Détection de véhicules fondée sur l'estimation de carte de probabilités

Ruddy Théodose[1,2], Dieumet Denis[1], Christophe Blanc[2], Thierry Chateau[2] and Paul Checchin[2]

[1] Sherpa Engineering, R&D Departement, F-92250 La Garenne-Colombes, France

[2] Institut Pascal, UMR 6602, Université Clermont Auvergne, CNRS, SIGMA Clermont, F-63000 Clermont-Ferrand, France

r.theodose@sherpa-eng.com

Résumé

La détection et la localisation des obstacles à partir d'un nuage de points LiDAR sont des sujets majeurs dans le domaine de la conduite autonome. Les méthodes qui deviennent de plus en plus précises opèrent toutes de bout en bout, c'est-à-dire en retournant directement les coordonnées des boîtes englobantes des objets d'intérêt qu'elles estiment correctes. Cette stratégie rend plus difficile le diagnostic en cas de défaillance. Nous présentons une approche simple qui produit d'une part une carte de probabilité intermédiaire permettant de sélectionner les régions d'intérêt pour un système a posteriori*, et de les visualiser lors de l'inférence. D'autre part, les véhicules correctement détectés sont représentés sous la forme d'ellipses. Un véhicule caché pourra ne pas être une ellipse complète, sa marque indiquera tout de même une présence. Grâce à cette représentation paramétrique des obstacles, une extraction directe des prédictions à partir de cette carte intermédiaire est possible. La simplicité architecturale de l'approche proposée, en comparaison des méthodes de l'état de l'art, ne l'empêche pas d'atteindre un haut niveau de performances sur le benchmark KITTI pour la détection d'obstacles en vue de dessus.*

Mots Clef

Apprentissage profond, LiDAR, détection d'objets.

Abstract

Detection and localization of obstacles from a LiDAR point cloud have become a major stake for autonomous driving. While methods become more and more precise, they operate in an end-to-end fashion, returning directly the final boxes whose probability is beyond some threshold. This strategy makes the diagnosis difficult in case of failure as it does not allow to identify which area of the scene caused the false prediction. The method introduced in this paper create first an intermediate probability map allowing the selection of a region of interest for a subsequent detection system and the visualization of the regions that got the focus of the algorithm. Moreover, the detected vehicles are represented as ellipses. A hidden vehicle may illustrate a truncated shape but the mark in the map still indicates a possible existence. Thanks to this parametric representation, extracting obstacles directly from the probability map is possible. With its simple architecture, the proposed algorithm reaches high levels of performance on the KITTI Object Dataset Benchmark.

Keywords

Deep learning, LiDAR, object detection.

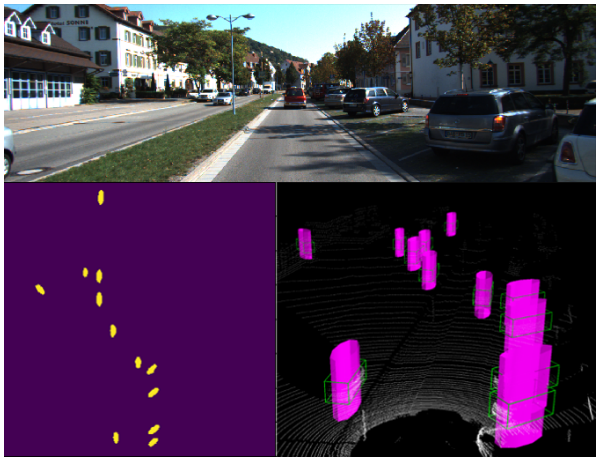


FIGURE 1 – Une carte de probabilité binarisée prédite en Bird's Eye View (bas gauche) est utilisée par un RPN (*Region Proposal Network*) pour prédire les ellipses finales (bas droite).

1 Introduction

Les réseaux de neurones artificiels ont permis de grandes avancées en vision par ordinateur telles que la segmentation sémantique, l'estimation de pose humaine ou la détection d'objets [21, 20, 13]. La dernière tâche est devenue l'une des plus critiques dans le développement d'agents autonomes en particulier des voitures. Le système de perception doit notamment analyser les environs de l'égo-véhicule, détecter et suivre les obstacles potentiellement mobiles qui pourraient interférer avec la trajectoire du véhicule avant de décider de la manœuvre à suivre. Ces dernières années, le capteur LiDAR (Light De-

