ARCHITECTURE LOGICIELLE ET MATÉRIELLE DES VOITURES AUTONOMES

AC20 : Semestre d'Automne 2020

RESUMÉ

Ce travail de recherche s'efforcera d'étudier les composants logiciels et matériels des voitures autonomes dans la perspective d'acquérir des connaissances scientifiques à leurs sujets.

Elios Cama

Etudiant en deuxième année de Tronc Commun à l'UBTM

Responsable UV : M. Mohamed TACHIKART

Enseignant suiveur : M. Alexandre Lombard

utbm
université de technologie
Belfort-Montbéliard

Remerciements

Avant de commencer ce travail de recherche, je tenais à témoigner ma reconnaissance envers mon école, l'Université de Technologie de Belfort-Montbéliard, de proposer des Unités de valeurs de recherche permettant à ses étudiants d'obtenir par eux-mêmes et dans de bonnes conditions les compétences qu'ils souhaitent acquérir dans le domaine de leur choix.

Je souhaitais aussi à remercier M. Mohamed TACHIKART pour m'avoir encouragé à choisir cet UV lors des campagnes d'inscriptions. Malgré les conditions exceptionnelles de la crise sanitaire, il a su se rendre disponible pour organiser au mieux le déroulement de cette UV.

Enfin, je remercie tout particulièrement M. Alexandre Lombard, mon professeur suiveur. Il m'a guidé tout au long de mes démarches de recherche et a toujours pris le temps de me répondre de manière précise et concise dans un temps limité. Toute l'aide qu'il m'a apporté le long de cette étude et de ce semestre m'a été très précieuse.

Table des matières

Introduction

Depuis l'arrivée de l'électricité et de la révolution industrielle, l'Homme développe de plus en plus d'outils et d'objets capables de réaliser automatiquement des tâches complexes. Un des plus grands défis de l'Humanité est donc d'automatiser toute les tâches qui lui sont difficiles dans son quotidien afin de réduire ses efforts et de maximiser son temps.

Avec, la création de la roue, l'Homme a toujours cherché à développer des véhicules plus performants. Cependant, depuis plus d'un siècle maintenant nous vivons dans un monde régit par la technologie et les ordinateurs, ce qui a poussé l'Homme à faire des avancées technologiques dans les voitures. Il ne se contente plus d'améliorer les composants matériels de celles-ci, mais intègre des logiciels et des technologies électroniques.

Depuis plusieurs dizaines d'années, un mythe est construit sur une voiture qui se conduirait toute seule. Grâce aux avancées technologiques de la dernière décennie, nous avons réussi à intégrer de plus en plus de technologie dans nos voitures, ce qui les a transformés en véritable ordinateurs roulants. Aujourd'hui, de tels véhicules autonomes sont réels et sillonnent les routes des Etats-Unis tous les jours.

Je me suis donc intéressé à ce qui faisait qu'une voiture était autonome, donc de sa partie matérielle ainsi que de sa partie logicielle. Ce choix m'est apparu naturellement car je suis tout particulièrement intéressé par les voitures et la technologie.

Cette recherche aborde l'étude de plusieurs composants afin de trouver, de la manière la plus juste, ceux qui sont propres à une technologie de voiture autonome. Tout d'abord nous nous intéresserons aux voitures autonomes dans un contexte historique et économique afin de bien maitriser les notions essentielles. Ensuite, nous verrons comment les voitures autonomes perçoivent leur environnement. Enfin, nous ferons l'étude de la prise de décision de ces véhicules.

I. Présentation générale des voitures autonomes

A. Histoire des voitures autonomes

a. Histoire et évolution

Commençons par rappeler la définition d'un véhicule autonome. Un véhicule autonome est un véhicule dont la conduite est en partie ou entièrement automatisée. Depuis presque un siècle, des technologies d'autonomie ont été développées. En effet, en 1939 General Motors présente la transmission automatique nommée Hydro Matic Drive. Puis, dix ans plus tard a été créé le système ABS qui permet une aide au freinage, mais aussi le régulateur de vitesse. Ce n'est qu'en 1977 qu'une équipe japonaise de la ville de Tsukuba fit rouler une voiture sans intervention humaine sur un circuit fermé à 30km/h, grâce à une technologie de reconnaissance du marquage au sol.

Figure 1: Voiture Tsuskuba 1977 [1]

Plus tard, en 1985, la DARPA (Defense Advanced Research Projects Agency), une organisation américaine finance le projet ALV (Autonomous Land Vehicle) qui aboutit à un véhicule permettant de suivre une route quelconque à 30km/h. Cette même année démarre le projet NavLab, des véhicules autonomes développés par le laboratoire de robotique de l'université de Pittsburgh. Un des véhicules de NavLab sera le premier à parcourir les Etats-Unis d'une côte à une autre de façon autonome en 1995. En 1987, la Commission européenne finança le programme Prometheus, un investissement de 800 millions d'euros contribuant au développement d'outils technologiques dédiés à la conduite autonome. Un des derniers gros projets fût celui d'un véhicule de l'équipe allemande de Daimler-Benz en 1995, qui parcouru le trajet Munich-Copenhague aller-retour avec un record de conduite sans assistance humaine de 158km. Presque quinze ans plus tard, un projet révolutionne le monde des véhicules autonomes, la Google Car.

Figure 2 : Google Car

En 2009 Google débute son projet d'Auto-Driving Car, qui équipait les Toyota Prius et des Lexus RX 450h de capteurs diverses afin de parvenir à les faire rouler de manière autonome. Mais ce qui a retenu l'attention des scientifiques, fût la Google Car, un prototype de voiture autonome de niveau 4 (je détaillerais précisément les niveaux d'autonomies dans la prochaine partie) qui ne proposait ni pédales ni volant. Depuis 2014, un nouveau système de conduite autonome fait son apparition chez un géant de l'automobile aux Etats-Unis, Tesla. Tesla lance Autopilot. Ses véhicules équipés de cette technologie sont aussi équipés de caméras, de capteurs ultrasons et d'un radar. Ce logiciel permet à la voiture de la marque de se déplacer de manière autonome en ville et sur autoroute, et surtout de pouvoir reconnaître la signalisation routière. Grâce à l'essor de la marque et à ses nombreuses mises à jour logiciel, l'Autopilot de Tesla est maintenant reconnu comme une des plus puissantes technologies de conduite autonome.

Figure 3 : AutoPilot sur une Tesla Model S

Finalement en 2016, le projet de Google Car est rebaptisé Waymo. Ces véhicules sont équipés de nombreux capteurs habituels mais par-dessus tout, d'un LIDAR, une technologie de plus en plus démocratisée que j'expliquerai par la suite. La marque devient le leader mondial des véhicules autonomes au fil des années grâce à l'accord de nombreux états des Etats Unis de pouvoir tester leurs véhicules en condition réelles pour les « entrainer ». Je vous expliquerai dans la troisième partie de cet axe pourquoi j'ai choisi de focaliser cette étude sur cette marque plutôt qu'une autre.

b. Législation et éthique

Dans le domaine de la législation des véhicules autonomes, les Etats Unis sont en avance par rapport au reste du monde. En effet, en 2017, vingt-huit états avaient déjà accepté la mise en circulation de véhicules 100% autonomes. En réalité ces autorisations peuvent être soumise à des conditions comme le fait qu'ils doivent être enregistrés, assurés et qu'ils aient un certificat de conformité données par le ministère des transports, c'est le cas du Nevada, premier état à avoir donné son accord en 2011. En France, un décret a été mis en application en 2019 afin de permettre également le test des voitures autonomes de niveau 4 (capable de circuler sans occupant) sur des trajets balisés.

En 2016, 36 461 personnes ont perdu la vie dans un accident de voiture aux Etats Unis. Et un total de 1,35 millions dans le monde entier. On sait que 94% des accidents aux USA sont dus aux choix des humains et non à cause de leur véhicule. 12 millions des américains ayant plus de 40 ans sont aveugles ou ont des problèmes de vision, 79% des personnes âgées vivent dans des endroits où la voiture est nécessaire pour se déplacer. Et plus de 54 heures de nos vies sont gâchées chaque année dans les bouchons. Un rapport du Ministère des Transports des Etats Unis affirment que la mise en circulation de voitures autonomes pourrait réduire les accidents de la route mortels de l'ordre de 94%. Ces véhicules qui enlèvent toute responsabilité du conducteur dans leurs choix.

Cependant, le 18 Mars 2018, un véhicule autonome de la marque Uber a tué une femme nommée Elaine Hersberg. C'était le premier accident mortel dans l'histoire des véhicules autonomes. L'accident s'est déroulé de nuit lorsque la femme en question s'est avancée sur la route en marchant à côté de son vélo. Le véhicule ne l'a pas détecté et l'a percuté fatalement. La détermination du responsable de l'accident est une tâche très complexe. Qui accuser, Uber, la conductrice présente dans la voiture ? Uber a suspendu ses activités de véhicules autonomes durant neuf mois après l'accident mais n'a pas été retenu coupable. A chaque accident, les enregistrements vidéo de la voiture sont envoyés directement aux assurances ainsi qu'aux autorités nécessaires comme la police et les juges.

Les voitures autonomes font aussi face à des dilemmes éthiques. En effet lors d'une situation où un accident est inévitable, que doit choisir la voiture entre épargner ses passagers ou épargner les piétons ? Une enquête menée par le MIT appelée le Moral Machine propose 13 situations dans lesquelles un véhicule va avoir un accident. L'utilisateur de l'enquête doit choisir s'il épargne les personnes qui marchent sur le passage piéton en tuant les passagers, ou l'inverse. Cette enquête a été réalisé dans plus de 200 pays, par des millions de personnes, ce qui a permis d'avoir des résultats très intéressants. En effet, le MIT est parvenu à distinguer trois groupes de régions dans le monde ou les résultats sont différents. Comme on peut le voir dans les graphiques ci-dessous.

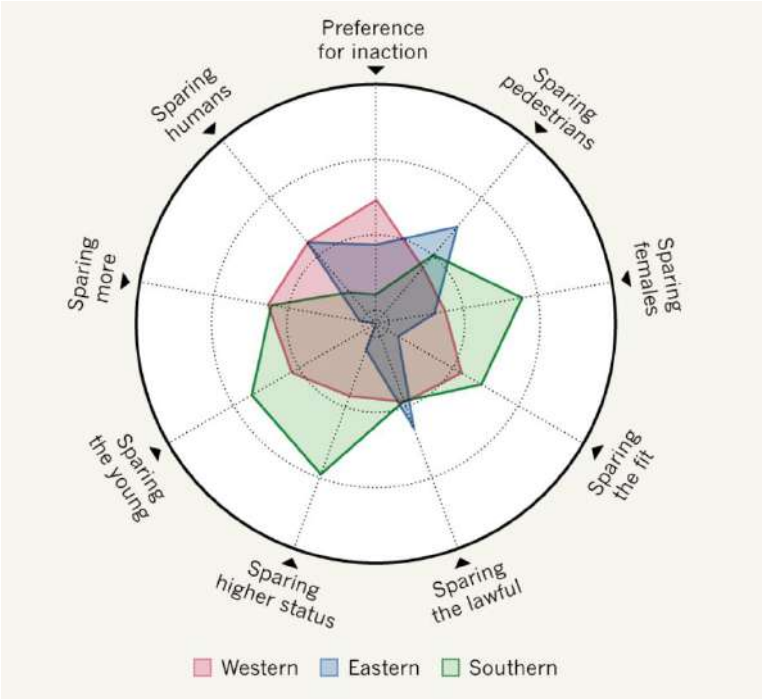


Figure 4 : Répartition des décisions dans le monde

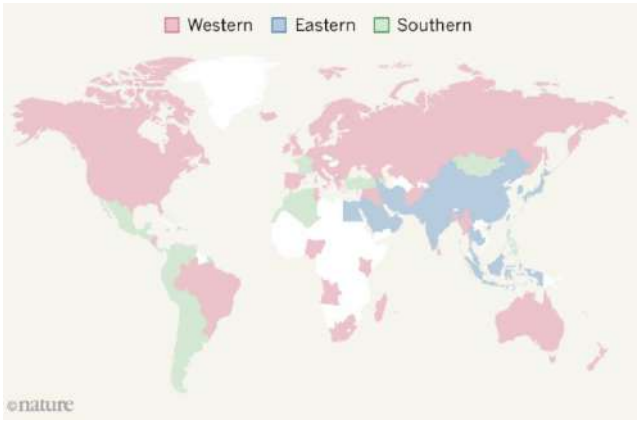


Figure 5 : Carte du monde associée avec la figure 4

On remarque que la culture économique et sociale des pays influence directement les choix éthiques. Certaines cultures tendent à sauver les femmes plutôt que les hommes ou d'autres les enfants plutôt que les personnes âgées. Cette enquête doit mener à une discussion des constructeurs de véhicules autonomes concernant la prise de décision dans des situations critiques.

B. Principes fondamentaux des voitures autonomes

Il existe de nombreux modèles de voitures autonomes ainsi que de nombreuses technologies les accompagnant. Afin de les classifier, la SAE International (Society of automotive engineers), une organisation américaine composée d'ingénieurs, de chefs d'entreprise, de professeurs et d'étudiants de plusieurs pays différents à travers le monde, a mis au point une liste de niveaux qui permet de grouper les véhicules autonomes. Ainsi, ils sont parvenus à six niveaux, allant de 0 à 5, de l'absence totale d'automatisation à l'automatisation totale.

Le niveau 0 correspond à un véhicule où le conducteur doit réaliser toutes les tâches de conduite, c'est-à-dire tourner le volant, freiner et accélérer. Il peut disposer d'un système d'aide au freinage d'urgence ABS car celui-ci ne sert pas à le piloter. La plupart des véhicules sont de niveaux 0 de nos jours.

Un véhicule de niveau 1 comprend plusieurs technologies d'aides à la conduite. En effet, il comporte un système de régulateur de vitesse automatique, une assistance au maintien sur la voie et peut permettre de garder une distance de sécurité avec un véhicule le devançant. Néanmoins, le conducteur doit être totalement en contrôle de son véhicule.

Ensuite, il y a les véhicules d'autonomie de niveau 2, avec qui beaucoup de personnes sont familières. Dans un pays comme les Etats Unis où les ventes de voitures premiums sont largement dominées par Tesla, une grande partie de la population utilise donc l'Autopilot. Cette technologie de Tesla et le Super Cruise system de General Motors sont tous les deux qualifiés de niveau 2. Le véhicule est capable de tourner, d'accélérer et de freiner tout seul, ce qui laisse un peu de liberté au conducteur. Cependant, il doit quand même avoir les mains sur le volant tout le temps.

Des véhicules de niveau 3 ne sont pas encore commercialisés. Ils sont capables de se piloter eux-mêmes dans n'importe quelle situation, ils sont avertis et conscients de l'environnement qui les entourent. Le conducteur doit pourvoir être prêt à reprendre le contrôle si le véhicule n'arrive pas à réaliser une tâche. Une voiture manufacturée et produite à grande échelle de niveau 3 devait être commercialisée en 2020, l'Audi A8L. Face à un manque d'avancées technologiques des concurrents et à une incompétence des états de pouvoir créer des nouvelles règles concernant cette technologie, Audi a préféré abandonner le projet et proposer une simple aide à la conduite sur ses véhicules.

La différence principale entre le niveau 3 et le niveau 4 est que les véhicules de ce niveau agissent tout seul dans presque toutes les situations, même dans les situations critiques. Le conducteur a encore l'option de pouvoir prendre le contrôle s'il le souhaite. La loi ne permet pas un déploiement total de ces véhicules, c'est pourquoi ils sont souvent utilisés dans des zones prédéfinies. Cette technologie est beaucoup utilisée dans des taxis autonomes chez des sociétés françaises comme NAVYA qui produit et vend des taxis et des navettes, mais aussi elle va se développer grâce à un accord entre Volvo et un géant de l'internet chinois, Baidu. Un accord qui va permettre le développement de taxi autonomes électriques en Chine. Enfin la société sur laquelle ce rapport s'appuie principalement, Waymo, qui propose des taxis autonomes en Arizona, mais surtout qui test ses véhicules de niveau 4 depuis plusieurs années, et qui ont parcouru plus de 10 millions de kilomètres.

