Détection et suivi temps-réel d'objets 3D pour la smart mobilité routière et ferroviaire

Alexandre Evain, Redouane Khemmar, Antoine Mauri, François Garnier, Messmer Messmer, Madjid Haddad, Rémi Boutteau, Sébastien Breteche, Sofiane Ahmedali

Détection et suivi temps-réel d'objets 3D pour la smart mobilité routière et ferroviaire

Alexandre Evain[1], Antoine Mauri[1], François Garnier[1], Messmer Kounouho[1], Redouane Khemmar[1,*]
Madjid Haddad[3], Rémi Boutteau[2], Sébastien Breteche[3], Sofiane Ahmedali[4]

[1] Normandie Univ, UNIROUEN, ESIGELEC, IRSEEM, 76000 Rouen, France
[2] Normandie Univ, UNIROUEN, UNILEHAVRE, INSA Rouen, LITIS, 76000 Rouen, France
[3] SEGULA Technologies, 19 rue d'Arras, 92000 Nanterre, France ;
[4] IBISC-EA4526, Université d'Evry Val d'Essonne Paris-Saclay, France ;

alexandre.evain@groupe-esigelec.org ; redouane.khemmar@esigelec.fr ;
remi.boutteau@univ-rouen.fr ; sofiane.ahmedali@univ-evry.fr
madjid.haddad@segula.fr ; sebastien.breteche@segula.fr ;

Résumé

La détection et le suivi d'objets tridimensionnels (3D) temps-réel est une tâche importante dans le cas des véhicules autonomes et de la mobilité intelligente routière et ferroviaire, afin de leur permettre d'analyser leur environnement à des fins de navigation et d'évitement d'obstacles. Dans cet article, nous essayons d'améliorer l'efficacité de la détection d'objets 3D monoculaire en utilisant la combinaison de jeux de données, la distillation de connaissances et la création d'un modèle léger. Tout d'abord, nous combinons des ensembles de données réelles et synthétiques pour augmenter la diversité et la richesse des données d'entraînement. Ensuite, nous utilisons la distillation des connaissances pour transférer les connaissances d'un grand modèle pré-entraîné vers un modèle plus petit et léger. Enfin, nous avons créé un modèle léger en sélectionnant les combinaisons de largeur, de profondeur et de résolution afin d'atteindre une complexité et un temps de calcul cibles. Nos expériences montrent que l'utilisation de chaque méthode améliore soit la précision, soit l'efficacité de notre modèle, sans inconvénient majeur. L'utilisation de toutes ces approches est particulièrement utile pour les environnements à ressources limitées, tels que les voitures à conduite autonome et les systèmes ferroviaires.

Mots Clef

Détection d'objets 3D ; distillation de connaissances ; estimation des boîtes englobantes 3D ; localisation d'objets ; estimation de la distance ; détection multi-objets 3D ; apprentissage profond ; mobilité intelligente.

Abstract

Three-dimensional (3D) real-time object detection and tracking is an important task in the case of autonomous vehicles and road and railway smart mobility in order to allow them to analyze their environment for navigation and obstacle avoidance purposes. In this paper, we try to improve the efficiency of 3D monocular object detection by using dataset combination, knowledge distillation, and creating a lightweight model. Firstly, we combine real and synthetic datasets to increase the diversity and richness of the training data. Then, we use knowledge distillation to transfer the knowledge from a large, pre-trained model to a smaller, lightweight model. Finally, we created a lightweight model by selecting the combinations of width, depth and resolution in order to reach a target complexity and computation time. Our experiments show that using each method improves either the accuracy or the efficiency of our model with no significant drawbacks. Using all these approaches is especially useful for resource-constrained environments, such as self-driving cars and railway systems.

Keywords

3D Object Detection ; Knowledge Distillation ; 3D Bounding Boxes Estimation ; Object Localization ; Distance Estimation ; 3D Multi-Object Detection ; Deep Learning ; Smart Mobility.

1 Introduction

Les véhicules autonomes sont de plus en plus présents dans notre vie quotidienne, ouvrant de nouvelles perspectives en matière de mobilité et de transport. Ces véhicules évoluent dans un environnement dynamique partagé par de nombreux autres utilisateurs. Les travaux de notre équipe sont directement liés à l'une des problématiques majeures de ce domaine, la perception de l'environnement pour les véhicules autonomes (voitures autonomoes, tramways autonomes et trains autonomes). Dans le contexte des applications de mobilité intelligente, la détection d'objets et

l'estimation précise de la profondeur sont nécessaires pour une navigation sûre. Une bonne perception de l'environnement nécessite de comprendre les points suivants :

— Cartographie et localisation : il s'agit de l'établissement de relations spatiales entre le véhicule et les objets statiques environnants.
— Détection et suivi des objets en mouvement
— Classification des objets (voiture, bus, piéton, cycliste, etc.) : Les véhicules doivent être capables de détecter précisément chaque classe différente, car chaque classe a son propre comportement, et l'identification correcte de chaque objet est une clé de la compréhension de l'environnement.

Dans le travail de notre équipe, nous nous concentrons sur la détection, le suivi et la classification des objets, qui sont cruciaux pour la compréhension de l'environnement. Les algorithmes d'apprentissage profond sont utilisés pour analyser les images et identifier les objets qu'elles contiennent. Les deux principales méthodes sont les algorithmes de classification et de localisation et la segmentation d'instance. Les algorithmes de classification et de localisation dessinent des boîtes de délimitation autour de chaque objet détecté, tandis que la segmentation d'instance attribue chaque pixel à une classe unique.