tection And Ranging) est devenu le capteur de référence pour la détection d'obstacles, de par sa capacité à délivrer des nuages de points retranscrivant la géométrie et la disposition de la scène. Cependant, les méthodes de détection en 3D et en Bird's Eye View (BEV ou vue de dessus) peinent à atteindre les performances de leurs homologues sur les images RGB, bien que certaines illustrent des performances impressionnantes [26, 6, 10]. Ceci est notamment dû au fait que les données LiDAR, pour un même champ de vision donné, ont une résolution bien plus faible. Parmi les avancées majeures des réseaux de neurones, l'utilisation de convolutions a grandement amélioré les résultats obtenus en détection 2D. Or, l'une des difficultés du traitement des nuages de points tient en leur format de liste non ordonnée qui n'est pas compatible avec l'emploi de convolutions. Il est possible de diviser les méthodes de détection BEV/3D en deux catégories : les méthodes unimodales utilisant du LiDAR uniquement et les méthodes multimodales couplant les données de plusieurs capteurs, en général LiDAR et caméra RGB. Elles délivrent directement des résultats finaux sans détailler plus explicitement quelles régions en sont la cause. Les probabilités liées à chaque ancre peuvent servir d'indicateur mais celles dont la probabilité est en deçà d'un seuil sont directement rejetées. Comme aucun système n'est parfait, obtenir une visualisation globale de ce que l'algorithme vise ou de la manière dont il analyse l'espace peut être utile, d'une part pour estimer les zones dangereuses, d'autre part pour mieux comprendre les causes d'un possible dysfonctionnement.

La méthode présentée estime de manière explicite une carte de probabilités en BEV pour observer plus facilement les détections (voir Fig. 1). Cette carte peut être considérée comme une grille d'occupation pour une classe spécifique d'objets. Un réseau de segmentation extrait à partir d'une grille de données cette carte de probabilités sur laquelle chaque véhicule est représenté par une tache elliptique. Cette carte est passée en entrée à un *Region Proposal Network* (RPN) pour extraire les boîtes englobantes des voitures. Il est également possible d'extraire les boîtes directement depuis la carte de probabilités par extraction d'ellipses, permettant ainsi l'obtention d'une redondance sur les résultats.

Dans cette étude, les objets d'intérêt sont limités aux voitures (pas de camions ni de motos). Les travaux se focalisent sur la détection en Bird's Eye View (BEV) sur données LiDAR uniquement. Malgré son architecture simple, l'algorithme présenté illustre de très bonnes performances sur le benchmark de détection KITTI tout en ajoutant la caractéristique de pouvoir délivrer un aperçu de l'état courant de la scène, potentiellement utilisable par des algorithmes de planification pour mieux réagir près de zones incertaines.

La suite du papier est organisée ainsi : la section 2 présente les méthodes existantes de l'état de l'art pour la détection BEV et 3D, la section 3 décrit les différentes parties de la méthode proposée, leurs liens et la mise en forme des données. La section 4 détaille les expériences menées et les résultats obtenus.

2 État de l'art

Cette section présente les méthodes existantes consacrées à la détection 3D/BEV. Elles peuvent être divisées en 3 catégories : les méthodes traitant des données 2D, les méthodes utilisant les nuages de points et les méthodes multimodales. La solution proposée utilisant un réseau de segmentation sémantique, une partie sera également consacrée à cette tâche.

2.1 Détection sur données 2D RGB

Les détecteurs 2D sur images RGB ont atteint un certain niveau de maturité au point qu'ils sont actuellement utilisées dans l'industrie voire implémentées au sein de bibliothèques standards. Aussi des travaux existants se focalisent sur l'estimation de caractéristiques 3D à partir de vues de caméras RGB. Mono3D [5] attribue des scores à des boîtes 3D candidates et leur projection sur image en utilisant des caractéristiques telles que la forme, le contexte ou la classe d'objets afin de sélectionner les meilleures boîtes. Deep3DBox [17] suppose que la projection d'une boîte 3D s'ajuste à celle proposée par un détecteur 2D. Deep MANTA [4] utilise un entraînement multi-tâches pour la détection du véhicule, la localisation des parties du véhicule, la caractérisation de leurs visibilité et l'estimation des dimensions en utilisant comme *a priori* une base de données de modèles 3D de véhicules.

2.2 Détection à partir de nuages de points

Les nuages de points 3D, par nature non structurés, ne sont pas adaptés aux réseaux de neurones sous leur forme la plus simple. La plupart des méthodes transforment ces données en des structures organisées. En général, une étape dite de discrétisation ou de voxelisation est réalisée. Une concaténation sur l'axe vertical de la grille permet notamment d'obtenir des Bird's Eye View des données qui sont des projections au sol de la grille. Dans [6, 3], des cartes de hauteur issues des différentes tranches d'une grille de voxels ainsi qu'une carte de réflectance et une carte de densité sont concaténées afin de former une entrée multicanaux pour leurs réseaux respectifs. Dans [1], les auteurs étendent les travaux réalisés avec YOLO [20] pour inférer des boîtes englobantes 3D à partir d'une seule carte de hauteur et d'une carte d'intensité.

