



1 Rue de la Chebarde, 63178 Aubière



Campus Universitaire des Cézeaux, 4 Avenue Blaise Pascal, 63178 Aubière

Rapport de projet
Projet de troisième année
Filière : F4 - Modélisation mathématique et science des données

SEGMENTATION 3D AUTOMATIQUE DE SCÈNES ROUTIÈRES À PARTIR D'IMAGES RADAR MMW ET D'UNE CAMÉRA PAR DEEP LEARNING

Présenté par : Mathieu Cayssol, Hamza El-Moufakkir

Tuteur ISIMA :
Violaine Antoine

Date de la soutenance :
21/03/2022

Tuteurs Institut Pascal :
Omar Ait Aider, Charles Antoine Noury

Durée du projet :
120 heures par étudiant

Remerciements

Je tiens à remercier Dr. Omar Ait Aider et Dr. Charles Antoine Noury pour leur accueil au sein de l’Institut Pascal ainsi que leur suivi tout au long du projet.

Je tiens aussi également à remercier Mme Violaine Antoine, responsable de la filière F4.

Nous remercions Mme Murielle MOUZAT, notre professeur de communication, de nous avoir présenté les éléments essentiels à l’élaboration d’un rapport de projet et d’une présentation orale également.

Enfin, nous souhaitons également remercier notre famille et nos amis.

Résumé

Pour la conduite autonome, il est essentiel d'utiliser différentes modalités de capture de scènes routières. Cela permet au véhicule d'anticiper et de prendre les bonnes décisions. Il est ainsi nécessaire de mettre en place une architecture combinant les informations des différents capteurs.

Dans le cadre de ce projet, nous présentons une architecture utilisant des données caméra et des données radars pour la détection d'objets et l'estimation de la distance dans le cadre de la conduite autonome. Notre architecture peut se décomposer en 3 grandes parties. La première consiste à détecter sémantiquement les objets sur une image radar à l'aide d'un réseau de neurones (U-Net). La seconde partie consiste à détecter sémantiquement les objets sur une image caméra en utilisant un réseau neuronal convolutif basé sur les régions (RCNN). La troisième partie consiste à associer les voitures détectées sur l'image caméra aux voitures détectées sur l'image radar. Nous utilisons un réseau de neurones entièrement connecté (FCNN).

Nous avons entraîné le réseau U-Net sur plus de 1556 images radars annotées de l'Institut Pascal. Pour le réseau RCNN, nous avons utilisé un modèle pré entraîné sur la base de données COCO contenant plus de 328K images. Pour finir, nous avons entraîné le FCNN sur 2065 couples issus des images caméra segmentées et images radars segmentées de l'Institut Pascal. Nous avons utilisé Python, Numpy, Tensorflow, Keras ainsi que Google Colab. Ces différentes étapes nous permettent de présenter une preuve de concept (PoC). Celle-ci peut être améliorée en termes de performances en utilisant les métriques proposées dans ce rapport.

Mot-clés : Machine learning, Deep learning, Computer Vision, Autonomous Driving, Radar, Camera.

Github : <https://github.com/MathieuCayssol/ComputerVision-Radar-Camera>

Abstract

For autonomous driving, it is essential to use different modalities to capture road scenes. This allows the vehicle to anticipate and make the right decisions. It is thus necessary to set up an architecture combining information from different sensors.

In this project, we present an architecture using camera and radar data for object detection and distance estimation in the context of autonomous driving. Our architecture can be decomposed into 3 main parts. The first part consists in semantically detecting objects on a radar image using a U-Net. The second part consists in semantically detecting objects on a camera image using a region-based convolutional neural network (RCNN). The third part consists in associating the cars detected on the camera image with the cars detected on the radar image. We use a fully connected neural network (FCNN).

We trained the U-Net on more than 1556 annotated radar images from the Institut Pascal. For the RCNN we used a pre-trained model on the COCO database containing more than 328K images. Finally, we trained the FCNN on 2065 pairs of segmented camera images and segmented radar images from the Institut Pascal. We used Python, Numpy, tensorflow, keras and Google Colab. These different steps allow us to present a proof of concept (PoC), which can be improved in terms of performance using the metrics proposed in this report.

Keywords : Machine learning, Deep learning, Computer Vision, Autonomous Driving, Radar, Camera.

Github : <https://github.com/MathieuCayssol/ComputerVision-Radar-Camera>

Table des figures

1	Organigramme de la recherche à l'Institut Pascal	2
2	Véhicule équipé d'un radar à ondes millimétriques (MMW) pour l'analyse du terrain	4
3	Exemple d'image radar (gauche). Notez que la direction avant du véhicule est marquée par la ligne noire. Vue de face du véhicule (à droite), telle qu'acquise par une caméra située au même endroit	4
4	Principe simplifié de capture d'une image à l'aide d'une caméra digitale (ou appareil photographique numérique)	7
5	Exemple de données provenant du jeu de données NuScenes montrant le nuage de points radar (rouge), les boîtes de vérité du sol en 3D (vert) et le nuage de points LIDAR (gris).	9
6	L'architecture de réseau proposée. Les entrées du réseau sont le nuage de points radar, l'image de la caméra et les boîtes d'ancrage 3D. Les propositions d'objets basées sur le radar sont générées à partir du nuage de points et fusionnées avec les caractéristiques de l'image pour améliorer la localisation des boîtes.	10
7	Architecture simplifiée pour combiner les images radars et les images caméra pour la conduite autonome	11
8	Les outils de travail	12
9	Google Colab	12
10	Diagramme de Gantt prévisionnel	13
11	Diagramme de Gantt réel	13
12	Tenseur d'une image RGB avec les canaux rouge, vert et bleu	14
13	Example d'image caméra prise à la voiture à Aubière	15
14	Example d'image radar de notre jeu de données	16
15	Image caméra ($I_{547,camera}$)	17
16	Segmentation associée à l'image caméra ($S_{547,camera}$)	17
17	Segmentation de 5 masques de la classe "voiture" de l'image 547	17
18	Segmentation des poteaux de l'image 547	18
19	Segmentation des arbres de l'image 547	18
20	Image radar n°200 notée $I_{200,radar}$	18
21	Image radar segmentée n°200 notée $S_{200,radar}$	18
22	L'encadré en rouge représente la zone d'intérêt de l'image radar	19
23	Organisation générale des données	20
24	Pipeline de transformation de données schématisée	21
25	Différentes stratégies de recadrage	22
26	Recadrage très restrictif : 160x80	22
27	Recadrage moyennement restrictif : 272x80	23

28	Recadrage peu restrictif : 368x80	23
29	Architecture du réseau de neurone U-Net [16] (La taille des tenseurs n'est pas représentatif. Dans notre cas, le tenseur est de taille 368x80)	25
30	Schéma pour le Jaccard Index ou IoU score	26
31	Schéma de Sørensen-Dice ou Dice coefficient	26
32	Influence du taux d'apprentissage sur la convergence vers le minimum de la fonction de coût	27
33	Prédiction d'image 272x80. En haut : prédiction du réseau U-Net. En bas : vérité terrain.	28
34	Prédiction d'image 160x80. En haut : prédiction du réseau U-Net. En bas : vérité terrain.	29
35	Architecture du réseau Mask-R-CNN	29
36	Image caméra ($I_{547,camera}$)	30
37	Segmentation associée à l'image caméra ($S_{547,camera}$)	30
38	Images radars et caméra segmentées voiture par voiture. En haut : segmentations radar. En bas : segmentations caméra correspondantes	31
39	Résultat après le recadrage : débordement de pixels sur la segmentation des images caméra	32
40	Pipeline de données pour le réseau de neurones réalisant la transformation géométrique caméra vers radar	33
41	Architecture du réseau de neurones complètement connecté	34
42	Architecture de DeepLearning associant les images radars et les images caméra pour la conduite autonome	36

Liste des tableaux

1	Résultats de la segmentation des images radars	28
---	--	----

Liste des abréviations

MMW *Millimeter Waves*

LIDAR *Light Detection and Ranging*

RPN *Region Proposal Network*

PoC *Proof of Concept*

FCNN *Fully Connected Neural Network*

R-CNN *Region-based Convolutional Neural Network*

Glossaire

Bounding-Box. Boîte englobante en Computer Vision.

Pipeline (Pipeline de données). Ensemble d'éléments de traitement de données en série où la sortie d'une élément est l'entrée du suivant.

Table des matières

Introduction	1
1 Contexte du projet	2
1.1 L’Institut Pascal	2
1.2 Présentation du projet	2
1.2.1 Explication du projet et contexte	2
1.2.2 Le radar à ondes millimétriques (MMW)	3
1.2.3 La détection et télémétrie par la lumière (LiDAR)	5
1.2.4 La différence entre le radar MMW et le LIDAR	5
1.2.5 La modalité d’image caméra	6
1.2.6 Conclusion sur les différentes modalités	7
1.3 État de l’art	8
1.3.1 Modalités radar/camera	8
1.3.2 Proposition d’architecture	10
1.4 Outils de développement	11
1.5 Organisation des tâches	12
2 Conception et réalisation	14
2.1 Compréhension des données	14
2.1.1 Les images caméra	14
2.1.2 Les images radar	15
2.1.3 La segmentation des images caméra	16
2.1.4 La segmentation des images radars	18
2.2 Détection sémantique de voitures sur les images radars	19
2.2.1 Pipeline de données	20
2.2.2 Modèle de deeplearning pour la détection sémantique	24
2.2.3 Résultats et performances	27
2.3 Détection sémantique de voitures sur les images caméra	29
2.4 Combinaison des images radars et images caméra	30
2.4.1 Création de la base d’entraînement	31
2.4.2 Réseau de neurones	33
2.5 Présentation du PoC	35
Conclusion	37
Références	38

Introduction

Plus de 95% des accidents de la route sont causés par des erreurs humaines. Nous sommes faillibles, distraits et potentiellement dangereux. D'après le rapport IDTechEx de 2021, sur plus de 187 accidents impliquant une voiture autonome, seulement 2 accidents avaient pour origine la mauvaise performance du système autonome.

Depuis une dizaine d'années, de nombreux constructeurs automobiles tels que Tesla, Toyota ou encore Volkswagen se sont intéressés au développement de voitures autonomes. Le développement d'une voiture autonome nécessite le croisement de nombreux domaines scientifiques tels que la physique, l'informatique, la robotique et l'intelligence artificielle. En effet, la voiture doit percevoir et analyser l'environnement pour ensuite prendre les bonnes décisions. Tout cela doit s'effectuer en temps réel, dans des délais très courts.

Dans le cadre de ce projet de troisième année, en collaboration avec l'Institut Pascal, nous nous sommes concentrés sur la partie intelligence artificielle. L'objectif est d'analyser les objets statiques et dynamiques présents sur la route à l'aide de modèles de Deep Learning. Plus précisément, il s'agit de segmenter des scènes routières à partir d'images caméra et d'échos radar. Pour réaliser ce projet, nous avons à disposition un grand nombre de données traitées et annotées de la part des équipes de l'Institut Pascal.

Dans un premier temps, nous allons contextualiser le projet et décrire les différentes modalités d'acquisition de données de la part du véhicule autonome. Nous discuterons également de l'état de l'art dans ce domaine et comment notre solution se positionne par rapport à celui-ci. Dans un second temps, nous développerons les différentes idées et réalisations au cours de ce projet. Pour finir, nous présenterons notre architecture finale qui constitue un premier prototype (Proof of Concept).

1 Contexte du projet

1.1 L’Institut Pascal

Placé sous la tutelle de l’Université Clermont Auvergne (UCA), du CNRS et de SIGMA Clermont, l’Institut Pascal est un laboratoire de recherche interdisciplinaire s’inscrivant dans des domaines stratégiques des Sciences de l’Ingénierie et des Systèmes : Génie des Procédés, Mécanique, Robotique, Physique des Sciences de l’Information, Santé.

Il regroupe 153 enseignants-chercheurs (UCA, SIGMA), 4 chercheurs (CNRS), 26 (P/MCU)-PH ainsi que 151 doctorants autour de 5 axes de recherche, eux-mêmes divisés en thèmes de recherche (Figure 1) :

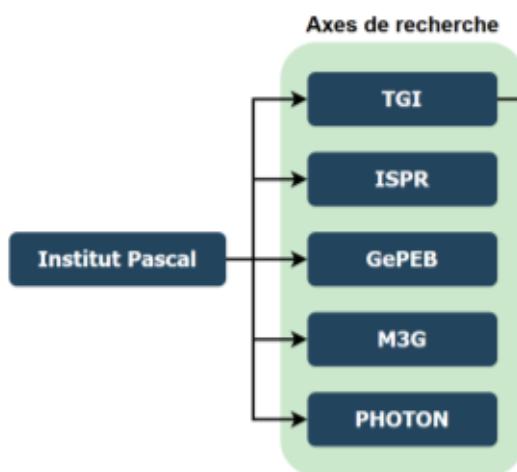


FIGURE 1 – Organigramme de la recherche à l’Institut Pascal

1.2 Présentation du projet

1.2.1 Explication du projet et contexte

Un véhicule intelligent est un véhicule ayant la capacité de décision et d’action. Il est ainsi capable de réaliser des tâches et des missions sans intervention humaine ou avec des interventions humaines minimales, et ce sur des routes ouvertes à la circulation publique. L’adjectif « intelligent » fait référence aux systèmes de transport ayant la capacité de se déplacer quasiment sans aucune intervention humaine. Les principaux exemples Ce sont les navettes, les voitures, les camions ou encore les bus.[1]

Néanmoins pour acquérir cette compétence de conduite autonome, le véhicule doit s’entraîner à percevoir et localiser les entités d’importance. Il s’agit donc de caractériser

en temps réel ce qui ne peut pas être enregistré statiquement par cartographie (piétons, véhicules, panneaux de signalisation). La finalité est de caractériser cet espace libre roulable dans lequel le véhicule peut naviguer. Il est donc nécessaire au véhicule intelligent de détecter et localiser les objets dynamiques.