Le dernier niveau d'autonomie est le niveau 5. Les véhicules de ce niveau ne proposent pas de volant ni de pédales, tout est automatisée. Le conducteur n'a plus besoin de se soucier de la conduite car le véhicule est censé avoir les mêmes compétences que lui mais avec plus d'expérience. Il y a eu quelques tests de voitures avec ne niveau d'autonomie, mais pour l'heure, aucune n'est disponible. Il faudra attendre encore quelques années avant de voir des prototypes fiables.

Nous avons vu les différents niveaux d'autonomie des voitures autonomes, nous allons maintenant nous intéresser à la marque sur laquelle l'étude est focalisée.

C. Pourquoi Waymo

a. Histoire de la marque

Tout a commencé lors de l'édition 2005 du DARPA Grand Challenge, une compétition mettant en jeu des véhicules terrestres sans pilotes et autonomes. Les gagnants de cette édition étaient une équipe d'ingénieurs de l'université de Stanford. Ils avaient ajouté à un Volkswagen Touareg des technologies qui lui avait permis de rouler de manière autonome à travers le désert de Mojave. Parmi ces ingénieurs, il y avait Sebastian Thrun. Celui-ci s'est entouré de Anthony Levandowski et de Mike Montemerlo pour commencer le projet de la Google Car. Ce n'est qu'en octobre 2010 que Google annonce avoir développé un système de pilotage automatique qu'ils avaient installé sur sept voitures. En 2014, Google X présente son premier prototype de voiture autonome. Il ne comporte ni pédales ni volant et atteint une vitesse maximale de 40km/h. Il est encore en phase de test à ce moment-là. Un an après, le véhicule obtient le droit de rouler sur les routes californiennes sous condition qu'un ingénieur soit à bord. L'année d'après, le projet se rebaptise Waymo et devient une filiale d'Alphabet. Après deux ans de tests, Waymo présente son nouveau service, Waymo One. Un service de taxis autonomes que l'on appelle via une application sur smartphone. Waymo a signé des partenariats avec des groupes comme Fiat Chrysler et Jaguar Land Rover pour installer ses composants sur des voitures de séries notamment la Chrysler Pacifica et la Jaguar I-Pace. Aujourd'hui de nombreux véhicule se déplacent sur les routes des Etats Unis, ils ont lancé leur dernier modèle de la Jaguar I-Pace de cinquième génération, le modèle que nous allons étudier.

Figure 6 : Gamme de véhicule de Waymo

Il existe aussi une branche de la marque nommée Waymo Via qui s'occupe de la livraison par camion. En effet depuis 2017 des tests de livraison par leurs camions autonomes sont en cours dans les états de l'Arizona et de la Californie.

b. Motivations de la marque

On pourrait se demander pourquoi une entreprise comme Google s'intéresse de si près aux voitures autonomes et quelles sont ses motivations. Tout d'abord, tous les géants de la technologie s'attaquent à ce marché car ils savent que celui-ci est en pleine expansion et que celui qui parviendra à produire un véhicule fiable en premier pourrait être le leader de ce marché très rapidement. Google a fait un constat basé sur plusieurs statistiques. Il y a eu 1,35 millions de morts causés par des accidents de voitures dans le monde en 2016. 2,4 millions de blessures sont dues aux accidents de voitures en 2015. Deux personnes sur trois seront impliquées dans un accident lié à l'alcool dans leur vie. Une augmentation de 6% du nombre de morts sur la route en 2016. Ils se sont aussi appuyés sur le fait qu'aux Etats Unis, douze millions des personnes de 40 ans et plus sont atteints de trouble de la vision ou de perte total de la vision. On sait aussi que 79% des personnes âgées vivent dans des endroits où ils sont dépendants d'un moyen de locomotion.

Le projet de la Google Car n'était au départ qu'un système comme celui de Tesla où le conducteur devait rester attentif à tout moment pour pouvoir reprendre le contrôle de la voiture. Cependant, après une expérience où plusieurs de leurs employés étaient amenés à essayer ce système, ils se sont rendu compte que le conducteur avait très rapidement une confiance totale dans les capacités d'autonomie du véhicule, donc qu'il s'occupait de ses affaires personnelles plutôt que de prêter attention à la route. Les employés rédigeaient des messages, se remaquillaient ou même parfois dormaient. C'est pour cela qu'ils se sont focalisés sur les voitures à autonomie complète. Enfin une étude de la sécurité routière des Etats Unis à montrer que 94% des accidents étaient dues par une erreur de l'humain et non du véhicule. Ainsi, le but principal de Waymo est de sauver des vies en proposant des véhicules autonomes qui retire totalement la responsabilité de l'Homme dans ses choix.

Enfin, Waymo propose son service de taxi autonome en libre-service en 2018. Waymo a remarqué que la plupart de la population n'utilisait sa voiture qu'une heure ou deux par jours, sinon elle restait dans un garage ou sur un parking. Partant de ce constat, l'idée qu'en un seul clic sur son smartphone on puisse commander une voiture qui nous emmène au travail et qui continue sa route tout au long de la journée et même de la nuit leur est parût intéressante. Ce système déployé à grande échelle permettrait à la population de ne plus dépenser d'argent dans un véhicule personnel, donc directement de moins produire de véhicules et surtout de moins polluer.

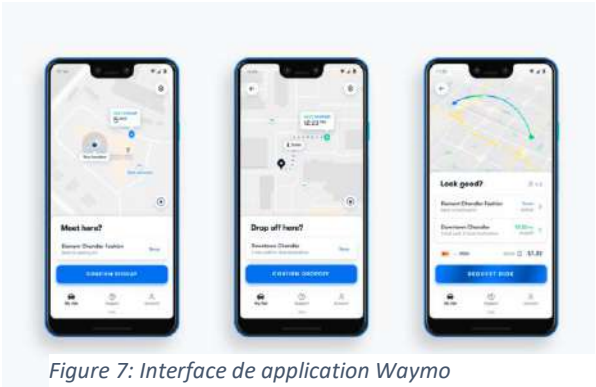


Figure 7: Interface de application Waymo

Figure 8: Taxi de Waymo One

c. Leader du marché

Si pour l'instant le marché des voitures autonomes est minoritaire, il pourrait représenter un marché de 55 millions de véhicules dans une vingtaine d'années. Les constructeurs américains devraient être en deuxième position tandis que l'Asie prendrait la première place. Cependant, ce sont actuellement les Etats Unis qui sont leader des innovations et des test réalisés dans les villes. Les véhicules avec un niveau d'autonomie de niveau 4 ou 5 ne devraient pas voir le jour dans une production massive avant les années 2030, cependant un facteur pourrait faire accélérer ce processus, les partenariats. En effet Waymo est très en avance sur ce point et nous l'a démontré en achetant des voitures à Chrysler et à Jaguar.

Actuellement la Californie est l'endroit où il y a le plus de tests de véhicules autonomes. En effet, 62 entreprises ont reçu l'accord pour tester la fiabilité de leurs véhicules, mais en contrepartie, ils doivent fournir leurs données à l'état. C'est encore ici Waymo qui domine le secteur car 64% de l'ensemble des kilomètres de tests parcourus dans l'état par des véhicules autonomes sont réalisés par ceux de Waymo.

Waymo domine dans un autre secteur, le taux d'intervention humaine lors de la conduite de ses véhicules. En effet entre décembre 2017 et 2018, les véhicules de Waymo ont parcourus environ deux millions de miles, et sur ceux-ci, les chauffeurs n'ont eu qu'à intervenir une fois tous les 11 018 miles. Son rival General Motors avec son système Cruise Automation a parcouru 447 621 miles avec une intervention tous les 5205 miles. Waymo propose des véhicules avec un taux de désengagement de 0.09 pour 1000 miles, vient ensuite General Motors avec 0.19 et Zoox avec un taux de 0.52. On pourrait penser que Apple propose des véhicules performants, mais ce n'est pas le cas, les premiers essais de la marque à la pomme totalisaient 871,65 désengagements tous les 1000 miles.

Enfin, Waymo pousse les ingénieurs et les développeurs à proposer des nouvelles solutions pour la marque. La société a aussi mis le code de leur logiciel en open-source sur GitHub afin que tout le monde puisse y accéder et s'en inspirer. Waymo organise aussi un évènement nommé le Waymo Open Dataset Challenges, qui propose à des équipes de personnes intéressées de venir répondre à plusieurs défis technologiques. Le but est d'aider la recherche dans l'avancement de la détection des machines et de la technologie des véhicules autonomes en générale.

Pour conclure, nous pouvons dire que Waymo surpasse ses concurrents dans la fiabilité, les tests, mais surtout en matière d'avancement technologique, ce sont toutes les raisons qui m'ont amené à focaliser mes recherches sur cette entreprise. Nous allons maintenant nous intéresser à la perception de l'environnement par ces voitures

II. Perception de l'environnement

A. Les capteurs

Dans cette partie nous allons nous intéresser aux différents capteurs présents sur la cinquième génération des véhicules de Waymo, et donc sur le Jaguar I-Pace équipé de tous ces capteurs. La voiture de Waymo est équipée de trois capteurs différents, des radars, des caméras et des LIDAR. Tous ces capteurs sont reliés à un ordinateur central situé dans la voiture, qui va permettre l'analyse de toutes les données envoyées continuellement par les capteurs. Les caméras et les LIDAR sont

étroitement liés c'est pour cela que je vais tout d'abord m'intéresser aux radars, puis dans un second temps aux deux autres.

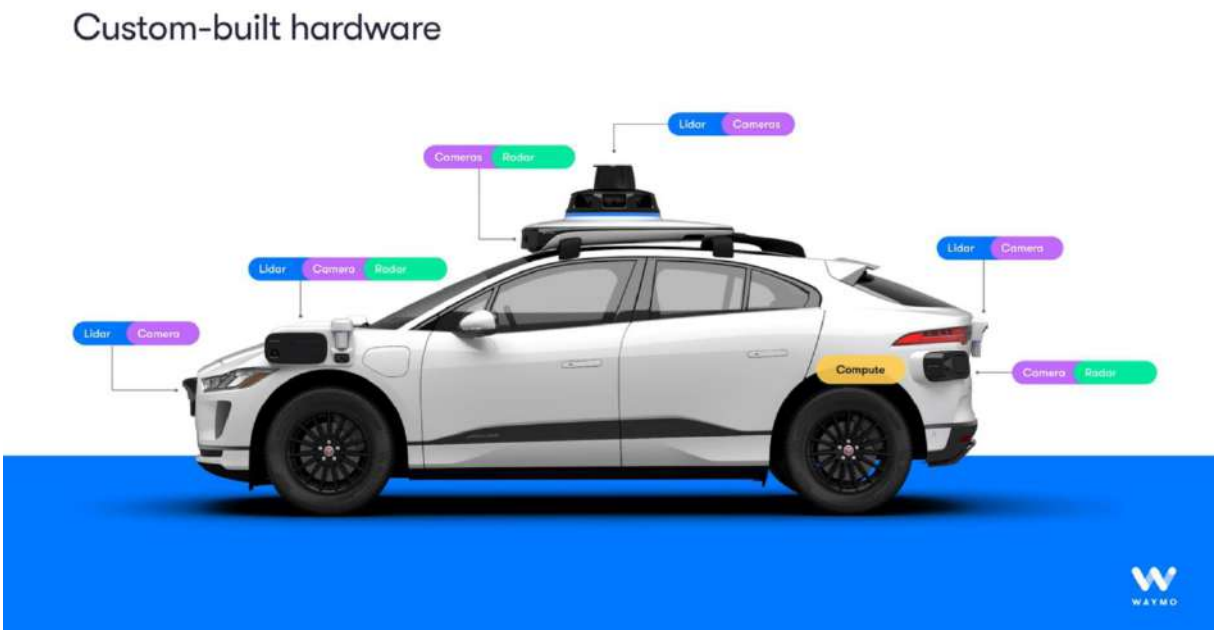


Figure 9 : Répartition des capteurs sur la voiture

a. Fonctionnement et disposition des radars

Commençons par faire un rappel sur le fonctionnement des radars. Un radar est un appareil qui émet une impulsion focalisée de micro-ondes. Une partie du faisceau énergétique rebondit sur l'objet visé, puis revient sur le radar pour qu'il puisse la mesurer. Un radar peut calculer une taille, une vitesse, une quantité mais aussi la direction d'un mouvement.

Waymo a équipé ses véhicules de radar dans un but bien précis. En effet, alors que les Lidar permettent de voir les objets et que les caméras permettent d'appréhender l'environnement du véhicule, les radars servent à mesurer d'une manière instantanée la vitesse des objets comme les voitures ou les piétons. L'avantage principal qu'un radar possède par rapport à une caméra par exemple, c'est qu'il peut fonctionner dans n'importe quelles conditions météorologiques. En effet, il ne se soucie pas de la pluie, du brouillard ou de la nuit. La marque a développé ses propres radars capables de détecter des objets à quelques centaines de mètres. Cela permet au véhicule d'appréhender plus rapidement ce qu'il va arriver et donc permet au conducteur d'être plus serein.

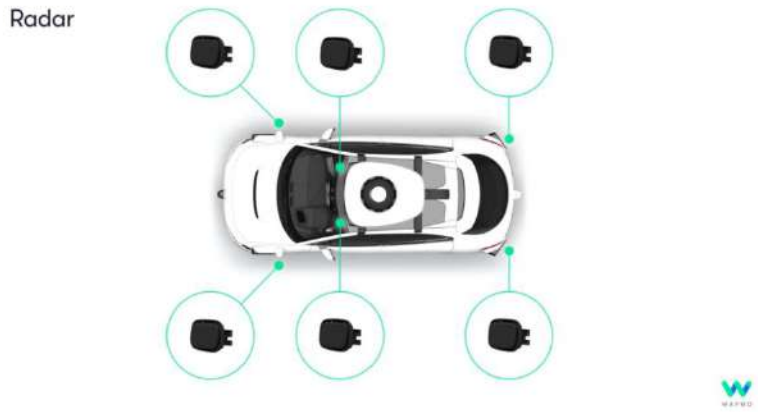


Figure 10: Répartition des radars

Le véhicule est donc équipé de six radars disposés tout autour de la voiture. Cette disposition géométrique simple permet aux radars de couvrir tous l'espace environnant. Ils sont arrivés à la conclusion qu'il fallait que les radars offrent une vue complète de l'environnement après une décennie de tests. Les radars de cette cinquième génération sont les premiers à proposer une imagerie avec une telle portée, définition et champ de vision adaptée aux voitures autonomes. Leurs performances sont encore améliorées grâce à la combinaison avec les données des caméras et des lidars.

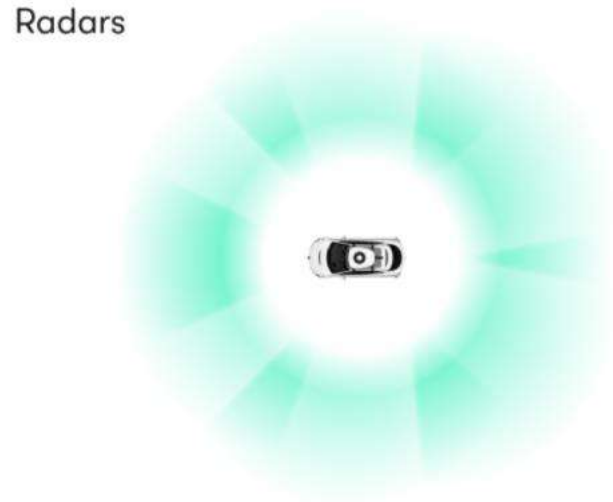


Figure 11 : Champ de vision des radars

b. Fonctionnement et disposition des caméras et des Lidars

Je ne vais pas expliquer en détail le fonctionnement d'une caméra numérique car ce ne serait pas pertinent dans le cadre de ce rapport. Il faut juste retenir qu'une caméra capture quelques dizaines d'images par seconde.

Cependant le fonctionnement d'un lidar est plus complexe. Son acronyme signifie « light detection and ranging », ce qui se traduit par la détection et l'estimation de la distance par la lumière. Le concept

d'utiliser la lumière pour faire des mesures n'est pas nouveau bien entendu, il est utilisé par exemple dans les radars de la police. Le fonctionnement d'un tel système est simple, un faisceau lumineux est envoyé, rebondit sur une surface puis est capté par l'émetteur. Un lidar est très souvent associé à un moteur qui permet de le faire tourner dans des directions variables afin de parcourir tout l'environnement de l'objet. Le temps que le faisceau met pour faire l'aller-retour est transformé ensuite en une distance avec un calcul très simple : distance = vitesse * temps. Les types de faisceau lumineux peuvent appartenir aux domaines des ultra-violets, du visible, ou de l'infrarouge. La particularité des Lidars est que plusieurs émissions sont envoyées à intervalles très courts. Ce procédé permet que chaque distance soit transformée en un pixel, et ces nombreux pixels forment entre eux un nuage de points qui peut être associé à une image composée de pixels. Cette image en 3D peut être ensuite analysée pour permettre de détecter des obstacles. Tout ce qui entoure le radar est représenté sous un nuage de points permettant de discerner les différentes formes et distances.