Bien que les deux méthodes aient leurs avantages, dans le contexte des voitures et des trains autonomes, la détection et la classification d'objet des objets est la méthode la plus pertinente, ce qui compte est d'identifier les obstacles, leur types, leur localisation et leurs dimensions pour estimer leur comportement. A l'inverse, la segmentation d'image est plus consommatrice en resource alors que les informations additionnelles fournies ne sont pas utiles pour estimer le comportement du véhicles. Pour cette raison, nous allons donc utiliser la détection et classification d'objets dans nos travaux, en particulier la détection et classification tridimensionnelle.

En plus de cela, des défis supplémentaires ont été identifiés. Le premier est la disponibilité de bases de données dans le domaine ferroviaire/routier, suffisamment riches (à la fois en termes de nombre d'images et de données de vérité terrain) pour permettre l'entraînement adéquat des méthodes d'apprentissage profond. Alors qu'il existe de nombreux ensembles de données couvrant le domaine routier, les ensembles de données couvrant le domaine ferroviaire pour la détection d'objets avec vérité terrain sont beaucoup moins nombreux.

Le deuxième défi identifié concerne la conception d'une solution de détection, de localisation et de prédiction de trajectoire temps-réel dans un environnement complexe et varié, compatible avec des ressources de traitement embarquées limitées. Les solutions d'optimisation doivent combiner les performances de détection sans altérer la vitesse du modèle d'inférence.

Notre objectif est d'apporter une réponse à ces deux défis en exploitant pleinement la combinaison de jeux de données, et en développant et améliorant un système de détection et de suivi d'objets 3D monoculaire capable de fournir des performances temps-réel, même sur des systèmes embarqués disposant de ressources de traitement limitées.

2 État de l'art

2.1 Méthodes de détection d'objets

Il existe de nombreux algorithmes de détection d'objets avec leurs propres approches uniques, parmi lesquels [35, 17] sont quelques-unes des méthodes de détection d'objets 3D monoculaires les plus récentes, tandis que [38, 37] sont les approches de l'état de l'art les plus performantes en 2021 dans ce domaine. Parmi ces approches de détection d'objets 3D monoculaires, [24] et [25] sont des méthodes spécifiquement destinées à atteindre des capacités temps-réel élevées comme les nôtres. Cependant, la première a été testée sur deux GPU RTX 1080 tandis que la seconde a utilisé deux GPU NVIDIA Tesla V100. Cela diffère de notre approche, qui utilise principalement des systèmes embarqués comme le NVIDIA Jetson TX2 pour ses évaluations de performance temps-réel comme dans le tableau 5.

YOLO [30] propose une méthode de détection d'objet en utilisant un seul réseau convolutif pour prédire la position des objets (région) et leur classe tout en gardant de bonnes performances de détection. C'est pour ses performances temps-réel que nous avons choisi de nous concentrer sur YOLO plutôt que sur d'autres méthodes de détection d'objets à plusieurs étapes. Les différentes versions de YOLO [31, 3, 20, 7] publiées depuis ont apporté de nombreuses améliorations à la version originale, à la fois en temps de calcul et en précision.

YOLO divise l'image d'entrée en un ensemble de grilles plus ou moins fines. Il applique ensuite des convolutions et prédit si un objet est situé dans une zone de la grille. Cependant, c'est l'agglomération de ces "bounding boxes" qui va nous permettre d'effectuer la détection finale de l'objet en appliquant une fonction NMS (Non-Maximal Suppression). C'est ce découpage sous forme de grille qui va permettre à YOLO de détecter aussi bien les petits que les grands objets sur l'image tout en gagnant en rapidité par rapport aux autres méthodes de détection. La version de YOLO que nous avons utilisée [27] était une méthode de détection 3D monoculaire construite sur YOLOv5 [20], qui remplace les bounding boxes originales de YOLOv5 (contenant des informations de classe ainsi que des informations de localisation dans l'image) par des bounding boxes hybrides, qui contiennent plus d'informations : distance de la caméra, centres 3D projetés sur le plan de l'image ainsi que dimensions et orientation 3D. L'ensemble de ces informations nous permet de localiser précisément la position et l'orientation d'un objet dans un environnement 3D. Nous avons apporté d'autres améliorations à cette méthode, qui vise à fournir une détection d'objets 3D monoculaire, l'accent étant mis sur les performances temps-réel.

En outre, pendant ou peu avant nos travaux, d'autres

versions de YOLO ont été publiées, telles que YOLOR,YOLOX,YOLOv6,YOLOv7. Parmi celles-ci, la plus intéressante est [6] qui offre des gains en temps et en performance sans aucun inconvénient. Bien qu'il n'ait pas été possible de l'utiliser au cours de notre travail, nous avons l'intention de porter notre adaptation de YOLOv5 à YOLOv7 afin de bénéficier des améliorations considérables de cette dernière.