Pour cela, il utilise un système de perception fournissant des informations provenants :

- de capteurs extéroceptifs embarqués
- de capteurs embarqués dans d'autres véhicules ou sur l'infrastructure dont les données sont transmises par télécommunication.

En se focalisant sur les informations provenant des capteurs extéroceptifs embarqués, on remarque qu'il s'agit d'une problématique largement étudiée dans le monde de la recherche depuis une trentaine d'années. Les principaux capteurs extéroceptifs sont :

- des caméras
- des capteurs radars
- des capteurs à ultrasons
- des capteurs LIDAR (laser à balayage)
- des capteurs de vision comme les caméras thermiques ou les caméras à événements (chaque pixel émet un signal lorsqu'il change d'état).

L'objectif est ensuite d'entraîner différents réseaux de neurones à détecter les objets dynamiques. Si plusieurs capteurs sont utilisés, il est alors nécessaire de mettre en place une architecture pour combiner ces différentes modalités. Une fois entraînés, les réseaux de neurones seront ensuite capables de détecter en temps réel les objets statiques et dynamiques environnant le véhicule. Dans le cadre de ce projet, nous devons combiner les données de 2 capteurs extéroceptifs : un radar à ondes millimétriques et une caméra. La combinaison de ces deux modalités est peu courante dans les papiers de recherche. Pourtant l'utilisation d'un radar présente de nombreux avantages dans l'industrie puisqu'il est bien moins coûteux que le LIDAR. Les voitures Tesla utilisent par exemple une multitude de radars. [5] Nous allons détailler ces différents capteurs extéroceptifs (caméra et radar) ainsi que le LIDAR puisque nous souhaitons obtenir des résultats similaires à la combinaison caméra+LIDAR , mais en utilisant un radar.

1.2.2 Le radar à ondes millimétriques (MMW)

Le radar à ondes millimétriques désigne un radar fonctionnant dans la bande des ondes millimétriques. En général, les ondes millimétriques sont des ondes électromagnétiques dans le domaine de fréquence de 30-300 GHz (longueur d'onde de 1-10 mm). La longueur d'onde de l'onde millimétrique se situe entre l'onde centimétrique et l'onde lumineuse.[7]

Le radar à ondes millimétriques présente donc les avantages de la navigation par micro-ondes et de la navigation photoélectrique. Les ondes millimétriques ont un large éventail d'applications dans les communications 5G, la télédétection par satellite, la navigation des missiles et les contre-mesures électroniques. Ces dernières années, les technologies connexes telles que la conception des circuits et la technologie des antennes ont été de plus en plus développées et mûries avec l'amélioration continue du niveau des composants.

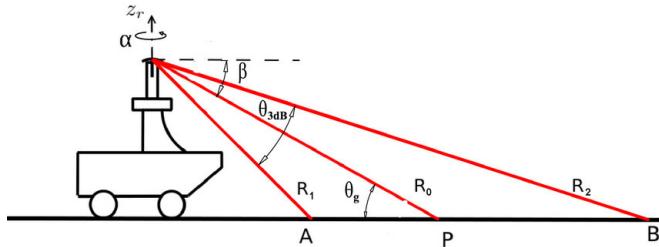


FIGURE 2 – Véhicule équipé d'un radar à ondes millimétriques (MMW) pour l'analyse du terrain

Après l'installation du radar MMW sur le véhicule, celui-ci commence à capturer des images avec une fréquence donnée (la même fréquence que celle de la caméra) afin d'obtenir un couple d'image radar et d'image caméra au même instant t .

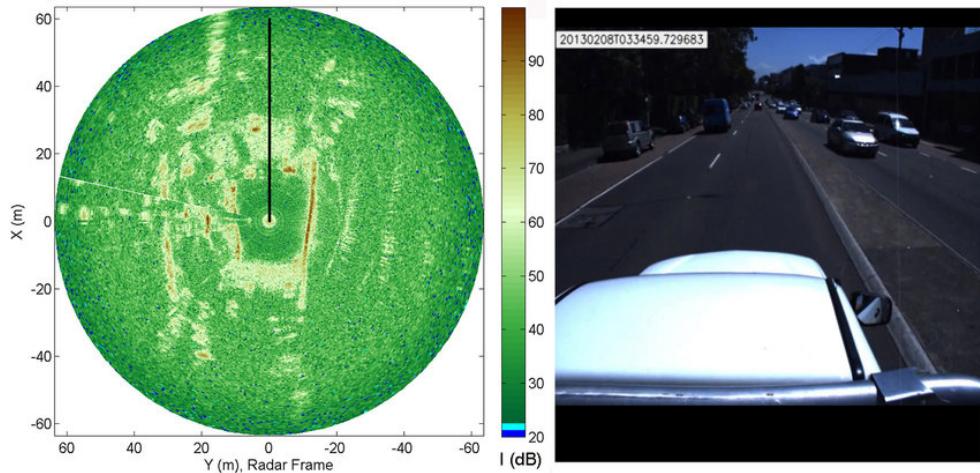


FIGURE 3 – Exemple d'image radar (gauche). Notez que la direction avant du véhicule est marquée par la ligne noire. Vue de face du véhicule (à droite), telle qu'acquise par une caméra située au même endroit

Le radar est une technologie efficace, néanmoins avec l'avancée technologique est apparu le LIDAR. Nous allons présenter ce dernier et expliquer les différences avec le radar.

1.2.3 La détection et télémétrie par la lumière (LiDAR)

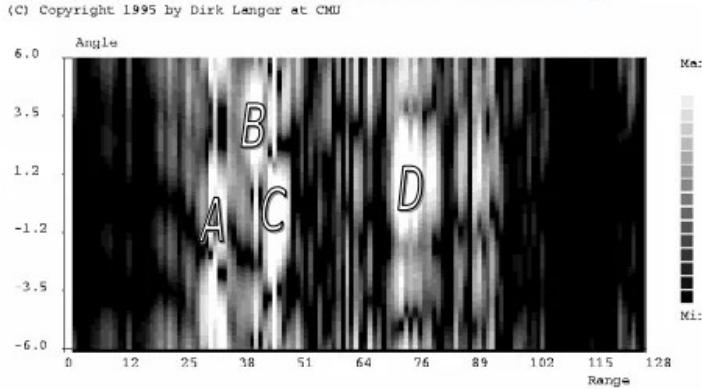
Light Detection and Ranging (LiDAR) est une technologie de télédétection basée sur la lumière. Le principe du LIDAR est très simple : pointer le rayon laser infrarouge sur une surface et mesurer le temps que met le laser à revenir à sa source. Un LIDAR ayant un angle de vision de 360° (en utilisant un miroir rotatif par exemple), il est possible d'obtenir un nuage de points de l'environnement. Ensuite, un logiciel spécifique réalise une image 3D qui reproduit la forme autour du nuage LIDAR avec une position précise dans l'espace.[11]

1.2.4 La différence entre le radar MMW et le LIDAR

Grâce à sa longueur d'onde, le RADAR peut détecter des objets à grande distance et à travers le brouillard ou les nuages. Mais sa résolution latérale est limitée par la taille de l'antenne. La résolution du RADAR standard est de plusieurs mètres à une distance de 100 mètres. Le LIDAR est une solution compacte qui permet un haut niveau de précision pour la cartographie 3D. La précision est généralement inférieure à 1cm pour plusieurs centaines de mètres. Les exemples d'application du LiDAR sont l'altimétrie laser ou encore la cartographie de contours. Le radar, en revanche, est utilisé pour les systèmes anti-collision des avions, le contrôle du trafic aérien ou encore l'astronomie radar.

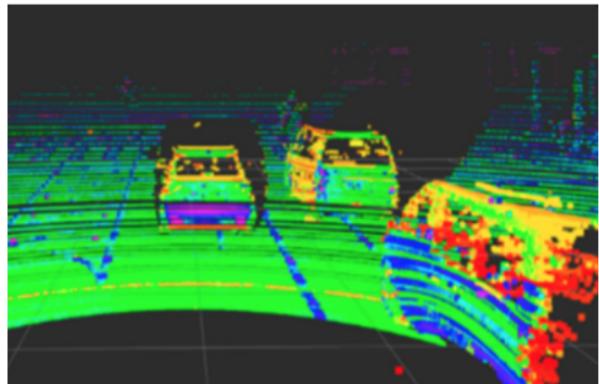
- Exemple des données radar MMW :

Données provenant d'un radar à balayage. En haut il s'agit d'une image caméra de la scène. L'image du bas correspond à un écho radar en coordonnées polaires. Les lettres sur l'écho correspondent aux emplacements marqués sur l'image caméra. La luminosité indique la force du retour. La voiture A est proche et au centre de l'écho radar (l'image vidéo ne s'étend pas aussi loin vers la droite que le radar) ; B est plus loin et à gauche ; C est un peu plus loin et est à peine visible au-dessus du toit de A ; D est beaucoup plus loin et à un relèvement entre A et B.



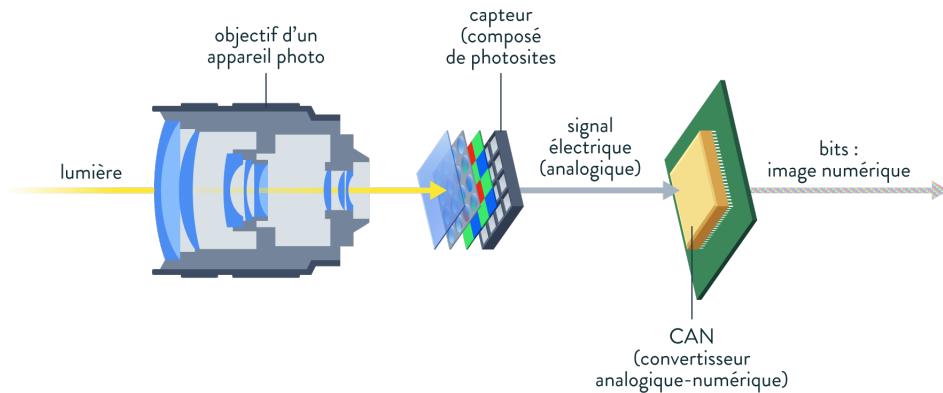
- Exemple des données LIDAR :

L'image du haut est une image caméra prise à partir d'une voiture. En bas, nous pouvons observer la scène environnante obtenue à partir d'un LIDAR. On remarque que l'image obtenue à partir du LIDAR est beaucoup facilement interprétable. En effet, le LIDAR permet d'obtenir des scènes tridimensionnelles de l'environnement. On discerne très facilement les deux voitures se trouvant devant le LIDAR et la caméra.



1.2.5 La modalité d'image caméra

L'imagerie numérique utilise des appareils, appelés caméra ou appareil photographique, contenant des réseaux de photodétecteurs électroniques pour produire des images focalisées par un objectif, par opposition à une exposition sur un film photographique. Les images capturées sont numérisées et stockées sous forme de fichier informatique prêt à être traité, visualisé, publié électroniquement. Lors de la prise d'une photographie, il s'agit d'une transformation géométrique consistant à projeter une scène tridimensionnelle sur un plan en deux dimensions. Cette représentation bidimensionnelle est obtenue sur le réseau de photodétecteurs (capteurs sur la figure). Ce signal analogique (ou électrique) est ensuite transformé en signal numérique à l'aide d'un CAN.



© SCHOOLMOUV

FIGURE 4 – Principe simplifié de capture d'une image à l'aide d'une caméra digitale (ou appareil photographique numérique)

Dans le cas d'une image RGB, le signal numérique est composé de 3 matrices de pixels, correspondant à nos 3 canaux de couleurs (rouge, vert et bleu).

1.2.6 Conclusion sur les différentes modalités

Après cette comparaison, on remarque bien que le radar MMW et le LIDAR sont deux techniques que nous pouvons utiliser dans le cadre de la segmentation automatique de scènes routières. Bien que la quantité d'information obtenue à partir du radar est moindre comparé au LIDAR, on observe que les véhicules autonomes sont majoritairement équipés de radars. En effet, le radar MMW est beaucoup moins cher que le LIDAR, et donc la plupart des entreprises qui travaillent dans le domaine des voitures autonomes préfèrent utiliser le radar MMW. L'objectif de ce projet est de développer une méthode de segmentation automatique de scènes routières à partir de paires d'images caméra et radar MMW. Les images radar permettent de percevoir les distances entre les objets et sont donc très utiles pour la navigation autonome sûre tandis que les images caméra permettent d'identifier la nature et le nombre d'objets présents sur une scène routière.

La combinaison des images radars et des images caméra pour la conduite autonome est un champ de recherche peu développé. Dans la partie suivante, nous allons présenter les différents éléments issus de notre état de l'art et nous allons ensuite proposer notre idée pour développer une architecture simplifiée.