Waymo a choisi une disposition spéciale de ses deux types de capteurs. La voiture dispose de 29 caméras disposées tout autour de la voiture. Cette disposition avec une caméra avant, une caméra arrière, une avant droite et gauche, une côté droit et gauche, une arrière droite et gauche et une sur le toit permet de couvrir toutes les zones environnantes possibles. Les lidars eux sont disposés de la manière suivante. Les quatre reprennent les positions des caméras avant, arrière, côté droit et côté gauche, et un lidar longue portée est situé sur le toit.

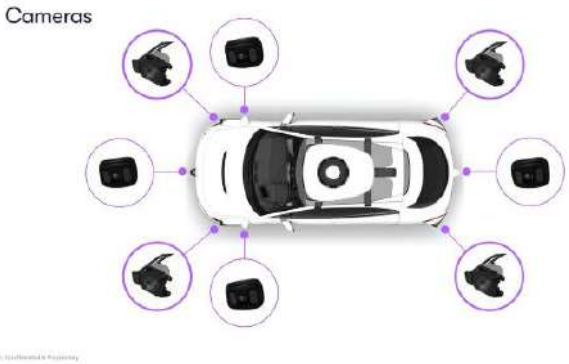


Figure 12: Répartition des caméras

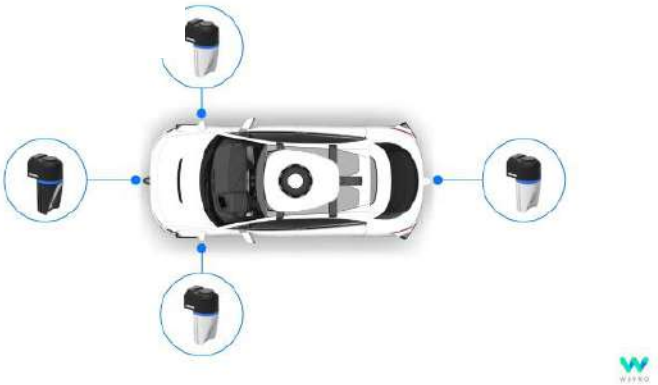


Figure 12: Répartition des lidars

Les caméras dont sont équipés le véhicule ont les caractéristiques suivantes. Les caméras avant, avant droite et avant gauche fournisse des vidéos en format 1920x1280 tandis que celle des côtés droit et gauche en fournissent de format 1920x1040. La caméra du dessus a une portée de 500 mètres. Les lidars développés et produits par Waymo nommés Laser Beam Honeycomb permettent d'avoir des nuages de millions de points et le moteur les faisant tourner peut atteindre des vitesses en milliers de tours par minutes. Ils ont un champ de vision horizontal de 360° et vertical de 95° ce qui lui permet de passer devant ses concurrents qui eux n'en proposent que de 30°.

Figure 14: Lidar HoneyComb seul

Figure 15 : Lidar HoneyComb sur une voiture

Enfin, les deux capteurs situés sur le toit, c'est-à-dire la caméra longue portée et le Lidar moyenne portée sont reliés pour que l'ordinateur puisse traiter leurs informations d'une manière globalisée que nous détaillerons dans la prochaine partie. C'est aussi le cas pour les caméras et lidars avant et arrière qui contribuent à une vision périmétrique, mais aussi pour ceux des côtés droits et gauches qui contribuent eux à une vision périphérique. En effet, en ce qui concerne la vision périmétrique, le lidar s'occupe de détecter les obstacles, la caméra permet de les identifier et de donner un contexte à la scène qu'elle perçoit avec du Machine Learning.

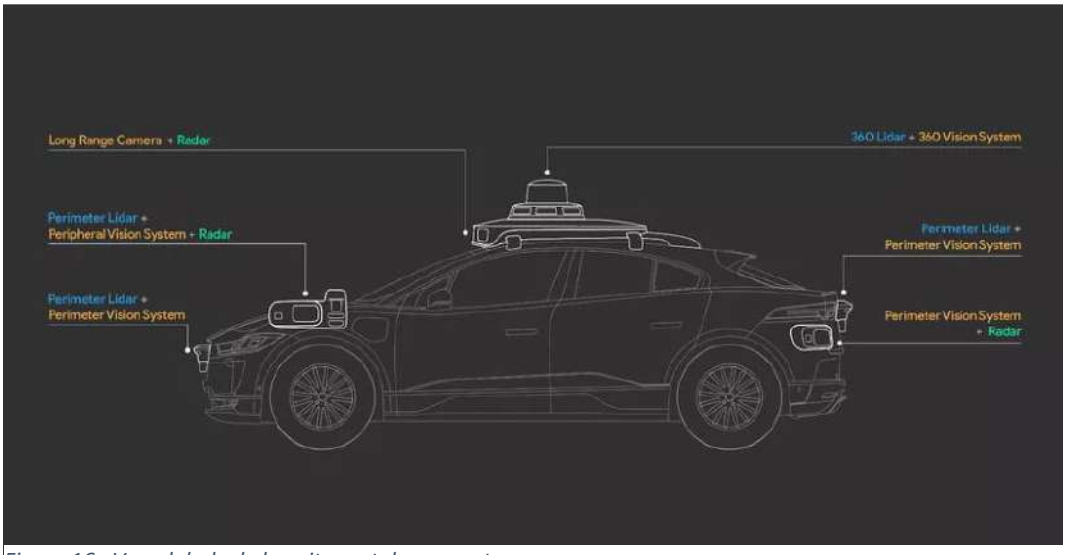


Figure 16 : Vue globale de la voiture et de ses capteurs

Nous avons vu ici les différents capteurs présents sur la voiture de Waymo, nous allons maintenant nous intéresser à leur utilisation avec la création de Labels.

B. Labels

Nous allons dans cette partie nous intéresser à la création des « bounding boxes », des boîtes créés grâce à du code à partir des données fournies par les caméras et les Lidars. La partie logicielle des voitures autonomes créer des bounding boxes labélisées pour délimiter et détecter les objets que les capteurs perçoivent. Celles-ci sont labélisées, c'est-à-dire que chacune d'entre elles est unique et possède ses propres informations.

Il existe plusieurs types d'objets pouvant être labélisés. Les véhicules, les piétons, les cyclistes, les panneaux de signalisations et enfin les objets inconnus. Tous les autres objets ne sont pas labélisés. Cependant, l'ordinateur reconnaît des zones qui ne doivent pas être labélisés, comme la voie à contre sens sur l'autoroute, ou un parking sur le bord de la route. Nous allons maintenant détailler les spécifications de ces labels pour les trois types principaux.

a. Les véhicules motorisés

Tout d'abord, il faut savoir que les objets considérés comme des véhicules motorisés sont les voitures, les camions, les bus et les motos. Les box des motos comprennent à la fois la moto et le motard. Elles intègrent les rétroviseurs mais pas les petits objets qui dépassent comme une antenne ou un attelage. Les véhicules tirant des remorques sont séparés en deux box, une pour le véhicule lui-même et une pour la remorque. Les véhicules comme les autres types d'objets peuvent être labélisés grâce à une caméra mais aussi grâce à un lidar. Voici les exemples des deux sortes.

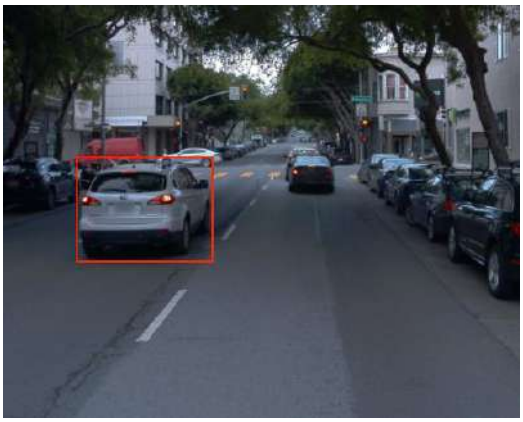


Figure 17: Label 2D d'un véhicule

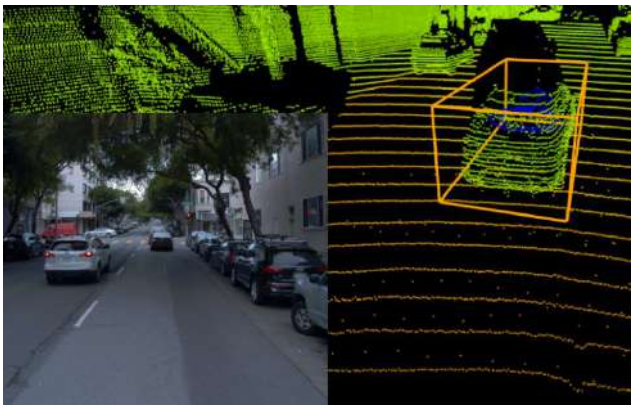


Figure 18: Label 3D d'un véhicule

On peut voit sur l'image de droite, qu'un label 2D a été créé autour de la voiture à partir d'une photo alors que sur l'image de droite, on remarque qu'un label 3D a été créé dans un nuage de points.

b. Les piétons

Nous allons maintenant nous intéresser à la création de labels pour des piétons. Comme énoncé précédemment, les piétons peuvent être labélisés par les caméras ou les lidars. Les personnes roulant sur des trottinettes, même électriques, des segways et des skateboards sont considérés comme des piétons. Les personnes présentes sur des publicités, dans les miroirs, et les mannequins de magasins ne sont pas considérés comme tels. Une personne portant dans ses bras un enfant ou un objet de moins de deux mètres de long sera dessiné dans une seule boxe. En revanche, si la personne pousse une poussette ou un cadi, alors deux box sont dessinées, une box piéton et une boxe objet inconnu. Voici les exemples des Label 2D et 3D pour les piétons.

Figure 19: Label 2D d'un piéton

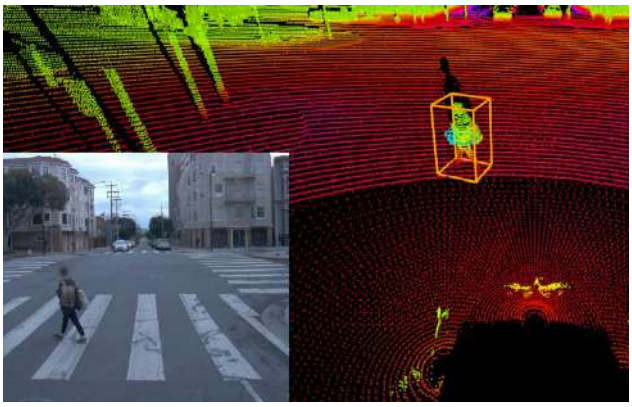


Figure 20: Label 3D d'un piéton

c. Les cyclistes

L'avant dernier objet pouvant être labélisé est le cycliste. En effet, chaque objet pouvant être détecter comme un cycliste sur son vélo est dessiné dans une box. Les vélos ne roulant pas ou étant à côté d'un piéton le poussant ne sont pas considérés comme des objets cyclistes. Les objets comme des sièges enfants ou des petites remorques de vélos sont inclus dans les box. Avec les deux exemples ci-dessous on remarque bien que le cycliste et le vélo sont associés.

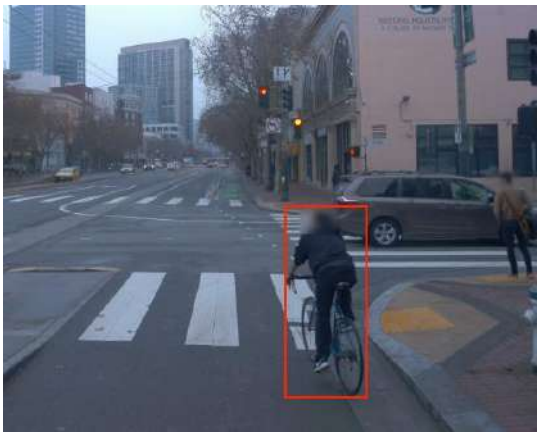


Figure 21: Label 2D d'un cycliste

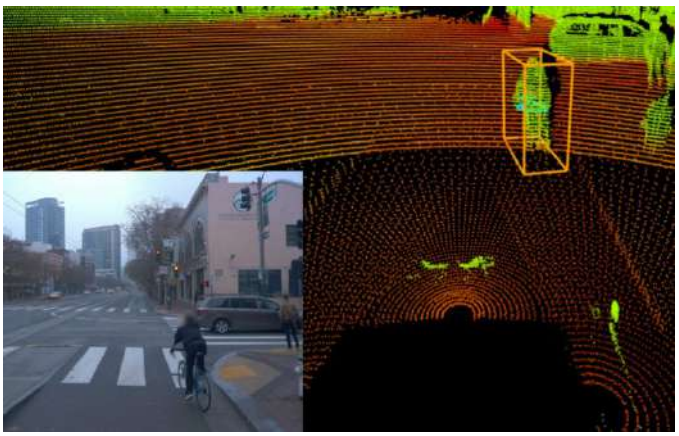


Figure 22: Label 3D d'un cycliste

d. Les panneaux de signalisations

Le dernier objet pouvant être labélisé est tout aussi important que les autres, il s'agit des panneaux de signalisation. Des box sont créées si les panneaux ont un rapport avec la circulation. En effet les panneaux de noms de rues et les pancartes de publicités ne sont pas labélisés. Le label est créé seulement à l'endroit de l'information, c'est-à-dire que le poteau n'est pas inclus dans la box. Voici un exemple de panneaux labélisés en 3D.

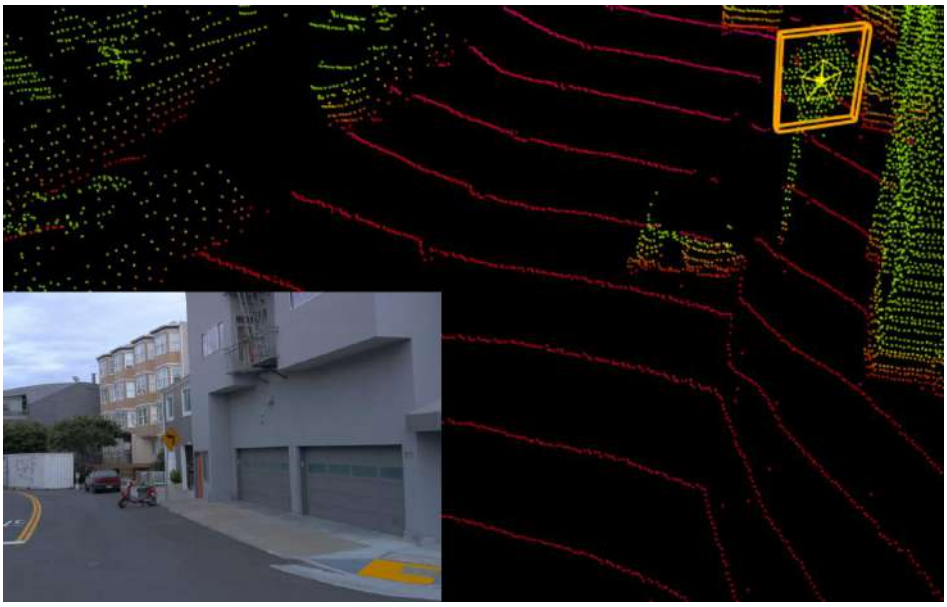


Figure 23: Label d'un panneau de signalisation

Ainsi, nous avons vu comment une voiture autonome discernait les différents objets environnants, en créant des bounding boxes à partir des données des caméras et des lidars. Nous allons maintenant nous intéresser à la prise de décision des ordinateurs de la voiture autonome.

III. Prise de décision du véhicule

A. Rappels et Data Augmentation

Afin de rouler d'une manière sécurisée, la voiture Waymo doit percevoir tout ce qui l'entoure. Les capteurs comme les caméras et les Lidars permettent ceci, néanmoins ils ne sont pas capables de fonctionner tout seul. En effet, il ne s'agit pas seulement de voir un objet, il s'agit de le reconnaître. Pour cela, l'ordinateur de la voiture doit traiter les données reçues des capteurs et par un algorithme, discerner et labéliser les objets. Cet algorithme utilise une technologie d'intelligence artificielle en réseau de neurones.

