de fusion de données pour la perception de l'environnement.

2.7 Conclusion

Dans ce chapitre, plusieurs approches utilisées pour la détection et l'identification des obstacles ont été présentées dans le cadre du stationnement autonome. Les différentes étapes du traitement des données visuelles ont permis de mettre en évidence l'importance des techniques de vision artificielle dans l'analyse de l'environnement du véhicule.

L'étude des méthodes de classification et des algorithmes de détection a montré l'intérêt des techniques d'apprentissage automatique et d'apprentissage profond pour améliorer la reconnaissance des objets. La combinaison des données issues des capteurs de distance et des systèmes de vision constitue une solution efficace pour renforcer la robustesse et la précision du système de stationnement autonome. Cette complémentarité contribue à améliorer la fiabilité des opérations de stationnement et représente une étape importante vers le développement de systèmes embarqués plus intelligents et plus sûrs.

CHAPITRE 3 : REALISATION EXPREMENTALE ET FUSION MULTI-CAPTEUR POUR LA DETECTION D'OBSTACLES

3.1 Introduction

Après avoir présenté les concepts théoriques liés à la détection d'obstacles et aux techniques de vision par ordinateur dans le chapitre précédent, ce chapitre est consacré à la réalisation pratique du système développé pour l'assistance au stationnement autonome.

Nous consacrons ce chapitre pour l'expérimentation du système de détection d'obstacles dédié pour la manœuvre du stationnement autonome. Dans un premier temps, notre approche est basée sur le descripteur HOG et le classifieur SVM qui est développée pour la classification des voitures et des cônes de signalisation. Par la suite, une autre approche basée sur le modèle YOLO est mise en œuvre pour la détection d'objets en temps réel. Enfin, l'implémentation du système sur la plateforme Raspberry Pi 4, la fusion des données issues des capteurs ainsi que les résultats expérimentaux sont exposés et interprétés.

3.2 Présentation générale du travail réalisé

Dans le cadre de ce travail, plusieurs approches de détection et de classification d'obstacles ont été développées et évaluées afin de concevoir un système d'assistance au stationnement autonome. Dans un premier temps, une base de données d'images contenant des voitures et des cônes de signalisation a été constituée. Ces images ont ensuite été prétraitées afin d'extraire leurs caractéristiques visuelles à l'aide du descripteur HOG (Histogram of Oriented Gradients). Les vecteurs de caractéristiques obtenus ont servi à l'apprentissage d'un classifieur SVM (Support Vector Machine) chargé de distinguer les différentes catégories d'obstacles.

Dans un second temps, une approche plus avancée basée sur le modèle de Deep Learning YOLO (You Only Look Once) a été mise en œuvre. Contrairement à l'approche HOG-SVM, YOLO permet de réaliser simultanément la détection, la localisation et la classification des objets présents dans une image. Une base de données annotée contenant principalement des voitures et des cônes a été utilisée pour l'entraînement et l'évaluation du modèle. Les performances des deux approches ont ensuite été comparées afin d'identifier la solution la plus adaptée aux contraintes du stationnement autonome.

Enfin, le modèle retenu a été déployé sur une plateforme embarquée Raspberry Pi 4 et intégré à un système de perception multi-capteurs comprenant une caméra USB, un capteur ultrasonique HC-SR04 et un micro-LiDAR TF-Luna. Cette architecture permet d'associer les informations issues de la vision artificielle aux mesures de distance afin d'améliorer la fiabilité de la détection d'obstacles en temps réel.

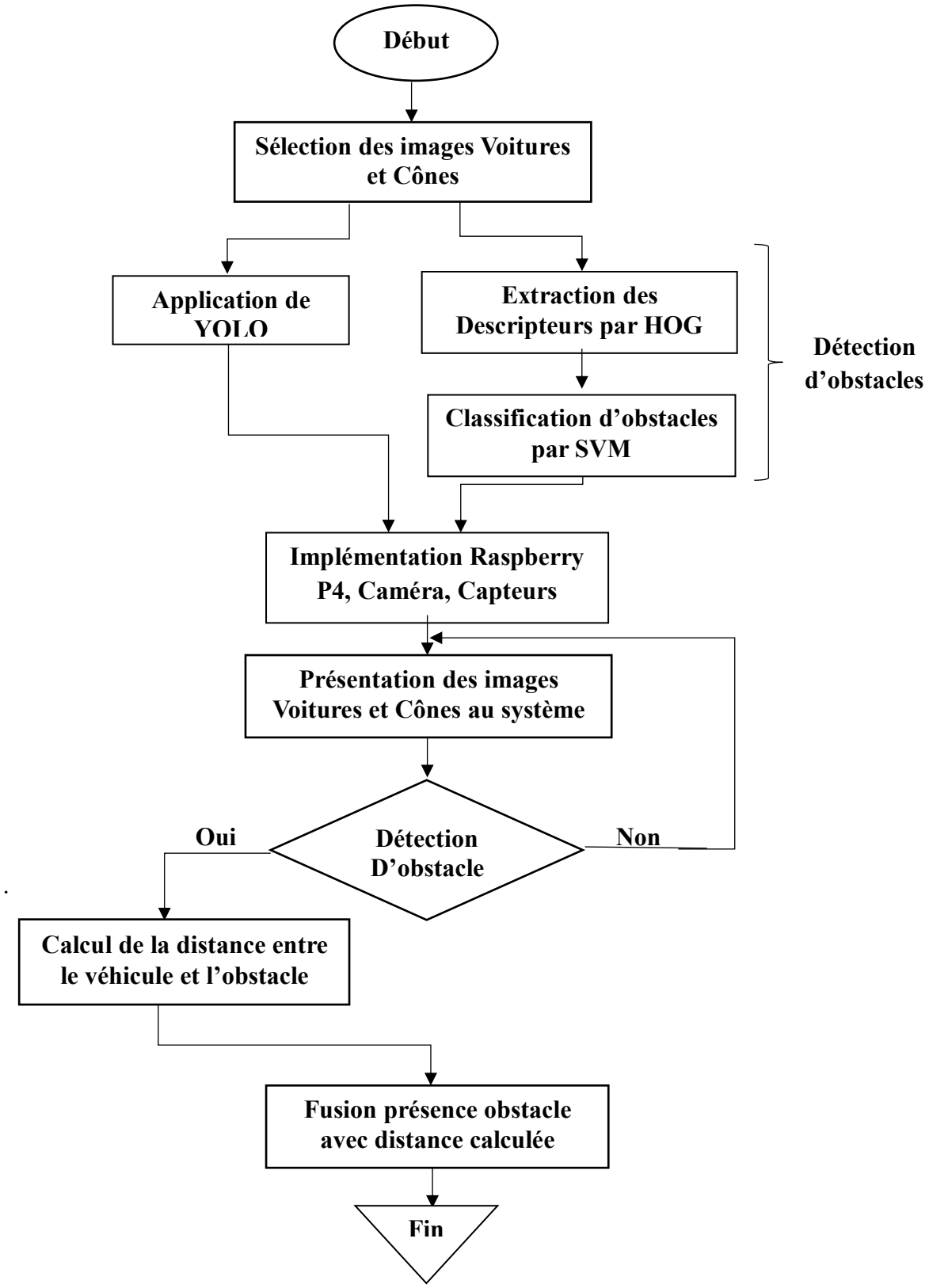

Figure 3. 1. Organigramme global de la méthodologie de détection et classification.

3.3 Développement du système basé sur HOG et SVM

3.3.1 Constitution de la base de données

La première étape de réalisation du système consiste à construire une base de données d'images destinée à l'apprentissage et au test du modèle de classification. Cette base de données contient deux catégories principales d'obstacles :

- **Les voitures :** obstacles dynamiques de grande taille.

Figure 3. 2. Exemple voiture.

- **Les cônes de signalisation** : obstacles statiques de petite taille.

Figure 3. 3. Exemple de cône.