2.2 Datasets

Le volume et la qualité des données d'un problème d'apprentissage restent une question fondamentale, de même que la structure du modèle à entraîner. Le domaine des voitures autonomes étant en plein développement, il dispose d'une multitude de jeux de données pour le domaine routier, comme KITTI [11] qui fournit des images provenant d'une caméra stéréoscopique, la profondeur de la scène mesurée par un LIDAR Velodyne ainsi que des annotations de véhicules et de piétons pour la détection d'objets, ou NuScenes [5] qui dispose de 6 caméras et d'un LIDAR avec un grand volume de données pouvant être utilisées pour la détection et le suivi 2D et 3D. Il existe également d'autres jeux de données disponibles comme CityScapes [26], Pascal VOC [9], ImageNet [19] ou MS-COCO [33], mais ils n'ont pas été utilisés lors de notre travail car ils ne contenaient pas les informations dont nous avions besoin. Nous utilisons KITTI et NuScenes principalement parce que ce sont deux jeux de données open-source dédiés à la détection 3D pour les véhicules autonomes et qu'ils contiennent tous deux des séquences de vidéos provenant d'environnements urbains.

En outre, il existe des simulateurs qui permettent l'acquisition et l'annotation automatiques, comme dans CARLA [8] ou SYNTHIA [12]. Cependant, le rendu graphique des simulateurs n'est pas toujours à la hauteur de la réalité, en raison de la qualité du rendu, des problèmes de texture, des graphismes datés et des décalages de domaine, et n'est pas assez photoréaliste pour nos méthodes une fois appliquées en conditions réelles. Une acquisition à partir de Grand Theft Auto V [32] (GTA V), un jeu vidéo à l'apparence photoréaliste, nous a permis d'obtenir un grand ensemble de données virtuelles pour la détection 3D pouvant être combiné avec nos ensembles de données précédents. Il existe plusieurs articles expliquant comment acquérir les données de ce jeu spécifique, comme celui de Richter [32] qui comprend des instructions et le code disponible à utiliser, et celui de Philipp Krähenbühl [23], un outil permettant d'extraire le code de rendu d'un jeu vidéo.

Notre équipe a également développé un jeu de données hybride rail-route réel-synthétique appelé ESRORAD [21]. Pour développer le jeu de données hybride route-rail, nous avons utilisé une combinaison de données synthétiques générées à l'aide de la GTA, ainsi que des enregistrements d'images réelles. Nous avons utilisé la GTA pour générer un grand nombre d'images de routes et de voies ferrées, ainsi que les étiquettes correspondantes et les données de vérité terrain. Pour compléter les données synthétiques par des exemples réels, nous avons également collecté un ensemble d'images réelles de routes et de voies ferrées en utilisant une voiture équipée de capteurs et en roulant sur une voie ferrée existante, ce qui a donné lieu à un ensemble de données hybride comprenant à la fois des données synthétiques et des données réelles.

Pour le domaine ferroviaire, le nombre de jeux de données disponibles est beaucoup plus faible : nous n'avons pour l'instant identifié que deux jeux de données, RailSem19 [29], et FRSign [15]. Railsem19 offre 8500 images pour la segmentation sémantique, FRSign a 100 000 images pour la reconnaissance des feux de signalisation ferroviaire, mais ces deux jeux de données ne sont pas pertinents dans le cas de la détection d'objets 3D. Le jeu de données ESRORAD est à ce jour le seul jeu de données offrant des données routières et ferroviaires librement disponibles avec des vérités de terrain. Bien que le thème de nos travaux est le domaine routier et ferroviaire, dans le cadre de cet article l'algorithme n'a été évalué que sur des datasets routiers afin de pouvoir le comparer avec les autres méthodes de l'état de l'art, nous comptons néanmoins le tester sur le dataset ESRORAD afin d'évaluer ses performances dans le milieur routier et ferroviaire très prochainement.

2.3 Tracking d'objets

Dans le contexte des voitures autonomes, connaître la position dans l'espace des objets qui entourent le véhicule n'est pas suffisant, car ils sont généralement en mouvement. Pour cette raison, nous utilisons le suivi d'objets pour anticiper leur comportement dynamique et prédire les situations dangereuses. Le suivi d'objets nous permet de faire l'association de détection, c'est-à-dire de reconnaître un objet déterminé entre deux images et de le suivre dans le temps. Ces outils utilisent des séquences d'images et font donc une extraction à la fois spatiale et temporelle (distance, vitesse, texture, etc.). Bien que très utiles, ces méthodes présentent des limites :

— L'occlusion et la troncature font que l'algorithme oublie l'objet et dès qu'il revient dans l'image l'algorithme le considère comme un nouvel objet.
— Les mouvements rapides de la caméra ou du point de vue peuvent amener l'algorithme à prédire des trajectoires erronées, car il attribue ces changements importants et soudains dans le cadre aux mouvements des objets.

Il existe plusieurs méthodes de MOT (Multiple-Object Tracking), certaines basées sur l'utilisation de filtres de Kalman, comme SORT [2] (Simple Online and Realtime Tracking), d'autres utilisant des CNN (Convolutional Neural Networks) comme DeepSORT [36] ou Trackformer [28]. Les solutions de suivi utilisant le Deep Learning effectuent souvent à la fois la détection 2D et le suivi des objets, ce qui les rend partiellement redondantes avec notre algorithme de détection d'objets déjà existant. Les performances temps-réel étant une question importante pour

nous, nous ne pouvons pas nous permettre d'effectuer une détection d'objet 2D en parallèle de notre détection d'objet 3D principale.