D'autres méthodes utilisent directement le formatage en grille de voxels. Dans [12], les auteurs étendent des mécanismes des détecteurs 2D en utilisant des convolutions 3D. Cependant, ces opérateurs sont denses et représentent des coûts calculatoires et en mémoire plus importants. Les auteurs de [7] améliorent les convolutions 3D en rajoutant un système de vote pour réaliser des opérations creuses. La plupart des approches définissent le contenu des cellules manuellement. Dans [26], un réseau préliminaire est appli-

qué aux cellules non vides pour apprendre des représentations optimales pour un problème spécifique avant d'utiliser ces représentations dans un RPN. A partir de ces travaux et partant du principe que la majorité des cellules est vide, les auteurs de [24] utilisent des opérateurs « creux » (*sparse convolutions*) [15, 9] pour réduire les coûts calculatoires. Les travaux de [11] étendent encore cette approche en encodant des colonnes entières au lieu de subdiviser l'axe vertical. Cela permet notamment de mieux représenter les objets ayant une forme verticale comme les piétons.

2.3 Détection par approche multimodale

La principale difficulté de l'utilisation conjointe d'un nuage de points et d'une image réside dans leurs différences de structure et de mode d'acquisition. Des méthodes telles que [6, 10] exploitent le fait qu'un nuage de points 3D peut être projeté afin de générer des images sous plusieurs points de vues. Les vues les plus utilisées sont les BEV, les projections cylindriques et les projections perspectives frontales (point de vue camera). Ces vues peuvent alors être traitées comme des images et les relations entre ces vues permettent d'estimer les caractéristiques 3D des détections. D'autres approches se focalisent sur l'utilisation de PointNet [19] pour la détection. Dans [18], un détecteur 2D RGB pré-entraîné produit des détections 2D et leurs frustums correspondant sur le nuage de points, puis un PointNet est appliqué au nuage de visible dans la détection pour segmenter les points et estimer la boîte 3D adéquate. Dans [23], les données RGB et 3D sont d'abord traitées séparément par un ResNet et un PointNet. Leurs résultats sont alors fusionnées par un réseau pour prédire les boîtes 3D correspondantes.

2.4 Segmentation Sémantique

La segmentation sémantique consiste en la classification des éléments d'une donnée, par exemple des pixels d'une image. L'un des points critiques de cette tâche est le respect des bordures et des régions. Une classification purement individuelle causerait trop de bruit dans les résultats. Des travaux ont été réalisés sur l'exploitation des relations entre les pixels. Les réseaux de segmentation tels que [2, 22, 16] fonctionnent sur un modèle encodeur/décoder. L'encodeur crée une représentation plus petite mais plus informative de la donnée d'entrée tandis que le décodeur doit reconstruire une donnée de même dimensions que l'entrée à partir de l'encodage. Afin de conserver l'information géométrique, des connexions sont réalisées entre les couches de chaque partie (voir Fig. 2). Des réseaux tels que [25] ajoutent également les représentations globales de la scène aux représentations locales, ce à différentes résolutions afin de renvoyer moins de classes aberrantes en fonction du contexte (par exemple, pas de bateaux sur une image représentant une zone urbaine).

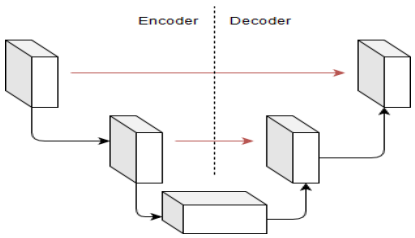


FIGURE 2 – Architecture Encodeur-Decodeur.

3 Architecture proposée

Dans cette section, les différents composants de l'architecture sont présentés. Après une description du formatage des données, les différentes parties du système de détection sont ensuite détaillées. Enfin, la mise en forme des vérités terrain est présentée.

3.1 Formatage des données d'entrées

Les nuages de points 3D peuvent être organisés de différentes façons. Une structure simple à visualiser est préférable. Le nuage de points est donc converti en grille de voxels. Cependant les convolutions 3D constituent un coût computationnel non négligeable. La totalité des véhicules est cependant positionnée sur un plan du sol à faible pente (pas forcément nulle toutefois), les tranches verticales de la grille de voxels sont considérées comme des canaux d'une image, autorisant ainsi l'utilisation de convolutions 2D plus standards.

Le système de coordonnées utilisé est le suivant : l'origine du repère est le centre du capteur LiDAR, l'axe x est orienté vers la droite, l'axe y vers le bas, l'axe z vers l'avant du véhicule. Chaque point 3D est décrit à travers ses coordonnées (x, y, z) et sa réflectance r. Le nuage est d'abord transformé en une grille régulière de taille $[n_x, n_y, n_z]$, chaque cellule représente un pavé de $[s_x, s_y, s_z]$ m^3. Chaque voxel est représenté sous la forme d'un vecteur de dimension 5 par un encodage statistique simple à partir des points en son sein. Les composantes de ce vecteur sont la hauteur maximale, la hauteur minimale, la hauteur moyenne, la densité normalisée et la réflectance maximum des points de la cellule (voir Fig. 3 et Fig. 4).