1.3 État de l'art

1.3.1 Modalités radar/camera

La détection d'objets et l'estimation de la profondeur constituent une partie cruciale du système de perception des véhicules autonomes. Les voitures autonomes modernes sont généralement équipées de plusieurs capteurs de perception tels que des caméras, des radars et des LIDAR. L'utilisation de plusieurs modalités de capteurs permet d'exploiter leurs propriétés complémentaires. Néanmoins, le processus de fusion multimodale rend la conception du système de perception plus difficile. Au cours des dernières années, de nombreuses méthodes de fusion de capteurs ont été proposées pour les applications de conduite autonome. La plupart des algorithmes de fusion de capteurs existants se concentrent sur la combinaison d'images RGB avec des nuages de points LIDAR 3D. Les LIDAR fournissent des informations plus précises que le radar sur la profondeur pouvant être utilisée pour la détection d'objets en 3D. Ceci est particulièrement utile dans les applications de conduite autonome où il est crucial de connaître la distance de tous les objets détectés pour un fonctionnement sûr.[12]

Alors que les LIDAR deviennent populaires dans les véhicules autonomes, les radars sont utilisés dans les véhicules autonomes et non autonomes depuis de nombreuses années comme un capteur de profondeur indispensable. Les radars fonctionnent en mesurant la réflexion des ondes radio sur les objets, et utilisent l'effet Doppler pour estimer la vitesse des objets. Bien que les radars fournissent des informations précises sur la distance et la vitesse, ils ne sont pas particulièrement efficaces pour classer les objets. C'est pourquoi la fusion des radars et d'autres capteurs, comme les caméras, est un sujet très intéressant pour les applications de conduite autonome. Un système de fusion radar-caméra peut fournir des informations précieuses sur la profondeur de tous les objets détectés dans un scénario de conduite autonome, tout en éliminant la nécessité d'une détection d'objets 3D coûteuse en calcul à l'aide de nuages de points LIDAR.

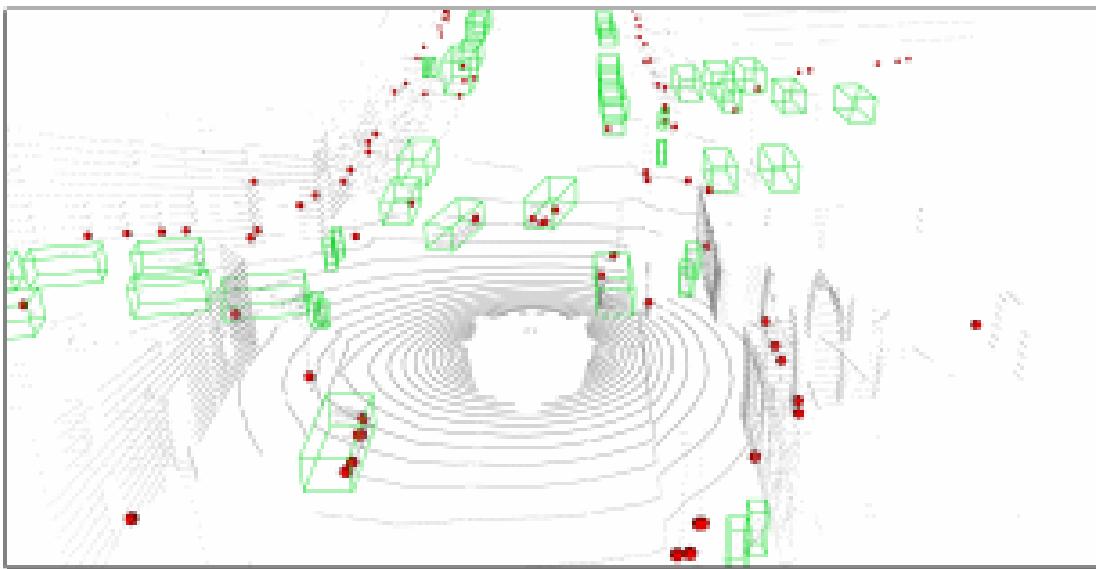


FIGURE 5 – Exemple de données provenant du jeu de données NuScenes montrant le nuage de points radar (rouge), les boîtes de vérité du sol en 3D (vert) et le nuage de points LIDAR (gris).

Pendant notre état de l'art, nous avons trouvé une architecture qui répond à ce que nous cherchions (fusion radar-caméra). Le réseau [13] prend des nuages de points radar et des images RGB en entrée et génère des propositions d'objets précises pour un cadre de détection d'objets en deux étapes. Ils adoptent une approche de fusion intermédiaire en fusionnant les données radar et image, où les sorties de chaque capteur sont d'abord traitées indépendamment, puis fusionnées à un stade ultérieur pour un traitement plus poussé. Plus précisément, ils utilisent d'abord les détections radar pour générer des propositions d'objets en 3D, puis ils mappent les propositions sur l'image et utilisent les caractéristiques de l'image extraites par un réseau fédérateur pour améliorer leur localisation. Ces propositions sont ensuite fusionnées avec les propositions basées sur l'image générées dans un RPN [15], et sont introduites dans la deuxième étape pour la classification. Toutes les propositions générées sont associées à une profondeur estimée, calculée soit directement à partir des détections radar, soit via une couche de régression de distance dans le réseau RPN.

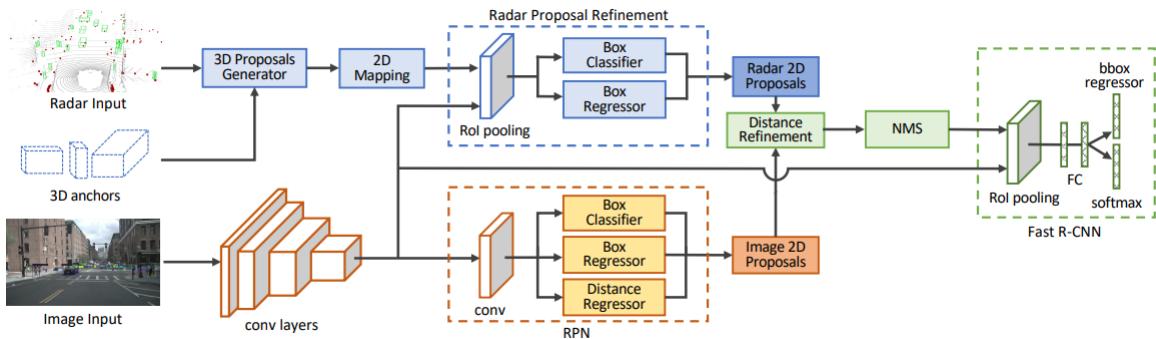


FIGURE 6 – L’architecture de réseau proposée. Les entrées du réseau sont le nuage de points radar, l’image de la caméra et les boîtes d’ancrage 3D. Les propositions d’objets basées sur le radar sont générées à partir du nuage de points et fusionnées avec les caractéristiques de l’image pour améliorer la localisation des boîtes.

D’un point de vue critique, on observe que cette architecture est extrêmement complexe et difficile à comprendre. Néanmoins, l’idée consistant à traiter de manière séparée les deux modalités (caméra/radar) puis à les combiner nous semble intéressante. Nous avons alors réfléchi à une architecture simplifiée et efficace pour résoudre ce problème.

1.3.2 Proposition d’architecture

Du fait de la complexité de cette architecture, nous nous sommes inspirés de cette idée pour développer une architecture plus simple et ayant des résultats similaires. Il s’agit de combiner les images radars et les images caméra. Pour cela, nous allons utiliser un réseau de neurones qui effectue la segmentation des voitures sur l’image caméra. Nous utilisons également un réseau de neurones qui segmente les voitures sur les images radars. Pour finir, nous allons essayer de combiner la sortie de ces deux réseaux de neurones afin d’obtenir une estimation précise de la distance pour chaque objet.

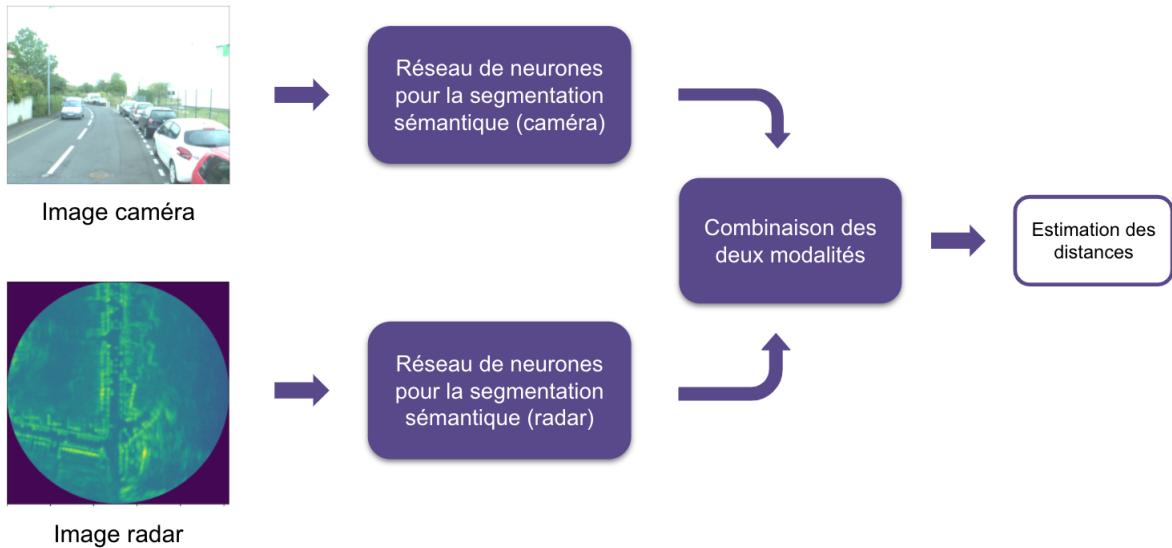


FIGURE 7 – Architecture simplifiée pour combiner les images radars et les images caméra pour la conduite autonome

Pour ce projet, nous nous sommes concentrés uniquement sur la segmentation des voitures. Ensuite, il sera possible d'appliquer cette méthodologie sur d'autres objets tels que les piétons, les poteaux, les vélos ...

1.4 Outils de développement

Dans le but de mettre en pratique cette architecture, nous avons décidé d'utiliser TensorFlow. Il s'agit d'une plateforme open source de bout en bout pour l'apprentissage automatique. Elle dispose d'un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires qui permet aux chercheurs de repousser les limites du Machine learning et aux développeurs de créer et de déployer facilement des applications basées sur le machine learning.[14] Nous avons choisi de travailler avec la surcouche Keras. Il s'agit d'une API d'apprentissage profond écrite en Python, et utilisant la plateforme d'apprentissage automatique TensorFlow. Elle a été développée pour permettre une expérimentation rapide. La capacité de passer d'une hypothèse ou d'une idée à un résultat concret nous semble essentiel.[9]



FIGURE 8 – Les outils de travail

La capacité de calcul de nos ordinateurs étant extrêmement limitée, nous avons décidé d'utiliser Google Colab, le service cloud de Jupyter Notebook Google. Cet outil est destiné à la formation et à l'expérimentation dans le domaine de l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud, sans avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur. L'utilisation de Google Colab nous permet également d'entraîner extrêmement rapidement nos modèles.[10]



FIGURE 9 – Google Colab

1.5 Organisation des tâches

Dans le but de mener à bien notre projet, nous avons commencé par nous former aux techniques de vision par ordinateur et au deep learning dans le domaine de l'imagerie. Nous avons passé une bonne partie du début du projet à comprendre les fondamentaux et à nous former sur les différentes techniques. Rapidement, nous avons réalisé une documentation de ces techniques et des papiers de recherche dans ce domaine. Nous mettons régulièrement à jour cette base documentaire pendant le projet. Des réunions régulières avec nos tuteurs entreprises ont eu lieu afin d'avancer sur le projet. La figure 10 et 11 représentent le diagramme de Gantt prévisionnel et réel du projet.

Nous pouvons nous apercevoir que le planning a été respecté dans l'ensemble. Les

Rapport de projet - F4

parties sur état de l'art, mise en place d'une documentation et compréhension des données ont été plus longues qu'initialement prévues.