2.4 Distillation de connaissances

La distillation de connaissances telle que définie dans [16] est une méthode d'optimisation des réseaux de neurones. Elle consiste à entraîner deux modèles en parallèle : un réseau professeur et un réseau élève. Le réseau professeur est un réseau déjà entraîné avec des couches larges et complexes, lui permettant d'obtenir des scores de précision élevés (en mAP, précision de position, et précision de dimensions). Le réseau élève est un réseau léger, avec des couches plus fines et moins profondes, ce qui lui permet d'avoir une vitesse d'exécution plus élevée par rapport au modèle de l'enseignant. Un article répertoriant les méthodes de distillation de connaissances [13] décrit les trois principaux types de schémas de distillation (distillation hors ligne, distillation en ligne et auto-distillation), ainsi que certains des derniers algorithmes de distillation, tels que [1] et [18]. Dans [14], les expériences ont montré que la distillation permettait de filtrer les variables d'entrée non pertinentes, d'améliorer la séparabilité des états et d'accroître la robustesse des modèles face aux variations de l'environnement.

3 Améliorations de la détection et suivi d'objets 3D

3.1 Détection d'objets 3D basée YOLOv5

Pour créer un algorithme de détection d'objets 3D à partir de YOLOv5, nous avons modifié l'architecture du réseau afin d'ajouter des sorties supplémentaires pour les informations 3D. Cela a impliqué l'ajout de nouvelles couches au réseau, capables de prédire le centre 3D de l'objet, ainsi que ses dimensions et son orientation, sur la base de la boîte de délimitation 2D prédite par YOLOv5. Nous avons également ajouté les fonctions de perte et les métriques d'entrainement afin de prendre en compte ces nouvelles données lors de l'entrainement. Ensuite, nous avons entraîné le réseau YOLOv5-3D [27] sur un ensemble de données d'images avec des étiquettes 3D correspondantes.

L'un des avantages de notre algorithme de détection 3D est qu'il possède une architecture à une seule étape et un nombre limité de couches, ce qui le rend très efficace et lui permet d'atteindre de bonnes performances temps-réel. Notre algorithme peut être exécuté sur un large éventail de dispositifs, y compris les systèmes embarqués à faible capacité de calcul tels que le Jetson TX2. Dans le domaine des véhicules autonomes, il est essentiel que les algorithmes utilisés pour la détection et la localisation d'objets puissent fonctionner temps-réel, car les décisions prises par ces algorithmes peuvent avoir des conséquences importantes, comme éviter les collisions avec d'autres objets par exemple. Le tableau 1 montre l'évaluation quantitative de YOLOv5 sur le jeu de données KITTI.

TABLE 1 – Évaluation quantitative de YOLOv5-3D sur KITTI après entraînement sur différentes bases de données

Training database	Training Resolution	Inference Resolution	2D Detection AP	2D Detection R	2D Detection mAP@0.5	Distance RE	Distance SRE	Distance RMSE	Distance log RMSE	Distance α_1	Distance α_2	Distance α_3	Dimension DS	Centre CS	Orientation OS
KITTI	672x244	672x244	0.897	0.752	0.732	0.0483	0.0987	1.82	0.0733	0.989	0.998	0.999	0.883	0.959	0.906
	672x244	1312x416	0.835	0.821	0.779	0.178	1.15	6.61	0.218	0.6	0.997	0.999	0.866	0.949	0.928
GTA	1312x768	1312x416	0.839	0.603	0.584	0.140	0.784	5.21	0.157	0.846	0.999	1.0	0.851	0.967	0.955
	1312x768	1312x768	0.823	0.593	0.571	0.114	0.584	4.63	0.134	0.913	1.0	1.0	0.848	0.962	0.955
NuScenes	1312x768	1312x416	0.857	0.678	0.697	0.455	5.97	14.7	0.378	0.0766	0.945	0.999	0.651	0.965	0.977
	1312x768	1312x768	0.843	0.642	0.635	0.0654	0.198	2.72	0.0776	0.994	1.0	1.0	0.654	0.964	0.986

L'une des principales limites de YOLOv5-3D est qu'il est extrêmement dépendant de la résolution des images d'entraînement. Si les images utilisées pendant l'inférence ont une résolution différente de celle des images utilisées pendant l'entraînement, cela affectera négativement la précision des différentes prédictions, en particulier la prédiction de la distance, comme on peut le voir dans le tableau 1. Dans ce tableau, nous pouvons voir que la RE (Relative Error) et la RMSE (Root Mean Square Error) sont toutes deux plus élevées après un changement de résolution entre l'entraînement et l'inférence.

En outre, la précision de notre algorithme dépend également des différences de paramètres intrinsèques de la caméra entre l'apprentissage et l'inférence. Ces paramètres comprennent la distance focale de la caméra, la taille des pixels sur le capteur de la caméra et d'autres facteurs qui affectent la façon dont la caméra capture les images. Si les paramètres intrinsèques de la caméra utilisés pour l'inférence sont différents de ceux utilisés lors de l'entrainement, la précision de l'algorithme peut être réduite.

Malgré ces limitations de YOLOv5-3D, il est possible de les atténuer facilement en modifiant les dimensions des images d'entrée, à la fois par le redimensionnement et le recadrage, ainsi qu'en utilisant la fonction undistort d'OpenCV [4] qui reprojette une image d'une matrice de caméra à une autre. Plutôt que d'essayer d'ajuster les images d'inférence aux résolutions et aux matrices de caméra des images d'entraînement, la meilleure solution consiste à prétraiter les données d'entraînement. Une fois que le modèle est entraîné avec des données prétraitées correspondant déjà à la résolution et à la matrice de la caméra de l'inférence, l'inférence fonctionnera comme prévu sans nécessiter de prétraitement supplémentaire. Il est donc nécessaire de connaitre à l'avance sur quelle caméra l'on compte faire fonctionner l'algorithme afin de pré-traiter les données d'entrainements pour que leurs dimensions et matrices correspondent à celles de la caméra d'inférence.