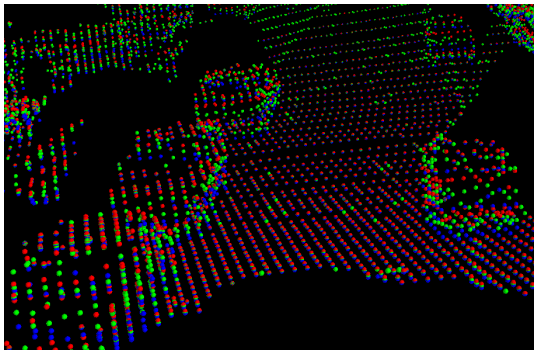


FIGURE 3 – Exemple de voxelisation. Les points rouges ont la hauteur maximale, les bleus ont la hauteur minimale, les verts la hauteur moyenne. Les cellules sont réparties régulièrement sur le plan XZ.

La grille de voxels, représentée par un tenseur de taille $[n_x, n_y, n_z, 5]$ est transformée en une pseudo-image de taille $[n_x, n_z, 5 \times n_y]$.

3.2 Architecture

La méthode peut être décomposée en deux parties : l'estimation de la carte de probabilité à l'aide d'un réseau de segmentation et l'extraction des résultats. Pour cette seconde partie, une approche basée modèle ou une solution à base de réseaux de neurones sont envisageables. Nous avons choisi ici d'utiliser un RPN *Region Proposal Network* (RPN) (voir Fig. 5). Ces deux parties sont décrites dans les paragraphes suivants. Le processus d'extraction d'ellipse sera également introduit.

Réseau de segmentation. Le rôle du réseau de segmentation est de produire une carte de score d'appartenance d'un objet de type BEV. La carte sortie est comparable à l'estimation d'une grille d'occupation mais pour une catégorie d'objet spécifique. Chaque cellule est soit occupée par l'objet d'intérêt, soit occupée par un autre objet ou vide. Une fonction sigmoïde est appliquée à la sortie du réseau pour restreindre les valeurs des scores entre 0 et 1.

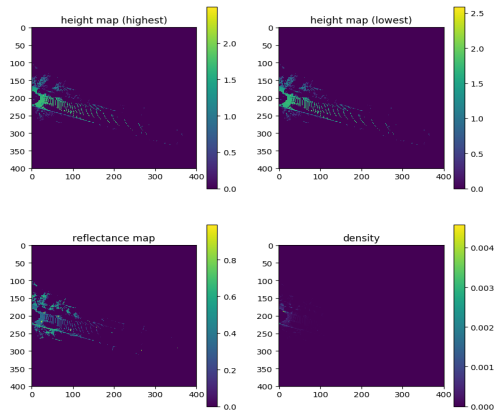


FIGURE 4 – Canaux de la grille de données, concaténés sur l'axe vertical.

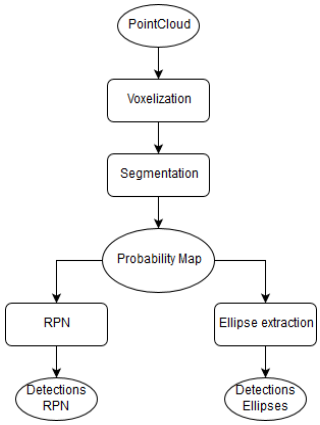


FIGURE 5 – Architecture du système.

Extraction d'ellipse. Les taches elliptiques présentent sur la carte de probabilités symbolisent un véhicule. Il est possible extraire ces ellipses avec leurs caractéristiques (centre (x_c, z_c), demi grand axe a, demi petit axe b, angle θ) grâce à une simple régression elliptique et de retrouver les boîtes englobantes correspondantes. Un exemple de processus simple d'extraction d'ellipses dans une image est décrit dans l'algorithme 1. La méthode choisie pour la régression est l'algorithme présenté dans [8]. Elle contraint la conique à être une ellipse (pas de parabole ou d'hyperbole). Elle a l'avantage d'être très rapide. Cependant, comme elle est appliquée sur les pixels de la carte, la précision des résultats dépendra notamment de la discrétisation choisie.