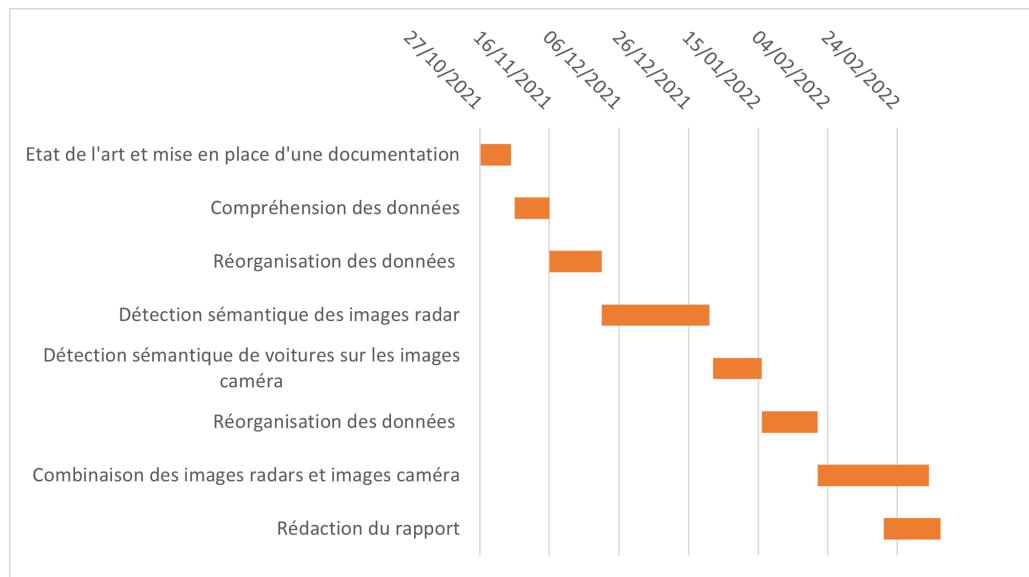


FIGURE 10 – Diagramme de Gantt prévisionnel

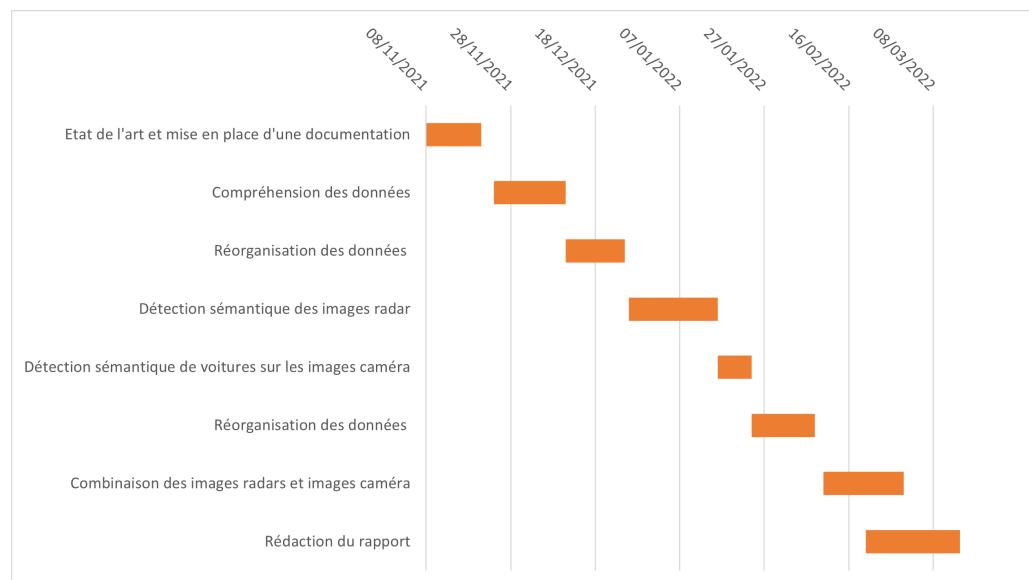


FIGURE 11 – Diagramme de Gantt réel

2 Conception et réalisation

2.1 Compréhension des données

Les données mis à notre disposition pour ce projet sont les données des images caméra, les données des images radars et les segmentations sémantiques associées. Notre jeu de données est composé de 3 trajectoires. Ces trajectoires correspondent à des séquences d'image capturées par la voiture de l'Institut Pascal sur les routes d'Aubière et de Clermont-Ferrand. Lorsque la voiture roule, elle capture à intervalle de temps régulier un couple d'images radar et caméra à l'instant t que l'on notera $(I_{t,camera}, I_{t,radar})$. A ce couple d'image, il est possible d'y associer un couple d'image segmenter, au même instant t , que l'on notera $(S_{t,camera}, S_{t,radar})$. Nous allons ainsi détailler chaque modalité d'image d'un point de vue physique et informatique.

2.1.1 Les images caméra

Dans le cas d'une image RGB, le signal numérique est composé de 3 matrices de pixels. Chaque valeur de pixels correspond aux nuances de rouge, de vert et de bleu. On peut représenter chaque canaux par une matrice de dimensions correspondant à la largeur et la hauteur de l'image. Chaque pixel de l'image va donc représenter une variable de la matrice, qui correspond à l'intensité de la couleur à ce pixel précis. Un pixel peut être définie via une variable comprise entre 0 et 255, correspondant à l'intensité de la couleur.

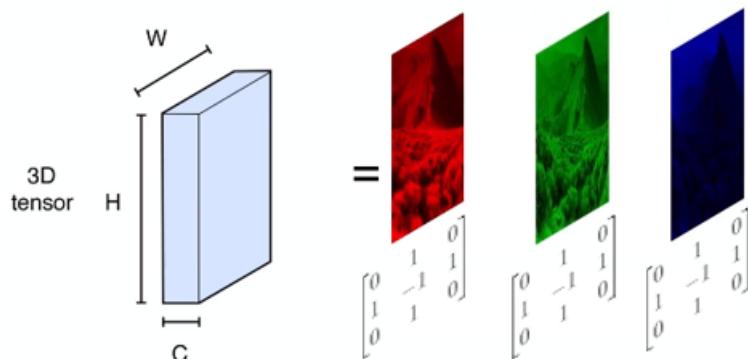


FIGURE 12 – Tenseur d'une image RGB avec les canaux rouge, vert et bleu

Dans notre cas, nous avons une image numérique suivant les 3 canaux rouge, vert et bleu. Plus précisément, nous avons des images RGB au format .tiff. Contrairement au format JPEG très courant, le format tiff est plus fidèle à la qualité de l'image et est souvent utilisé en photographie professionnelle. Cependant, ce format demande plus d'espace de stockage du fait de l'absence de compression. Voici un exemple d'image caméra :



FIGURE 13 – Example d’image caméra prise à la voiture à Aubière

Nous travaillons avec un set de $N = 1556$ images caméra de dimension 1280x1024. Au total, nous avons un tenseur de données de dimensions : $(N, pixels_x, pixels_y, channels) = (1556, 1280, 1024, 3)$.

2.1.2 Les images radar

Le radar est un système de détection qui utilise des ondes radio pour déterminer la distance, l’angle ou la vitesse d’objets. Un système radar se compose d’un émetteur produisant des ondes électromagnétiques dans le domaine des radio ou des micro-ondes, d’une antenne d’émission, d’une antenne de réception (souvent la même antenne est utilisée pour l’émission et la réception) et d’un récepteur et d’un processeur pour déterminer les propriétés du ou des objets.. Les ondes radio (pulsées ou continues) de l’émetteur se réfléchissent sur l’objet et reviennent au récepteur, donnant des informations sur la localisation et la vitesse de l’objet. Dans le cadre du projet, nous avons à disposition des images radars représentant les objets entourant la voiture. L’intensité de l’onde réceptionnée par le radar dépend des propriétés physiques des matériaux étant traversés ou réfléchissant l’onde. La perspective adoptée est la *Bird’s-eye view* (vu à vol d’oiseau) avec la voiture au centre de l’image radar. La modalité de scan est circulaire autour de la voiture et suivant le plan ($x,y,z = constant$) avec $constant$ correspondant à la hauteur du radar par rapport au plan de la route. Généralement, les objets métalliques ont une intensité très élevée tandis que l’air (aucun objet) à une intensité considérée comme nulle.

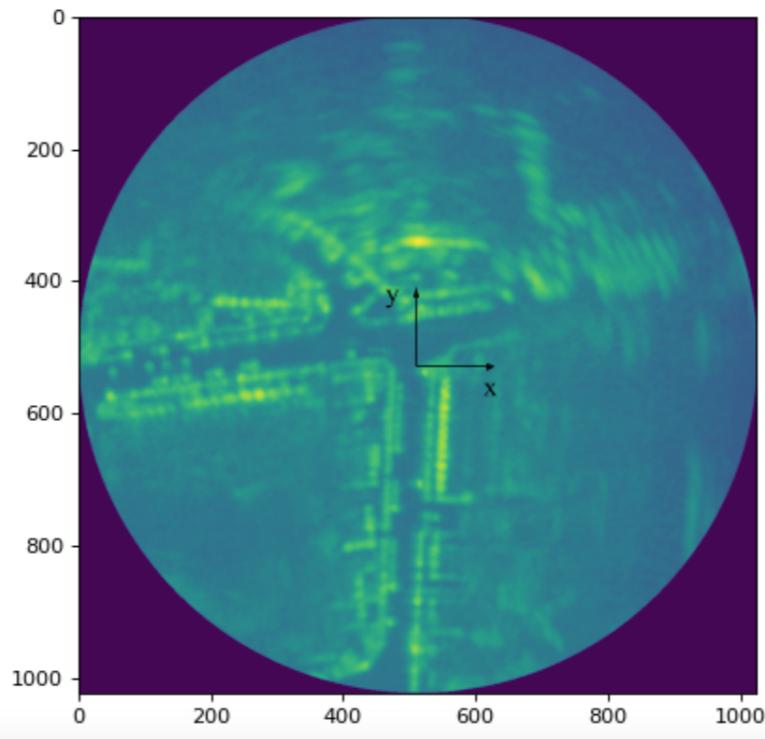


FIGURE 14 – Example d'image radar de notre jeu de données

Dans cet exemple, la voiture équipée du radar et de la caméra se trouve au centre de l'image, elle est donc à un croisement entre deux routes. Contrairement à l'image caméra, l'information n'est plus sur 3 canaux mais sur un seul. La valeur de chaque pixel correspond à l'intensité de l'onde réfléchie par l'objet se trouvant à cette position (x,y) dans le plan de l'onde diffusée. Nous travaillons avec un set de $N = 1556$ images radar de dimension 1025x1025. Au total, nous avons un tenseur de données de dimensions : $(N, pixels_x, pixels_y, channels) = (1556, 1025, 1025, 1)$.

2.1.3 La segmentation des images caméra

La segmentation d'images est le processus de partitionnement d'une image numérique en plusieurs segments d'image, également appelés régions d'image constituant un ensemble de pixels. Le résultat de la segmentation d'une image est un ensemble de segments qui couvrent collectivement la totalité de l'image, ou un ensemble de contours extraits de l'image. Il existe deux types de segmentation :

- La **segmentation sémantique** qui détermine, pour chaque pixel, la classe d'appartenance de l'objet. Par exemple, si il y a plusieurs voitures sur l'image, celles-ci vont être segmentées comme étant un objet appartenant à la classe voiture. Le reste sera identifié comme étant l'arrière-plan.

- La **segmentation par instance** qui détermine, pour chaque pixel, une classe d'appartenance de l'objet. Elle détecte chaque objet d'intérêt distinct dans l'image. Par exemple, si il y a plusieurs voitures dans l'image, chaque voiture est segmentée comme un objet individuel.

Pour ce projet, nous nous intéressons principalement à la segmentation par instance : les objets appartenant à la même classe (voiture par exemple) sont numérotés de 1 à n . Voici un exemple de couple d'image caméra et d'image caméra segmenté (notée $I_{547,camera}$ et $S_{547,camera}$) à l'aide de l'outil annotator de l'Institut Pascal.



FIGURE 15 – Image caméra ($I_{547,camera}$)

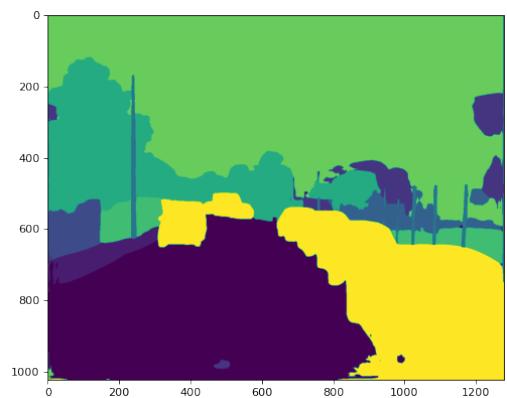


FIGURE 16 – Segmentation associée à l'image caméra ($S_{547,camera}$)

On peut ensuite décomposer les objets d'une classe distinctement. Par exemple, on peut afficher le masque des 5 voitures :

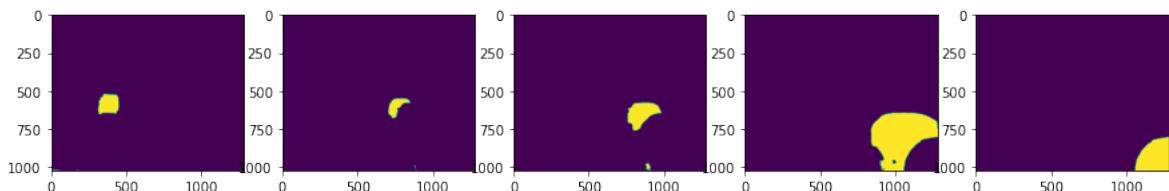


FIGURE 17 – Segmentation de 5 masques de la classe "voiture" de l'image 547

Pour une classe en particulier (ici la classe voiture), les pixels ont des valeurs distinctes. La valeur des pixels pour les images de 1 à 5 lorsqu'il y a une voiture sont (13001, 13002, 13003, 13004, 13005) respectivement tandis que le reste des pixels de l'image ont 0 pour valeur. La segmentation peut également se faire sur d'autres éléments de l'image tels que les poteaux ou encore les arbres. Les valeurs des pixels sont alors 5000 et 8000 respectivement.

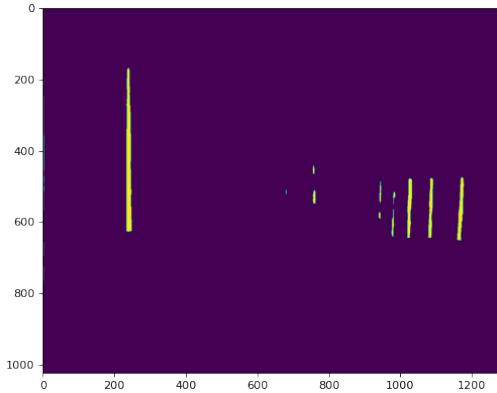


FIGURE 18 – Segmentation des poteaux de l'image 547

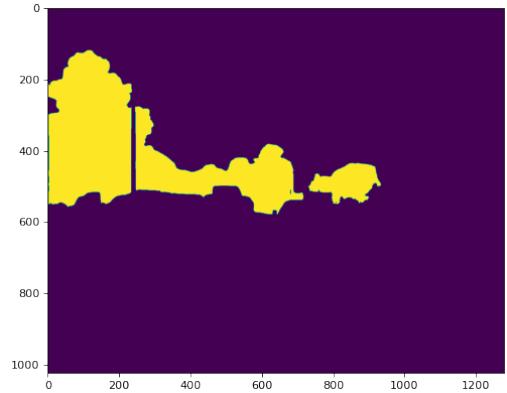


FIGURE 19 – Segmentation des arbres de l'image 547

Pour ce projet, nous nous sommes principalement concentrées sur la segmentation des voitures. L'objectif est ensuite d'étendre les techniques utilisées pour les voitures aux autres éléments de l'environnement. Pour l'instant, nous avons un ensemble de $N = 1556$ images segmentées de dimension 1280x1024. Au total, nous avons un tenseur de données de dimensions : $(N, \text{pixels}_x, \text{pixels}_y, \text{nb_voitures}) = (1556, 1280, 1024, \text{nb_voitures})$ où nb_voitures est le nombre de voitures distinctes sur l'image.

