Stage d'application

UE PRO

Rapport de stage d'exécution - LIRIS

Auteur :
M. Julien GAUBIL

Encadrant :
M. Emmanuel DELLANDREA

24/05/2021 - 27/08/2021
LIRIS
40 Av. Guy de Collongue, 69130 Écully

Remerciements

Je tiens tout d'abord à remercier l'ECL, école d'ingénieur-es qui permet à ses étudiants de réaliser ces stages stages d'application très formateurs lors de leur deuxième année. Les conseils donnés pour la rédaction de ce rapport de stage ont été précieux.

Je remercie Emmanuel DELLANDREA qui m'a permis d'obtenir ce stage au LIRIS, et a été mon tuteur de stage. Il m'a en effet proposé un sujet qui correspondait à mon projet professionnel, et m'a donné les ressources pour pouvoir l'appréhender. Je le remercie vivement pour son accueil au sein de son équipe et pour avoir pris de son temps ainsi que du temps de son équipe pour m'aider dans mes tâches quotidiennes. Mes remerciements s'adressent aussi à Mr. CHEN qui m'a, lui aussi, accordé de son temps et sa confiance. Leur expérience s'est avérée importante pour m'aider à progresser et me mettre dans les meilleures dispositions.

Enfin, je tiens particulièrement à remercier tous les membres du laboratoire avec lesquels j'ai pu échanger pour leur esprit d'entraide : ils m'ont aiguillé sur le travail à accomplir et n'ont pas hésité à répondre à mes questions.

Table des matières

Résumé

Ce stage d'application a été effectué dans le domaine de la recherche en Machine Learning. Cela correspond à mon projet professionnel, ce stage a donc été le moyen de découvrir ce métier et son fonctionnement.

Le stage s'inscrit dans les problématiques de compréhension de scènes, où il s'agit de créer des algorithmes pour permettre aux ordinateurs de comprendre une scène visuelle qui leur est présentée. La compréhension passe notamment par la détection des objets et leur reconnaissance. Le problème de recherche de ce stage vise également à estimer la pose des objets détectés. Pour cela, des algorithmes dits d'apprentissage automatique (*Machine Learning*) sont utilisés.

Abstract

The context of this internship was the academic research in Machine Learning. Since it corresponds to my professional project, this internship has been very useful to help me understanding this job and its functioning.

The internship tackles the problematic of Scene Understanding, that consists in creating algorithms to help computers understanding a visual scene. The understanding often requires object detection and recognition. This internship also aims at estimating the pose of detected objects. To perform those tasks, Machine Learning algorithms are used.

1 Etat de l'art

1.1 Contexte du stage

Le stage s'inscrit dans le cadre du projet Chiron auquel participe le LIRIS. Ce projet est commun avec l'université Japonaise de Nagoya ainsi que l'université allemande TU Darmstadt.

Il s'agit d'un projet qui vise à concevoir un robot capable d'aider les humains et de comprendre la scène qui l'entoure (repérer les objets, les suivre, détecter comment les saisir). Le projet Chiron comprend aussi une partie de prédiction d'intentions, l'objectif n'étant pas d'avoir un robot complètement autonome.

Le scénario d'utilisation est celui d'un patient alité dans un hôpital qui commande à un robot d'aller chercher pour lui un produit au supermarché de l'hôpital. Le patient dispose d'une interface pour contrôler le robot, a priori un gant. Il est donc capable d'initier le mouvement, mais le robot doit comprendre son intention et terminer le mouvement. Pour cela, il doit être capable de comprendre la scène.

Le LIRIS est en charge de cette partie de compréhension de scènes via des algorithmes d'apprentissage automatique en vision par ordinateur. Le problème se restreint à la reconnaissance de la scène dans le magasin pour saisir les objets.