Afin d'améliorer notre modèle, nous avons décidé de créer une version légère destinée à être utilisée dans des systèmes embarqués tels que le Jetson TX2. L'objectif de ce modèle est d'améliorer les performances temps-réel et de s'exécuter rapidement, même sur des systèmes aux ressources limitées. Nous utilisons le Jetson TX2 comme référence pendant notre développement, car il s'agit d'un système embarqué puissant qui est couramment utilisé dans une variété d'applications. En concevant le modèle pour qu'il fonctionne bien sur le Jetson TX2, nous pouvons nous assurer qu'il fonctionnera bien sur un large éventail de systèmes embarqués avec des spécifications similaires, plutôt

que d'être limité aux seuls systèmes haut de gamme.

Un réseau qui a trop de paramètres aura un temps de calcul beaucoup plus élevé tout en ayant une plus grande tendance à se surentraîner, ce qui dégrade considérablement sa précision. A l'inverse, un modèle qui n'a pas assez de paramètres n'apprendra pas bien et aura une précision moindre. Dans notre réseau, le nombre de paramètres définissant sa complexité dépend largement du nombre de couches et de filtres. La vitesse et la précision de notre approche dépendront également de la résolution des images d'entrée. Un travail a déjà été effectué pour obtenir un ensemble optimal (profondeur, largeur, résolution de l'image) dans EfficientNet [22]. Les auteurs ont déterminé que la complexité du modèle, mesurée en Flops, était liée à la largeur (w), la profondeur (d), et la résolution de l'image (r) comme décrit dans l'équation 1 :

$$Flops \sim w^2 \times r^2 \times d \tag{1}$$

Afin de créer un modèle encore plus léger, nous avons choisi d'adopter une approche similaire. Tout d'abord, nous avons défini la complexité de notre modèle avec l'objectif d'obtenir un temps de calcul de $33ms/image$ sur le Jetson TX2. La complexité choisie est d'environ 10 GFlops. Nous sélectionnons ensuite toutes les combinaisons (largeur, profondeur, résolution) qui nous permettent d'obtenir cette même complexité. Enfin, nous entraînons chacun de ces modèles pendant 20 époques sur GTA. Si nous définissons $w = 1$ et $d = 1$ comme la largeur et la hauteur du modèle "Large" et $r = 1$ pour définir une image de résolution 1280×720, le modèle optimal que nous avons trouvé a la combinaison ($w = 0.4$, $d = 0.5$, $r = \frac{576}{1280} = 0.45$). Nous avons ensuite adopté la même procédure pour entraîner les autres modèles (Large, Medium, etc.).

3.2 Suivi d'objets et visualisation

Dans le contexte des voitures autonomes, connaître la position dans l'espace des objets entourant le véhicule n'est pas suffisant, car les autres véhicules sont généralement en mouvement et les collisions doivent être évitées. Par conséquent, afin d'anticiper leur comportement et leurs trajectoires pour éviter les situations dangereuses, nous avons utilisé des solutions de suivi d'objets multiples : nous avons intégré et modifié l'algorithme SORT [2] à cette fin : SORT ne fonctionne que pour la détection 2D, et nous avons modifié son algorithme de Kalman pour passer de la 2D à la 3D. Les résultats sont visibles sur la Figure 1.

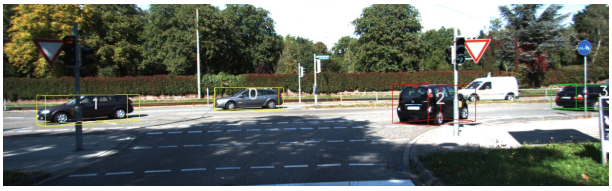

FIGURE 1 – Suivi d'objets avec boîtes de délimitation 3D et numéros de suivi

De plus, afin d'avoir une meilleure visibilité sur les trajectoires des objets environnants, nous avons réalisé un outil de visualisation vu en Fig.2 qui permet d'observer facilement la localisation et la trajectoire des objets suivis.

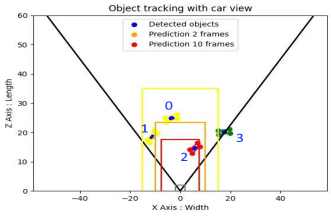

FIGURE 2 – Visualisation du suivi des objets en 2D et des trajectoires prédites des objets

Une fois les véhicules représentés sur le plan 2D, nous avons ajouté l'estimation de leur trajectoire obtenue par notre version modifiée de SORT afin de visualiser s'il existe un risque de collision entre les véhicules et notre propre voiture. Nous avons ensuite modélisé trois zones (jaune, orange et rouge) pour modéliser différents niveaux de danger en fonction de la proximité de l'obstacle avec OpenCV. La couleur des boîtes englobantes sera associée à la zone de danger pour plus de clarté.

3.3 Combinaison de jeux de données

La combinaison et l'expansion de l'ensemble de données d'entraînement ont un double objectif : tout d'abord, augmenter la quantité de données d'entraînement signifie augmenter la diversité des situations représentées, ce qui permet au modèle de s'adapter à ces nouvelles situations représentées. En outre, l'augmentation des données d'entraînement permet d'éviter le sur-apprentissage, et l'entraînement provenant de différents ensembles de données présente l'avantage supplémentaire d'atténuer les biais liés aux ensembles de données, tels que les conditions d'éclairage spécifiques ou la surreprésentation des scènes.