Les images ont été réparties selon les différentes classes d'obstacles afin de faciliter leur traitement et leur étiquetage. Chaque image est associée à une classe représentant le type d'obstacle correspondant.

3.3.2 Extractions des descripteurs HOG

Après la collecte des images, une étape d'extraction de caractéristiques a été réalisée à l'aide du descripteur HOG. Cette méthode repose sur l'analyse locale des gradients d'intensité dans l'image, permettant de capturer des informations sur la forme et la structure des objets. Pour chaque image, les étapes suivantes ont été appliquées :

- Conversion de l'image en niveaux de gris afin de réduire la complexité des données et de mettre en évidence les informations de forme.
- Redimensionnement des images à une taille uniforme de 128 × 128 pixels.
- Calcul des gradients d'intensité en chaque pixel.
- Quantification des orientations des gradients dans des cellules locales.
- Normalisation des histogrammes dans des blocs de cellules.

3.3.3 Classification par SVM

Le classifieur SVM a été entraîné sur les descripteurs HOG extraits précédemment afin de distinguer les deux classes d'obstacles. Le SVM à noyau linéaire a été retenu pour sa capacité à trouver un hyperplan de séparation optimal entre les classes dans l'espace des caractéristiques.

La base de données a été divisée en deux ensembles :

- 70 % des images pour l'apprentissage.
- 30 % des images pour le test.

Afin de maintenir une répartition équilibrée des classes dans les deux ensembles, une division stratifiée (stratified split) a été appliquée. Cette méthode garantit le respect des proportions initiales de chaque classe dans les sous- ensembles d'apprentissage et de test.

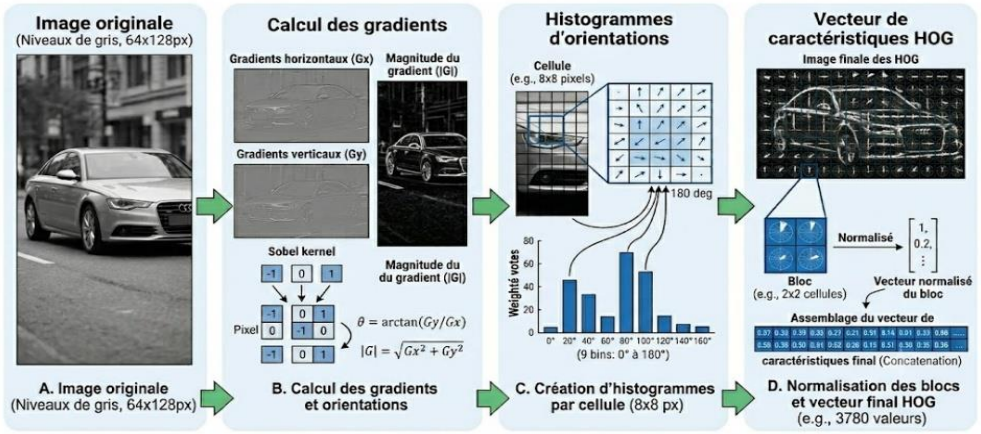

Figure 3. 4. Étapes de l'extraction des caractéristiques HOG.

3.3.4 Détection avec YOLO

En complément de l'approche HOG+SVM, le modèle YOLO a été déployé pour assurer une détection d'obstacles en temps réel. Contrairement à la méthode classique qui procède en deux étapes distinctes (extraction + classification), YOLO réalise simultanément la détection et la classification des objets dans une seule passe grâce aux mécanismes d'apprentissage profond.

Le modèle est capable de reconnaître jusqu'à 24 classes d'objets (voitures, cônes, piétons, animaux, et.), avec une localisation précise des objets via des boîtes englobantes (Bounding Boxes) accompagnées du nom de la classe et du score de confiance associé. Pour le déploiement sur la Raspberry Pi 4, le modèle YOLO a été utilisé via le framework Ultralytics, permettant une détection en temps réel des objets avec une bonne performance sur une architecture embarquée à ressources limitées.

3.4 Base de données

3.4.1 Base de données pour SVM

La base de données constituée pour l'entraînement du classifieur HOG+SVM comprend 117 images au total, réparties entre deux classes :

Tableau 3. 1. Répartition de la base de données pour HOG+SVM

Utilisation	**Pourcentage**	**Nombre total d'images**	**Voitures**	**Cônes**
Apprentissage (train)	70%	81	34	47

Test	30%	36	15	21
Total	100%	117	49	68

Les images ont été collectées dans des conditions variées (angles de vue, éclairage, distances) afin d'améliorer la capacité de généralisation du modèle. Chaque image a été prétraitée par conversion en niveaux de gris et redimensionnement à une dimension fixe, puis transformée en descripteur HOG avant d'être fournie au classifieur SVM.

3.4.2 Base de données pour YOLO

Pour le déploiement et l'évaluation du modèle YOLO, une base de données distincte a été utilisée. Le modèle YOLO exploité dans ce travail a été pré-entraîné sur un large ensemble de données incluant plusieurs catégories d'objets. Pour l'application au contexte du stationnement autonome, le modèle a été adapté pour cibler prioritairement les voitures et les cônes de signalisation.

Le format d'annotation YOLO requiert, pour chaque image, un fichier texte associé contenant les coordonnées des boîtes englobantes normalisées. Le tableau suivant présente les 24 classes utilisées :

Tableau 3. 2. Classes et identifiants de la base de données YOLO (24 classes).

Etiquette	**Classes**	**Etiquette**	**Classes**	**Etiquette**	**Classes**
0	Cône	8	Stop sign	16	Backpack
1	Car (Voiture)	9	Parking meter	17	Umbrella
2	Bicycle	10	Bench	18	Handbag
3	Motorcycle	11	Cat	19	Potted plant
4	Bus	12	Dog	20	Refrigerator
5	Truck	13	Horse	21	Keyboard
6	Traffic light	14	Sheep	22	Cell phone
7	Fire hydrant	15	Cow	23	Person

La base de données YOLO comprend 2 121 images utiles et 9 418 instances annotées au total. 315 images corrompues ont été détectées et automatiquement ignorées lors de l'entraînement. Ça veut dire que :

- Les 2 121 images utiles ; c'est le nombre total de photos que le modèle a pu réellement lire et utiliser pour apprendre. Chaque photo est une scène complète qui peut contenir un ou plusieurs objets.
- Les 9 418 instances annotées ; c'est le nombre total d'objets dessinés (annotés avec des boîtes englobantes) dans toutes ces images. Par exemple, si une seule photo contient 2 voitures et 1 cône, ça compte pour 3 instances. Donc en moyenne, chaque image contient environ 4 à 5 objets annotés.
- Les 315 images corrompues ; ce sont des fichiers images qui existent dans le dossier mais qui sont illisibles ou vides (pas de données pixel réelles). Elles représentent environ 13 % du dataset original. YOLO les a détectées automatiquement au démarrage de l'entraînement et les a ignorées, donc elles n'ont eu aucun impact sur le modèle.

Les tests du modèle YOLO ont été réalisés en conditions réelles à l'aide d'une caméra USB connectée à la Raspberry Pi 4, capturant des images à une résolution de 640×480 pixels. Les scénarios de test ont été conçus pour évaluer :

- La capacité de détection en présence de plusieurs obstacles simultanément.
- La robustesse face aux variations d'éclairage et aux occultations partielles.
- La précision des boîtes englobantes sur les objets détectés.
- La vitesse de traitement (inférence en temps réel) sur la plateforme embarquée.