2.1.4 La segmentation des images radars

La segmentation des images radar suit le même processus que la segmentation des images caméra. Néanmoins, la segmentation n'est présente que sur la classe voiture. De plus, les images radars ont été annotées à la main par les équipes de l'Institut Pascal. Voici un exemple d'image radar et de la segmentation associée :

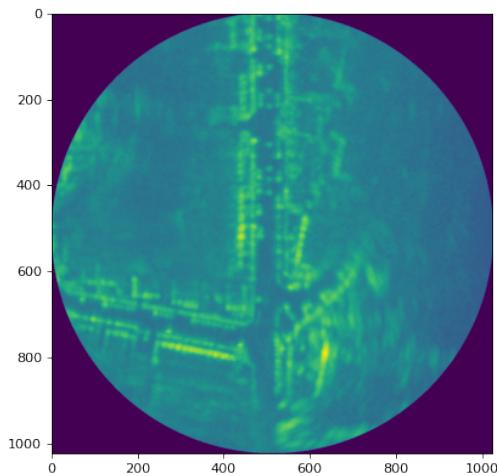


FIGURE 20 – Image radar n°200 notée $I_{200, \text{radar}}$

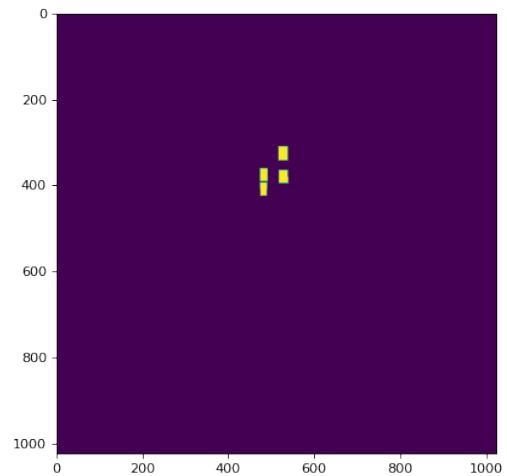


FIGURE 21 – Image radar segmentée n°200 notée $S_{200, \text{radar}}$

La segmentation des voitures sur les images radars a été réalisée en s'aidant des images de la caméra. Ainsi, une voiture détectée par le radar se trouvant derrière la voiture de l'Institut Pascal n'est pas segmentée. Seules les voitures visibles sur la caméra sont segmentées sur les images radars. On comprend donc qu'une grande partie de l'information des images radars ne sont pas utiles. Toutes les informations derrière la voiture de l'Institut Pascal n'est pas utile. De plus, nous nous intéressons uniquement à la route et aux trottoirs. Cela nous permet de délimiter une zone d'intérêt sur l'image radar et d'entraîner plus efficacement notre réseau (moins de bruit dans l'image).

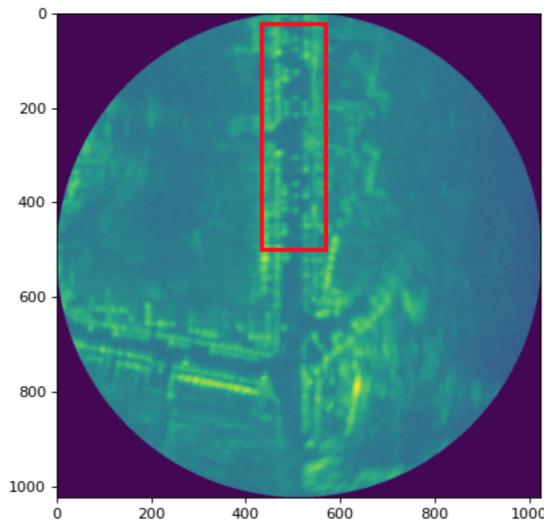


FIGURE 22 – L'encadré en rouge représente la zone d'intérêt de l'image radar

Nous avons un ensemble de $N = 1556$ images segmentées de dimension 1025x1025. Au total, nous avons un tenseur de données de dimensions : $(N, pixels_x, pixels_y, nb_voitures) = (1556, 1025, 1025, nb_voitures)$ où $nb_voitures$ est le nombre de voitures segmentées à la main sur une image (similaire aux données segmentées de l'image caméra).

2.2 Détection sémantique de voitures sur les images radars

L'objectif est de détecter sémantiquement la forme et la taille des voitures à partir d'une image radar. Il s'agit donc de prédire les voitures présentes sur une image radar, sans aide humaine. Pour ce faire, nous allons utiliser des techniques de deep learning qui consiste à détecter sémantique des objets sur une image. Dans un premier, il est nécessaire de formater les données suivant le bon format et la bonne modalité. Dans un second temps, nous devons entraîner le modèle sur un jeu d'entraînement $(I_{t,radar}, S_{t,radar})$ constitué d'une image radar et de la segmentation associée. Pour terminer, nous devons évaluer la performance de notre modèle et trouver des pistes d'amélioration.

2.2.1 Pipeline de données

2.2.1.1 Réorganisation des données

Les données fournies par les équipes de l’Institut Pascal sont 3 dossiers contenant les différentes images et annotations.

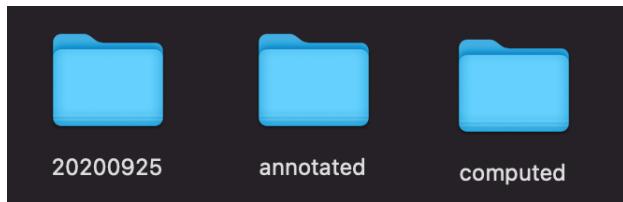


FIGURE 23 – Organisation générale des données

- Le dossier **20200925** contient l’intégralité des images caméra au format .tiff. On ne s’intéresse pas à ce dossier dans cette partie.
- Le dossier **annotated** contient les données annotées des images caméra et des images radars.
- Le dossier **computed** contient les images radars.

Dans le dossier **annotated**, nous nous intéressons principalement à deux dossiers : **traj1_mask_noury** et **traj3_mask_coly**. Ces deux dossiers contiennent respectivement 568 et 989 fichiers *npz*. Chaque fichier correspond à une image radar annotée. Le format de fichier *.npz* est une archive zippée de fichiers portant le nom des variables qu’ils contiennent. L’archive n’est pas compressée et chaque fichier de l’archive contient une variable au format *.npy*. Ainsi, pour récupérer une image radar annotée, il suffit de lire le fichier *npz* en sélectionnant la bonne variable (*/"arr_0"/* dans notre cas).

Dans le dossier **computed**, nous nous intéressons principalement à deux dossiers : **traj1/synchronized_scans** et **traj3/synchronized_scans**. Ces deux dossiers contiennent respectivement 568 et 989 fichiers *npz*. Chaque fichier correspond à une image radar. Ainsi, pour récupérer une image radar, il suffit de lire le fichier *npz* en sélectionnant la bonne variable (*/"scan"/* dans notre cas).

Pour nous simplifier la tâche, nous avons utilisé un script Python permettant de convertir toutes ces données sous la forme d’un dossier appelé X contenant les 1556 images radars au format npy et un dossier appelé Y contenant les 1556 images radars annotées au format npy. Pour se faire, nous avons utilisé un fichier texte qui indique l’adresse de toutes les images radars et les images radars annotées correspondantes. A partir de ce fichier texte, nous avons transféré l’intégralité des données sous forme des deux dossiers X

et Y. Dans le dossier X, les fichiers sont nommés : (*scan_0.npy*, ..., *scan_1555.npy*). Dans le dossier Y, les fichiers sont nommés (*annoted_0.npy*, ..., *annoted_1555.npy*). Il était également nécessaire d'appliquer une rotation sur les images radars puisqu'elles étaient orientées de +90 degrés par rapport à l'image annotée correspondante.

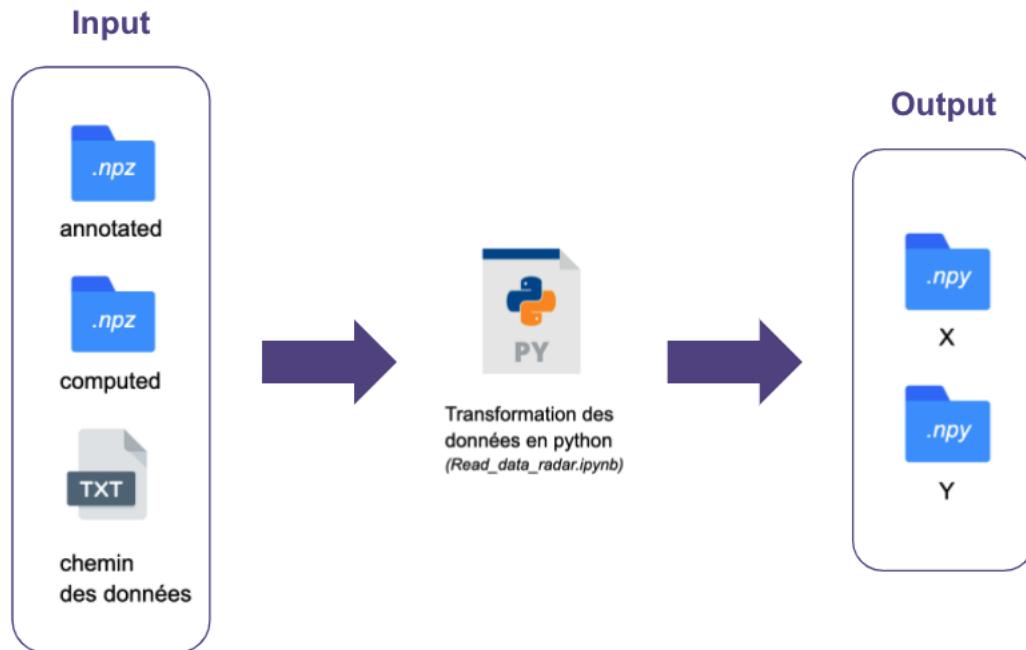


FIGURE 24 – Piepline de transformation de données schématisée

Dans ce fichier python, nous avons utilisé les librairies suivantes : Numpy (module pour le calcul scientifique), zipfile (module d'outils pour créer, lire, écrire, ajouter et lister un fichier ZIP), PIL (librairie d'imagerie) et Matplotlib (Bibliothèque pour afficher des graphiques). Nous avons choisi d'utiliser le langage Python pour la facilité d'utilisation et la flexibilité d'utilisation. De plus, il est très utilisé en machine learning et en deep learning, proposant notamment de nombreuses librairies gratuites. Le choix des librairies se fait généralement sur l'efficacité du module, la rapidité de prise en main et la disponibilité de la documentation/échanges entre membres de la communauté.

2.2.1.2 Recadrage des images

Nous avons pu voir dans la section 2.1.4 que seulement une petite zone de l'image radar nous intéresse. En effet, c'est uniquement dans cette zone que la segmentation des images radars a été réalisée. Pour cela, il est nécessaire de les recadrer avant d'entraîner un réseau de neurones. Pour se faire, nous avons déterminé deux manières simplistes de recadrer l'image :

- Un recadrage très restrictif : 160 pixels de longeur et 80 pixels de largeur.

- Un recadrage moyennement restrictif : 272 pixels de longeur et 80 pixels de largeur.
- Un recadrage peu restrictif : 368 pixels de longeur et 80 pixels de largeur.

On utilise les différentes stratégies de recadrage suivantes :

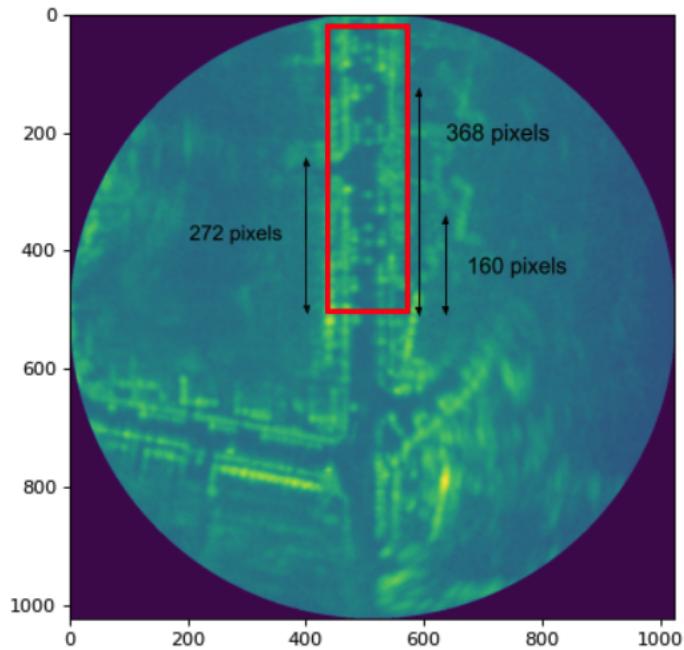


FIGURE 25 – Différentes stratégies de recadrage

On obtient ainsi les paires d’images radars et radars annotées (en haut les images radars, en bas les images radars annotées correspondantes).