La technologie utilisée est un modèle de d'apprentissage profond tout en un, qui fait notamment de l'estimation de poses : détection des objets, classification et estimation de leur translation et rotation par rapport à la caméra. L'objectif est que le robot puisse s'adapter à différentes conditions sans avoir à l'entraîner de nouveau (scénario d'Incremental Learning). De plus, les données sur lesquelles seront entraînées le modèle seront des données synthétiques. En effet, il est coûteux en temps d'obtenir des données d'entraînement réelles précisément annotées par des humains tandis qu'il est possible de générer un grand volume de données synthétiques. Permettre à un modèle entraîné sur des données synthétiques d'être performant sur des données réelles pose alors un autre défi, communément appelé *Domain Shift*.

Le stage se place dans le contexte de l'arrivée prochaine d'un doctorant au LIRIS sur le projet. Les objectifs du stage sont alors :

— réaliser un état de l'art des méthodes d'apprentissage automatique pour la détection d'objets et l'estimation de poses, qui sont deux tâches clefs pour la compréhension de la scène dans le scénario d'application,
— implémenter un modèle qui servira de point de comparaison (*baseline*) à

partir de l'état de l'art.

1.2 Principes scientifiques utilisés

La sous-section suivante se consacre à une rapide introduction aux réseaux de neurones artificiels et à leurs application à la segmentation d'instances et l'estimation de poses 6D. Le lecteur familier de ces notions pourra directement se rendre à la sous-section suivante qui détaille les solutions techniques retenues.

1.2.1 Réseaux de neurones

Les réseaux de neurones sont des algorithmes de Machine Learning qui ont pour vocation de reproduire le fonctionnement de l'apprentissage par un cerveau biologique. Pour cela, les réseaux de neurones sont composés d'un élément de base que sont les neurones articiels, qui visent à imiter le fonctionnement des neurones biologiques.

Les neurones sont ensuites rassemblés en couches. Les neurones d'une même couche ont alors tous la même fonction d'activation. On distingue 3 types de couches :

— *couche d'entrée* : en bleu sur l'image 1, elle possède autant de neurones que les données en entrée ont de variables. Exemple : images 20x10 en entrée = 200 neurones sur la couche d'entrée.

— *couche de sortie* : en vert sur l'image 1. Elle contient la contient K neurones, où K est le nombre de classes du problème.

— *couches cachées* : en orange sur l'image 1. Comprise entre la couche d'entrée et de sortie, chacune contient N_i neurones. Les connexions entre les neurones de deux couches consécutives sont représentées par une matrice dont les éléments sont les poids du modèle. Chaque couche possède une fonction d'activation qui s'applique aux valeurs en entrée de chacun des neurones de la couche. Plusieurs choix pour la fonction d'activation sont possibles.

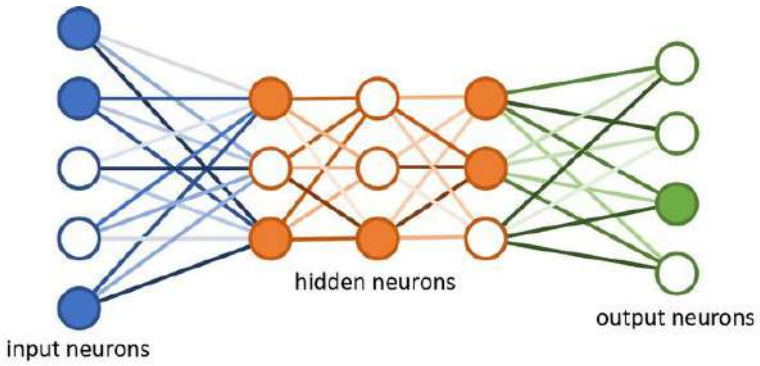

FIGURE 1 – Schéma simplifié d'un réseau de neurones

Les paramètres d'un réseau de neurones sont donc les poids qui représentent les connexions entre ces neurones.

D'un point de vue technique, il est également possible de distinguer deux types de couches :

— les couches dites *fully-connected*, représentées sur la figure 2. Chaque neurone d'une couche fully-connected est connecté à tous les neurones de la couche précedente ;
— les couches pour lesquelles chaque neurone n'est connecté qu'à une partie des neurones de la couche précédente. C'est le cas des couches dites de convolution, représentées figure 3, utilisées dans les réseaux de neurones convolutifs.