Avant toute chose, j'aimerai rappeler comment est-ce qu'une intelligence artificielle s'entraîne et apprend. Premièrement, il faut savoir qu'une IA est un ensemble de techniques crées par l'Homme dans le but d'imiter une intelligence humaine dans un processus. Cependant, dans un exemple comme la reconnaissance d'image comme celui que nous étudions ici, il est impossible de créer un algorithme qui reconnaisse parfaitement les images sans jamais avoir été entraîné. Une analogie peut donc se faire avec un nouveau-né. En effet, un nouveau-né ne peut ni marcher ni écrire ni lire à la naissance. Il

faut qu'il passe par des phases d'apprentissages où il apprendra de ses erreurs avant de pouvoir maîtriser une tâche. C'est le même scénario chez les algorithmes d'intelligence artificielle. En effet, tout comme un professeur ou un parent pourrait faire, un ingénieur dis à l'algorithme ce qu'il faut apprendre et comment l'apprendre. Dans le cas étudié, l'apprentissage est supervisé. Il faut savoir qu'ici, l'objectif du réseau de neurones est de faire de la classification. C'est-à-dire qu'à partir d'une image ou d'un nuage de points donnés, le réseau de neurones (RN) doit pouvoir savoir à quelle catégorie l'image se rattache. Le principe est simple et repose sur une logique de boîte noire que nous détaillerons plus tard, on donne une image, donc un tableau de pixels en entrée, puis en retour on reçoit une catégorie d'objet représentée sous la forme d'un vecteur.

Comme énoncé précédemment, l'ordinateur a besoin de s'entraîner pour pouvoir discerner toutes sortes d'objets ou de personnes. La base d'images sur laquelle il s'appuie est directement tirée de toutes les données que les capteurs ont recueillis lors des essais en réel des voitures. Cependant, même si nous pouvons penser que le nombre d'images ou de nuages de points est conséquent, ce n'est pas le cas. L'ordinateur a besoin de beaucoup plus de données pour pouvoir être fiable et donc s'entraîner correctement. Un des problèmes d'un entraînement sur un nombre limité d'images est l'overfitting. Ce phénomène apparaît quand l'ordinateur apprend sur un dataset limité. . Il se traduit par des prédictions non pertinentes sur des images extérieures à celles contenues dans le dataset Un dataset est un ensemble des données. Deux solutions étaient envisageables avant l'arrivée de la Data Augmentation (DA). Tout d'abord d'augmenter la taille du dataset, mais cela nécessitait un travail de récupération et de labélisation conséquent. La deuxième option était de réduire les paramètres de reconnaissance mais cela restait encore très complexe.

Pour s'entraîner, on le laisse deviner tout seul la catégorie de l'image, puis on mesure ses résultats en les comparant aux données réelles. Ensuite, on applique un autre algorithme qui permet qui modifie le réseau de neurones pour minimiser les erreurs. A force de corrections, le RN parvient à prédire efficacement la catégorie d'une image. Cependant, plus la base de données est grande, plus il sera précis. De plus, plus les données sont optimisées pour le rendre efficace, plus il sera précis et fiable. C'est là que la Data Augmentation prend du sens.

La DA est un procédé qui permet d'augmenter de façon conséquente la taille du dataset à travers un algorithme de manipulation d'image. On modifie l'aspect des images comme la luminosité ou la rotation afin d'obtenir de multiples images à partir d'une seule. Nous allons maintenant nous intéresser à deux technologies, RandAugment et PPBA (Progressive Population Based Augmentation). Les deux sont des algorithmes qui permettent de trouver automatiquement des combinaisons de DA permettant de trouver des modèles pour entraîner les algorithmes de reconnaissance d'objets. C'est-à-dire que ce sont des algorithmes qui sont capables de créer de manière autonome des stratégies de DA, donc d'augmentation d'images, afin que l'IA puisse s'entraîner sur une base d'images qui va rendre plus efficace son entraînement. Enfin, on évalue les performances du réseau de neurones en lui présentant une banque d'images inconnues, et on mesure son taux d'erreur concernant la reconnaissance d'objets. RandAugment s'applique sur des données d'images récupérées par les caméras, tandis que PPBA, lui, sur des nuages de points récupérés par des LIDAR.

B. RandAugment

Dans cette partie, nous allons nous intéresser à toutes les transformations qu'une image peut subir, puis par la suite nous nous intéresserons au fonctionnement global de l'algorithme.

a. Transformations

RandAugment (RA) est une technologie servant à trouver des méthodes de DA de manière automatique. Elle n'est pas la première à être utilisée dans des voitures autonomes. En effet, d'autres ont vu le jour, et RA s'en est inspirée. AutoAugment est une technologie qui a été beaucoup utilisée mais qui proposait une phase de recherche dans un serveur proxy séparé. Une phase de recherche est l'endroit où on instancie tous les paramètres visant à transformer une image. Le fait d'avoir une phase de recherche dans un proxy comportait des désavantages comme une exécution des programmes plus longues, donc une consommation des ressources de l'ordinateur plus importante. C'est pour cela que RA a décidé d'abandonner cette logique pour intégrer tous les paramètres directement dans l'algorithme en les réduisant en taille. RA s'est aussi inspiré de PBA (Population Based Augmentation), une technologie qui avait fait remarquer qu'avoir une seule variable de magnitude pour toutes les transformations était optimale, nous expliquerons cela par la suite. Comme énoncé précédemment la phase de recherche n'est plus séparée. Le nombre de paramètres donc de transformations possibles appliquées à l'image est de K = 14 :

- identity
- autoContrast
- equalize
- rotate
- solarize
- color
- posterize
- contrast
- brightness
- sharpness
- shear-x
- shear-y
- translate-x
- translate-y

Figure 24: Liste des transformations possibles

Ces paramètres sont donc l'identité de l'image, toujours vraie, la rotation, la postérisation, une technique consistant à réduire le nombre de couleurs dans l'image, la netteté, la translations sur l'axe X. Mais aussi le contraste automatique, l'inversion des tons, le contraste, la transvection sur l'axe X et la translation sur l'axe Y. Enfin, il y a l'égalisation, la variation du nombre de couleur, la luminosité et la transvection sur l'axe Y. Nous allons maintenant détailler le code de ces treize transformations (l'identité ne change pas), grâce au projet GitHub open-source disponible.

Nous devons tout d'abord expliquer une fonction qui est beaucoup utilisée. La fonction wrap prend en paramètre une image. Elle utilise la fonction « shape » contenue dans le module TensorFlow appelé « tf ». Cette fonction retourne la forme du tenseur donc de l'image dans une liste. Ensuite la fonction « ones » est appelée, elle créer un tenseur avec toutes les valeurs instanciées à 1. Donc la variable « extended channel » contient maintenant un tenseur de la même forme que l'image mais avec des valeurs de 1. On utilise ensuite la fonction « concat » qui sert à concaténer deux tenseurs sur une même dimension. On applique cette fonction aux deux tenseurs « image » et « extended_channel ». Ces deux vecteurs maintenant concaténés sont associés à la variable « extended » qui est retournée par la fonction wrap. Cette fonction permet de ne pas avoir de vide là où l'image après modification n'existe plus. Elle crée des pixels dans les endroits vides

```
def wrap(image):
  """Returns 'image' with an extra channel set to all 1s."""
  shape = tf.shape(image)
  extended_channel = tf.ones([shape[0], shape[1], 1], image.dtype)
  extended = tf.concat([image, extended_channel], 2)
  return extended
```

Figure 25: Fonction wrap

Rotation :

Figure 26 : Image ayant subi une rotation

La fonction rotate prend trois paramètres, l'image, le degré de rotation et une variable « replace ». L'image est un tenseur codé sur 8 bits. Le degré est un flottant, si sa valeur est positive alors l'image tourne dans le sens des aiguilles d'une montre et inversement. « Replace » contient des valeurs de tenseur d'une seule dimension pour remplir les pixels vides causés par la rotation. La fonction convertit les degrés en radian. Elle utilise la fonction rotate contenue dans le module « tensorflow.contrib ». Cette fonction permet de faire tourner une image par rapport à un angle donné. Cette fonction rotate prend en paramètre « wrap(image) » donc l'image de base avec des pixels pour combler ceux causés par la rotation, et la variable « radians » instanciés avant. Enfin, la fonction retourne le résultat de « image » et « replace » passé en paramètre par la fonction unwrap. Donc l'image tournée par rapport à un angle « degrees » et dont les pixels censés être vides sont grisés.

```python
def rotate(image, degrees, replace):
  """Rotates the image by degrees either clockwise or counterclockwise.

  Args:
    image: An image Tensor of type uint8.
    degrees: Float, a scalar angle in degrees to rotate all images by. If
      degrees is positive the image will be rotated clockwise otherwise it will
      be rotated counterclockwise.
    replace: A one or three value 1D tensor to fill empty pixels caused by
      the rotate operation.

  Returns:
    The rotated version of image.
  """
  # Convert from degrees to radians.
  degrees_to_radians = math.pi / 180.0
  radians = degrees * degrees_to_radians

  # In practice, we should randomize the rotation degrees by flipping
  # it negatively half the time, but that's done on 'degrees' outside
  # of the function.
  image = contrib_image.rotate(wrap(image), radians)
  return unwrap(image, replace)
```

Figure 27: Fonction rotate

Postérisation :

Figure 28: Image postérisée

La deuxième transformation est la postérisation. Une fonction qui modifie le nombre de couleurs disponibles dans l'image. Plus spécifiquement, elle réduit le nombre de bit pour chaque couleur (rouge, vert, bleu). Elle est l'équivalente de la fonction Posterize dans la librairie Python PIL, une fonction plus facile à aborder. Elle prend donc deux arguments, l'image et un entier appelé « bits ». Elle retourne l'image avec des couleurs codés en « (8 – bits) » bits.

PIL.ImageOps.posterize(*image, bits*)

Reduce the number of bits for each color channel.

Parameters:
- **image** – The image to posterize.
- **bits** – The number of bits to keep for each channel (1-8).

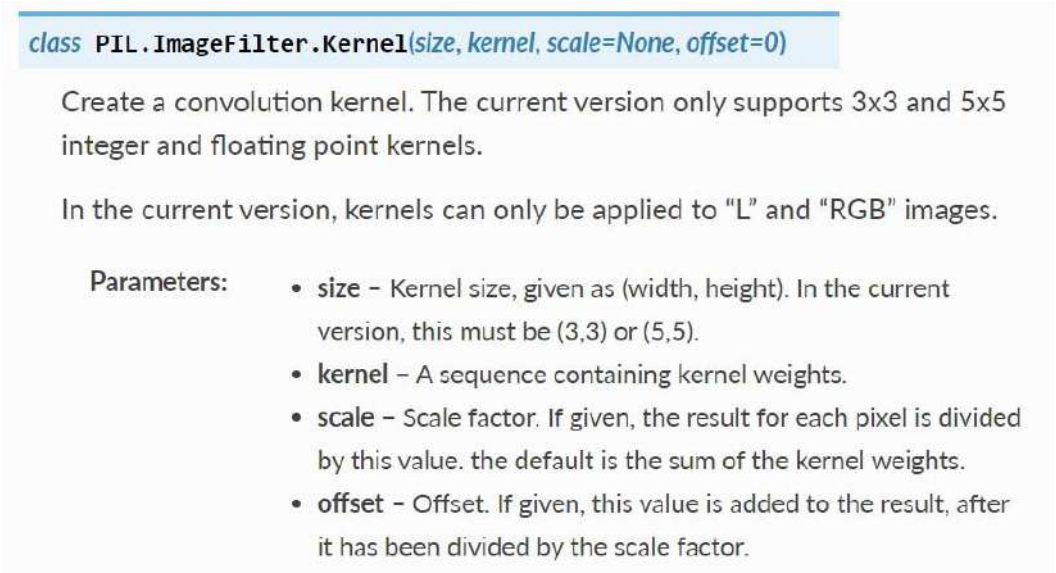
Returns: An image.

Figure 29: Fonction posterize de PIL

Netteté :

Figure 30 : Image avec une netteté augmentée

La troisième transformation est la netteté. Elle prend en paramètre une image et une variable appelée « factor ». On fait tout d'abord une copie de l'image de base afin d'avoir une sauvegarde. On caste ensuite l'image dans une variable flottante de 32 bits. Avec la fonction expand_dims, on retourne un tenseur d'une longueur de 1. Le reste de la fonction s'apparente à la méthode Kernel situé dans la librairie PIL, et plus simple à comprendre.



class **PIL.ImageFilter.Kernel**(*size, kernel, scale=None, offset=0*)

Create a convolution kernel. The current version only supports 3x3 and 5x5 integer and floating point kernels.

In the current version, kernels can only be applied to "L" and "RGB" images.

Parameters:

- **size** – Kernel size, given as (width, height). In the current version, this must be (3,3) or (5,5).
- **kernel** – A sequence containing kernel weights.
- **scale** – Scale factor. If given, the result for each pixel is divided by this value. the default is the sum of the kernel weights.
- **offset** – Offset. If given, this value is added to the result, after it has been divided by the scale factor.

Figure 31 : Fonction Kernel de PIL

Translation en X et Y :

Figure 32 : Image ayant subi une translation sur X

Figure 33 : Image ayant subi une translation sur Y

La fonction translate X permet de déplacer une image sur l'axe X selon un nombre de pixels donnés. On utilise la fonction translate du module « tensorflow », qui décale l'image selon un vecteur ou plusieurs. Ici les arguments de cette fonction sont wrap(image) et le vecteur[-pixels,0]. Cela permet de décaler l'image de moins la valeur de pixels passé en argument et d'avoir des pixels là où l'image n'existe plus. Ensuite on retourne l'image décalé en faisant appel à la fonction unwrap qui grise les pixels autour de l'image. Pour la fonction translate-Y, le code est le même sauf le vecteur qui devient [0, -pixels] pour avoir des valeurs que sur l'axe Y.

```
def translate_x(image, pixels, replace):
  """Equivalent of PIL Translate in X dimension."""
  image = contrib_image.translate(wrap(image), [-pixels, 0])
  return unwrap(image, replace)
```

Figure 34 : Fonction translate

Contraste Automatique :

Figure 35 : Image avec un contraste automatique

La fonction de contraste automatique est importée depuis la fonction éponyme de la librairie PIL. Cette fonction calcule un histogramme de la luminosité de l'image. Elle trouve le pixel le plus sombre et le pixel le plus clair. Elle transforme l'image afin que le pixel le plus sombre devienne du noir et le plus clair devienne du blanc. Elle retourne donc l'image transformée. Ce procédé permet de mieux discerner les formes avec une différence importante entre les parties claires et sombres.

PIL.ImageOps.autocontrast(*image, cutoff=0, ignore=None*)

Maximize (normalize) image contrast. This function calculates a histogram of the input image, removes **cutoff** percent of the lightest and darkest pixels from the histogram, and remaps the image so that the darkest pixel becomes black (0), and the lightest becomes white (255).

Parameters:
- **image** – The image to process.
- **cutoff** – How many percent to cut off from the histogram.
- **ignore** – The background pixel value (use None for no background).

Returns: An image.