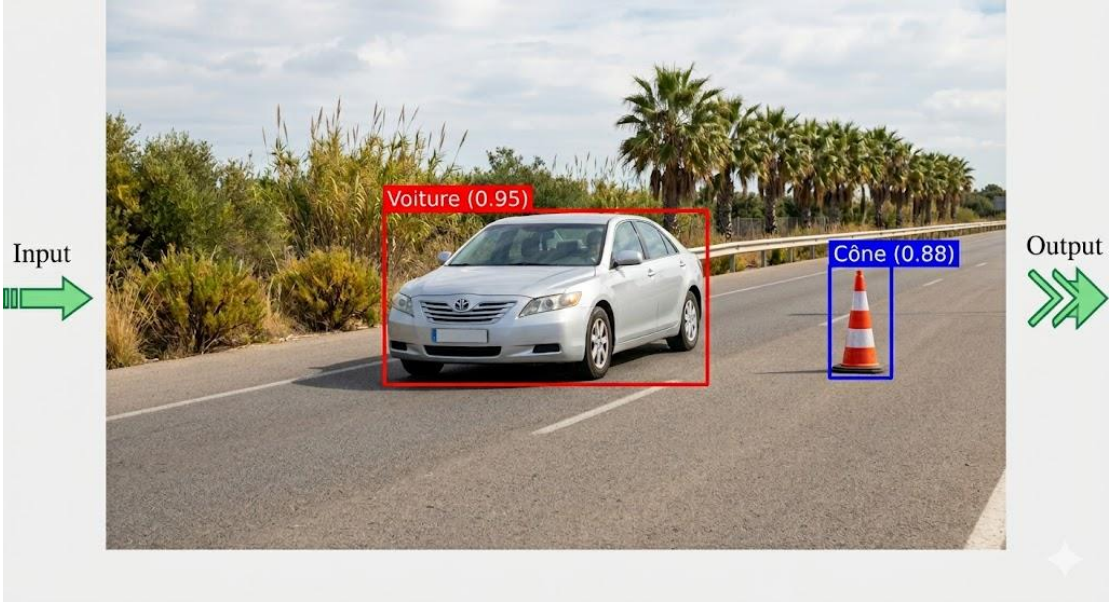

Figure 3. 5 . Exemple de détection YOLO. (AI)

3.5 Implémentation sur Raspberry Pi 4

3.5.1 Matériels utilisés

La réalisation du système de détection d'obstacles et d'assistance au stationnement repose sur plusieurs composants matériels travaillant de manière complémentaire.

Tableau 3. 3. Composants matériels du système embarqué.

Composants	**Rôle dans le système**	**Caractéristiques principales**
Raspberry Pi 4 Model B	Unité centrale de traitement	Quad-Core 1,5 GHz, 8 Go RAM, GPIO / USB / UART.
Caméra USB	Acquisition du flux vidéo (données 2D)	Résolution 640 × 480 px, flux continu
Capteur HC-SR04	Mesure ultrasonique de distance (données 1D)	Portée 2 cm – 4 m, calcul par temps de vol acoustique
Micro-Lidar-TF-Luna	Mesure optique de distance (données 1D)	Portée jusqu'à 8 m, interface UART 115 200 bauds

L'association de ces différents composants permet de disposer d'un système de perception complet combinant la vision artificielle et la mesure de distance. Cette architecture matérielle constitue la base de la stratégie de fusion de données mise en œuvre pour améliorer la sécurité et la fiabilité du stationnement autonome.

La carte Raspberry Pi 4 Model B constitue l'unité centrale du système. Son processeur Quad-Core cadencé à 1,5 GHz et ses interfaces de communication intégrées permettent de connecter simultanément la caméra, le capteur ultrasonique et le micro-LiDAR, tout en assurant l'exécution du modèle YOLOv8n et l'affichage des résultats en temps réel.

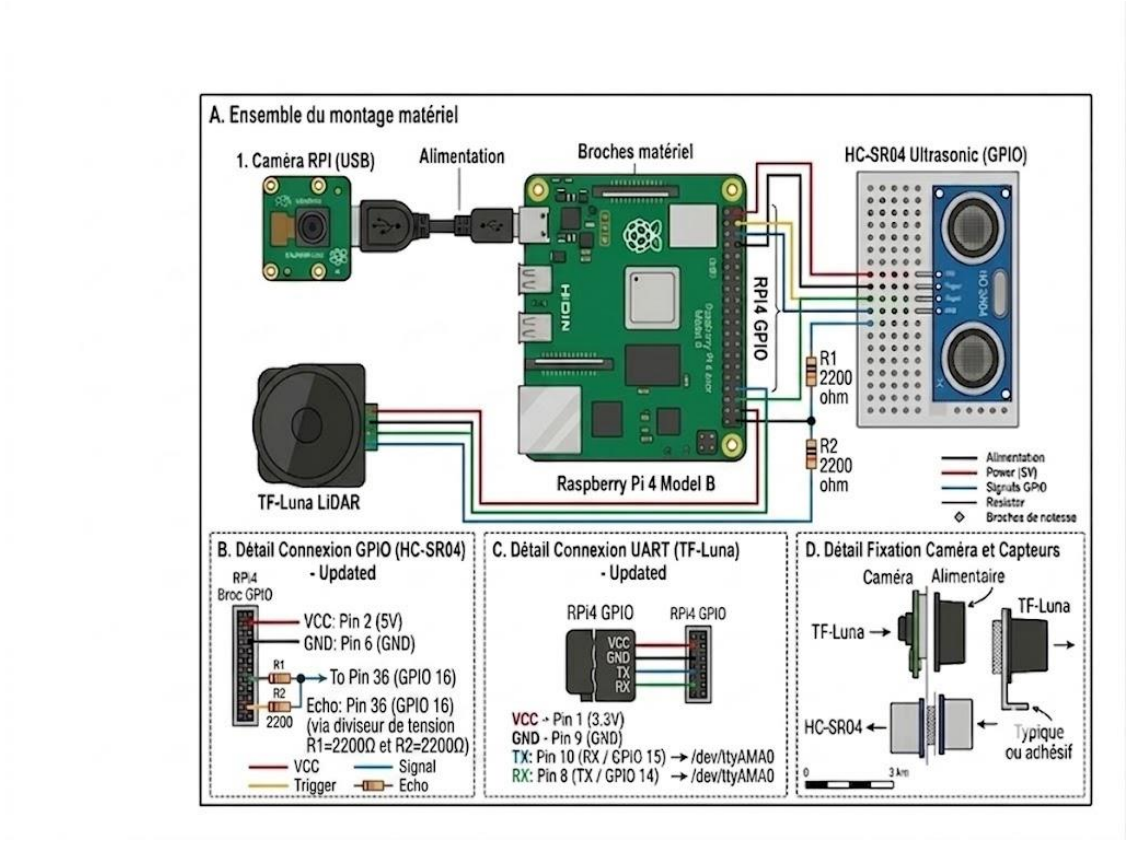

Figure 3. 6. Architecture matérielle du système embarqué.

3.5.2 Environnement logiciel

L'implémentation algorithmique du système a été entièrement réalisée en langage Python. Ce choix est motivé par la simplicité de développement offerte par ce langage ainsi que par la disponibilité de nombreuses bibliothèques dédiées à la vision par ordinateur, à l'intelligence artificielle et à l'interfaçage matériel. Les principales bibliothèques utilisées sont :

- **OpenCV (Open Source Computer Vision Library)** : utilisée pour l'acquisition des images à partir de la caméra USB, le traitement des images ainsi que l'affichage des résultats de détection, notamment les boîtes englobantes et les informations textuelles.
- **Ultralytics YOLO** : utilisée pour charger le modèle YOLO et réaliser la détection ainsi que la classification des obstacles en temps réel.
- **RPi.GPIO** : utilisée pour configurer et piloter les broches GPIO de la Raspberry Pi 4 afin d'acquérir les mesures du capteur ultrasonique.
- **Threading** : bibliothèque native de Python permettant l'exécution parallèle de plusieurs tâches, notamment la lecture continue des données du capteur ultrasonique et le traitement des images, afin d'améliorer la réactivité du système.