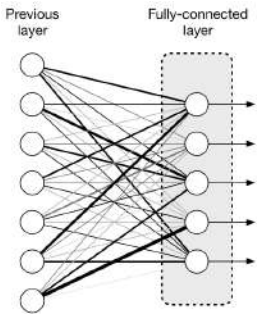

FIGURE 2 – Couche fully-connected dans un réseau de neurones

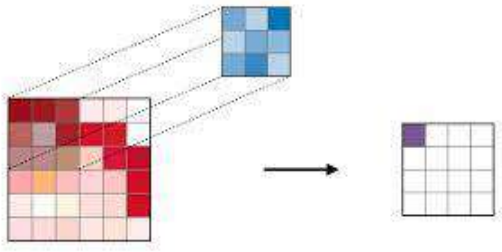

FIGURE 3 – Couche de convolution dans un réseau de neurones

L'apprentissage se découpe généralement en 2 phases :

1. **l'entraînement** : à l'aide du calcul d'une fonction de coût et de l'algorithme de descente du gradient, les poids du réseau sont modifiés jusqu'à converger vers des valeurs qui minimisent la fonction de coût sur le dataset.
2. **la généralisation/inférence** : une fois entraîné, le réseau est prêt à faire des prédictions sur des instances qui lui sont inconnues.

1.2.2 Détection d'objets et estimation de poses

La détection d'objets et l'estimation de poses sont deux domaines de la Computer Vision qui ont d'importantes applications en Robotique, et donc pour le projet Chiron.

Il s'agit, étant donné la représentation d'une scène, d'arriver à détecter les objets présents au sein de la scène, et d'estimer leur translation et rotation par rapport à la caméra.

Le passage de coordonnées $C = (X, Y, Z)$ dans le repère de la caméra aux coordonnées $c = (x, y, z)$ dans le plan image est souvent modélisé par :

$$c = K \times R_t \times C$$

où

$$R_t = \begin{bmatrix} & R_{3\times3} & & t_x \\ & & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ et } K = \begin{bmatrix} f_x & S_{uv} & c_u & 0 \\ 0 & f_y & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

avec $R = R_{3\times3}$ est la matrice de rotation qui sera estimée par le modèle d'estimation de pose, et $T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$ la translation. K est la matrice des paramètres intrinsèques de la caméra.

La représentation d'une scène se compose classiquement d'images, de cartes de profondeur, et de nuages de points appartenant à des modèles 3D d'objets.

Les applications en Robotique de la détection d'objets et l'estimation de poses posent le problème de la capacité de ces modèles à tourner en temps réel. En effet, lors de la phase d'inférence, il faut idéalement que le modèle soit capable de détecter les objets au sein de la scène et prédire leur pose avec une vitesse

suffisante pour permettre au robot d'interagir avec son environnement.

Les datasets et métriques classiquement utilisées en détection d'objet et segmentation d'instances sont présentés en détail dans l'annexe.

1.3 Segmentation d'instances

Cette sous-section vise à proposer l'état de l'art des méthodes d'apprentissage profond de segmentation d'instances.

1.3.1 BlendMask

Le modèle **BlendMask** a été introduit dans le papier [1], présenté à la conférence CVPR 2020. Le code relatif à son implémentation est disponible à : https://github.com/aim-uofa/AdelaiDet/.
Le modèle est également présenté dans une vidéo Youtube intéressante.

BlendMask est une méthode de segmentation d'instances qui propose pour chaque objet détecté un masque binaire. Elle permet de meilleures performances que le modèle très répandu Mask R-CNN, en étant plus efficace. Notamment, le temps d'inférence de la solution n'est pas proportionnel au nombre d'instances dans l'image. BlendMask est donc très intéressante pour des applications en temps réel.
Bien que ce ne soit pas explicitement présenté dans le papier, le modèle est capable de classifier les instances qu'il rencontre, ce qui est nécessaire pour le projet Chiron.