Figure 36 : Fonction autocontraste de PIL

Inversion des tons :

Figure 37 : Image avec tons inversés

La fonction solarize prend en paramètre l'image et une variable « threshold » d'un entier donné. Tous les pixels en dessous de cette variable sont inversés. Cette fonction retourne donc l'image avec tous les pixels en dessous de 128 inversés.

```
def solarize(image, threshold=128):
  # For each pixel in the image, select the pixel
  # if the value is less than the threshold.
  # Otherwise, subtract 255 from the pixel.
  return tf.where(image < threshold, image, 255 - image)
```

Figure 38 : Fonction solarize

Contraste :

Figure 39 : Image tirée vers un contraste moyen

Figure 40 : Image tirée vers un contraste naturel

Cette fonction prend en paramètre une image, et un facteur. Tout d'abord l'image est transformée en nuance de gris et affectée à une variable « degenerate ». Cette même variable est ensuite castée en un entier de 32 bits. On créer ensuite un histogramme comprenant toutes les valeurs des pixels de l'image en niveau de gris. On calcule ensuite la moyenne des valeurs des pixels en divisant la somme de tous les pixels par 256. On créer ensuite une image de la taille de l'image passé en paramètre avec tous les pixels de la valeur moyenne calculée avant. Enfin, la fonction retourne le mélange entre l'image avec les pixels moyens, l'image de base, avec une variable factor qui selon sa valeur déterminera si le résultat du mélange sera plus tiré de l'image de base ou de l'image moyenne.

```
def contrast(image, factor):
  """Equivalent of PIL Contrast."""
  degenerate = tf.image.rgb_to_grayscale(image)
  # Cast before calling tf.histogram.
  degenerate = tf.cast(degenerate, tf.int32)

  # Compute the grayscale histogram, then compute the mean pixel value,
  # and create a constant image size of that value.  Use that as the
  # blending degenerate target of the original image.
  hist = tf.histogram_fixed_width(degenerate, [0, 255], nbins=256)
  mean = tf.reduce_sum(tf.cast(hist, tf.float32)) / 256.0
  degenerate = tf.ones_like(degenerate, dtype=tf.float32) * mean
  degenerate = tf.clip_by_value(degenerate, 0.0, 255.0)
  degenerate = tf.image.grayscale_to_rgb(tf.cast(degenerate, tf.uint8))
  return blend(degenerate, image, factor)
```

Figure 41 : Fonction contrast

Cisaillement en X et Y :

Figure 42 : Image avec cisaillement horizontal

Figure 43 : Image avec cisaillement vertical

La transformation shear peut se traduire par un effet de cisaillement horizontal. La partie supérieure est décalée vers la droite et la partie inférieure est décalée vers la gauche. On utilise la fonction transform du module tensorflow. Cette fonction prend en paramètre une image, et un vecteur. Ce vecteur de taille 8 sert à transformer les coordonnées (x,y) de la manière suivante : on a le vecteur [a0,a1,a2,b0,b1,b2,c0,c1] et les points (x',y') = ((a0x + a1y + a2)/k , (b0x + b1y +b2)/k) avec k=c0x +c1y +1. Ici, le vecteur est [1, level,0,0,1,0,0,0] avec « level » un argument de la fonction shear-x. Donc l'image est décalée sur x car a0 et b0 sont multipliés par x. La fonction transform prend donc l'image « wrappé », et ce vecteur, ce qui permet de la cisaillé en x et de combler les pixels. Le return de la fonction shear-x s'occupe de colorer ces pixels en gris et de retourner l'image cisaillée. Pour shear-y, seul le vecteur change.

```
def shear_x(image, level, replace):
  """Equivalent of PIL Shearing in X dimension."""
  # Shear parallel to x axis is a projective transform
  # with a matrix form of:
  # [1  level
  #  0  1].
  image = contrib_image.transform(
      wrap(image), [1., level, 0., 0., 1., 0., 0., 0.])
  return unwrap(image, replace)
```

Figure 44 : Fonction shear_x

Egalisation :

Figure 45 : Image avec couleurs égalisées

L'égalisation des couleurs permet d'uniformiser les couleurs de l'image de sorte qu'aucune prenne le dessus sur l'une ou l'autre. Cette fonction est très lourde à expliquer et s'apparente à la fonction equalize de la librairie PIL. Cette fonction dessine un histogramme de l'image, met à niveau tous les pixels puis en redessinant le tenseur parvient à retourner une image égalisée.

Syntax: *PIL.ImageOps.equalize(image, mask=None)*

Parameters:

image: *The image to equalize.*

mask: *An optional mask. If given, only the pixels selected by the mask are included in the analysis.*

Returns: *An image.*

Figure 46 : Fontion equalize de PIL

Nuance de gris :

Figure 47 : Image en nuance de gris

La fonction color sert à transformer l'image en nuance de gris. On transforme l'image en nuance de gris, puis on la retransforme en couleur (RGB). La fonction retourne un mélange entre l'image de base, l'image en modifié en nuance de gris et selon un facteur qui influencera laquelle des deux images prendra le dessus.

```
def color(image, factor):
  """Equivalent of PIL Color."""
  degenerate = tf.image.grayscale_to_rgb(tf.image.rgb_to_grayscale(image))
  return blend(degenerate, image, factor)
```

Figure 48 : Fonction color

Luminosité :

Figure 49 : Image avec luminosité élevée

Cette fonction contrôle la luminosité de l'image. En effet, on créer un tenseur de la forme de l'image à qui on lui associe que des valeurs égales à 0. Ensuite la fonction renvoie un mélange entre l'image de base et le l'image « zéro » puis à l'aide d'un facteur détermine qui des deux images va le plus influencer l'image finale pour avoir ou non beaucoup de luminosité.

```
def brightness(image, factor):
  """Equivalent of PIL Brightness."""
  degenerate = tf.zeros_like(image)
  return blend(degenerate, image, factor)
```

Figure 50 : Fonction brightness

En appliquant chaque transformation avec une probabilité de 1/K, avec N transformations, on obtient K^N méthodes de DataAugmentation. Ensuite il y a un paramètre appelé magnitude qui correspond au degré de distorsion de chaque transformation. On s'est rendu compte avec la PBA que ces paramètres évoluaient de manière similaire durant l'exécution des algorithmes. Donc il a été décidé qu'une seule variable globale M suffirait à paramétrer toutes les transformations. Ce paramètre M est une constante ce qui permet d'avoir un seul « hyper » paramètre global. Ainsi l'algorithme de RandAugment s'écrit en quelques lignes et ne prend seulement que trois paramètres, l'image, le nombre de transformations à appliquer « num_layers », et le degré de transformation, la magnitude.

b. Algorithme

```
def distort_image_with_randaugment(image, num_layers, magnitude):

  replace_value = [128] * 3
  tf.logging.info('Using RandAug.')
  augmentation_hparams = contrib_training.HParams(
      cutout_const=40, translate_const=100)
  available_ops = [
      'AutoContrast', 'Equalize', 'Invert', 'Rotate', 'Posterize',
      'Solarize', 'Color', 'Contrast', 'Brightness', 'Sharpness',
      'ShearX', 'ShearY', 'TranslateX', 'TranslateY', 'Cutout', 'SolarizeAdd']

  for layer_num in range(num_layers):
    op_to_select = tf.random_uniform(
        [], maxval=len(available_ops), dtype=tf.int32)
    random_magnitude = float(magnitude)
    with tf.name_scope('randaug_layer_{}'.format(layer_num)):
      for (i, op_name) in enumerate(available_ops):
        prob = tf.random_uniform([], minval=0.2, maxval=0.8, dtype=tf.float32)
        func, _, args = _parse_policy_info(op_name, prob, random_magnitude,
                                           replace_value, augmentation_hparams)
        image = tf.cond(
            tf.equal(i, op_to_select),
            # pylint:disable=g-long-lambda
            lambda selected_func=func, selected_args=args: selected_func(
                image, *selected_args),
            # pylint:enable=g-long-lambda
            lambda: image)
  return image
```

Figure 51 : Algorithme RandAugment

Tout d'abord on a une liste de toutes les transformations appelée « available_ops » ce qui peut se traduire par opérations disponibles. Ensuite, on a une boucle for qui se répète autant de fois que la valeur de « num_layers ». A chaque itération on sélectionne une transformation et une magnitude aléatoire grâce à la fonction random_uniform. On entre ensuite dans une autre boucle for qui permet à chaque itération applique la transformation à l'image, puis la retourne modifiée.

Voici un exemple d'images « augmentées » par RandAugment. L'algorithme a été exécuté avec un nombre N = 2 de transformations et avec des degrés de magnitude M différent pour chacun des trois exemples.

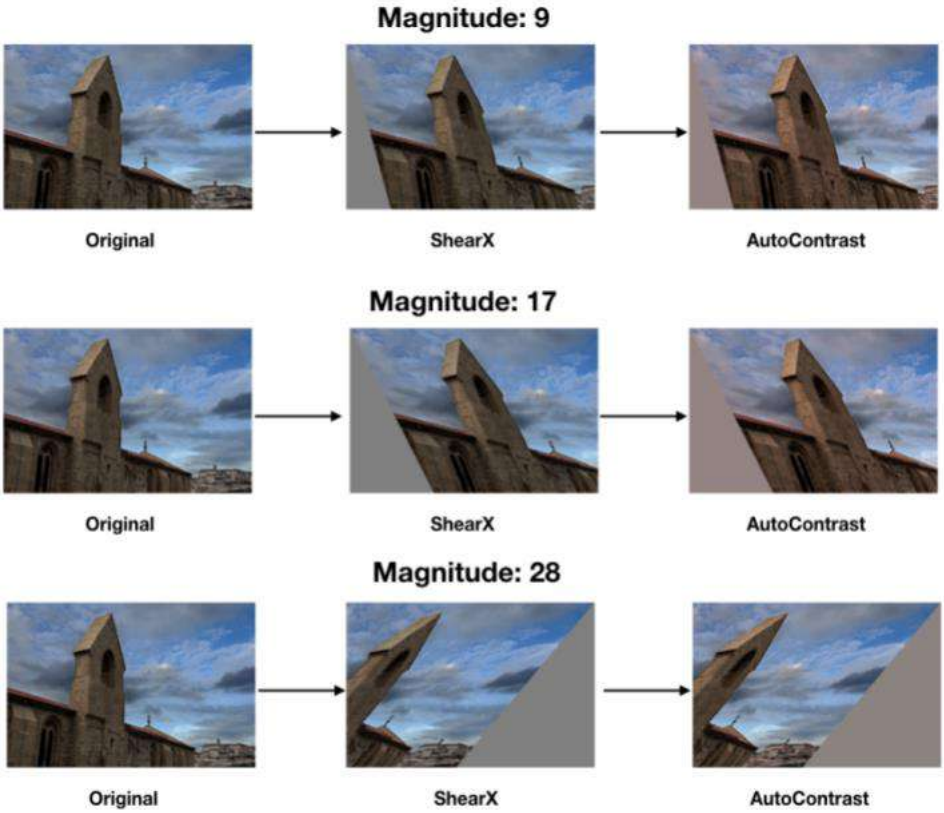


Figure 52 : Exemple de transformations successives

On remarque que dans les trois exemples la photo subit déjà une transvection en X, puis un contraste automatique, ce qui fait deux transformations. On remarque ensuite que plus la magnitude est élevée, plus les transformations sont importantes.

Nous avons vu toutes les propriétés et le fonctionnement de RandAugment, nous allons maintenant réaliser l'étude de PPBA.

C. Progressive Population Based Augmentation

Dans cette partie, nous allons tout d'abord expliquer le principe général de la PPBA, puis nous étudierons les différentes transformations de nuages de points. Enfin, nous nous intéresserons à ses algorithmes mathématiques.

a. Transformations

Dans le paragraphe précédent, nous avons montré qu'utilisé de la Data Augmentation dans la reconnaissance d'objet en 2D était très efficace. Beaucoup de travaux ont été menés concernant la DA sur les images mais peu se sont focalisés sur la reconnaissance d'objets dans des nuages de points. On ne peut pas appliquer les mêmes stratégies de DA en 3D car tout d'abord parce que l'espace de recherche est d'une tout autre grandeur, de plus cela nécessite des méthodes de recherche beaucoup

plus élaborées et efficaces. PPBA est un algorithme qui apprend à optimiser des stratégies de Data Augmentation de façon que les algorithmes de reconnaissance de nuages de points s'entraînent sur des données qui les rendent plus efficaces. Sa stratégie est la suivante. On génère aléatoirement des stratégies d'augmentation, chaque stratégie étant composée d'un ensemble d'opérations fixes déterminées aléatoirement au départ. Ensuite, on utilise chaque stratégie pour créer des données augmentées qui serviront à entraîner des réseaux de neurones différent. Puis, on évalue la performance de chacun des réseaux de neurones qui viennent d'être entraînés avec la métrique Omega. Enfin, on retient les stratégies qui ont abouti au meilleur apprentissage et on les fusionne. On suppose alors qu'on obtient une stratégie d'augmentation de données plus performante dans le sens où elle a généré des données ayant permis à un RN d'apprendre efficacement.

Ses initiales veulent dire Progressive Population Based Augmentation. De la reconnaissance d'objets entraîné par cet algorithme est dix fois plus rapide que la reconnaissance entrainée par d'autres modèles. Nous allons maintenant détailler son fonctionnement.

Le PPBA est composé de deux entités, un espace de recherche spécialisé dans la DA pour les nuages de points, et un algorithme d'optimisation des paramètres de DA. Un espace de recherche en Intelligence artificielle est un endroit où toutes les solutions à un problème sont exposées.

L'espace de recherche de la PPBA propose les huit transformations possibles applicables à un nuage de points pour créer une méthode de DA. Chaque transformation N est associée à une probabilité et à des paramètres particuliers. Les huit transformations sont les suivantes :

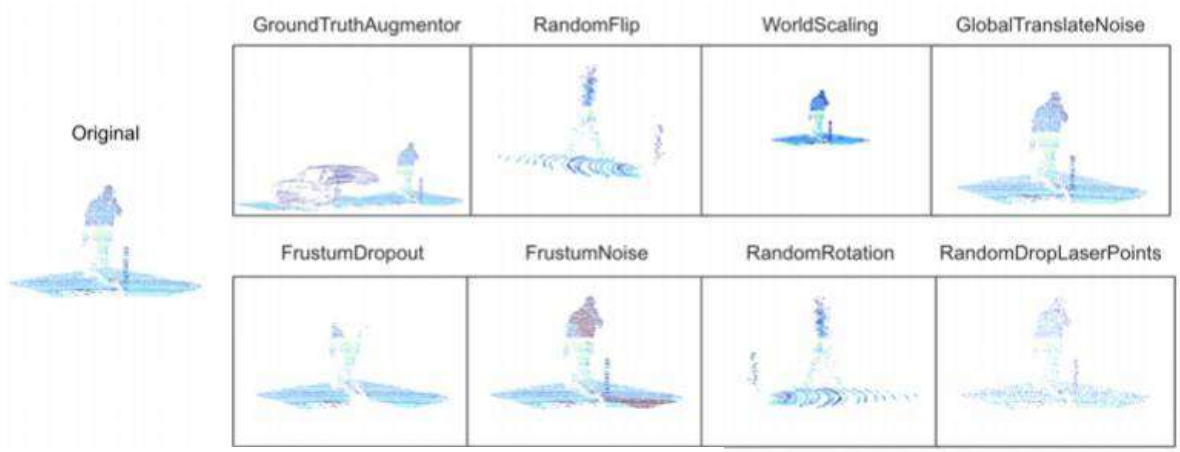


Figure 53 : Visualisation des huit transformations possibles en PPBA

GroudTruthAugmentator permet d'augmenter le nombre de bounding boxes à afficher dans le nuage de points par rapport aux données récupérées. Il contient quatre paramètres qui consiste à déterminer si un véhicule, un piéton, un cycliste ou un autre objet va être labélisé ou non. Le range permet de dire que l'objet est soit labélisé soit non.

RandomFlip permet d'inverser tous les points du nuage selon l'axe Y. Son paramètre est la probabilité d'inverser ou non.

La transformation du WorldScalling met à l'échelle toutes les boxes et les points. Le paramètre que cette fonction prend est l'intervalle des valeurs que peut prendre l'échelle.

La RandomRotation permet de faire tourner toutes les boxes et tous les points de manière aléatoire selon un angle donné compris entre 0 et l'angle maximum donné en paramètre.

Le GlobalTranslateNoise permet de faire une translation de toutes les données selon les trois axes X, Y et Z. Elle prend en paramètre les trois valeurs de déviation de l'image pour les trois axes.

Ensuite, FrustrumDropout, tous les points sont convertis en coordonnées cylindriques, puis un point est selectionné au hasard. Tous les points autour de lui avec un angle, un phi, un thêta et une distance sont alors supprimés. La fonction prend l'angle thêta, l'angle phi, un rayon, la probabilité de supprimer un point ou non et le type de suppression.

FrustumNoise reprend le même schéma que FrustumDroupout mais au lieu de supprimer des points permet d'en ajouter aléatoirement grâce au bruit Gaussien.