Les différences entre les données fournies par les jeux de données KITTI et GTA V sont présentées dans le tableau 2. Comme nous pouvons le constater, GTAV nous fournit environ 10 fois le nombre d'images fournies par KITTI. Comme la majorité des données annotées proviennent de notre jeu de données personnalisé GTA V, nous avons ajusté les données de KITTI pour suivre le formatage des données de GTA V, en les redimensionnant et en les recadrant, à l'aide de la bibliothèque OpenCV [4].

TABLE 2 – Répartition des données entre les datasets KITTI et GTA

	Real data	**Synthetic data**	**Combined data**
Dataset	KITTI	GTA V	KITTI +GTA V
N° of Pictures	11,193	110,271	121,464
Resolution	1224x370	1280x720	1280x720
N° of classes	3 (cars, pedestrians, cyclists)	3 (cars, pedestrians, cyclists)	3 (cars, pedestrians, cyclists)

Comme l'ensemble de données combiné d'origine présentait une grande disproportion entre les images synthé-

tiques et réelles, et entre l'ensemble de données d'entraînement et l'ensemble de données de test, nous avons créé un autre ensemble de données combiné avec cette fois-ci seulement 50% de données synthétiques, afin de voir si la disproportion affecte négativement ou non la précision du modèle entraîné. Les nouvelles combinaisons de jeux de données sont présentées dans le Tableau 3. Tous les ensembles de données seront utilisés pour entraîner la version étendue de l'algorithme YOLOv5-3D [27]. Pour des résultats supplémentaires, nous comparons également le jeu de données synthétique-réel 50% avec le jeu de données KITTI lorsque les deux utilisent un pré-entraînement de 15 époques sur KITTI. Quel que soit le jeu de données, pour pouvoir les comparer efficacement, nous devons utiliser le même dataset de validation, qui est extrait de KITTI.

TABLE 3 – Comparaison entre le dataset KITTI et des datasets combinant à la fois KITTI et des données synthétiques

	Training split	Validation split (KITTI)	Hyper-parameters	Model/Pre-training
Combo	114688	3769	GTA V	Large
Combo 50% synthetic	7424 synthetic + 7424 real	3769	GTA V	Large/last_15
KITTI (reference)	7424	3769	KITTI	Large
KITTI	7424	3769	KITTI	Large/large_15

4 Intégration et validation

Afin d'effectuer des tests sur le terrain et de voir comment les algorithmes se comportent en conditions réelles, l'étape suivante du travail effectué a été l'intégration de l'algorithme Yolov5 modifié sur un véhicule routier, dans ce cas, une Citroën AMI (figure 4). Nous avons utilisé le véhicule AMI, équipé d'une Jetson TX2 et d'une caméra RGB.

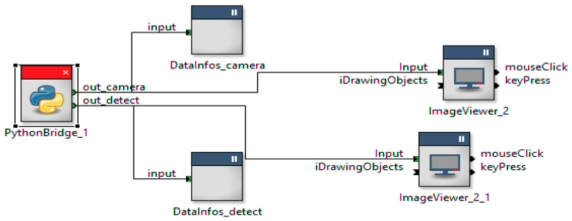

FIGURE 3 – Schéma RTMaps. Nous avons utilisé la bibliothèque "Python Bridge" de RTMaps afin d'intégrer l'algorithme Yolov5 sur le Jetson TX2.

Pour utiliser le code sur le Jetson, nous avons utilisé RTMaps (Figure 3), un environnement d'exécution permettant aux utilisateurs d'enregistrer et de rejouer des données provenant de capteurs de véhicules et de bus.

Une fois toutes les modifications et installations sur le véhicule routier effectuées, nous sommes passés aux tests, dans les zones urbaines de Rouen. Nous avons choisi une trajectoire qui nous permettait de rencontrer un maximum de véhicules, de personnes et de vélos et qui nous permettait également de rouler jusqu'à 40km/h ; ce qui nous a permis de tester l'algorithme dans des conditions réelles de travail, comme le montre la figure 4.

FIGURE 4 – L'algorithme testé en conditions réelles sur une Jetson embarqué

5 Résultats expérimentaux

5.1 Combinaison de Datasets

Pour voir l'effet de la combinaison des jeux de données, nous avons entraîné notre modèle sur les différents modèles combinés et de référence décrits dans le Tableau 3. Les résultats sont visibles dans la Figure 5.

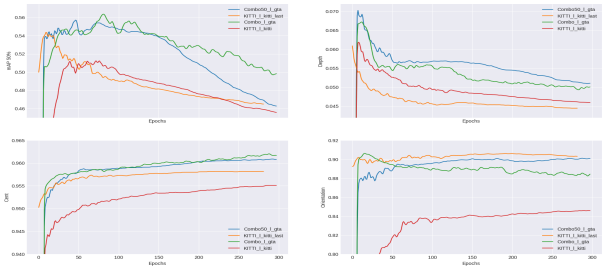

FIGURE 5 – Comparaison des résultats d'inférence à l'aide des modèles précédents

Ici, nous pouvons voir en vert et en bleu les performances du modèle entraîné sur des modèles combinés (combinaison synthétique-réel à 50% avec pré-entraînement en bleu et combinaison normale en vert), tandis qu'en orange et rouge sont les résultats avec la base de données KITTI originale (avec pré-entraînement en orange, sans pré-entraînement en rouge). Nous comparons ces modèles sur 4 métriques : mAP (mean Average Precision), la précision de la profondeur, la précision de la position du centre et enfin la précision de l'orientation. Les modèles combinés permettent d'améliorer les performances en matière de mAP, de précision et de position centrale, seule l'orientation n'a pas connu d'amélioration, c'est le seul cas où le modèle entraîné sur le jeu de données KITTI avec pré-entraînement a obtenu des performances légèrement supérieures à celles des modèles entraînés sur les jeux de données combinés. Néanmoins, étant donné que la combinaison a permis d'améliorer la mAP, la position du centre et la précision de la profondeur, nous pouvons conclure que l'utilisation de données synthétiques en plus des données réelles améliore les performances, malgré la nature artificielle de ces données.