La solution consiste en un module (un *blender*, mixeur) qui incorpore des informations riches au niveau de l'instance et des features précises au niveau des pixels. Cela utilise un mécanisme d'attention. Les masques proposés sont donc de meilleure qualité que Mask R-CNN, notamment parce que le module de segmentation utilise des features de meilleure résolution.

L'architecture est composée de 3 modules distincts représentés figure 4 :

— un module de **détection** qui permet de prédire des bounding boxes pour les objets détectés et les attention maps correspondantes. Il est composé d'un réseau backbone ainsi que d'un Feature Pyramidal Network (FPN), puis de sous-modules appelés *towers* qui sont rajoutées à la suite du FPN.

En sortie de ces towers, une couche de convolution permet de prédire une bounding box pour chaque objet détecté et les attention maps correspondantes.

— un module de **prédiction** de masques partiels, appelé *bottom module*. Ces masques partiels sont appelés *bases* et vont être fusionnés avec pondération grâce aux attentions maps produites dans le module de détection. Ce module est composé du module détecteur de DeepLabV3+ et utilise en entrée les features produites par le backbone, ou les features produites par le FPN.

— un module de **fusion** appelé *blender* qui fusionne les bases produites dans le bottom module en les pondérant grâce aux attention maps produites dans le module de détection.

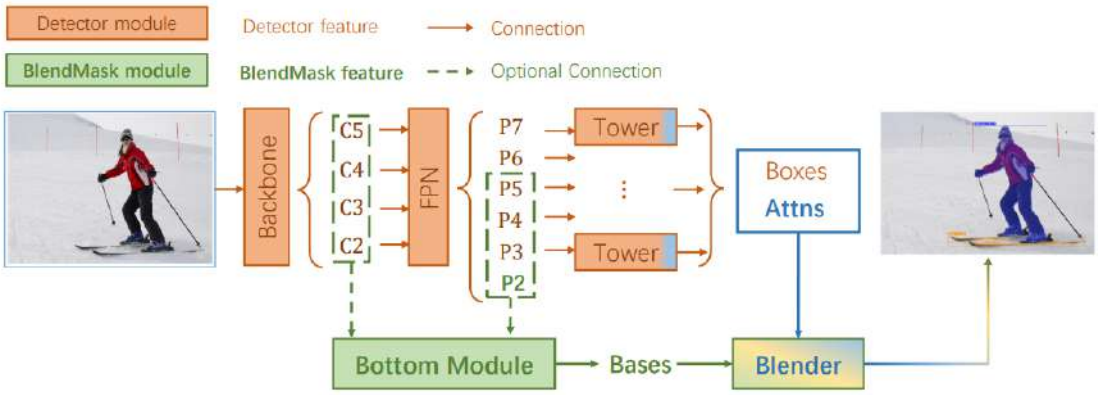

FIGURE 4 – Architecture de BlendMask

La figure 5 permet d'illustrer la fusion (réalisée dans le module blender) des bases en pondérant avec les attention maps produites par le module de détection.