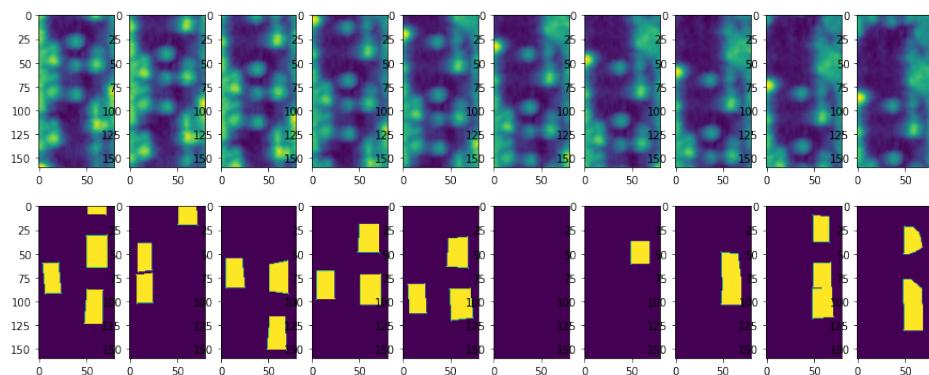


FIGURE 26 – Recadrage très restrictif : 160x80

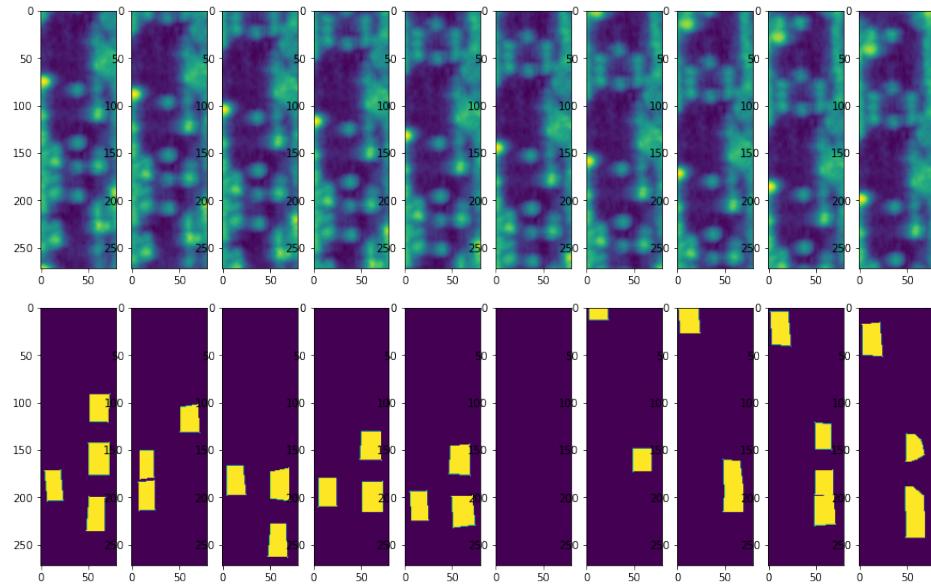


FIGURE 27 – Recadrage moyennement restrictif : 272x80

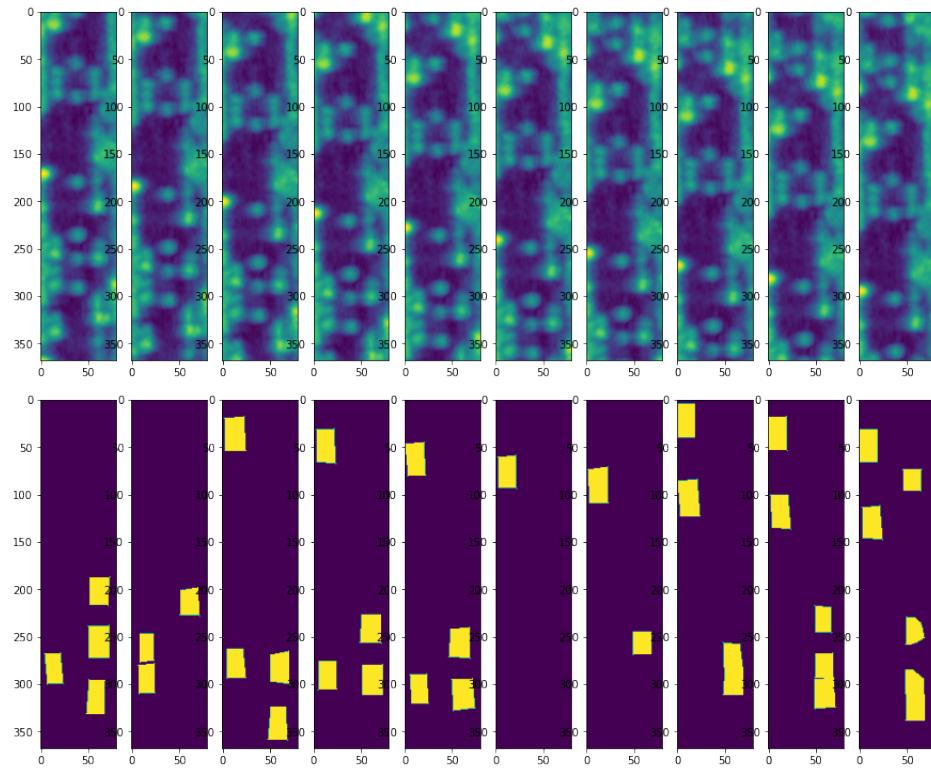


FIGURE 28 – Recadrage peu restrictif : 368x80

Nous avons utilisé ces différentes modalités de données pour entraîner notre réseau de neurones. Pour finir, nous avons normalisé les données pour avoir chaque pixel de l'image radar $\in [0, 1]$.

Soit X_t la valeur des pixels de l'image radar $I_{t, \text{radar}}$.

$\forall t \in 1, \dots, 1556$

$$X_{t,new} = \frac{X_t - \max(X_t)}{\max(X_t) - \min(X_t)}$$

2.2.2 Modèle de deeplearning pour la détection sémantique

L'objectif est d'obtenir à partir d'une image radar, une image segmenté associée. Ainsi, l'entrée du modèle est de la même taille que la sortie. Pour ce faire, nous avons utilisé un papier de recherche listant les différentes méthodes de segmentation sémantique [17].

2.2.2.1 Architecture et implémentation du réseau U-Net

Nous nous sommes intéressés au réseau de neurone U-Net [16] car il présente de nombreux avantages :

- Il nécessite peu de données annotées.
- L'entraînement est très rapide car il n'y a pas beaucoup de paramètres (environ 2 Millions).
- La performance était l'une des meilleures pour la segmentation sémantique en 2015.

Ce réseau correspond bien à notre problème car nous avons seulement 1556 images annotées, une puissance de calcul limitée (Google Colab gratuit) et nous souhaitons avoir une bonne détection des objets sur l'image. De plus, U-Net est un réseau de référence dans le domaine de la segmentation sémantique avec plus de 38 000 citations. L'architecture du réseau U-Net est la suivante :

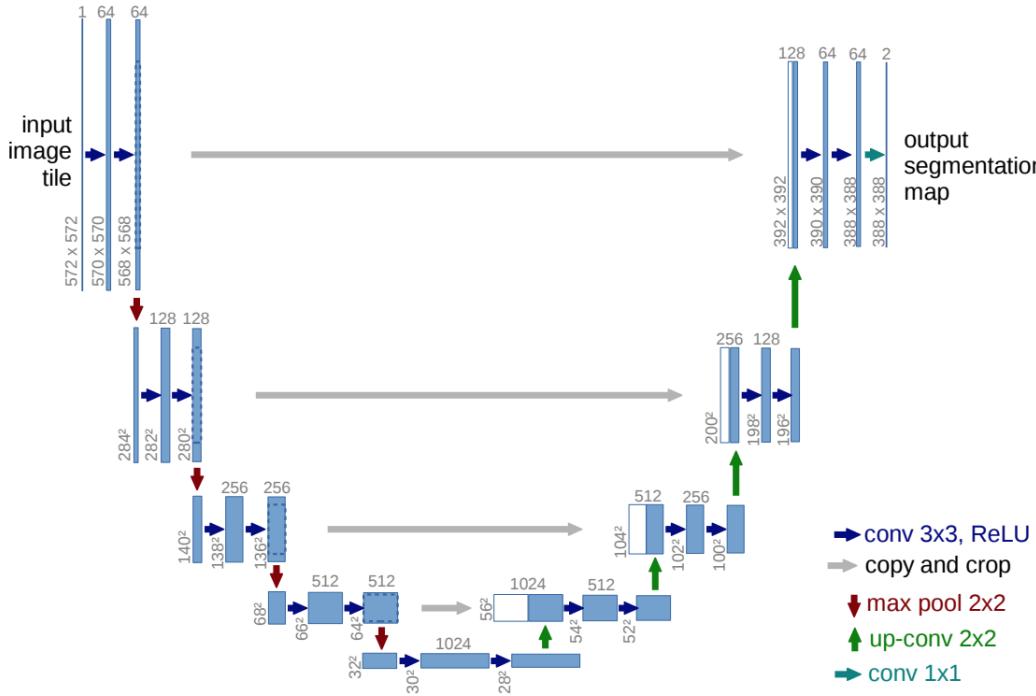


FIGURE 29 – Architecture du réseau de neurone U-Net [16] (La taille des tenseurs n'est pas représentatif. Dans notre cas, le tenseur est de taille 368x80)

Le réseau se compose d'un chemin de contraction et d'un chemin d'expansion, ce qui lui donne une architecture en forme de U. Le chemin de contraction est un réseau convolutif typique qui consiste en l'application répétée de convolutions, de *ReLU* et de *MaxPooling*. Pendant la contraction, l'information spatiale (taille de l'image) est réduite tandis que l'information sur les caractéristiques est augmentée (augmentation du nombre de features). La voie d'expansion combine les informations spatiales et de caractéristiques par une séquence de convolutions ascendantes et de concaténations avec des caractéristiques à haute résolution provenant de la voie de contraction (*skip connections*).

Pour implémenter le réseau U-Net, nous avons utilisé l'interface Keras. Keras est une API de réseaux de neurones de haut niveau, écrite en Python et interfaçable avec TensorFlow. Elle permet d'écrire très rapidement et facilement une grande diversité de réseaux de neurones. Nous avons également ajouté des fonctionnalités telles que des couches de dropout entre les couches de convolutions. [2]. Nous avons ajouté en annexe l'implémentation du code ainsi que les successions de couches pour les différentes entrées.

2.2.2.2 Métriques pour évaluer la qualité de la prédiction

La segmentation sémantique n'utilise pas des métriques usuelles pour mesurer la performance des prédictions. En effet, il est nécessaire d'évaluer si la position et la forme formée par les pixels prédits en sortie correspondent le plus fidèlement possible à l'image

annotée. Nous avons utilisé deux métriques courantes pour la segmentation sémantique :

- IoU (Intersection over Union) ou Jaccard index : elle décrit la zone de chevauchement entre la segmentation prédite et la vérité terrain divisée par la zone d'union entre la segmentation prédite et la vérité terrain. Pour une segmentation binaire (deux classes) ou multi-classes, l'IoU moyen de l'image est calculé en prenant l'IoU de chaque classe et en faisant la moyenne.
- Le coefficient Sørensen-Dice : il est égal à 2x la zone de chevauchement divisée par le nombre total de pixels dans les deux images.

La formule du Jaccard index ou IoU :

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|} = \frac{TP}{TP + FP + FN}$$

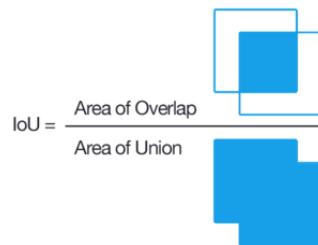


FIGURE 30 – Schéma pour le Jaccard Index ou IoU score

La formule de Sørensen-Dice ou Dice coefficient :

$$DSC(X,Y) = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2TP}{2TP + FP + FN}$$

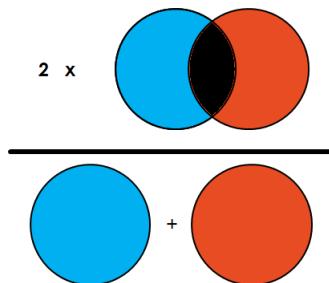


FIGURE 31 – Schéma de Sørensen-Dice ou Dice coefficient

Nous avons implémenté ces deux métriques en utilisant la librairie Numpy et la fonctionnalité backend de keras. Cela permet d'avoir une compatibilité avec la librairie keras (et donc notre réseau de neurones).