Data : Probabilty Map MP, patch size s, threshold peaks T_{peaks}, threshold valid T_{valid}
Result : List of BEV detections $bev_{detections}$

$Peaks \leftarrow$ findPeaks (MP, T_{peaks});
$H_{binary} \leftarrow$ binarize (MP, T_{valid});
$bev_{detections} \leftarrow$ [];

foreach *p in Peaks* **do**
  $patch \leftarrow$ extractPatchAroundPeak (H_{binary}, p);
  $cleanPatch \leftarrow$ floodFill ($patch$, p);
  $cleanPatchGradNorm \leftarrow$ norm (grad ($cleanPatch$));
  $ellipse \leftarrow$ fitEllipse ($cleanPatchGradNorm$);
  $boxParameters \leftarrow$ extractInfoEllipse ($ellipse$);
  $bev_{detections}$.insert ($boxParameters$);
end

Algorithme 1 : Exemple de procédure pour extraire les ellipses d'une carte de probabilités.

Region Proposal Network. La carte obtenue par le réseau de segmentation est passée au RPN pour classifier chaque ancre et enclencher la régression pour ajuster au mieux les paramètres à la vérité terrain. L'architecture du RPN est inspirée de celle décrite dans [26]. Sa composition est illustrée dans la figure 6.

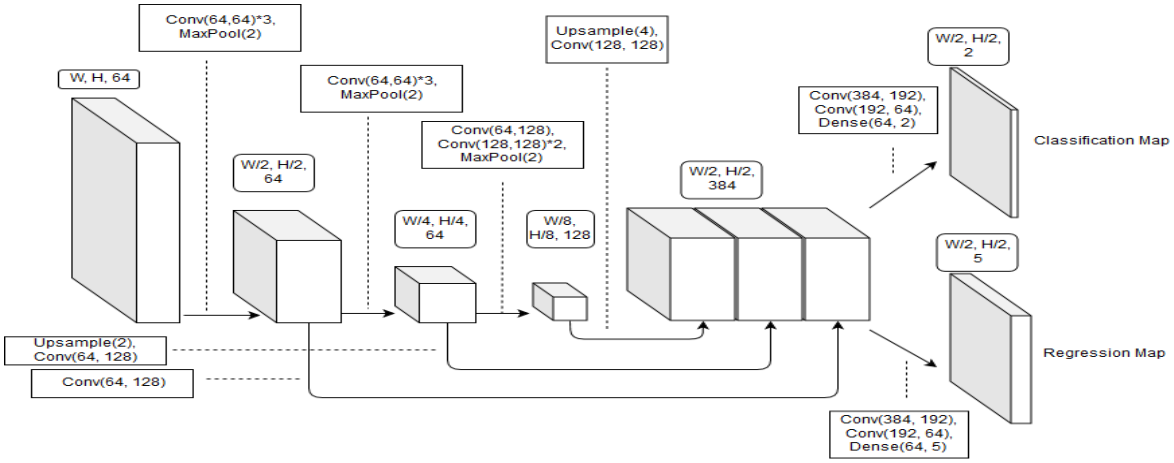


FIGURE 6 – *Region Proposal Network* (RPN).

3.3 Gestion de la vérité terrain

Les deux sous-systèmes de l'architecture sont différents dans leur structure et leurs objectifs. Leurs vérités terrains respectives sont donc formatées différemment. Cette section détaille le formatage des vérités terrains.

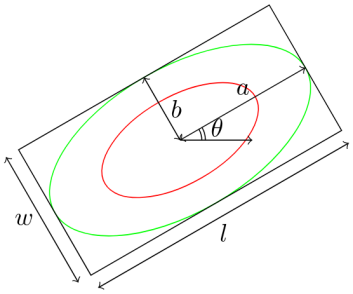


FIGURE 7 – Transformation de la boîte englobante orientée en ellipse. L'ellipse rouge est une version réduite de l'ellipse inscrite verte.

Réseau de segmentation. Le rôle d'un réseau de segmentation étant de classifier chacun des pixels d'une image, la vérité terrain doit définir quels pixels sont occupés par des voitures. En détection, les obstacles sont généralement représentés par des boîtes englobantes. Ici, les obstacles sont définis par une ellipse inscrite dans la boîte englobante (Fig. 7). Cela permet de décrire l'objet par une forme paramétrique simple. Une hypothèse est posée quant à ces ellipses : il n'y a aucune intersection entre celles-ci. Le facteur de réduction appliqué à l'ellipse inscrite permet de renforcer les écarts entre les obstacles (exemple : zones de stationnement). La scène est alors illustrée comme un ensemble de taches elliptiques avec différentes tailles et orientations.