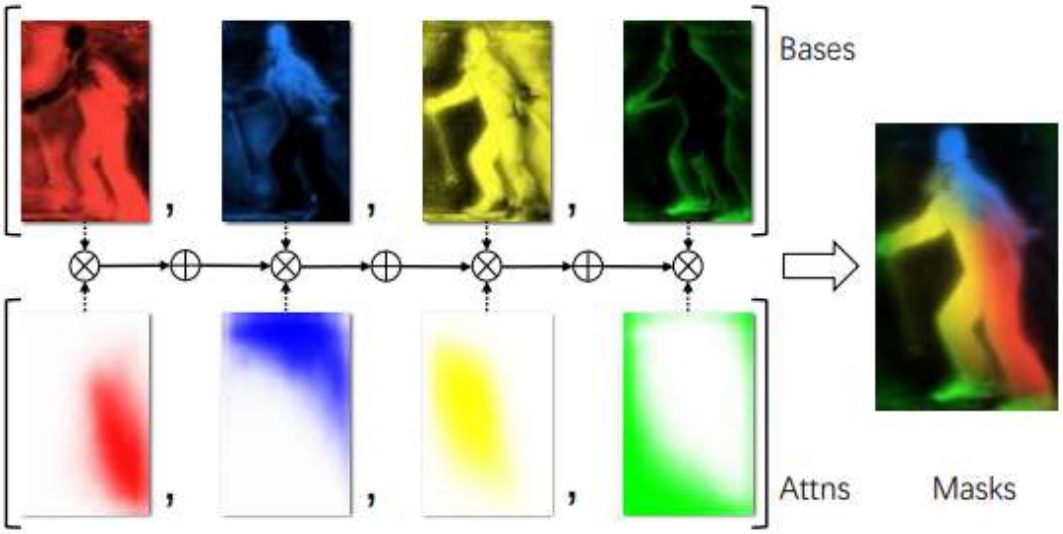

FIGURE 5 – Concaténation des bases pondérées par l'attention

Une étude qualitative montre que les masques de BlendMask sont plus précis que ceux produits par Mask R-CNN. Un exemple est présenté figure 25

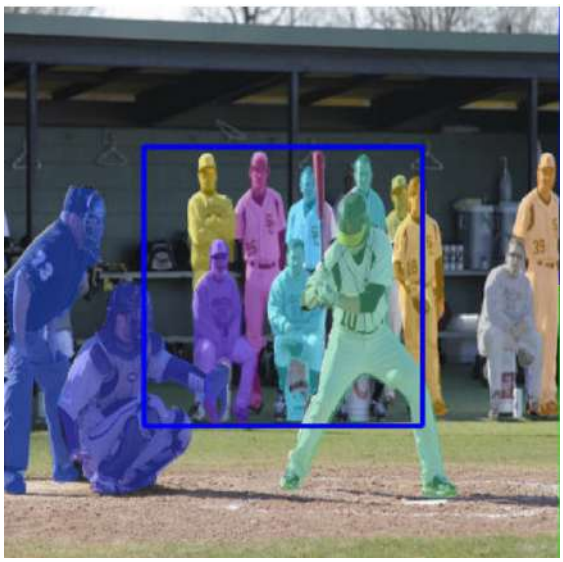

FIGURE 6 – Prédiction de masques par BlendMask sur MS-COCO

Une application en temps réel de BlendMask sur des scènes de films est présentée ici, et témoigne de la capacité du modèle à être efficace en temps réel

en produisant des masques précis. De plus, cette vidéo montre que BlendMask est effectivement capable d'effectuer une segmentation d'instances en identifiant les objets.

Les résultats détaillés de l'étude menée sur le modèle BlendMask sont présentés en annexe. Le modèle BlendMask sera plus tard comparé aux autres modèles de segmentation d'instances étudiés.

1.3.2 CenterMask

Le modèle **CenterMask** a été introduit dans le papier [2], présenté à la conférence CVPR 2020. Le code relatif à son implémentation est disponible à : https://github.com/youngwanLEE/CenterMask

CenterMask est un modèle destiné à être placé à la suite d'un réseau détecteur d'objet d'un certain type (appelé anchor-free) pour réaliser une segmentation d'instances. Il est utilisé à la suite du détecteur anchor-free FCOS. Les auteurs proposent un réseau backbone amélioré, VoVNetv2, qui présente de meilleurs résultats que ResNet et DenseNet.

L'architecture de CenterMask est composée de 3 parties principales (cf figure 7) :

- un réseau backbone pour l'extraction de features suivi d'un Feature Pyramidal Network (FPN), dont les features maps renvoyées sont de résolutions différentes.
- un réseau FCOS de détection d'objets anchor-free, proposal-free.
- un module appelé SAG-Mask, qui utilise un mécanisme d'attention pour produire les masques de segmentation pixel par pixel. Le mécanisme d'attention permet de mettre en valeur les pixels les plus intéressants et de supprimer le bruit.