3.5.3 Architecture logicielle Multi-threading

L'exécution simultanée du modèle YOLO et des capteurs de distance nécessite une architecture logicielle asynchrone. En effet, l'inférence du modèle de Deep Learning exige un temps de calcul CPU relativement important qui, dans une exécution séquentielle classique, bloquerait l'acquisition des données capteurs. Trois threads parallèles synchronisés par des verrous threading.Lock ont donc été mis en place :

Figure 3. 7. Architecture multi-thread du système embarqué.

La distance ultrasonique est calculée selon la relation :

$$d = \frac{v \times t}{2} \qquad \textit{Equation 3. 1}$$

Où :

- d est la distance mesurée (m), v = 343 m/s la vitesse du son dans l'air à température ambiante, et t le temps aller-retour de l'onde ultrasonique (s).

3.5.4 Fusion des données 1D et 2D

La fusion de données 1D et 2D consiste à combiner des informations provenant de capteurs de nature différente afin d'obtenir une perception plus complète et plus fiable de l'environnement. Les données 2D fournissent généralement des informations visuelles telles que la détection et la localisation des objets dans une image, tandis que les données 1D apportent des mesures scalaires, comme la distance entre le capteur et l'obstacle.

Tableau 3. 4 . Logique de décision selon la distance mesurée.

Distance mesurée d	**Statut du système**	**Couleur affichée**	**Action du système**
$d \geq 100cm$	SAFE	Vert	Zone sécurisée, aucune alerte.
$50cm \leq d < 100cm$	WARNING	Orange	Approche d'un obstacle, alerte préventive.
$d < 50cm$	DANGER ! STOP !	Rouge	Alerte critique et affichage de « OBSTACLE PROCHE ! »

La distance finale est déterminée par la fonction d'arbitrage dynamique get_best_distance, qui applique trois règles de priorité :

- **Priorité en zone aveugle ($\boldsymbol{d < 20\ cm}$) :** L'ultrasonique est prioritaire, le LiDAR perdant en fiabilité en dessous de cette limite.
- **Principe du consensus ($\boldsymbol{\Delta d < 30\ cm}$) :** Si les deux capteurs retournent simultanément des valeurs viables, le système calcule leur écart absolu :

  $\boldsymbol{\Delta d = |d_{LIDAR} - d_{Us}|}$ Si cet écart est inférieur à 30 cm, le système valide la cohérence des mesures et sélectionne, par mesure de sécurité automobile, la valeur minimale.

- **Arbitrage optique** ($\Delta \boldsymbol{d} \geq \mathbf{30}\ \boldsymbol{cm}$) **:** En cas de forte divergence ou de perturbation acoustique, le système élimine l'erreur de l'ultrason et fait prioritairement confiance à la précision et à la forte directivité du faisceau laser du LiDAR.

3.6 Résultats expérimentaux :

3.6.1 Résultats du classifieur HOG-SVM

Les performances du classifieur HOG-SVM ont été évaluées sur 36 images de test non utilisées lors de l'entraînement. Les résultats sont exprimés à travers la matrice de confusion et les métriques standardisées.

Définition de chaque identificateur :

- VP (vrai positif) Image de voiture correctement classifiée comme voiture.
- VN (vrai négatif) Image de cône correctement classifiée comme cône.
- FP (faux positif) Image de cône classifiée incorrectement comme voiture.
- FN (faux négatif) Image de voiture classifiée incorrectement comme cône.

Tableau 3. 5. Matrice de confusion du classifieur HOG-SVM

	Voiture	**Cône**	**Total réel**
Réel : Voiture	12 VP	3 FN	15
Réel : Cône	1 FP	20 VN	21
Total	13	23	36

Tableau 3. 6. Indicateurs métriques du classifieur HOG-SVM

Indicateur	**Formule**	**Valeur**
Exactitude (accurancy)	(VP+VN)/Total	88,89%
Précision (precision)	VP/(VP+FP)	92,31%
Rappel (recall)	VP/(VP+FN)	80,00%
Spécificité	VN/(VN+FP)	95,24%

Score F1	2*(P*R)/(P+R)	85,71%

Ces résultats montrent que le classifieur HOG-SVM atteint une exactitude globale de 88,89 % sur la base de test. La précision élevée (92,31 %) et la spécificité de 95,24 % indiquent un faible taux de fausse alarme. Le rappel de 80,00 % révèle que trois voitures ont été mal classées comme des cônes, principalement en raison de variations d'angle de vue et de ressemblances morphologiques. Le score F1 de 85,71 % confirme un équilibre satisfaisant entre précision et rappel dans des conditions contrôlées.

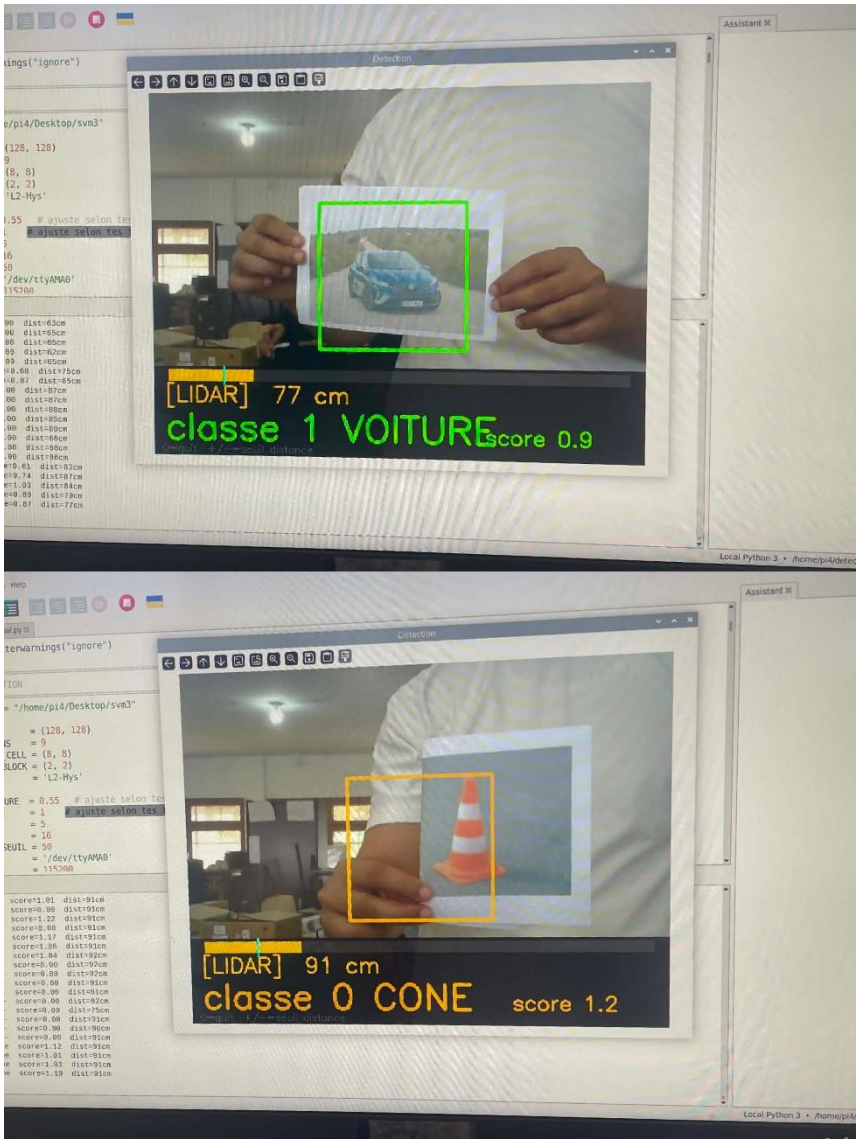

Figure 3. 8. Photo réelle des résultats du classifieur HOG-SVM.

3.6.2 Résultats du modèle YOLOv8n

Tableau 3. 7. Résultats globaux YOLOv8n.