RPN. Le RPN défini renvoie la classification de chacune des ancres ainsi que la correction à effectuer pour s'ajuster à la vérité terrain la plus proche de leur position. Comme les voitures ont des dimensions assez proches, les ancres définies ont toutes les mêmes dimensions et sont régulièrement espacées sur le sous-ensemble de $\mathbb{R}^3$ choisi. Soient $(x^a, z^a, w^a, l^a, \theta^a)$ les paramètres d'une ancre. La classification et la régression sont déterminées en même temps que les ellipses du réseau de segmentation. Un obstacle assimilé par une ellipse (x_c, z_c, a, b, θ) peut également s'écrire sous forme d'une conique (A, B, C, D, E, F) si pour un point (x, z) de celle-ci, on a :

$$Q(x,z) = Ax^2 + Bxz + Cz^2 + Dx + Ez + F = 0 \quad (1)$$

L'ellipse étant une forme fermée, une ancre est définie comme positive si son centre est à l'intérieur de celle-ci : $Q(x^a, z^a) \leq 0$.
Concernant les cibles de correction, si $(x^g, z^g, w^g, l^g, \theta^g)$ sont les paramètres de la vérité terrain et $d^a = \sqrt{l^{a2} + w^{a2}}$ la diagonale de la base d'une ancre, elles sont définies par :

$$\Delta x = \frac{x^g - x^a}{d^a}, \Delta z = \frac{z^g - z^a}{d^a}, \quad (2)$$

$$\Delta w = \log(\frac{w^g}{w^a}), \Delta l = \log(\frac{l^g}{l^a}), \quad (3)$$

$$\Delta\theta = \theta^g - \theta^a \quad (4)$$

4 Expérimentations

4.1 Jeu de données et critère d'évaluation

L'algorithme a été testé sur les données provenant de la base KITTI pour la détection d'objets. La base contenant 7481 échantillons d'entraînement, elle est découpée en un jeu de *training* et un jeu de *validation*, chacun comptabilisant la moitié des données de la base originale. L'*Average Precision* est utilisée comme métrique de comparaison. Le score d'*Intersection of Union* (IoU) est fixé à 0.7.

4.2 Détails d'implémentation

Données d'entrée. Pour les données d'entrées, l'espace des recherches est restreint à $[-40.0, 40.0] \times [-1.0, 3.0] \times$

$[0.0, 80.0]$ (les distances sont en m). Les dimensions de la grille sont $(n_x, n_y, n_z) = (400, 7, 400)$; chaque voxel correspond à un pavé de taille $(s_x \times s_y \times s_z) = (0.2 \times 0.57 \times 0.2)$ m^3.

Ancres. La taille de la carte des ancres est fixée à $(200, 200)$, les centres des ancres sont donc espacées de 0.40 m sur les axes x et z. Un seul type d'ancres est utilisée, son angle est de 0 degrés, ses dimensions sont $(w, l) = (1.6, 3.9)$ m.
Les sens de la carte de segmentation et de la carte d'ancres sont différents. Sur la carte de segmentation, plusieurs pixels sont nécessaires pour la définition d'une détection. Sur la carte des ancres, un pixel représente une hypothèse (ou un ensemble d'hypothèses si plusieurs ancres sont définies) sur l'obstacle à détecter, un seul pixel peut donc suffire.

Extraction d'ellipses. Le *patch* extrait autour de chaque pic fait 40 pixels de côté. Le seuil pour les pics T_{peaks} est fixé à 0.95 tandis qu'un pixel est considéré comme valide si sa valeur est supérieure à $T_{valid} = 0.1$. Pour accélérer le processus, une carte de visites a été créée pour éviter les traitements similaires : si un pic est détecté dans une région déjà visitée, il est ignoré.

Fonction de perte. La fonction perte utilisée pour l'entraînement est :

$$L_{total} = L_{seg} + L_{cls} + L_{reg} \tag{5}$$

avec L_{seg} se référant à la fonction perte pour l'estimation de la carte de probabilités issue du réseau de segmentation, L_{cls} est la fonction pour la classification des ancres et L_{reg} la fonction pour la régression des cibles.
La majorité des pixels de la carte de segmentation et la majorité des ancres sont liées à des zones de « background ». L_{seg} et L_{cls} sont définies en tant que fonctions dites « Focal Loss » [14], conçues pour mieux gérer les déséquilibres de classe qu'une fonction perte par entropie croisée. La « Focal Loss » est définie comme :

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{6}$$

avec p_t la probabilité estimée par le modèle, α_t est un terme de modulation lui-même dépendant d'un terme α, γ et α sont les paramètres de la fonction. (α, γ) sont fixés à $(0.25, 2)$. La valeur pour une image consiste en la moyenne de cette fonction appliquée à tous les pixels.
La fonction perte pour la régression peut elle-même être décomposée en deux sous-fonctions :

$$L_{reg} = L_{xzlw} + L_{\theta} \tag{7}$$

où L_{xzlw} est une fonction Smooth L1 sur la différence entre les cibles estimées et les cibles voulues. La fonction Smooth L1 est définie par :

$$SmoothL1(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & sinon. \end{cases} \tag{8}$$