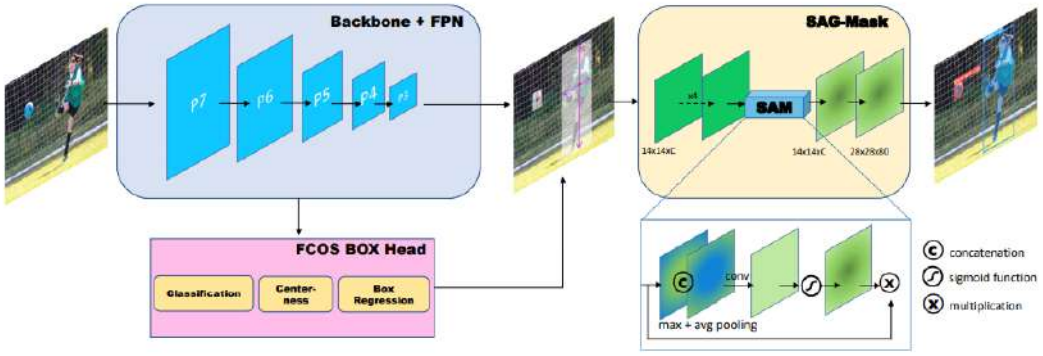

FIGURE 7 – Architecture générale du modèle CenterMask

En sortie du module RoIAlign, les features passent à travers 4 couches de convolution, puis un sous-module appelé SAM qui se compose de deux opérations successives d'average et max pooling suivies d'une concaténation. Ensuite, une convolution 3×3 est appliquée, puis la fonction sigmoïde, afin d'obtenir les attention maps qui peuvent pondérer les features maps obtenues avant le sous-module SAM.

Cela permet de pondérer grâce à l'attention spatiale l'importance des différentes régions des features maps en entrée de SAM. Une déconvolution est enfin appliquée, puis une convolution 1×1 permet d'obtenir les masques de segmentation.

Pour la segmentation d'instances en temps réel, une version lightweight (allégée) de CenterMask, CenterMask-Lite est proposée. Cette implémentation réduit le backbone, les « heads » utilisées par FCOS pour la prédiction de bounding boxes ainsi que les heads utilisées pour la production des masques. Cette version est beaucoup plus rapide (>30 fps contre 10 fps pour CenterMask) tout en conservant de bonnes performances : les meilleures pour un modèle real-time.

Enfin, la qualité des masques produits par CenterMask paraît bonne sur les exemples fournis dans le papier. Le modèle semble capable de bien segmenter les objets même lorsqu'ils sont présents en grand nombre dans la scène, ainsi qu'il est possible de le voir figure 28.

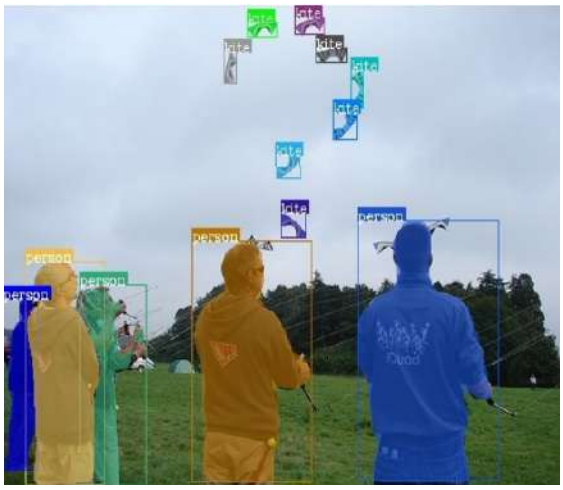

FIGURE 8 – Exemple de masques produits par CenterMask sur MS-COCO

Les résultats détaillés de l'étude menée sur le modèle CenterMask sont présentés en annexe. Le modèle CenterMask sera plus tard comparé aux autres modèles de segmentation d'instances étudiés.