Métrique / Indicateur	Précision	Rappel	mAP@50	mAP@50-95	Score F1	Vitesse d'inférence
Valeur	87,3%	70 ,8%	78,6%	61,9%	78,1%	2,5 ms/image
Observation	Faible taux de fausse alarme global	Certains objets difficiles non détectés	Bonne performance à seuil IoU 50 %	Marge d'amélioration à seuils IoU stricts	Bon équilibre précision / rappel global.	Compatible avec les exigences temps réel

Tableau 3. 8. Métriques de performance par classe.

	Classe	Images	Instances	Précision	Rappel	mAp@50	mAp@50-95
1	Person	241	5069	93,5%	68,8%	83,8%	62,3 %
2	Bicycle	70	160	80,3%	61,9%	73,8%	54,7%
3	Car	244	858	82,3%	54,7%	68,3%	46,1%
4	Motorcycle	75	171	78,5%	63,7%	73,4%	52,2%
5	Bus	75	105	94,0%	75,2%	86,0%	73,7%
6	Truck	118	186	78,9%	61,8%	74,4%	56,9%
7	Traffic light	88	260	89,1%	47,3%	56,3%	34,6%
8	Fire hydrant	22	26	94,0%	73,1%	80,4%	69,1%
9	Stop sign	24	26	92,2%	90,7%	90,9%	77,5%
10	Parking meter	17	30	84,3%	56,7%	61,9%	49,0%
11	Bench	110	158	92,9%	53,8%	67,6%	50,7%
12	Cat	81	89	97,7%	95,3%	99,0%	88,6%

13	Dog	84	109	84,1%	85,3%	90,9%	75,5%
14	Horse	58	119	91,8%	80,7%	89,2%	70,6%
15	Sheep	31	190	82,0%	79,4%	86,2%	63,5%
16	Cow	46	137	91,6%	87,6%	93,3%	75,2%
17	Backpack	62	166	69,9%	41,9%	50,7%	34,2%
18	Umbrella	85	194	85,6%	73,7%	81,7%	63,3%
19	Handbag	114	239	81,2%	41,8%	53,3%	37,1%
20	Potted plant	78	123	88,2%	84,6%	88,9%	66,3%
21	Refrigerator	33	39	84,8%	94,9%	93,7%	81,4%
22	Keyboard	96	140	92,7%	45,7%	52,9%	39,4%
23	Cell phone	49	50	90,9%	89,8%	93,8%	79,8%
24	Cone	220	774	96,2%	91,7%	96,9%	83,5%
-	Moyenne	2121	9418	87,3%	70,8%	78,6%	61,9%

Classes prioritaires — Cône : mAP@50 = 96,9 %, mAP@50-95 = 83,5 %, Précision 96,2 %, Rappel 91,7 %. Voiture (car) : mAP@50 = 68,3 %, mAP@50-95 = 46,1 %, Précision 82,3 %, Rappel 54,7 %.

Tableau 3. 9. Classement des 24 classes par mAP@50-95.

Rang	**Classe**	**mAP@50-95**	**Niveau**
1	Cat	88,6%	Excellent
2	Cône	83,5%	Excellent
3	Refrigerator	81,4%	Excellent
4	Cell phone	79,8%	Bon
5	Stop sign	77,5%	Bon
6	Dog	75,5%	Bon
7	Cow	75,2%	Bon
8	Bus	73,7%	Bon

9	Horse	70,6%	Moyen
10	Fire hydrant	69,1%	Moyen
11	Potted plant	66,3%	Moyen
12	Umbrella	63,5%	Moyen
13	Sheep	63,5%	Moyen
14	Person	62,3%	Moyen
15	Truck	56,9%	Moyen
16	Bicycle	54,7%	Moyen
17	Motorcycle	52,2%	Moyen
18	Bench	50,7%	Moyen
19	Parking meter	49,0%	Faible
20	Car	46,1%	Faible
21	Keyboard	39,4%	Faible
22	Handbag	37,1%	Faible
23	Traffic light	34,6%	Faible
24	backpack	34,2%	Faible

Les meilleures performances sont obtenues pour les classes cat (88,6 %), cone (83,5 %) et refrigerator (81,4 %), tandis que les classes backpack (34,2 %), traffic light (34,6 %) et handbag (37,1 %) restent les plus faibles, en raison de leur grande variabilité d'apparence et de leur nombre limité d'instances d'entraînement.

Figure 3. 9. Photo réelle des résultats de YOLOv8n.

3.6.3 Comparaison HOG-SVM / YOLO + Fusion multi-capteurs

Afin d'évaluer les performances des méthodes étudiées dans ce travail, une comparaison a été réalisée entre l'approche classique basée sur le descripteur HOG associé au classifieur SVM et l'approche basée sur le modèle de Deep Learning YOLO couplé à une stratégie de fusion multi-capteurs. Le tableau suivant présente les principales différences observées entre les deux approches.

Tableau 3. 10 Comparaison des deux approches

Critère	HOG + SVM	YOLO + Fusion multi-capteurs
Extraction des caractéristiques	Manuelle à partir du descripteur HOG	Automatique par réseau de neurones convolutifs
Détection des objets	Classification basée sur les caractéristiques extraites	Détection et classification simultanées
Gestion de plusieurs objets	Limitée	Possible dans une même image
Sensibilité aux variations d'éclairage	Élevée	Plus faible grâce à l'apprentissage profond
Localisation des objets	Précision variable	Boîtes englobantes plus précises
Intégration des capteurs de distance	Peu adaptée	Intégration directe dans le système
Fonctionnement temps réel	Limité sur des scènes complexes	Adapté au traitement embarqué après optimisation

3.7 Interprétation des résultats

Ce troisième chapitre a présenté la conception, la réalisation et la validation expérimentale du système de détection d'obstacles développé dans le cadre de ce mémoire. Deux approches complémentaires ont été mises en œuvre, évaluées et comparées.

La première approche, basée sur les descripteurs HOG associés au classifieur SVM, a permis d'obtenir une exactitude de 88,89 % sur la base de test binaire (voiture / cône), avec une précision de 92,31 % et une spécificité de 95,24 %. Ces résultats sont satisfaisants dans des conditions contrôlées, mais les limites de la méthode — sensibilité aux variations d'éclairage, absence de détection multi-objets, extraction manuelle des caractéristiques — ont motivé le développement d'une approche plus avancée.

La seconde approche, reposant sur le modèle YOLOv8n entraîné sur 24 classes et déployé sur Raspberry Pi 4, offre une détection plus robuste et adaptée aux exigences du temps réel. Le modèle atteint une mAP@50 de 78,6 % globalement et 96,9 % sur la classe cône spécifiquement.

L'intégration d'une architecture logicielle multi-thread et la fusion des données de la caméra, du capteur ultrasonique HC-SR04 et du micro-LiDAR TF-Luna ont renforcé la fiabilité globale du système, assurant la continuité de la surveillance même dans des conditions visuelles dégradées.

Au regard de l'ensemble des résultats obtenus, l'architecture YOLO associée à la fusion multi-capteurs constitue la solution la plus robuste et la plus adaptée aux contraintes d'un système embarqué destiné à l'assistance au stationnement autonome.

3.8 Conclusion

En somme, ce chapitre a permis de valider expérimentalement les performances de nos algorithmes et l'efficacité de notre stratégie de fusion multi-capteurs. Si l'approche HOG-SVM a montré des limites factuelles, le couplage entre YOLOv8n et nos capteurs de distance (HC-SR04 et TF-Luna) remplit pleinement les exigences de précision et de temps réel fixées pour notre système embarqué. Ces résultats probants valident l'ensemble des travaux menés tout au long de ce projet et nous permettent d'ouvrir la voie à la conclusion générale de ce mémoire.

CONCLUSION GENERALE

Conclusion générale :

Ce mémoire a présenté la conception et la réalisation d'un système embarqué de détection d'obstacles dédié à l'assistance au stationnement autonome. L'ensemble des travaux menés, depuis l'étude théorique jusqu'à la validation expérimentale, a permis de répondre à quelques exigences stationnement autonome.