5.2 Distillation de connaissances

La distillation des connaissances est une méthode d'optimisation, comme nous l'avons vu précédemment. Nous avons parallélisé deux réseaux, l'un en inférence, l'autre en apprentissage, et nous avons ajouté un nouveau type de fonction de perte au réseau étudiant à partir des prédictions faites par le réseau enseignant. Cela permet au réseau étudiant d'apprendre à partir des vérités de base et de la perte du réseau enseignant. Nous avons ensuite procédé à l'apprentissage du réseau étudiant à partir du réseau enseignant.

TABLE 4 – Comparaison entre un modèle distillé et un modèle régulier, sur le jeu de données KITTI, sur un GPU RTX 3080

Method	GTA pre-training	Resolution	[val] IOU 0.7			[val] IOU 0.5			Time /img (ms)	Memory Consumption
			Easy	Mod	Hard	Easy	Mod	Hard		
Small		672x224	7.29	5.48	5.34	44.35	31.80	29.91	1.5	1.6 GB
		1312x416	15.52	13.27	13.19	48.24	36.14	31.78	4.4	1.7 GB
	X	1312x416	15.59	13.50	13.14	49.71	38.02	32.70	4.4	1.6 GB
Small (with distillation)	X	672x224	16.60	13.49	12.01	53.62	35.97	33.32	1.5	1.7 GB

Comme on peut le voir dans le Tableau 4, notre modèle de distillation n'a pas conduit à une augmentation des performances temps-réel, mais il a néanmoins permis d'augmenter la précision : comme on peut le voir sur la Figure 6, le modèle avec distillation (en bleu) a obtenu un mAP plus élevé et une erreur relative plus faible, tout en conservant les mêmes performances temps-réel (Tableau 4). Il s'agit d'un compromis intéressant, car il nous permet d'améliorer la précision de notre modèle sans sacrifier la vitesse.

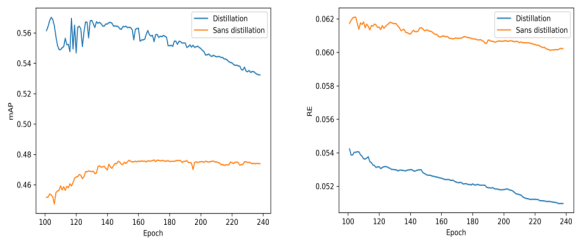

FIGURE 6 – Comparaison de l'entraînement sur KITTI entre un modèle distillé et un modèle classique.

5.3 Détection d'objets 3D : Modèle Lightweight

Nous avons d'abord entraîné YOLOv5 avec les mêmes dimensions que ce nouveau modèle sur le jeu de données COCO pendant 300 époques. Les poids pré-entraînés des premières couches sont ensuite transférés à notre modèle pour 15 époques d'entraînement sur GTA. *In fine*, nous entraînons ce modèle sur les jeux de données KITTI et NuScenes. Ce nouveau modèle correspond au meilleur compromis entre faible temps de calcul et précision, comme le montre le tableau 5.

Database (Model)	Resolution	2D Detection			Distance						Dimensions	Centre	Orientation	# Parameters	time/img
		AP	R	mAP@0.5	RE	SRE	RMSE	log RMSE	α_1	α_2	α_3	DS	CS	OS		
KITTI (Small)	672x224	0.722	0.520	0.500	0.069	0.163	2.27	0.094	0.978	0.997	0.999	0.863	0.942	0.879	7.3M	70ms
KITTI (LW)	608x192	0.642	0.518	0.469	0.056	0.133	2.15	0.081	0.985	0.998	0.999	0.871	0.953	0.863	6.0M	28ms
NuScenes (LW)	608x352	0.527	0.431	0.374	0.054	0.205	3.11	0.074	0.985	1	1	0.838	0.962	0.920	6.0M	28ms

TABLE 5 – Résultats quantitatifs de notre nouveau modèle lightweight sur une Jetson TX2. Les modèles testés ont été entraînés sur KITTI et NuScenes.

Les résultats de la Table 5 sont particulièrement prometteurs car ils nous permettent d'atteindre un temps d'inférence de 28ms sur le Jetson TX2 (correspondant à un frame-rate de 35 Frame-Per-Second). Par rapport au modèle léger, nous sommes passés d'un temps d'inférence de 70 ms à 28 ms (une amélioration relative de 60%), tout en ne passant que d'une valeur de mAP@0.5 de 0,500 à 0,469 (une diminution de 6,2%)

6 Conclusion

En conclusion, nous avons amélioré la détection d'objets en 3D en utilisant des techniques telles que la distillation des connaissances, la combinaison des ensembles de données et la création de modèles légers. Ces techniques ont permis d'améliorer les performances et la précision sans modifier les algorithmes eux-mêmes. En combinant plusieurs ensembles de données, le modèle a été exposé à un ensemble de données plus large et plus diversifié, ce qui a conduit à de meilleures performances. L'utilisation de la distillation des connaissances a permis de transférer les connaissances d'un modèle pré-entraîné plus important à un modèle plus petit et plus efficace. Dans l'ensemble, ce travail a démontré l'efficacité de ces techniques pour améliorer les performances des modèles de détection d'objets en 3D. Chaque approche, prise séparément, apporte des améliorations au modèle, que ce soit en termes de performances ou de précision, et les combiner et les utiliser ensemble est une étape essentielle vers le développement de systèmes de détection d'objets 3D plus efficaces et plus performants pour une variété d'applications.