Enfin, RandomDropout supprime aléatoirement des points, et prend en paramètre la probabilité d'en supprimer ou non.

b. Algorithme

Voici un algorithme de PPBA en langage algorithmique. Cet algorithme suit la métrique Ω pour un modèle θ. Un modèle θ est mesuré selon ses performances d'après une métrique Ω. Cette métrique est définie à l'avance par les ingénieurs avec une très grande précision car elle détermine si oui ou non la reconnaissance d'objet a été efficace. Un modèle est meilleur si ses paramètres d'opérations d'augmentations λ sont bons. Ce paramètre est défini de cette manière : $\lambda = (\lambda_t)_{t=1}^{T}$ avec t le nombre de l'itération actuelle pendant l'exécution de l'algorithme. Les paramètres sont nommés λ et sont donc définis à chaque itération donc « tour » de la boucle for. Avec cet algorithme on doit pouvoir parvenir à la meilleure valeur de λ, la valeur de λ pour laquelle Ω (θ) est à son maximum donc quand le modèle est le meilleur, soit : $\lambda^* = \arg\max_{\lambda \in \Lambda^T} \Omega(\theta)$. Pour faire simple, on cherche à connaître les paramètres ayant permis d'avoir le meilleur modèle évalué. A l'inverse, l'optimisation du modèle passe par une fonction objective différentiable. Une fonction qui suit l'algorithme du gradient stochastique, une méthode de descente itérative utilisée pour la minimalisation d'une fonction objective. L'optimisation du modèle se traduit par : $\theta^* = arg \min_{\theta \epsilon \Theta} L(X, Y, \lambda^t)$. X et Y étant les coordonnées des labels. L'entraînement est divisé en N itérations durant lesquelles chaque modèles θ avec des différents λ^t sont entrainés et évalué par Ω. A la première itération, tous les modèles et les paramètres sont initialisés de manière aléatoire. Les modèles sont placés dans des populations P. Après la première itération, les modèles sont sélectionnés dans la phase d'exploitation pour avoir été les meilleurs. La phase suivante est celle d'exploration où un sous-ensemble des opérations d'augmentation vont être exploré pour être optimisés en mutant les paramètres des modèles parents tandis que ceux restants seront hérités. La phase d'exploitation permet de garder les bons modèles et de remplacer les mauvais.

Voyons maintenant l'algorithme de PPBA en question afin de l'expliquer en détail.

Input: data and label pairs $(\mathcal{X}, \mathcal{Y})$
Search Space: $\mathcal{S} = \{op_i : params_i\}_{i=1}^{n}$
Set $t = 0$, $num\_ops = 2$, population $\mathcal{P} = \{\}$, best params and metrics for each operation $historical\_op\_params = \{\}$
while $t \neq \mathcal{N}$ **do**
  for θ_i^t in $\{\theta_1^t, \theta_2^t, ..., \theta_{\mathcal{M}}^t\}$ (asynchronously in parallel) **do**
    # Initialize models and augmentation parameters in current iteration

    if $t == 0$ **then**
      $op\_params_i^t$ = Random.sample($\mathcal{S}$, num_ops)
      Initialize θ_i^t, λ_i^t, $params$ of $op\_params_i^t$
      Update λ_i^t with $op\_params_i^t$
    else
      Initialize θ_i^t with the weights of $winner_i^{t-1}$
      Update λ_i^t with λ_i^{t-1} and $op\_params_i^t$
    end if
    # Train and evaluate models, and update the population
    Update θ_i^t according to formular (2)
    Compute metric $\Omega_i^t = \Omega(\theta_i^t)$
    Update $historical\_op\_params$ with $op\_params_i^t$ and Ω_i^t
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\theta_i^t\}$
    # Replace inferior augmentation parameters with better ones
    $winner_i^t \leftarrow$ Compete(θ_i^t, Random.sample($\mathcal{P}$))
    if $winner_i^t \neq \theta_i^t$ **then**
      $op\_params_i^{t+1} \leftarrow$ Mutate($winner_i^t$'s op_params, $historical\_op\_params$)
    else
      $op\_params_i^{t+1} \leftarrow op\_params_i^t$
    end if
  end for
  $t \leftarrow t + 1$
end while

Figure 54 : Algorithme de PPBA

Les entrées de l'algorithme sont les données et les coordonnées d'un label. On intègre ensuite l'espace de recherche. La liste de toutes les transformations et de leurs paramètres appelés respectivement « op » et « params » sont affectés à la variable S. On instancie ensuite les variables « t », « num_ops », une variable P qui contiendra une liste de tous les modèles présent dans une population, puis une variable « historical_op_params » qui servira à stocker les meilleurs paramètres lambda et meilleure métrique oméga pour chaque transformation.

On entre ensuite dans une boucle « tant que » qui continue pour un nombre N d'itérations donc de transformations. On entre ensuite dans une boucle « for » qui parcoure tous les modèles à chaque itération « t ».

La première étape consiste à initialiser les modèles avec les paramètres. Si c'est la première itération alors, on affecte à « op_params » un paramètre aléatoire grâce à la fonction sample du module Random qui permet de sélectionner un des paramètres de la deuxième transformation aléatoirement dans la liste S. Ensuite, on initialise les données de cette variable « op_params » puis on met à jour son paramètre lambda. Sinon, si ce n'est pas la première opération, alors on initialise et on met à jour les paramètres du modèle théta avec ceux du gagnant de l'itération précédent, c'est-à-dire celui avec les meilleures performances.

La deuxième phase de l'algorithme se présente de la manière suivante. On met à jour le modèle thêta avec la formule d'optimisation énoncée précédemment, puis on calcule ses performances avec la fonction oméga. On affecte « op_params » et la fonction oméga à la variable « historical_op_params » qui contient maintenant les données du modèle le plus performant de l'itération. Ensuite, on l'ajoute à la population P.

Vient ensuite la dernière phase où on recherche le modèle le plus performant de la population. On met en comparaison le modèle le plus performant de la population P « winner » avec un des modèles venant d'être créé, sélectionné aléatoirement. Puis, si le modèle n'est pas aussi bon que « winner », alors on remplace ses paramètres par ceux de winner pour la prochaine itération. Sinon on garde les mêmes paramètres pour le modèle suivant.

La variable t est incrémentée de 1 dont la boucle while passe au tour suivant. La boucle s'arrête quand t est égal au nombre de transformations N instancié au début.

Nous allons maintenant étudier un schéma simplifié de l'exécution du PPBA.

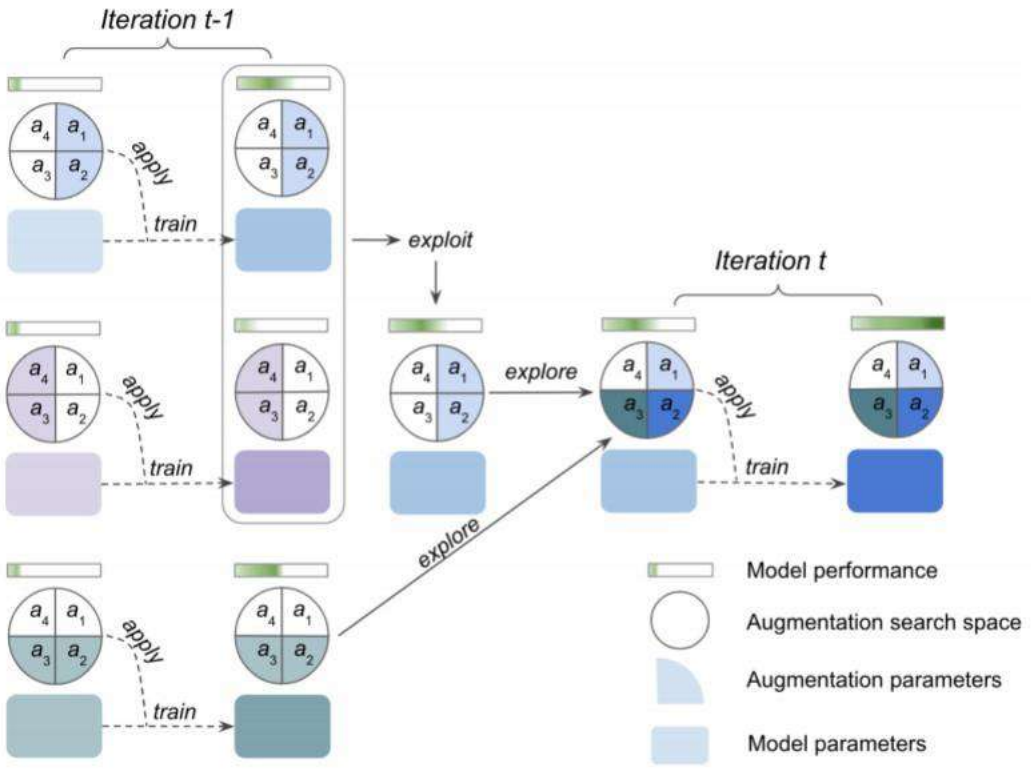


Figure 55 : Schéma d'un exemple de PPBA

On dispose ici de quatre paramètres d'augmentations a1, a2, a3 et a4. On entraîne ces paramètres deux par deux. Les trois modèles font partis d'une population. On remarque que les combinaisons (a1, a2) et (a3, a2) présentent des performances plus importantes que celles de la combinaison (a4, a3). Ces deux combinaisons ont alors passés la phase de l'exploitation et sont maintenant dans la phase d'exploration. On applique les trois paramètres gagnants (a1, a2, a3) à un modèle que l'on entraîne et on remarque que ce modèle a les performances les plus hautes possibles donc prend place en tant que « winner ». Ce modèle est fondé sur la loi de l'évolution avec la sélection de chromosomes présentant des caractères avantageux au fil des générations au sein d'une population.

On se rend compte que la Data Augmentation pour les nuages de points est une tâche très ardue. Chaque transformations contient plusieurs paramètres avec des intervalles optimaux inconnus. Afin de mieux traiter ces paramètres, seulement une faible proportion est modifiée, et les paramètres des itérations précédentes sont réutilisés. En effet, les meilleurs paramètres de chaque itération sont adoptés par les modèles de la génération suivante. Donc à chaque itération l'espace de recherche est diminué. Nous allons maintenant nous intéresser à la phase d'exploration. Voici son algorithme.

Algorithm 2 Exploration Based on Historical Data

Input: $op\_params = \{op_i : params_i\}_{i=1}^{num\_ops}$, best params and metric for each operation *historical_op_params*
Search Space: $\mathcal{S} = \{(op_i, params_i)\}_{i=1}^{n}$
Set $exploration\_rate = 0.8$, $selected\_ops = []$, $new\_op\_params = \{\}$
if Random(0, 1) < *exploration_rate* **then**
  selected_ops = *op_params*.Keys()
else
  selected_ops = Random.sample($\mathcal{S}$.Key(), *num_ops*)
end if
for i in Range(*num_ops*) **do**
  # Choose augmentation parameters, which successors will mutate
  # to generate new parameters
  if *selected_ops*[*i*] in *op_params*.Keys() **then**
    parent_params = *op_params*[*selected_ops*[*i*]]
  else if *selected_ops*[*i*] in *historical_op_params*.Keys() **then**
    parent_params = *historical_op_params*[*selected_ops*[*i*]]
  else
    Initialize *parent_params* randomly
  end if
  new_op_params[*selected_ops*[*i*]] = MutateParams(*parent_params*)
end for

Figure 57 : Algorithme d'amélioration des paramètres

On remarque que les entrées sont les variable « op_params » contenant toutes les transformations avec leurs paramètres similaires à « S » dans l'algorithme précédent, et les paramètres « historical_op_params ». On affecte à S la liste de toutes les opérations, on crée deux variables « selected_ops » et « new_op_params » destiné respectivement à accueillir l'opération sélectionnée et les nouveaux paramètres de la transformation. Puis, on a la variable « exploration_rate instanciée à la valeur 0,8. Enfin, une boucle if permet de sélectionner un nombre aléatoire entre 0 et 1 et de le comparer à « exploration_rate ». Si sa valeur est inférieure à « exploration_rate », alors les paramètres sélectionnés (donc ceux de l'itération en cours) sont affectés à une variable « selected_ops ». Sinon, cette variable prend des valeurs aléatoires de paramètres. La boucle if se termine.

On entre ensuite dans une boucle for qui permet la succession des paramètres pour l'itération suivante. Cette boucle se répète autant de fois que le nombre de transformations N. Une boucle if permet de distinguer trois cas de successions. Le premier permet de transmettre les paramètres du modèle parent aux successeurs. Le deuxième permet de transmettre les paramètres de

« historical_op_params » aux successeurs donc les meilleurs. Et enfin, le troisième permet de transmettre des paramètres aléatoires. On peut illustrer cet algorithme avec le schéma suivant.

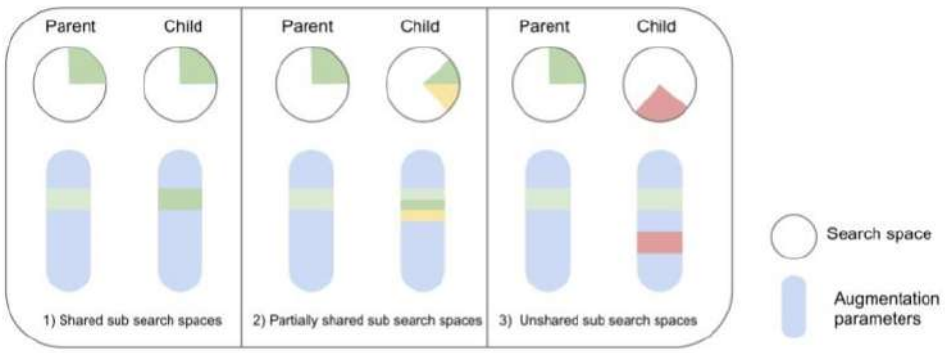


Figure 58 : Visualisation des trois cas de successions de paramètres

Encore une fois on retrouve une similarité avec les schémas de chromosomes. Dans le premier cas, les paramètres du parent sont transmis au modèle enfant. Dans le deuxième, les paramètres que l'enfant possède sont ceux des meilleurs modèles précédents. Enfin dans le dernier, on remarque une attribution aléatoire des paramètres.

En conclusion, nous pouvons dire que la PPBA est un algorithme qui s'inspire de la loi de l'évolution pour entraîner des modèles de DA afin d'améliorer la reconnaissance d'objets dans des nuages de points. Il fonctionne dans une boucle permettant, à chaque itération, d'appliquer une transformation sur le nuage de points, avec des paramètres précis, puis d'évaluer son efficacité sur le modèle. Cette technologie est de nos jours utilisées dans les ordinateurs des voitures de Waymo et dans bien d'autres applications. Elle contribue à rendre la conduite des véhicules plus précises et donc plus sécurisante pour les passagers et le monde extérieur.

Nous avons donc vu ici tout le fonctionnement de la Progressive Population Based Augmentation, nous allons maintenant nous intéresser à la simulation des voitures autonomes.

D. Simulation

Nous avons vu précédemment deux algorithmes qui permettaient d'augmenter les capacités de détection des voitures autonomes pour les lidar et pour les caméras. Nous allons maintenant nous intéresser à la simulation réelle et logicielle. A eux trois, ils forment l'ensemble qui permet à la voiture d'être meilleure. C'est-à-dire un entraînement rigoureux des logiciels pour les capteurs mais aussi un entraînement plus générale des comportements de la voiture dans des situations particulières. La voiture de Waymo est donc prête à rouler de façon sécurisée grâce à ces trois aspects. Nous allons dans un premier temps nous intéresser à la simulation irl (in real life) des véhicules, puis dans un second temps, nous traiterons de la simulation logicielle.

a. Le Castle

Nous savons que Waymo fait rouler ses véhicules sur des routes publiques dans plusieurs états des Etats Unis depuis plusieurs années maintenant, mais nous ne savons pas où-est-ce qu'ils sont entraînés. Quoi de mieux qu'une ville artificielle pour pouvoir simuler des situations réelles ? C'est l'idée que Waymo a eu il y a de ça plusieurs années lorsqu'ils ont réhabilité une ancienne zone militaire en ville artificielle. Cette ville s'appelle le Castle et est situé en Californie dans un endroit tenu à l'abris de regards, presque secret. Elle est formée d'avenues, de carrefours, d'un rocade et même de rond points.