L_{θ} est une légère variation de L_{xzlw}. Le sinus de la différence entre les cibles voulues $\Delta\theta_t$ et les cibles estimées $\Delta\theta_p$ est utilisé.

$$L_{\theta}(\Delta\theta_t, \Delta\theta_p) = SmoothL1(\sin(\Delta\theta_t - \Delta\theta_p)) \tag{9}$$

Avec cette fonction, le réseau n'infère pas directement l'angle mais la direction. En valeur absolue, un sinus de 1 indiquera une orthogonalité tandis que 0 indiquera une colinéarité. L'hypothèse réalisée est que l'orientation avant/arrière peut être donnée par un classifieur sur les données ou déterminée sur une séquence grâce aux vitesses (mis à part les cas nécessitant une marche arrière comme un stationnement).

Paramètres de l'entraînement. Le réseau de segmentation utilisé est un U-Net [22]. Le réseau complet est entraîné à l'aide d'un *Adam optimizer* sur 150 *epoches*, avec un *learning rate* de base de 0.01 et un facteur de décroissance de 0.8 appliqué toutes les 15 *epoches*.

4.3 Résultats et discussion

Les échantillons de la base de données d'entraînement sont dès le départ triés aléatoirement (aucune relation entre deux échantillons consécutifs). Le jeu de données d'entraînement est constitué des 3741 premiers échantillons. Les échantillons utilisés par les autres méthodes ne sont pas fournis.

TABLE 1 – Performances pour la détection en BEV des voitures sur les données de validation. Valeurs en *Average Precision* (AP) (en %).

Méthode	*Easy*	*Moderate*	*Hard*
MV3D[6]	86.55	78.10	76.67
Frustum Pointnet [18]	88.16	84.02	76.44
VoxelNet [26]	89.60	84.81	78.57
SECOND [24]	**89.96**	**87.07**	**79.66**
Notre approche (Ellipse)	79.86	81.74	75.55
Notre approche (RPN)	72.83	70.93	72.47

Comme le montre le tableau 1, le système proposé atteint un niveau de performance élevé malgré sa simplicité algorithmique. Plusieurs points peuvent être notés :

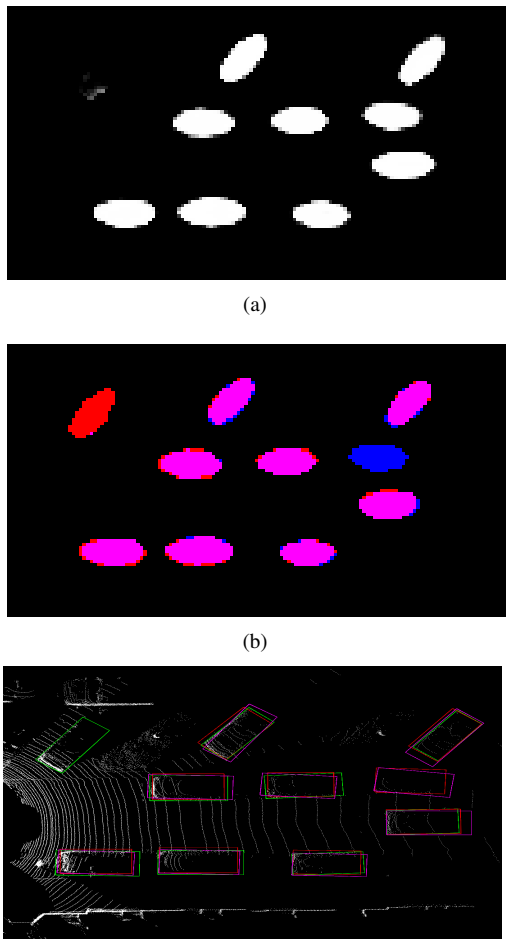
- Sur la ligne « ellipse », les cas « intermédiaires » ont de meilleurs résultats que les cas « faciles ». La principale hypothèse est liée à l'utilisation de réseau de segmentation et à la position des cas simples. Le *padding* utilisé par le U-Net peut affecter la qualité des segmentations sur les bords de l'image. Avec la configuration choisie, les cas « faciles » sont situés sur le bord gauche de l'image.
- Les performances du RPN sont moins bonnes que celles issues de l'extraction d'ellipse. Dans le système présenté, la carte des probabilités produite par

le réseau de segmentation est directement passée au RPN, sans aucune autre information. Sur les données de validation, le U-Net est appliqué sur des données dont il ne s'est pas servi pour s'entraîner, la sortie sera donc dégradée. Ces cartes « dégradées » sont passées au RPN qui, en fin d'entraînement, ne s'est entraîné que sur des données « propres ». Une solution possible serait de découpler les deux systèmes lors de l'entraînement, en utilisant en complément de la carte de probabilités des données indépendantes, par exemple réutiliser la grille d'entrée pour compenser la perte d'information.