2.2.2.3 Paramètres d'entraînement du réseau de neurones

Pour l'entraînement du réseau de neurones, nous avons utilisé différents paramètres et techniques afin d'obtenir les meilleures performances possibles. Nous avons commencé par séparer le jeu de données en set d'entraînement (90%) et en set de test (10%). Cela nous permet de mesurer la qualité de la prédiction sur un jeu de données inconnu du réseau de neurones. Les paramètres usuelles utilisées sont les suivants :

- L'optimiseur : Adam [3]
- Fonction de coût : Binary cross entropy [4]
- Batch-size : 10
- Epochs : 60

Nous avons également ajouté des paramètres permettant d'améliorer la performance du réseau de neurones :

- Augmentation des données : nous avons ajouté des images avec une rotation d'angle de ± 5 degrés.
- ReduceLROnPlateau : fonction Keras permettant de réduire le taux d'apprentissage quand la valeur de la fonction de coût stagne sur plusieurs epochs successives. [18] Cela permet d'obtenir un taux d'apprentissage optimal pendant l'entraînement (graphe du milieu) :

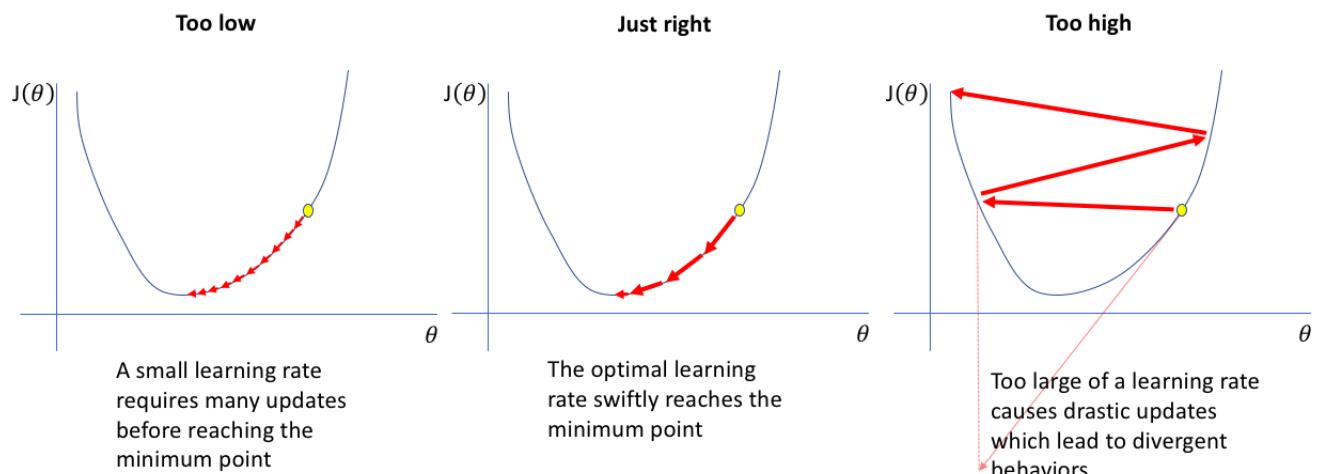


FIGURE 32 – Influence du taux d'apprentissage sur la convergence vers le minimum de la fonction de coût

2.2.3 Résultats et performances

Pour nos 3 tailles de cropping d'image, nous obtenons les résultats suivants :

Taille de l'image d'entrée	Dice coefficient	IoU mean
160x80	71,36%	62,06%
272x80	66,21%	53,70%
368x80	60,59%	48,05%

TABLE 1 – Résultats de la segmentation des images radars

Nous avons enregistré les poids du réseau de neurones dans un fichier .h5 pour les 3 formats d'image. Si on souhaite prédire la localisation des voitures à partir d'une image radar, il suffit de charger le fichier de poids en Python et d'utiliser la méthode *predict()* sur les images.

Voici le résultat obtenu sur quelques images du set de données test (pour une image de taille 272x80 et 160x80) :

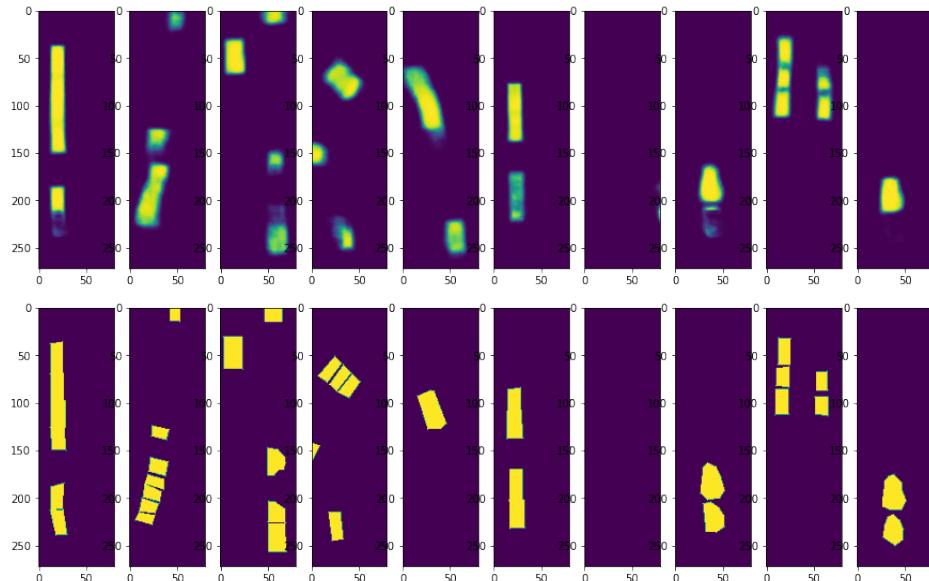


FIGURE 33 – Prédiction d'image 272x80. En haut : prédiction du réseau U-Net. En bas : vérité terrain.

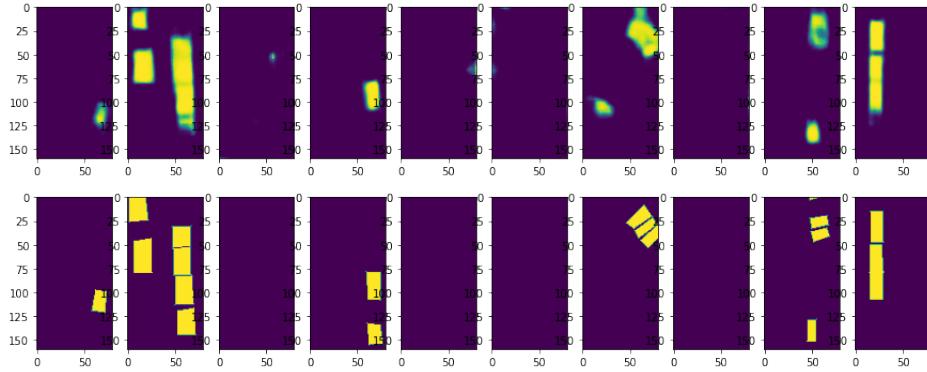


FIGURE 34 – Prédiction d’image 160x80. En haut : prédiction du réseau U-Net. En bas : vérité terrain.

2.3 Détection sémantique de voitures sur les images caméra

La détection sémantique des objets sur l’image caméra a déjà été réalisée par les équipes de l’Institut Pascal à l’aide de l’outil annotator. Néanmoins, nous avons proposé un outil de détection d’objet et de segmentation sémantique (sémantique et par instance) permettant d’obtenir de meilleures performances. Il s’agit du réseau de neurones Mask R-CNN [8]. Un R-CNN commence par appliquer un mécanisme appelé recherche sélective pour extraire les régions d’intérêt (ROI), où chaque ROI est un rectangle qui peut représenter la limite d’un objet dans l’image. Selon le scénario, il peut y avoir jusqu’à deux mille ROI. Ensuite, chaque ROI est soumis à un réseau neuronal pour produire des caractéristiques de sortie. Pour les caractéristiques de sortie de chaque ROI, une collection de classificateurs automatiques à vecteur de support est utilisée pour déterminer le type d’objet (le cas échéant) contenu dans le ROI. [6]. Mask R-CNN fonctionne sur ce principe tout en ajoutant la segmentation par instance. Voici l’architecture du réseau de neurones Mask R-CNN :

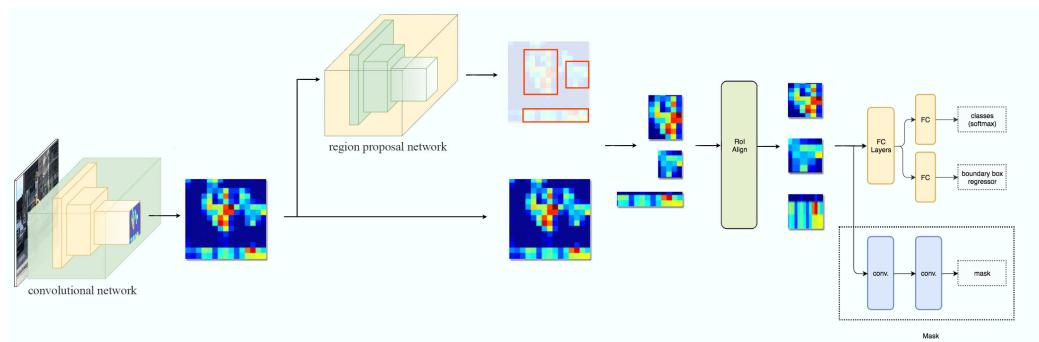


FIGURE 35 – Architecture du réseau Mask-R-CNN

Pour implémenter ce réseau, nous avons utilisé une librairie haut niveau en Keras et Tensorflow : https://github.com/matterport/Mask_RCNN. Il s’agit de l’implémentation

principale de ce réseau (21K stars sur github) et bénéficiant d'une large communauté (accès facile à la documentation). Nous avons utilisé la version pré entraînée du réseau sur la base de données MS COCO (328K images). Voici les résultats obtenus sur nos images :



FIGURE 36 – Image caméra
($I_{547,camera}$)



FIGURE 37 – Segmentation associée à l'image caméra ($S_{547,camera}$)

2.4 Combinaison des images radars et images caméra

L'objectif est maintenant de combiner la détection de voitures à l'aide des images radars et des images caméras. En effet, la détection sur l'image radar permet d'obtenir une estimation de la distance à laquelle se trouve les différents véhicules tandis que l'image caméra permet d'identifier précisément la forme et le type de l'objet (identique à la vision d'un humain). Pour chaque scène au temps t , nous avons sur les images caméra et les images radars plusieurs objets (véhicules dans notre cas). L'objectif est d'associer chaque véhicule détecté sur l'image caméra aux véhicules détectés sur l'image radar. On utilise l'approximation qu'un véhicule sur une image caméra est identifié par sa boîte englobante (Bounding Box) : c'est-à-dire les 4 coordonnées $x_{min}, x_{max}, y_{min}, y_{max}$. En considérant qu'il y a i véhicules sur une image caméra au temps t , on utilise la notation suivante :

$$\forall t \in \{1, \dots, 1556\}, \forall i \in \{1, \dots, nb \text{ voitures}\}$$

avec $BOXi, t, camera = (C_{xmin}, C_{xmax}, C_{ymin}, C_{ymax})$

On utilise également l'approximation qu'un véhicule sur une image radar est identifié par sa boîte englobante (Bounding Box) : c'est-à-dire les 4 coordonnées $x_{min}, x_{max}, y_{min}, y_{max}$. En considérant qu'il y a i véhicules sur une image caméra au temps t , on utilise la notation suivante :

$$\forall t \in \{1, \dots, 1556\}, \forall i \in \{1, \dots, nb \text{ voitures}\}$$

avec $BOX_i, t, radar = (R_{xmin}, R_{xmax}, R_{ymin}, R_{ymax})$

Le passage de la bounding box caméra à la bounding box radar est une transformation géométrique. En effet, une image caméra est la projection d'une scène tridimensionnelle sur un plan, selon une certaine perspective. Cette perspective est inhérente aux caractéristiques physiques de la caméra. A contrario, l'image radar est issue de la propagation d'une onde. Ainsi, les distances entre les objets sur l'image radar est la même partout, il n'y a pas d'effet de perspective. Il est donc très facile d'estimer la distance entre les objets sur une image radar (simple mise à l'échelle) tandis qu'il est difficile voire impossible d'estimer la distance des objets sur une image caméra (à cause de la perspective).

Notre objectif est de créer un réseau de neurones capable de prédire la bounding box d'une image radar ($R_{xmin}, R_{xmax}, R_{ymin}, R_{ymax}$) à partir de la bounding box d'une image caméra ($C_{xmin}, C_{xmax}, C_{ymin}, C_{ymax}$). Pour cela, il est nécessaire de créer une base d'entraînement solide afin d'obtenir les meilleurs résultats possibles.

2.4.1 Création de la base d'entraînement

Identiquement à la partie 2.2.2 sur la détection sémantique de voitures sur les images radars, il est nécessaire de créer un script python qui permet de transformer les données au format adéquat pour l'entraînement de notre réseau de neurones. En effet, à partir des images radars segmentées et de la segmentation des images caméra, nous devons estimer les boundingbox correspondantes.

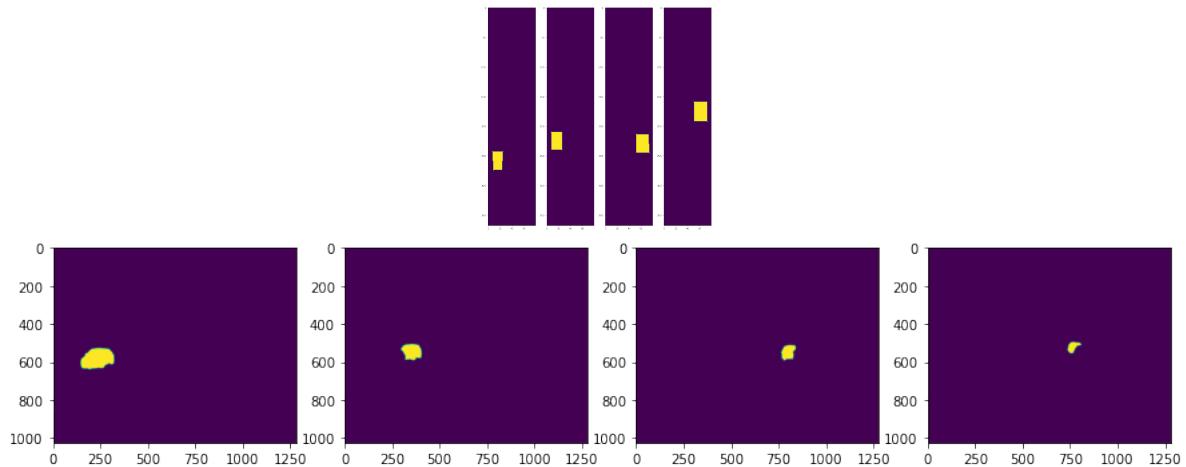


FIGURE 38 – Images radars et caméra segmentées voiture par voiture. En haut : segmentations radar. En bas : segmentations caméra correspondantes

Pour déterminer la bounding box des images radars, il est nécessaire de recadrer l'image. Il faut donc prendre l'ordonnée et l'abscisse des pixels extrêmes (min et max) ayant une valeur non nulle. Une fonction python en Numpy permet de faire ce recadrage.