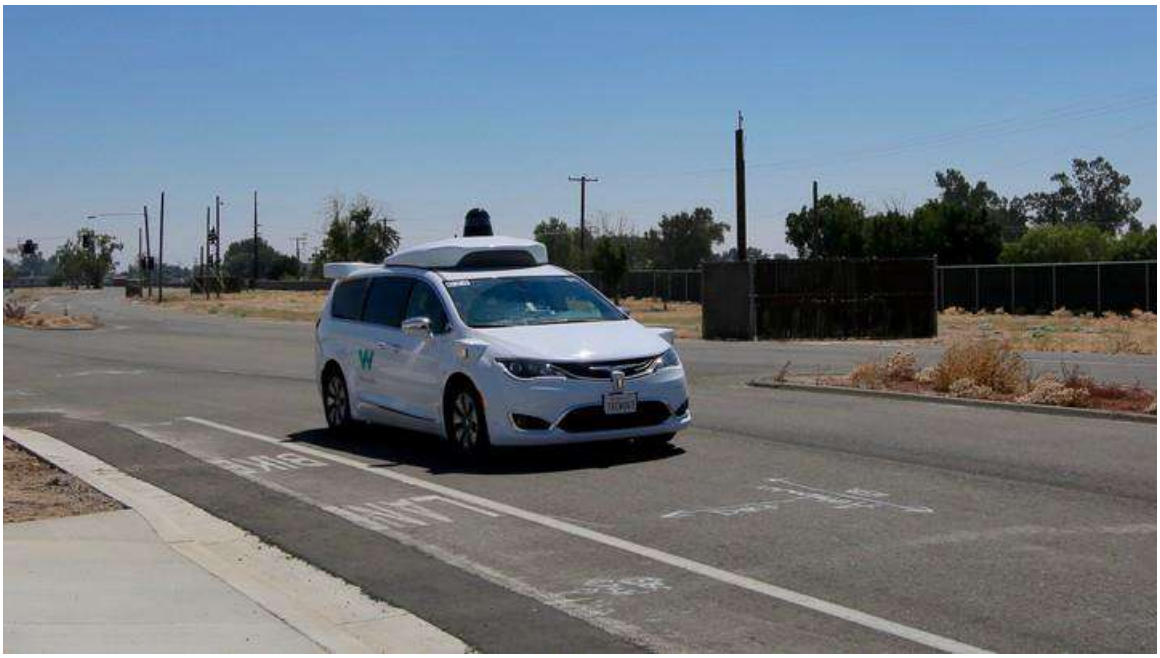


Figure 59 : Chrysler Pacifica sur une route du Castle

Cette ville est façonnable à la guise des équipes de Waymo. Ils peuvent modifier les routes et les obstacle de façon infinie afin de reproduire des scénarios destinés à augmenter les capacités de la voiture. Ils possèdent des centaines de véhicules allant du camion benne au simple vélo et surtout des accessoires comme des plots ou des faux piétons.

Figure 60 : Mise en position d'obstacles pour entraîner la voiture

Depuis sa création, Waymo à accumuler pas moins de 40000 scénarios de test complets. De plus chaque scénario amène à des dizaines d'autres variations grâce à un processus appelé le fuzzing que nous détaillerons tout de suite après. Les ingénieurs recréent des situations jamais vues sur les routes que leur véhicules ont traversées. Ce sont des situations pouvant être exceptionnelles et très délicates à gérer même pour l'homme. Grâce à ses tests, ils observent comment réagissent les véhicules et peuvent adapter leur comportement à travers des modifications logicielles et même matérielles parfois. Ils peuvent tester leur partie Hardware dans des conditions difficiles comme des tunnels de pluie, des chambres thermiques, des dos d'ânes et des pentes. Tout cela permet de compléter les tests en simulation sur ordinateur.

b. Fuzzing

Dans le Castle, il y a des bureaux où travaillent des dizaines d'ingénieurs. Certains s'intéressent à modéliser les scénarios sur des logiciels. Les logiciels utilisés sont Carcraft et XView. Le premier sert à modéliser un monde virtuel en 3D pour entraîner les voitures, le deuxième sert à avoir un retour sur toutes les données perçues par les capteurs. Ces mondes 3D sont directement tirés des données des capteurs. En effet ils modélisent le Castle en entier et des villes entières comme Phoenix afin que les voitures puissent s'entraîner dans des environnements presque réels.

Figure 61 : Plan de Phoenix en visualisation 3D dans CarCraft

L'avantage de ces logiciels est de pouvoir choisir un scénario précis et d'entraîner la voiture sur ce même scénario dans des centaines de variations différentes en un temps très court grâce à la possibilité de modeler l'environnement. Donc le fonctionnement de la simulation repose sur trois principes. Modélisation d'un environnement, création de centaines de variations d'une situation, et enfin application de la nouvelle compétence acquises à tous les véhicules. Nous allons maintenant nous intéresser au fuzzing.

Le fuzzing est une technique de test logiciel de Black Box. Le Black Box est une méthode de test de logiciel qui ne prend pas en compte la structure du code interne, mais que les entrées et les sorties.

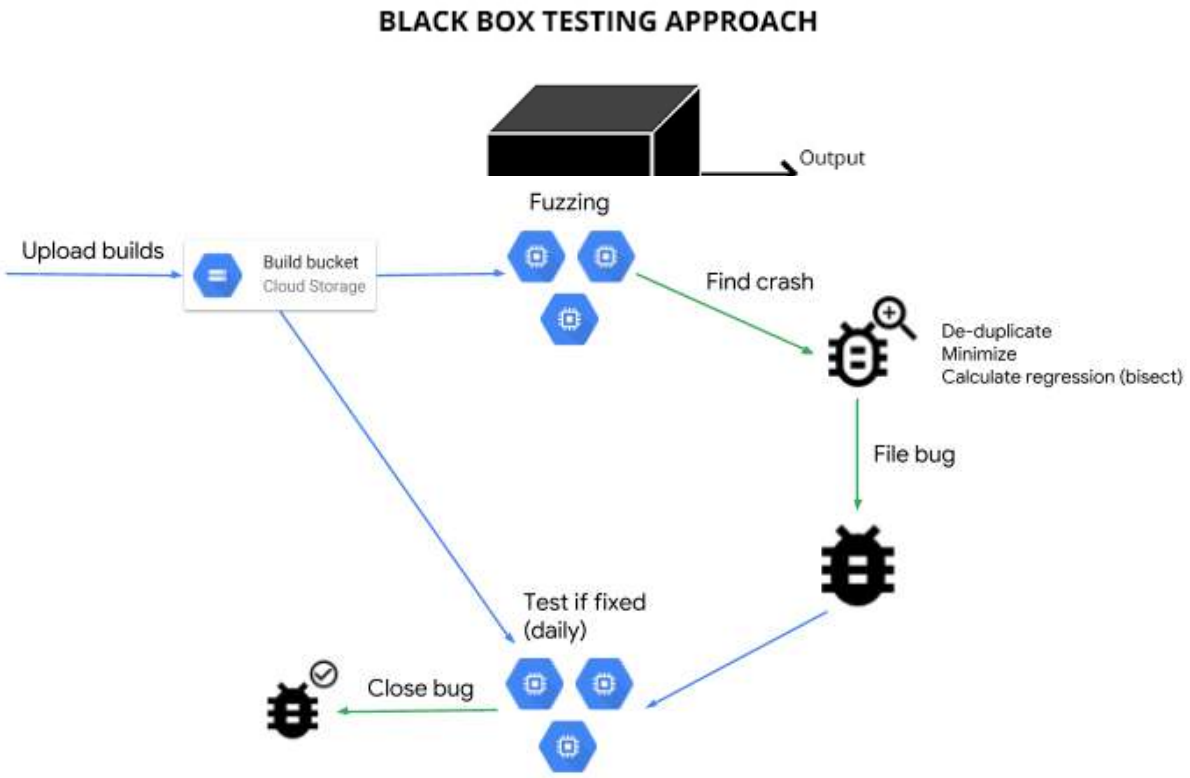


Figure 63 : Schéma de fuzzing de ClusterFuzz

Le fuzzing lui consiste à injecter des informations erronées dans un algorithme afin que de voir comment il réagit, pour trouver des erreurs. Si le logiciel retourne une erreur ou qu'il s'arrête à la suite d'une implémentation inhabituelle, alors c'est qu'il n'est pas préparé à toutes les situations possibles. Voici un schéma d'un processus standard de fuzzing.

Google a mis tout son projet sur le fuzzing en open source sur GitHub. Ce projet s'appelle ClusterFuzz. Il se compose de deux éléments principaux qui sont une instance sur App Engine, une plateforme de Google permettant d'héberger des applications web, et des bots de fuzzing. Le fait d'utiliser App Engine permet d'avoir une interface affichant toutes les données et les stat du fuzzing directement sur internet. Les bots sont des algorithmes qui permettent de réaliser des tâches bien précises de façon automatique. Je vais m'intéresser à quelques une des principales tâches mais je ne vais pas détailler leur code car chacune d'entre elles présentent plusieurs centaines de lignes de codes, ce qui rendrait l'explication peu pertinentes.

La première tâche est 'fuzz'. Elle consiste à démarrer une 'FuzzingSession', donc un fuzzing. Une FuzzingSession est initialisée par un nom, un type de fuzzing, des paramètres de fuzzing aléatoire et des variables plus générales comme l'endroit où envoyer les données. Un corpus est un fichier qui contient un ensemble d'input destiné à l'objet sur lequel le fuzzing est exécuté. Le corpus est donc synchronisé et ensuite l'algorithme est exécuté et retourne et sauvegarde toutes les données nécessaires.

La seconde tâche se nomme 'progression' et vérifie si un test case est réparé ou pas. Un testcase est une entrée pour le fuzzing qui provoque un bug dans une situation. On regarde si pour un testcase

donné sur un type de fuzzing donné, si une erreur apparaît, et sinon on trouve et on récupère les paramètres qui ont été modifiés pour que le testcase puisse être restauré.

La tâche suivante est la régression. Elle consiste à trouver le moment où un bug est apparu dans le fuzzing. Plus précisément elle analyse les 'commit' et retourne un intervalle du premier commit au dernier commit. Les commit sont ici des 'revision', des nombres qui permettent d'identifier un build spécifique dans un programme, similaire à un git hash.

La quatrième tâche s'appelle 'minimization' et consiste à réduire un testcase. En effet le but est de minimiser la taille d'un testcase tout en s'assurant qu'il produise les mêmes effets. Le but est d'éliminer toute sortes d'informations pouvant potentiellement être nuisible si non retirée.

L'avant dernière est le 'corpus_pruning'. Celle-ci permet de réduire la taille du corpus. Lorsqu'on lance un fuzzing, à la fin beaucoup de fichier sont inutiles et prennent de l'espace et de la mémoire pour rien. La seule manière de les éliminer est de réduire la taille du corpus à son minimum.

La dernière opération est l'analyse. On crée un testcase manuellement et on l'incorpore dans le fuzzing. On regarde ensuite si le système crash ou pas.

Maintenant que nous avons vu comment fonctionnait un fuzzing, nous allons en étudier un exemple. Nous allons étudier un carrefour avec 4 stops dans la ville de Phoenix. La ville ayant déjà été modélisée en situation de conduite réelle, on arrive à recréer le carrefour dans Carcraft. Le carrefour est d'abord vide, puis les ingénieurs peuvent y ajouter tous les obstacles possibles à leur guises. Des piétons, des cyclistes, mais aussi des véhicules. Les obstacles générés adoptent des comportements de circulation normaux.

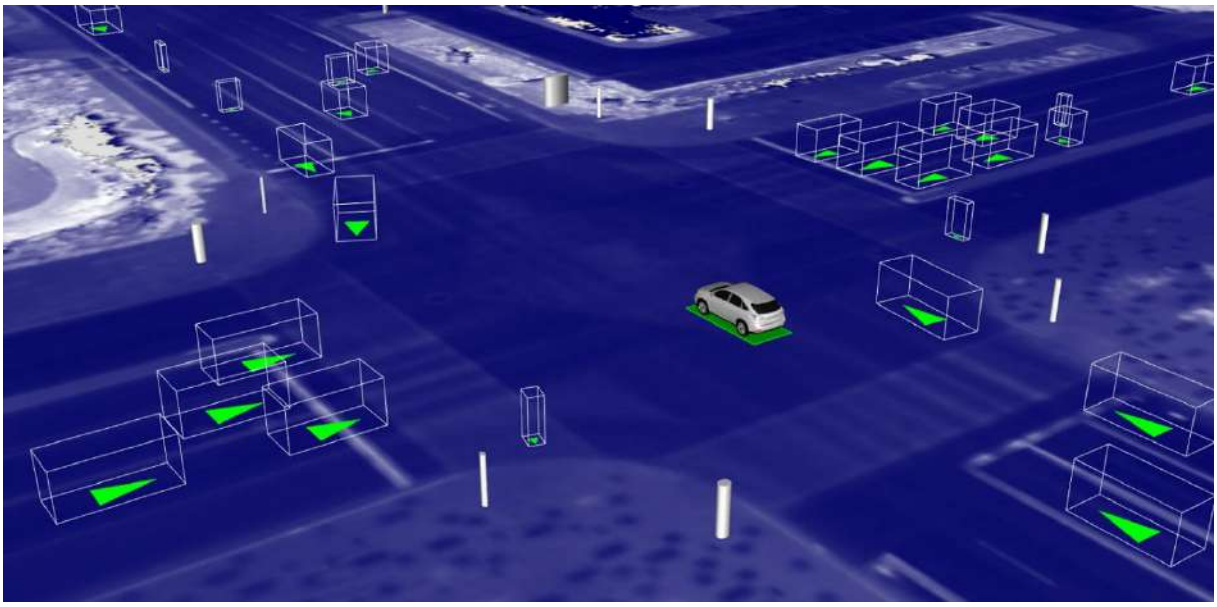


Figure 64 : Visualisation d'un carrefour sur CarCraft

Afin d'avoir des calculs moins lourds, ils ne demandent pas au véhicule de simulation de reconnaître les obstacles, ils indiquent directement au véhicule leurs coordonnées. Maintenant qu'ils ont le scénario avec leur voiture qui doit traverser un carrefour, ils doivent l'analyser et c'est là que le fuzzing rentre en jeu. En effet dans cet exemple ce n'est pas moins de 800 variations qui ont été explorées. Ils regroupent leur données dans un graphique particulier.

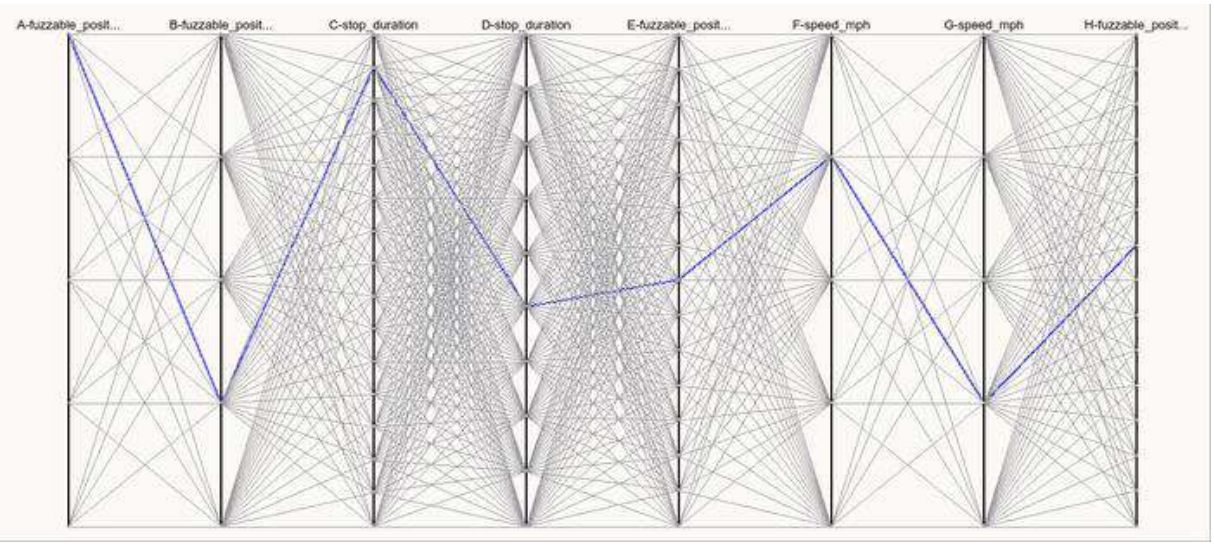


Figure 65 : Graphique en lacets d'un fuzzing

En effet comme nous pouvons le voir, chaque fil correspond à une situation donc à un chemin de la voiture. Il y a des variables qui définissent le chemin que la voiture a pris, et qui le caractérise. On remarque ici des positions précises, des temps d'arrêts ainsi que des vitesses. La suite du travail réside dans l'analyse de ce graphique. Les ingénieurs recherchent les situations où la voiture s'est arrêtée, où le temps de décision a été trop long ou alors quand le profil de freinage était anormal. Ici on a deux simulations en même temps, la première est représentée par la voiture blanche qui a passé ce carrefour de manière fluide et sans s'arrêter, la deuxième est représentée par la box derrière elle en pointillé qui elle s'est arrêtée.