Dans un premier temps, une approche classique combinant le descripteur HOG et le classifieur SVM a été développée pour la classification binaire des obstacles (voitures et cônes de signalisation). Cette méthode a atteint une exactitude de 88,89 % sur la base de test, avec une précision de 92,31 % et une spécificité de 95,24 %, démontrant sa pertinence dans des conditions contrôlées. Toutefois, ses limites intrinsèques — sensibilité aux variations d'éclairage, extraction manuelle des caractéristiques et incapacité à détecter simultanément plusieurs objets — ont motivé le développement d'une approche plus avancée.

Dans un second temps, le modèle de Deep Learning YOLOv8n a été entraîné sur une base de données de 24 classes et déployé sur la plateforme embarquée Raspberry Pi 4. Les résultats obtenus sont significativement supérieurs, avec une mAP@50 globale de 78,6 %, atteignant 96,9 % sur la classe cône — obstacle prioritaire dans le contexte du stationnement. La vitesse d'inférence de 2,5 ms par image confirme la compatibilité du modèle avec les exigences d'un traitement en temps réel sur architecture embarquée.

Afin de renforcer la fiabilité de la perception, une architecture de fusion multi-capteurs a été mise en place, combinant les données visuelles de la caméra USB avec les mesures de distance fournies par le capteur ultrasonique HC-SR04 et le micro-LiDAR TF-Luna. Une logique d'arbitrage dynamique a été développée pour sélectionner en permanence la mesure la plus fiable selon les conditions d'utilisation. Cette fusion confère au système une robustesse accrue, notamment dans les situations où la perception visuelle seule se révèle insuffisante.

L'ensemble des résultats expérimentaux valide l'approche proposée et confirme que la combinaison de YOLOv8n avec la fusion multi-capteurs constitue une solution robuste, précise et adaptée aux contraintes d'un système embarqué d'assistance au stationnement autonome.

En perspective, plusieurs axes d'amélioration peuvent être envisagés pour enrichir ce travail. L'intégration d'un module de planification de trajectoire permettrait de passer d'un système de détection à un système d'assistance complet. L'enrichissement de la base de données d'entraînement, notamment pour les classes présentant de faibles performances (feux de

circulation,…), contribuerait à améliorer la robustesse du modèle dans des environnements urbains diversifiés. Enfin, le passage à des architectures matérielles plus puissantes ou l'adoption de techniques d'optimisation telles que la quantification des modèles permettrait d'accroître encore la vitesse de traitement et d'ouvrir la voie à des applications de conduite plus avancées.

REFERENCES BIBLIOGRAPHIQUES

Références bibliographiques

[1] Houenou, Adam. Calcul de trajectoires pour la préconisation de manœuvres automobiles sur la base d'une perception multi-capteur : application à l'évitement de collision. HAL Thèses. [En ligne] 2013. https://theses.hal.science/tel-00977389/file/These_UTC_Adam_Houenou.pdf.

[2] PARK'UP Systems. Parking de surface. PARK'UP Systems. [En ligne] https://theses.hal.science/tel-00977389/file/These_UTC_Adam_Houenou.pdf .

[3] Université Badji Mokhtar Annaba. Etude, conception et réalisation d'un système de stationnement intelligent. Bibliothèque université Annaba. [En ligne] 2022. https://theses-algerie.com/9105486751418249/memoire-de-master/universite-abderrahmane-mira-bejaia/conception-d-un-syst%C3%A8me-de-stationnement-intelligent-en-utilisant-des solutions-ido-et-big-data#google_vignette.

[4] Lift système. Parking automatique. Lift système. [En ligne] https://theses.hal.science/tel-00977389/file/These_UTC_Adam_Houenou.pdf .

[5] SAE International. Taxonomy and Definitions for Terms Related to Driving Automation systems for On-Road Motor Vehicles. SAE International. [En ligne] SAE International, 30 avril 2021. https://www.sae.org/standards/j3016_202104-taxonomy-definitions-terms-related-driving-automation-systems-road-motor-vehicles .

[6] état de vaud - direction générale de l'enseignement obligatoire. Rapport P6 2020/04. Lausanne : état de vaud, 2020.

[7] Nio, Marceau. Stationnement autonome : le nouveau terrain de jeu des constructeurs. Eco Motors News. [En ligne] Eco Motors News , 23 Janvier 2026. https://www.ecomotorsnews.com/expertises/stationnement-autonome-le-nouveau-terrain-de-jeu-des-constructeurs .

[8] Jouvin, Bernard. Stationnement autonome : la fin des créneaux cauchemardesques, c'est pour bientÔt ? le Dauphiné libéré. [En ligne] Le Dauphiné libéré, 22 septembre 2021. https://www.ledauphine.com/magazine-automobile/2021/09/22/stationnement-autonome-la-fin-des-creneaux-cauchemardesques-c-est-pour-bientot.

[9] Zhang, Peizhi, et al. Reinforcement Learning-Based End-to-End Parking for automatic Parking System. 2019, Vol. 19, 18.

[10] Bassy, Curtis. Comment réussir son stationnement : nos conseils. Ornikar. [En ligne] Ornikar, 13 02 2026. : https://www.ornikar.com/permis/conseils-conduite/stationnement/reussir?srsltid=AfmBOoqdzU0LxEAyUuk610TSS257Wm3Ssd1dJ8ABbjQj65Loe_AZdZi .

[11] Bradski G, Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media; 2008.

[12] Bauer D, Woods G, Chodorowski A, Manutchehr-Danai M. Energy-Efficient HOG-based Object Detection at 1080HD 60 fps with Multi-Scale Support. In : IEEE International Workshop on Signal Processing Systems (SiPS) ; 2019.

[13] Dalal N, Triggs B. Histograms of Oriented Gradients for Human Detection. In : Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) ; 2005.

[14] ML Science. Histogram of Oriented Gradients [En ligne]. Disponible sur : https://www.ml-science.com/histogram-of-oriented-gradients [Consulté en 2026].

[15] Towards AI. Fully Explained SVM Classification with Python [En ligne]. Disponible sur : https://pub.towardsai.net/fully-explained-svm-classification-with-python-eda124997bcd [Consulté en 2026].

[16] Liora. Méthode Kernel : tout savoir [En ligne]. Disponible sur : https://liora.io/methode-kernel-tout-savoir [Consulté en 2026].

[17] Liora. Définition d'un dataset [En ligne]. Disponible sur : https://liora.io/dataset-definition [Consulté en 2026].

[18] Blent AI. Détection d'images avec YOLO et TensorFlow [En ligne]. Disponible sur : https://blent.ai/blog/a/detection-images-yolo-tensorflow [Consulté en 2026].

[19] La Revue IA. YOLO : le framework de référence en vision par ordinateur [En ligne]. Disponible sur : https://larevueia.fr/yolo/ [Consulté en 2026].

[20] ResearchGate. Overview of the YOLOv8 model architecture [En ligne]. Disponible sur : https://www.researchgate.net/figure/Overview-of-the-Yolov8-model-architecture-The-head-neck-and-backbone-are-the-three_fig3_384068378 [Consulté en 2026].

ANNEXES

Annexe A- Fiches techniques des composants

A.1 Raspberry Pi 4 model B

Carte de calcul embarqué utilisée comme unité centrale de traitement (CPU/GPU) pour l'exécution des algorithmes HOG+SVM et YOLOv8n, la fusion de capteurs et la gestion des entrées/sorties GPIO.