La figure 8 illustre un cas sur des données de validations. Le réseau n'a pas détecté le premier véhicule incliné, apparaissant en rouge sur la figure 8(b), malgré un nombre de points important. Cependant la carte de probabilités (Fig. 8(a)) montre l'une légère trace à la position du véhicule, indiquant une possible présence.

Sur la figure 8(c), la prédiction s'ajuste globalement à la vérité terrain. Le bruit est principalement causé par la discrétisation de la carte de probabilités lors de l'extraction d'ellipses, les valeurs altérées passées au RPN et la précision du réseau de segmentation pour estimer les taches (il a de légères erreurs sur les bords des ellipses, voir Fig. 8(b)). La « fausse » détection quant à elle représente une vraie voiture, absente des annotations de la base.

En termes de temps de traitement, les deux réseaux sont exécutés en 30 ms tandis que l'extraction d'ellipse sur un cas avec une dizaine d'ellipses prend 3 ms.



(a) (b) (c)

FIGURE 8 – Exemple sur les données de validation. (a) La carte de probabilités. (b) Différence entre la vérité terrain et la carte estimée binarisée (seuil à 0.5). Le magenta signifie vrai positif, le rouge faux négatif et le bleu faux positif. (c) Détections sous formes de boîtes. Les vertes représentent la vérité terrain, les rouges les boîtes issues de l'extraction d'ellipse, les magenta les boîtes issues du RPN.

5 Conclusion

La méthode présentée dans le papier estime des détections en BEV à partir de nuages de points LiDAR suivant deux approches : une approche centrée traitement d'image et une approche RPN. Le réseau de segmentation prédit une carte de probabilités qui est utilisée en tant qu'indicateur de zones d'intérêt par le réseau de détection. Cette carte permet également de visualiser les régions qui ont donc suscité l'attention du réseau avant même les détections finales et éventuellement de repérer les zones potentiellement dangereuses car masquant probablement un véhicule. L'encodage utilisé actuellement est simple et rapide mais une grande partie de l'information géométrique est perdue lors de la discrétisation. L'utilisation d'encodages plus complexes tels que l'intégration d'attributs géométriques (planarité, sphéricité, etc.) ou l'apprentissage d'un encodage fera l'objet d'expériences ultérieures.

Remerciements

Ces travaux bénéficient du soutien de l'ANRT par le biais d'une bourse CIFRE avec la société Sherpa Engineering.

Références

[1] Waleed Ali, Sherif Abdelkarim, Mohamed Zahran, Mahmoud Zidan, and Ahmad El Sallab. YOLO3D : End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud. In *Conference : ECCV 2018 : "3D Reconstruction meets Semantics" workshop*, 2018.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Tr. on PAMI*, 39(12) :2481–2495, Dec 2017.

[3] Jorge Beltran, Carlos Guindel, Francisco Miguel Moreno, Daniel Cruzado, Fernando Garcia, and Arturo de la Escalera. BirdNet : a 3D Object Detection Framework from LiDAR information. *arXiv preprint arXiv :1805.01195*, May 2018.

[4] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep MANTA : A Coarse-to-fine Many-Task Network for

joint 2D and 3D vehicle analysis from monocular image. In *CVPR*, pages 2040–2049, 2017.

[5] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, pages 2147–2156, 2016.

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, page 3, 2017.

[7] M. Engelcke, D. Rao, D. Zeng Wang, C. Hay Tong, and I. Posner. Vote3Deep : Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. In *ICRA*, June 2017.

[8] Andrew Fitzgibbon, Maurizio Pilu, and Robert B Fisher. Direct least square fitting of ellipses. *IEEE Tr. on PAMI*, 21(5) :476–480, 1999.

[9] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, June 2018.

[10] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. *arXiv preprint arXiv :1712.02294*, 2017.

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars : Fast encoders for object detection from point clouds. *arXiv preprint arXiv :1812.05784*, 2018.

[12] B. Li. 3D fully convolutional network for vehicle detection in point cloud. In *IROS*, pages 1513–1518, Sept 2017.

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[14] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Tr. on PAMI*, 2018.

[15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *CVPR*, pages 806–814, 2015.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[17] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d Bounding Box Estimation Using Deep Learning and Geometry. In *CVPR*, 2017.

[18] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection From RGB-D Data. In *CVPR*, June 2018.

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet : Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv :1612.00593*, 2016.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN : Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[23] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion : Deep sensor fusion for 3d bounding box estimation. In *CVPR*, June 2018.

[24] Yan Yan, Yuxing Mao, and Bo Li. SECOND : Sparsely Embedded Convolutional Detection. *Sensors*, xs18(10) :33–37, 2018.

[25] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.

[26] Yin Zhou and Oncel Tuzel. VoxelNet : End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, June 2018.