Cependant, des améliorations sont encore possibles. Nous prévoyons d'implémenter la prédiction de l'orientation verticale dans notre modèle, de changer la méthode de suivi de SORT à une méthode plus moderne telle que DeepSORT [36] ou Trackformer [28], et d'utiliser davantage les méthodes d'apprentissage par transfert. L'ajout de la prédiction de l'orientation verticale aidera le modèle à mieux comprendre la structure 3D de la scène, ce qui conduira à des résultats plus robustes. L'utilisation d'une méthode de suivi moderne peut améliorer la précision, mais peut également avoir un impact négatif sur les performances temps-réel. Enfin, l'utilisation de méthodes d'apprentissage par transfert nous permettra d'améliorer encore les performances de notre modèle de détection d'objets en 3D.

Références

[1] U. Asif, J. Tang, and S. Harrer. Ensemble knowledge distillation for learning improved and efficient networks, 2019.

[2] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016.

[3] A. Bochkovskiy, C. Wang, and H. M. Liao. Yolov4 : Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[5] H. Caesar, V. Bankiti, et al. nuscenes : A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[6] W. Chien-Yao, B. Alexey, and L. H.-Y. Mark. Yolov7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[7] L. Chuyi, L. Lulu, J. Hongliang, et al. Yolov6 : A single-stage object detection framework for industrial applications, 2022.

[8] A. Dosovitskiy, G. Ros, et al. CARLA : An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[9] M. Everingham, L. Van Gool, et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2) :303–338, June 2010.

[10] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX : exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving ? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] R. German, S. Laura, M. Joanna, et al. The synthia dataset : A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation : A survey. *International Journal of Computer Vision*, 2021.

[14] V. Guillet. *Distillation de réseaux de neurones pour la généralisation et le transfert en apprentissage par renforcement*. PhD thesis, 2022. Thèse de doctorat dirigée par Aguilar Melchor, Carlos et Rachelson, Emmanuel Informatique et Télécommunications Toulouse, ISAE 2022.

[15] J. Harb, N. Rébéna, R. Chosidow, et al. Frsign : A large-scale traffic light dataset for autonomous trains. *CoRR*, abs/2002.05665, 2020.

[16] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. 2015.

[17] K.-C. Huang, T.-H. Wu, H.-T. Su, and W. H. Hsu. Monodtr : Monocular 3d object detection with depth-aware transformer, 2022.

[18] M. S. Iman, M. Farajtaba, et al. Improved knowledge distillation via teacher assistant. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04), 2020.

[19] D. Jia, D. Wei, S. Richard, et al. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[20] G. Jocher, A. Chaurasia, A. Stoken, et al. ultralytics/yolov5 : v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations, 2022.

[21] R. Khemmar, A. Mauri, et al. Road and railway smart mobility : a high-definition ground truth hybrid dataset. *Sensors*, 22(10) :3922, 2022.

[22] B. Koonce. Efficientnet. In *Convolutional neural networks with swift for tensorflow*, pages 109–123. Springer, 2021.

[23] P. Krähenbühl. Free supervision from video games. In *CVPR*, 2018.

[24] P. Li, H. Zhao, P. Liu, and F. Cao. Rtm3d : Real-time monocular 3d detection from object keypoints for autonomous driving, 2020.

[25] Z. Liu, D. Zhou, F. Lu, J. Fang, and L. Zhang. Autoshape : Real-time shape-aware monocular 3d object detection, 2021.

[26] C. Marius, O. Mohamed, et al. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] A. Mauri, R. Khemmar, et al. Lightweight convolutional neural network for real-time 3d object detection in road and railway environments. *Journal of Real-Time Image Processing*, 19(3) :499–516, Jun 2022.

[28] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer. Trackformer : Multi-object tracking with transformers. *CoRR*, abs/2101.02702, 2021.

[29] Z. Oliver, M. Markus, et al. Railsem19 : A dataset for semantic rail scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[30] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once : Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[31] J. Redmon and A. Farhadi. Yolov3 : An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[32] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data : Ground truth from computer games. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[33] L. Tsung-Yi, M. Michael, B. Serge, et al. Microsoft coco : Common objects in context, 2014.

[34] C. Wang, I. Yeh, and H. M. Liao. You only learn one representation : Unified network for multiple tasks. *CoRR*, abs/2105.04206, 2021.

[35] T. Wang, J. Pang, and D. Lin. Monocular 3d object detection with depth from motion, 2022.

[36] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017.

[37] Y. Zhang, J. Lu, and J. Zhou. Objects are different : Flexible monocular 3d object detection, 2021.

[38] Y. Zhou, Y. He, H. Zhu, C. Wang, H. Li, and Q. Jiang. Monocular 3d object detection : An extrinsic parameter free approach, 2021.