Figure 66 : Simulation 3D d'une situation sur CarCraft

Ils analysent chacune d'entre elles et corrige ce qui n'allait pas pour rendre la voiture plus compétente. Ensuite, ils testent ce changement en situation réelle dans le Castle pour confirmer ou non si cela présente une amélioration. Enfin, cette amélioration est incorporée dans tous les ordinateurs des voitures de Waymo. C'est grâce à ce processus que la simulation est effectuée et que les voitures savent de plus en plus conduire de façon autonome et sécuritaire.

Nous avons fait ici l'étude de la simulation en réel et en virtuel, deux étapes qui contribuent à un entraînement efficace et obligatoire des voitures autonomes.

IV. Conclusion

Ce travail de recherche m'a permis d'étudier les composants matériels et logiciels des voitures autonomes, et surtout sur un modèle de voiture de la marque Waymo, dans le but d'acquérir des connaissances scientifiques sur le sujet.

Tout d'abord nous avons chercher à comprendre l'évolution de cette technologie à travers les âges, et nous nous sommes rendu compte de la modernité de celle-ci. Ensuite nous avons vu les six caractéristiques qui catégorisent l'ensemble des voitures autonomes dans le monde. Enfin nos nous sommes intéressés à la marque Waymo, à son histoire et à ses motivations.

Dans un second temps, nous nous sommes penchés sur l'aspect de la perception de l'environnement par la voiture autonome. Nous avons tout d'abord vu le rôle ainsi que la disposition et le fonctionnement de tous les capteurs présent sur la voiture, pour nous intéresser ensuite à leur utilisation et comment ils arrivent à créer des Label 2D et 3D.

En considérant le fait que je souhaite poursuivre mes études dans la branche Informatique de L'UTBM, et que ce travail s'inscrit dans un contexte d'informatique, il m'a semblé indispensable de considérer l'étude de profonde des logiciels des voitures autonomes. Cette étude s'inscrit dans le cadre de la troisième et dernière partie de cette étude concernant la prise de décision du véhicule. Tout d'abord, nous nous sommes intéressés au concept de Data Augmentation, puis nous avons deux technologies permettant d'aider à entraîner les réseaux de neurones de la voiture. Finalement, nous avons fini par des recherches concernant la simulation.

Depuis des décennies, un mythe sur les voitures se conduisant toutes seules est construit dans les mentalités de chaque société dans le monde, et ce mythe se rapproche rapidement de la réalité. En effet, grâce aux avancées technologiques qui croissent de manière exponentielle, il ne reste plus que quelques années avant que la plupart des constructeurs automobiles proposent leur premier véhicule 100% autonome. Bien sûr il faudra un certain temps avant que ces véhicules dominent le marché de la vente de voitures dans le monde, mais lorsque ce temps arrivera, de grandes améliorations verront le jour. En effet, on suppose que la mortalité routière va décroitre, que le temps en voiture sera optimisé au mieux, qu'il n'y ait plus d'embouteillages, que des métiers pénibles de routiers soit remplacés, et surtout que l'électrique prenne le dessus sur les énergies fossiles. Ce sont peut-être encore des mythes, mais dans l'attente de leur réalisation, il faut que les états du monde entier améliorent leur politique envers les tests de ces véhicules.

V. Références bibliographiques

Illustrations:

Figure 1:

« Where to? A History of Autonomous Vehicles ». 2014. CHM. 8 mai 2014. https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/.

Figure 2:

« Google Patent Reveals How Self-Driving Cars May Communicate ». s. d. Time. https://time.com/4129247/google-self-driving-cars-patent/.

Figure 3 :

« Elon Musk: "Tesla rijden volgend jaar volledig autonoom" ». 2019. Autovisie. 25 octobre 2019. https://www.autovisie.nl/nieuws/elon-musk-tesla-rijden-volgend-jaar-volledig-autonoom/.

Figure 4 :

Maxmen, Amy. 2018. « Self-Driving Car Dilemmas Reveal That Moral Choices Are Not Universal ». *Nature* 562 (7728): 469-70. https://doi.org/10.1038/d41586-018-07135-0.

Figure 5:

Maxmen, Amy. 2018. « Self-Driving Car Dilemmas Reveal That Moral Choices Are Not Universal ». *Nature* 562 (7728): 469-70. https://doi.org/10.1038/d41586-018-07135-0.

Figure 6 :

« Home – Waymo ». s. d. Consulté le 10 Octobre 2020. https://waymo.com/.

Figure 7:

Claudel, Maxime. 2018. « Google lance Waymo One, son service de taxis autonomes ». Numerama. 6 décembre 2018. https://www.numerama.com/vroom/445594-google-lance-waymo-one-son-service-de-taxis-autonomes.html.

Figure 8:

Millon, Louise. 2018. « Waymo lance son service commercial des véhicules autonomes aux USA ». *Presse-citron* (blog). 5 décembre 2018. https://www.presse-citron.net/waymo-voitures-autonomes-ca-y-est-sont-en-service-etats-unis/.

Figures 9,10,11,12:

Waymo.2020. *Designing the Waymo Driver*. https://www.youtube.com/watch?v=o8rCOKSDMcg&t=191s.

Figure 14:

« Lidar - Laser Bear Honeycomb ». s. d. Waymo. https://waymo.com/lidar/.

Figure 15 :

« Véhicules autonomes : Waymo commercialise son propre capteur LiDAR ». s. d. Génération-NT. https://www.generation-nt.com/waymo-lidar-laser-bear-honeycomb-vehicule-autonome-actualite-1962896.html

Figure 16:

« Waypoint - The official Waymo blog: Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments ». s. d. Waymo Blog. https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html.

Figure 17,18,19,20,21,22,23 :

« Waymo-Research/Waymo-Open-Dataset ». s. d. GitHub. https://github.com/waymo-research/waymo-open-dataset.

Figure 24 :

https://arxiv.org/pdf/1909.13719.pdf

Figures 25,27,29,31,34,36,38,41,44,46,48,50,52:

« tpu/autoaugment.py at master · tensorflow/tpu ». s. d. https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/autoaugment.py.

Figures 26,28,30,32,33,35,36,37,39,40,42,43,45,47,49:

« Waymo Secures $750 Million in its Latest Funding Round, Totaling $3 Billion Since March ». s. d. https://www.futurecar.com/article-3927-1.html.

Figure 51 :

« tpu/autoaugment.py at master · tensorflow/tpu ». s. d. https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/autoaugment.py.

Figures 53-58:

https://arxiv.org/pdf/2004.00831.pdf

Figure 59:

« Waymo Racks up 4 Million Self-Driven Miles ». s. d. *TechCrunch* (blog). 2020. https://social.techcrunch.com/2017/11/27/waymo-racks-up-4-million-self-driven-miles/.

Figure 60 :

Stevens, Tim. s. d. « See Waymo's Autonomous Pacifica Cruise through Castle ». Roadshow. https://www.cnet.com/roadshow/pictures/waymo-castle/.

Figures 61,64,65,66 :

Madrigal, Story by Alexis C. s. d. « Inside Waymo's Secret World for Training Self-Driving Cars ». *The Atlantic*. https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

Figure 62:

Resources, Test Automation. 2018. « 5 Software Testing Methods ». Medium. 31 janvier 2018. https://medium.com/@testautomation/5-software-testing-methods-3a45a9426fdd.

Figure 63 :

« fuzz+pic.png (690×349) ». s. d. https://1.bp.blogspot.com/--Z0-izz5-vk/XFyXWwdn_PI/AAAAAAAAMmc/RonWQGzxzFsPLb1o6qmsuPGyvZ9nStRewCLcBGAs/s1600/fuzz%2Bpic.png.

Sources

[1] « Why we're pursuing full autonomy - YouTube ». https://www.youtube.com/watch?v=6ePWBBrWSzo&feature=youtu.be.

[2] ClusterFuzz. « Architecture ». /clusterfuzz/architecture/.

[3] Motor Authority. « Audi Gives up on Level 3 Autonomous Driver-Assist System in A8 ». https://www.motorauthority.com/news/1127984_audi-gives-up-on-level-3-autonomous-driver-assist-system-in-a8.

[4] « Automated Driving Systems A Vision for Safety.pdf ». https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf.

[5] « Calibrage géométrique d'une caméra ou d'un capteur de vision stéréoscopique - Calibrage ». http://www.optique-ingenieur.org/fr/cours/OPI_fr_M04_C01/co/Contenu08.html.

[6] CEA. « La voiture autonome ». Articles & Dossiers:L'essentiel sur. CEA/Découvrir & Comprendre. CEA, 22 novembre 2017. https://www.cea.fr/comprendre/Pages/nouvelles-technologies/essentiel-sur-voiture-autonome.aspx.

[7] Cheng, Shuyang, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, et al. « Improving 3D Object Detection through Progressive Population Based Augmentation ». *arXiv:2004.00831 [cs]*, 16 juillet 2020. http://arxiv.org/abs/2004.00831.

[8] Nouveaux Medias. « Comment apprend une intelligence artificielle ? », 11 juin 2020. https://nouveauxmedias.fr/comment-apprend-une-intelligence-artificielle/.

[9] Google Cloud. « Créer un système de recherche de similarités des représentations vectorielles continues en temps réel ». https://cloud.google.com/solutions/machine-learning/building-real-time-embeddings-similarity-matching-system?hl=fr.

[10] Cubuk, Ekin D., Barret Zoph, Jonathon Shlens, et Quoc V. Le. « RandAugment: Practical automated data augmentation with a reduced search space ». *arXiv:1909.13719 [cs]*, 13 novembre 2019. http://arxiv.org/abs/1909.13719.

[11] « Data – Waymo ». https://waymo.com/open/data/.

[12] Data Analytics Post. « Des capteurs multiples, divers et complémentaires », 7 février 2020. https://dataanalyticspost.com/sur-les-voitures-autonomes-des-capteurs-multiples-divers-et-complementaires/.

[13] « Fuzzing | OWASP ». https://owasp.org/www-community/Fuzzing.

[14] ClusterFuzz. « Glossary ». /clusterfuzz/reference/glossary/.

[15] Han, Jia Cheng, et Zhi Quan Zhou. « Metamorphic Fuzz Testing of Autonomous Vehicles ». In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 380-85. Seoul Republic of Korea: ACM, 2020. https://doi.org/10.1145/3387940.3392252.

[16] ———. « Metamorphic Fuzz Testing of Autonomous Vehicles ». In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 380-85. Seoul Republic of Korea: ACM, 2020. https://doi.org/10.1145/3387940.3392252.

[17] Waymo. « Home ». https://waymo.com/.

[18] « How autonomous vehicles could save over 350K lives in the US and millions worldwide | ZDNet ». https://www.zdnet.com/article/how-autonomous-vehicles-could-save-over-350k-lives-in-the-us-and-millions-worldwide/.

[19] « How Do Radars Work? | Earth Observing Laboratory ». https://www.eol.ucar.edu/content/how-do-radars-work.

[20] Google AI Blog. « Improving Deep Learning Performance with AutoAugment ». http://ai.googleblog.com/2018/06/improving-deep-learning-performance.html.

[21] « Inside Waymo's Secret World for Training Self-Driving Cars - The Atlantic ». https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.

[22] Wayve. « Learning to Drive in a Day ». https://wayve.ailearning-to-drive-in-a-day-with-reinforcement-learning.

[23] TensorFlow. « Module: tf | TensorFlow Core v2.3.1 ». https://www.tensorflow.org/api_docs/python/tf?hl=fr.

[24] « numpy-tensor-slicing slides ». https://lisaong.github.io/mldds-courseware/01_GettingStarted/numpy-tensor-slicing.slides.html.

[25] « Prosecutors find Uber not criminally liable in 2018 Arizona self-driving crash that killed a pedestrian | TechCrunch ».https://techcrunch.com/2019/03/05/prosecutors-find-uber-not-criminally-liable-in-2018-arizona-self-driving-crash-that-killed-a-pedestrian/?guccounter=1.

[26] GeeksforGeeks. « Python PIL | ImageOps.Equalize() Method », 27 juin 2019. https://www.geeksforgeeks.org/python-pil-imageops-equalize-method/.

[27] « Python PIL | ImageOps.equalize() method - GeeksforGeeks ». https://www.geeksforgeeks.org/python-pil-imageops-equalize-method/.

[28] « Regulating Autonomous Vehicles ». https://www.ncsl.org/research/transportation/regulating-autonomous-vehicles.aspx.

[29] « Reinforcement Learning ». In *Wikipedia*, 5 décembre 2020. https://en.wikipedia.org/w/index.php?title=Reinforcement_learning&oldid=992544107.

[30] « Salon de Francfort : Waymo explique pourquoi il a abandonné la conduite semi-autonome ». https://www.futura-sciences.com/tech/actualites/voiture-electrique-salon-francfort-waymo-explique-il-abandonne-conduite-semi-autonome-77583/.

[31] Schmelzer, Ron. « What Happens When Self-Driving Cars Kill People? » Forbes. https://www.forbes.com/sites/cognitiveworld/2019/09/26/what-happens-with-self-driving-cars-kill-people/.

[32] GeeksforGeeks. « Search Algorithms in AI », 14 janvier 2019. https://www.geeksforgeeks.org/search-algorithms-in-ai/.

[33] « Self-driving car dilemmas reveal that moral choices are not universal ». https://www.nature.com/articles/d41586-018-07135-0.

[34] « Sun et al. - 2020 - Scalability in Perception for Autonomous Driving .pdf ». https://openaccess.thecvf.com/content_CVPR_2020/papers/Sun_Scalability_in_Perception_for_Autonomous_Driving_Waymo_Open_Dataset_CVPR_2020_paper.pdf.

[35] Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, et al. « Scalability in Perception for Autonomous Driving: Waymo Open Dataset ». In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2443-51. Seattle, WA, USA: IEEE, 2020. https://doi.org/10.1109/CVPR42600.2020.00252.

[36] F-Secure Blog. « Super Awesome Fuzzing, Part One », 22 juin 2017. https://blog.f-secure.com/super-awesome-fuzzing-part-one/.

[37] GitHub. « Tensorflow/Tpu ». https://github.com/tensorflow/tpu.

[38] « The 6 Levels of Vehicle Autonomy Explained | Synopsys Automotive ». https://www.synopsys.com/automotive/autonomous-driving-levels.html.

[39] « The Only Way is Ethics for Self-Driving Cars | Automotive IQ ». https://www.automotive-iq.com/autonomous-drive/columns/the-only-way-is-ethics-for-self-driving-cars.

[40] « Voiture autonome : Waymo reste le leader incontesté ». https://www.assurland.com/assurance-blog/assurance-auto-actualite/essais-de-vehicules-autonomes-waymo-loin-devant-apple-bon-dernier_132492.html.

[41] « Voitures autonomes : 55 millions de ventes d'ici 2040 ? » https://www.bfmtv.com/economie/entreprises/transports/voitures-autonomes-55-millions-de-ventes-d-ici-2040_AN-201907090216.html.

[42] GitHub. « Waymo-Research/Waymo-Open-Dataset ». https://github.com/waymo-research/waymo-open-dataset.

[43] « Waypoint - The official Waymo blog: How Evolutionary Selection Can Train More Capable Self-Driving Cars ».https://blog.waymo.com/2019/08/how-evolutionary-selection-can-train.html.

[44] Waymo Blog. « Waypoint - The official Waymo blog: Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments ». https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html.

[45] « Waypoint - The official Waymo blog: The Waymo Driver's training regimen: How structured testing prepares our self-driving technology for the real world ». https://blog.waymo.com/2020/09/the-waymo-drivers-training-regime.html.

[46] « Waypoint - The official Waymo blog: Using automated data augmentation to advance our Waymo Driver ». https://blog.waymo.com/2020/04/using-automated-data-augmentation-to.html.

[47] Waymo Blog. « Waypoint - The official Waymo blog: Using automated data augmentation to advance our Waymo Driver ». https://blog.waymo.com/2020/04/using-automated-data-augmentation-to.html.

[48] « What is BLACK Box Testing? Techniques, Example & Types ». https://www.guru99.com/black-box-testing.html.

[49] GitHub. « Google/Clusterfuzz ». https://github.com/google/clusterfuzz.