Néanmoins, pour les images caméra, il est plus difficile d'appliquer cette technique. En effet, la segmentation issue d'Annotator n'est pas totalement exacte et il arrive parfois que des pixels débordent sur le bord de l'image ou encore à des positions aléatoires sur l'image. Voici un exemple de problème de débordements de pixels :

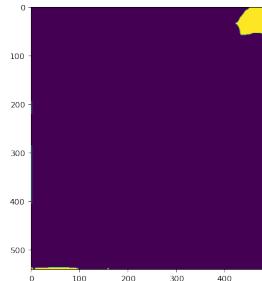


FIGURE 39 – Résultat après le recadrage : débordement de pixels sur la segmentation des images caméra

Pour faire face à ce problème de débordement de pixels, nous allons simplement éliminer les images dont le ratio entre le nombre de pixels non nuls et la taille de l'image est supérieur à 0.5.

$$Ratio = \frac{\text{Nombre de pixels non nuls apres le recadrage}}{\text{Nombre de pixels dans l'image recadree}}$$

Si $Ratio > 0.5$ alors on conserve le couple radar-caméra pour l'entraînement du réseau de neurones. Sinon, on utilise pas le couple dans notre dataset.

Cela traduit le fait que la boîte englobante doit contenir plus de 50% de pixels non nuls. Si une image ne respecte pas cette condition, alors on ne l'utilisera pas pour entraîner notre réseau de neurones. Idéalement, il serait souhaitable de retraiter automatiquement les données ayant des dépassements de pixels. Nous avons effectué toutes ces transformations et vérifications en utilisant un fichier python sous forme de notebook. Voici le pipeline de données.

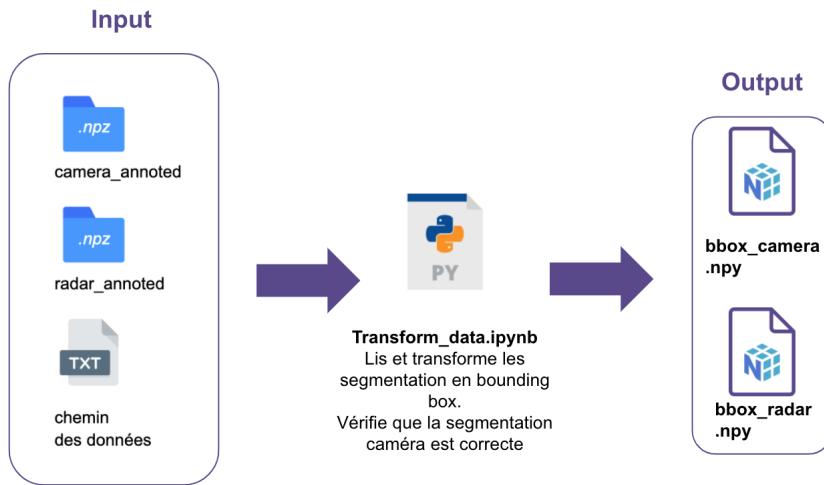


FIGURE 40 – Pipeline de données pour le réseau de neurones réalisant la transformation géométrique caméra vers radar

Au total, notre jeu de données est composé de 2065 exemples comprenant chacun 4 coordonnées caméra et 4 coordonnées radar. C'est à l'aide de ce jeu de données que nous allons entraîner notre réseau de neurones pour prédire efficacement les 4 coordonnées radars à partir des 4 coordonnées caméra.

2.4.2 Réseau de neurones

La transformation géométrique passant de l'image caméra à l'image radar n'est pas une transformation géométrique linéaire. Ainsi, il est nécessaire de créer un réseau de neurones qui permet d'approximer une fonction non linéaire (ie : fonction qui permet de faire la transformation géométrique de 4 coordonnées de la caméra à 4 coordonnées dans le radar). Pour cela nous avons utilisé un réseau de neurones entièrement connecté avec une fonction d'activation Relu. Voici les différentes couches :

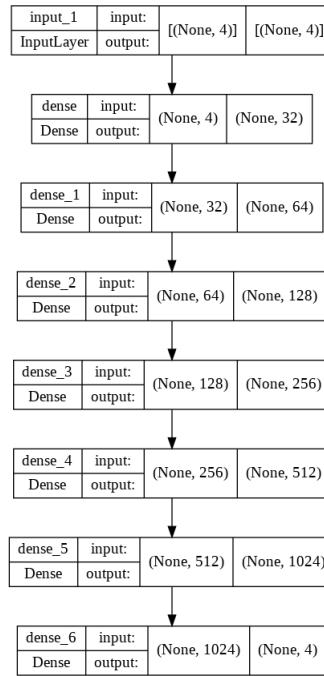


FIGURE 41 – Architecture du réseau de neurones complétement connecté

2.4.2.1 Paramètres d’entraînement du réseau de neurones

Pour l’entraînement du réseau de neurones, nous avons utilisé différents paramètres et techniques afin d’obtenir les meilleures performances possibles. Nous avons commencé par séparer le jeu de données en set d’entraînement (80%) et en set de test (20%). Cela nous permet de mesurer la qualité de la prédiction sur un jeu de données inconnu du réseau de neurones. Les paramètres usuels utilisés sont les suivants :

- L’optimiseur : Adam [3]
- Fonction de coût : Erreur quadratique moyenne

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2,$$

avec n le nombre d’éléments, Y_i la vérité terrain, \hat{Y}_i la valeur prédite.

- Batch-size : 10
- Epochs : 60

Nous avons également ajouté des paramètres permettant d’améliorer la performance du réseau de neurones :

- ReduceLROnPlateau : fonction Keras permettant de réduire le taux d’apprentissage quand la valeur de la fonction de coût stagne sur plusieurs epochs successives.

[18] Cela permet d'obtenir un taux d'apprentissage optimal pendant l'entraînement (graphe du milieu) :

2.4.2.2 Métrique pour évaluer la performance

Pour évaluer la performance de notre réseau de neurones, nous utilisons le RSME. Cette métrique est simplement la racine carré du MSE :

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n}}$$

avec n le nombre d'éléments, Y_i la vérité terrain, \hat{Y}_i la valeur prédictée.

Cette métrique a l'avantage d'être très facilement interprétable puisqu'elle est dans la même unité de mesure que Y_i . Dans notre cas, l'unité de mesure est le pixel. On distingue également l'erreur suivant l'axe de la route (axe y) et l'axe orthogonal à la route (axe x) présenté dans la Figure 11 de la partie 2.1.2. On mesure l'erreur en pixel et également en pourcentage par rapport aux dimensions de l'image (368 x 80).

$$RMSE_x = 5 \text{ pixels.} \quad RMSE_x(\%) = \frac{RMSE_x}{80} = 6,25\%$$

$$RMSE_y = 21 \text{ pixels.} \quad RMSE_y(\%) = \frac{RMSE_y}{368} = 6\%$$

Il s'agit d'une performance assez satisfaisante. Celle-ci peut encore être améliorée en entraînant le réseau de neurones sur une plus grande quantité de données.

2.5 Présentation du PoC

Fort de cette expérience et de ce travail, nous pouvons présenter une Proof of Concept. Il s'agit d'une architecture de 3 réseaux de neurones et différentes transformations géométriques.

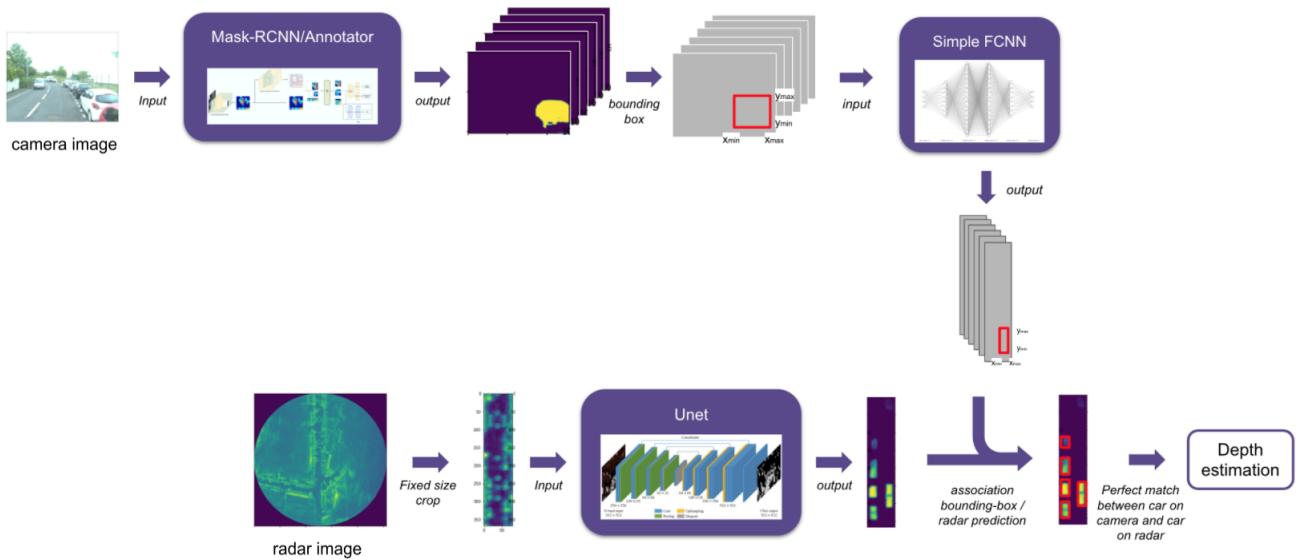


FIGURE 42 – Architecture de DeepLearning associant les images radars et les images caméra pour la conduite autonome

Notre architecture consiste en trois grandes parties. La première partie consiste à détecter sémantiquement les voitures sur l'image caméra. On obtient plusieurs masques. Il est ensuite nécessaire d'extraire les bounding box de ces masques. La deuxième partie consiste à recadrer l'image radar puis à détecter sémantiquement les voitures sur celle-ci. La troisième partie consiste à transformer géométriquement les boundingbox dans la base caméra en bounding box dans la base radar. Il suffit ensuite d'associer à chaque bounding box une des voitures détectées sur l'image radar. On peut donc obtenir une estimation de la distance des voitures se présentant sur la route. Il est possible de retrouver cette image en grand format sur le github du projet : <https://github.com/MathieuCayssol/ComputerVision-Radar-Camera>

Conclusion et perspectives

Le premier objectif a été d'entraîner un réseau de neurones convolutionnel (U-Net) à détecter sémantiquement les voitures sur des images radars. Nous avons dû comprendre, traiter et retravailler la donnée ainsi qu'optimiser les paramètres d'entraînement afin d'obtenir les meilleures performances possibles. Nous avons également choisi des métriques cohérentes pour estimer la bonne ou mauvaise performance de notre modèle (Dice Coefficient et IoU).

Le second objectif a été de proposer des outils de segmentation sémantique de voitures à partir d'images caméra. Nous avons utilisé un modèle de deep learning (Mask-RCNN) pré entraîné sur plus de 328K images. Ce modèle a de meilleures performances comparativement à la segmentation Annotator.

Le troisième objectif a été de combiner les images radars et les images caméra. Il s'agit d'associer les voitures détectées sur l'image caméra aux voitures prédites sur l'image radar. Nous avons transformé la segmentation sémantique de voitures en boîte englobante de voiture afin d'entraîner un réseau de neurones. Ce réseau de neurones entièrement connecté (FCNN) permet de prédire la boîte englobante d'une voiture sur une image radar à partir de la boîte englobante d'une voiture sur une image caméra. Nous avons également estimé la performance de notre réseau de neurones en utilisant le RSME.

Pour chaque sous objectifs, nous avons préalablement réorganisé les données de l'Institut Pascal afin de permettre d'obtenir un format adéquat à l'entraînement de nos réseaux de neurones.

Pour conclure, nous avons créé notre propre architecture de réseaux de neurones. Grâce à cette architecture, nous pouvons combiner efficacement les images caméra et les images radars afin d'estimer avec précision la distance entre les objets et la voiture autonome.

Références

- [1] Philippe Bonnifait and Clement Zinoune. Véhicule intelligent.
- [2] Shaofeng Cai, Wei Wang Yao Shu, and Gang Chen. Efficient and effective dropout for deep convolutional neural networks. 2020.
- [3] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On empirical comparisons of optimizers for deep learning. 2020.
- [4] ML Community. Loss function : ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html.
- [5] Wikipedia Community. Tesla autopilot.
- [6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014.
- [7] Sensing Handbook of Terahertz Technology for Imaging and Communications. Millimeter-wave radar.
- [8] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. 2018.
- [9] mattdangerw. Keras : Deep learning for humans.
- [10] Henri Michel. C'est quoi google colab ?
- [11] Léa Moussy. Lidar vs. radar.
- [12] Ramin Nabati. Radar-camera sensor fusion for joint object detection and distance estimation in autonomous vehicles.
- [13] Ramin Nabati and Hairong Qi. Radar-camera sensor fusion for joint object detection and distance estimation in autonomous vehicles. 2020.
- [14] Google open source. Tensorflow.
- [15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. 2016.
- [16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. 2015.
- [17] Farhana Sultana, Abu Sufiana, and Paramartha Duttab. Evolution of image segmentation using deep convolutional neural network : A survey. 2020.
- [18] Manzil Zaheer and Sashank J. Reddi. Adaptive methods for nonconvex optimization. 2018.