Tableau A. 1. Fiche technique Raspberry Pi 4 model B

Paramètre	**Valeur**
Processeur	ARM Cortex-A72 @ 1,8 GHz — Quad-Core 64 bits
RAM	8 Go LPDDR4X-3200
GPU	VideoCore VI (OpenGL ES 3.x, H.265 4K decode)
Connectivité	2× USB 3.0, 2× USB 2.0, 2× HDMI micro, Ethernet Gigabit, WiFi 802.11ac, BT 5.0
GPIO	40 broches (UART, I2C, SPI, PWM, 5V, 3V3, GND)
Alimentation	5 V / 3 A via USB-C (15 W minimum recommandé)
Alimentation	Raspberry Pi OS (Bullseye) 64 bits
Température de fonct.	0 °C à 85 °C
Dimension	85 mm × 56 mm × 17 mm

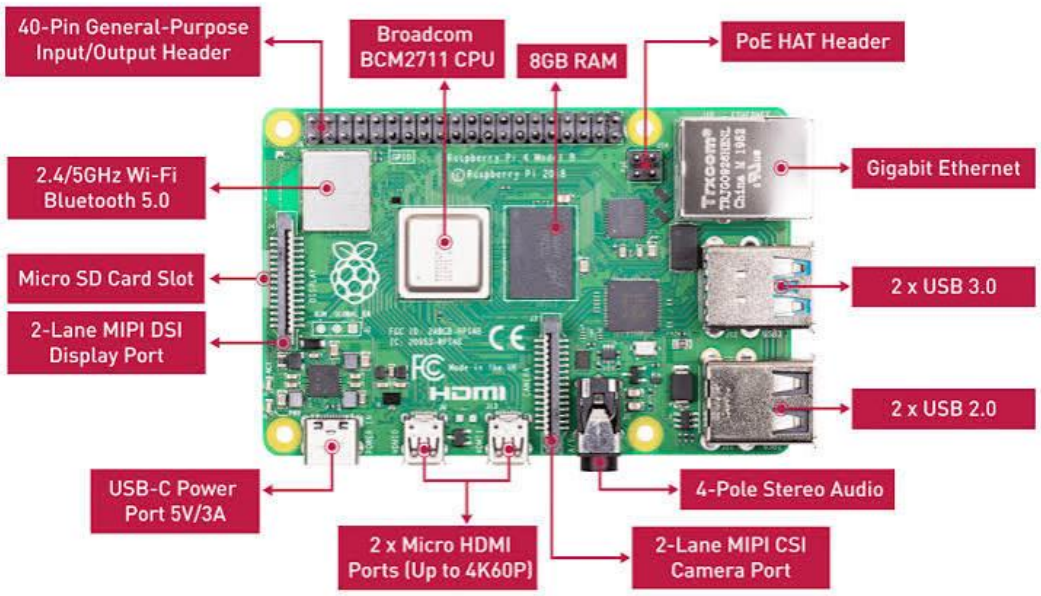

Figure A. 1. Raspberry Pi 4 model b

A.2 Capteur Ultrasonique HC-SR04

Utilisé pour la détection de proximité à courte portée (< 50 cm). Opère en parallèle avec le TF-Luna grâce à la stratégie de fusion de capteurs.

Tableau A. 2. Fiche technique de capteur HC-SR04

Paramètre	**Valeur**
Tension d'alimentation	5 VDC
Courant de veille	< 2 mA
Courant en fonctionnement	15 mA
Fréquence ultrasonique	40 kHz
Plage de mesure	2 cm — 400 cm
Résolution	0,3 cm
Angle d'ouverture	< 15°

Signal TRIG	Impulsion TTL 10 µs (niveau haut 5 V)
Signal ECHO	5 V (pont diviseur requis pour GPIO 3,3 V du RPi)
Formule distance	Distance (cm) = durée_ECHO (s) × 17 150
Dimensions	45 mm × 20 mm × 15 mm
GPIO utilisés (projet)	TRIG → GPIO 5 (pin 29) | ECHO → GPIO 16 (pin 36)

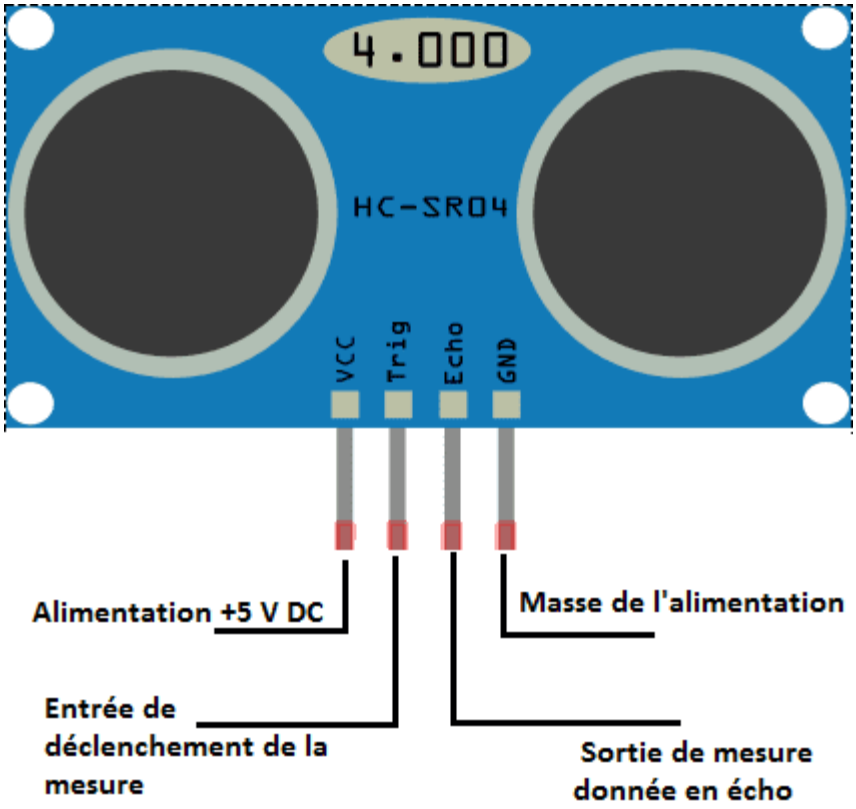

Figure A. 2. HC-SR04

A.3 Capteur LiDAR TF-Luna

Télémètre infrarouge ToF (Time-of-Flight), utilisé pour les mesures de distance longue portée dans la fusion de capteurs.

Tableau A. 3.Fiche technique de capteur LiDAR TF-Luna

Paramètre	**Valeur**
Tension d'alimentation	5 VDC ± 0,1 V
Courant moyen	130 mA (typ.) / 350 mA (max.)
Plage de mesure	0,2 m — 8 m (conditions normales)
Précision	± 6 cm (0,2–3 m) | ± 2 % (> 3 m)
Fréquence de trame	1 — 250 Hz (par défaut 100 Hz)
Longueur d'onde laser	850 nm (infrarouge, classe 1 — œil sûr)
Interface	UART — 115 200 baud, 8 bits, pas de parité, 1 bit stop
Format de trame	9 octets : 0x59 0x59 | Dist_L | Dist_H | Str_L | Str_H | Temp_L | Temp_H | CS
Condition de validité	Strength > 100 et 20 ≤ dist ≤ 800 cm (filtrage logiciel)
Température de fonct.	-10 °C à 60 °C
Connexion RPi (projet)	Rouge → 5V (pin 2) | Bleu1 → TXD GPIO14 (pin 8) | Bleu2 → RXD GPIO15 (pin 10) | Noir → GND
Port série utilisé	/dev/ttyAMA0 — 115 200 baud
Dimensions	35 mm × 21,25 mm × 13,5 mm

Figure A. 3 . LiDAR TF-Luna

A.4 Caméra USB (640 × 480)

Caméra USB générique utilisée pour l'acquisition d'images en temps réel, alimentant les pipelines HOG+SVM et YOLOv8n.

Tableau A. 4. Fiche technique du caméra USB (680*480)

Paramètre	**Valeur**
Interface	USB 2.0 (UVC — USB Video Class)
Résolution utilisée	640 × 480 pixels (VGA)
Fréquence d'images	30 fps (configuré via CAP_PROP_FPS)
Espace de couleur	BGR (OpenCV) → Niveaux de gris pour HOG

Pilote	v4l2 (Video4Linux) — natif Raspberry Pi OS
API utilisée	cv2.VideoCapture(index) — index 0 à 4 (auto-détection)

Annexe B — Le coût des composants

Les prix sont des estimations basées sur le marché algérien (DZD) au moment du développement du projet. Les prix peuvent varier selon le fournisseur.

Tableau B. 1. Tableau des coûts

Composant / Matériel	**Qté**	**Prix Unitaire**	**Total**
Raspberry Pi 4 Model B (4 GB)	1	~3 7000 DA	~3 7000 DA
Capteur ultrasonique HC-SR04	1	~650 DA	~650 DA
Capteur LiDAR TF-Luna (UART)	1	~6 800 DA	~6 800 DA
Caméra USB (640×480)	1	~2 800 DA	~2 800 DA
Câbles Dupont / Fils de liaison	1	~250 DA	~250 DA
Circuit imprimé	1	~400 DA	~400 DA
Pont diviseur de tension (résistances)	2	~100 DA	~100 DA
TOTAL GÉNÉRAL (estimation)			~4 8000 DA

Annexe C — Etapes de programmation et bibliothèques Python utilisées

Annexe C.1 — Étapes de Programmation : Pipeline HOG et SVM

Le programme HOG+SVM suit un pipeline séquentiel en 7 étapes, de la collecte des données jusqu'au déploiement temps réel sur Raspberry Pi 4.

C.1.1 Vue d'ensemble du pipeline

Tableau C. 1. Les étapes de programmation HOG+SVM

N°	Étape	Sortie produite
1	Collecte et organisation des images	117 images classées (49 voitures / 68 cônes)
2	Prétraitement des images	Images 128×128 px en niveaux de gris
3	Extraction des descripteurs HOG	Vecteurs de 8 100 valeurs par image
4	Normalisation (StandardScaler)	Features centrées-réduites (μ=0, σ=1)
5	Entraînement SVM (kernel linéaire)	Modèle svm_model.pkl sauvegardé
6	Évaluation sur jeu de test	Matrice de confusion + métriques (88,89 % accuracy)
7	Déploiement temps réel sur RPi 4	Classification en continu via caméra USB

C.1.2 Bibliothèques utilisées — HOG et SVM

Tableau C. 2. Bibliothèques utilisées dans HOG+SVM

Bibliothèque	Version	Rôle
cv2 (OpenCV)	4.x	Lecture/écriture d'images, conversion BGR→Gris, redimensionnement

numpy (np)	≥1.21	Manipulation des vecteurs HOG, calcul matriciel, statistiques
joblib	≥1.1	Sauvegarde / chargement des fichiers .pkl (SVM, Scaler, Classes)
skimage.feature.hog	scikit-image ≥0.19	Extraction du descripteur HOG (9 orientations, cellules 8×8)
sklearn.svm.SVC	scikit-learn ≥1.0	Entraînement du classifieur SVM à noyau linéaire
sklearn.preprocessing.StandardScaler	scikit-learn ≥1.0	Normalisation des vecteurs HOG (moyenne=0, écart-type=1)
sklearn.model_selection.train_test_split	scikit-learn ≥1.0	Division stratifiée 70% train / 30% test
sklearn.metrics	scikit-learn ≥1.0	Calcul des métriques : accuracy, precision, recall, F1, matrice de confusion
os / glob	stdlib	Parcours des dossiers, chargement des images par classe
time	stdlib	Mesure du temps d'entraînement et d'inférence

Commande d'installation :

pip install opencv-python numpy joblib scikit-image scikit-learn RPi.GPIO

Annexe C.2 — Étapes de Programmation : Pipeline YOLOv8n

Le pipeline YOLOv8n se déroule en deux phases distinctes : la phase d'entraînement (sur Google Colab avec GPU Tesla T4) et la phase de déploiement (sur Raspberry Pi 4).

C.2.1 Vue d'ensemble du pipeline

Tableau C. 3. Les étapes de programmation YOLOv8n

N°	Phase	Étape	Sortie produite
1	Préparation	Collecte et annotation du dataset	2 121 images annotées, format YOLO (data.yaml)
2	Entraînement	Entraînement YOLOv8n (50 époques)	Fichier best.pt (poids optimaux)
3	Évaluation	Calcul des métriques sur val set	mAP@50=78,6 %, mAP@50-95=61,9 %
4	Optimisation	Conversion NCNN (ARM optimisé)	Dossier 24classes_ncnn_model/
5	Déploiement	Chargement sur Raspberry Pi 4	Inférence temps réel ≤ 2,5 ms/image
6	Intégration	Fusion caméra + HC-SR04 + TF-Luna	Système complet avec alertes SAFE/WARNING/DANGER

C.2.2Bibliothèques utilisées — YOLOv8n

Tableau C. 4. Bibliothèques utilisées dans YOLOv8n

Bibliothèque	Version	Rôle
ultralytics (YOLO)	≥8.0	Chargement, entraînement, évaluation et exportation du modèle YOLOv8n
cv2 (OpenCV)	4.x	Capture vidéo USB, dessin des bounding boxes, affichage temps réel

os	stdlib	Gestion des chemins, variable QT_QPA_PLATFORM pour l'affichage
threading	stdlib	Thread parallèle pour la lecture du capteur HC-SR04
time	stdlib	Temporisation GPIO, timeout mesure ultrasonique, compteur FPS
RPi.GPIO	≥0.7	Contrôle GPIO : TRIG (pin 5) et ECHO (pin 16) du HC-SR04
torch / torchvision	≥2.0 (dép. YOLO)	Backend deep learning pour l'entraînement sur GPU (Tesla T4 / Google Colab)
yaml	stdlib	Lecture du fichier de configuration data.yaml (classes, chemins dataset)

Commandes d'installation :

Sur Google Colab (entraînement)

pip install ultralytics torch torchvision

Sur Raspberry Pi 4 (déploiement)

pip install ultralytics opencv-python RPi.GPIO

Annexe D- Métriques d'évaluation de la détection d'objets (mAP)

Dans le domaine de la vision par ordinateur et de la détection d'objets (comme avec les modèles YOLO), la métrique principale utilisée pour évaluer les performances d'un modèle est la mAP (mean Average Precision). Elle repose sur le calcul de l'IoU (Intersection over Union), qui mesure le taux de superposition entre la boîte englobante prédite par le modèle et la boîte réelle (*Ground Truth*).

D.1 mAP@50 (ou mAP@0.5)

La métrique **mAP@50** représente la précision moyenne globale du modèle lorsque le seuil d'IoU est fixé à **50 %** (0.5).

- **Fonctionnement :** Une prédiction est considérée comme un "Vrai Positif" (correcte) si elle chevauche la réalité d'au moins 50 %. Si le chevauchement est inférieur à 50 %, elle est comptée comme un "Faux Positif".
- **Interprétation :** C'est une métrique historiquement très utilisée (notamment dans le jeu de données PASCAL VOC). Elle est considérée comme indulgente, car elle évalue principalement la capacité du modèle à détecter la présence d'un objet et sa position globale, sans exiger que le détourage de la boîte soit parfaitement ajusté au pixel près.

D.2 mAP@50-95 (ou mAP@0.5:0.95)

La métrique **mAP@50-95** est une moyenne des mAP calculées sur 10 seuils d'IoU différents, allant de 50 % à 95 % avec un pas de 5 % :

$$\text{mAP@50-95} \equiv \frac{mAP@50 + mAP@55 + \dots mAP@90 + mAP@95}{10}$$

- **Fonctionnement :** Elle pénalise sévèrement les boîtes englobantes qui sont mal ajustées. Pour obtenir un score élevé à un seuil d'IoU de 95 %, la boîte prédite doit correspondre presque parfaitement à la réalité.
- **Interprétation :** C'est la métrique standard de référence aujourd'hui. Elle est beaucoup plus stricte et complète que la mAP@50, car elle valide à la fois l'exactitude de la classification de l'objet et la très haute précision géométrique de sa localisation.