1.3.3 MEInst

Le modèle **MEInst** a été introduit dans le papier [3], présenté à la conférence CVPR 2020. Le code relatif à son implémentation est disponible à : https://github.com/aim-uofa/AdelaiDet/

Le modèle Mask Encoding based Instance segmentation (MEInst) est un modèle de segmentation d'instances qui peut être mis à la suite d'un réseau détecteur d'objets pour performer la segmentation en masques. Le réseau détecteur utilisé est FCOS, qui est rapide, performant, et d'un type particulier appelé anchor-free. MEInst vise à encoder le masque 2D de chaque instance en un vecteur de plus petite dimension, en exploitant les redondances dans la représentation 2D du masque. Pour réaliser cette réduction de dimension, une Analyse en Composantes Principales est réalisée.

L'architecture du modèle MEInst est présentée sur la figure 9 :

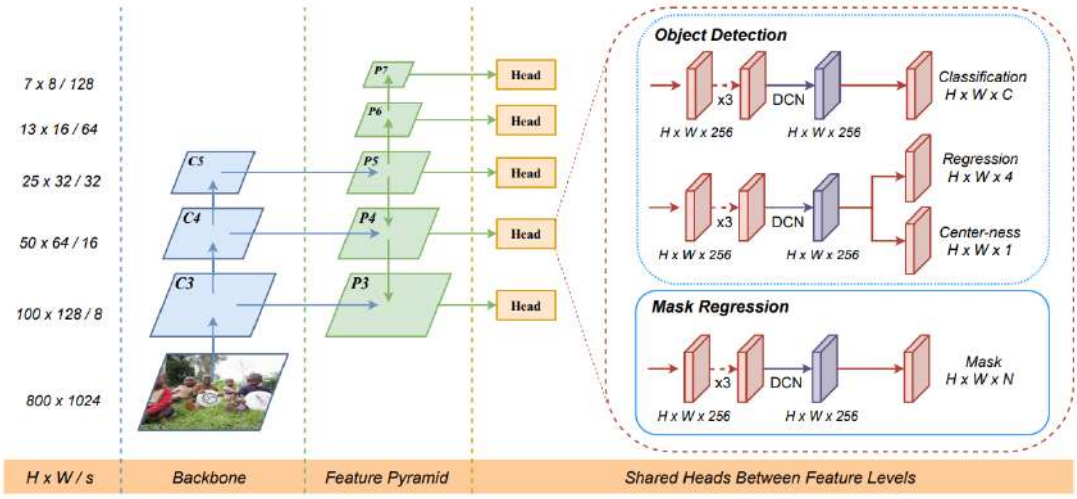

FIGURE 9 – Architecture générale du modèle MEInst

Le premier module est composé d'un réseau backbone et d'un Feature Pyramidal Network (FPN) pour la feature extraction. Le backbone utilisé est ResNet-50. Ensuite, le détecteur d'objets FCOS est utilisé afin de détecter les objets et déterminer leurs bounding boxes. Ce détecteur se compose de modules appelés *heads* constitués de deux branches, la première dédiée à la classification de l'objet détecté, la seconde dédiée à l'estimation de la centerness et la régression des contours de la bounding box.

Dans MEInst, FCOS est légèrement modifié pour améliorer ses résultats, des convolutions déformables sont utilisées dans la dernière couche de convolution des *heads*, afin de réduire le nombre de Faux-Positifs prédits.

Une branche supplémentaire est également rajoutée aux *heads* de FCOS. Cette branche réalise la prédiction des coefficients des masques de segmentation. L'objectif de MEInst est d'obtenir une représentation plus compacte des masques, partant du fait que seuls les pixels sur les contours des masques sont réellement significatifs, les pixels à l'intérieur comportent beaucoup de redondances. L'objectif est donc de passer d'un masque $M \in \mathbb{R}^{H \times W}$ à un vecteur qui représente ce masque, $u \in \mathbb{R}^n$ où $n << H.W$.

Pour cela, lors de l'entraînement, les masques ground-truth sont extraits puis mis sous forme de vecteur $v \in \mathbb{R}^{H.W}$. Une Analyse en Composantes Principales est alors réalisée avec pour objectif de déterminer les matrices T et W telles que $v = T.u$ et $\tilde{u} = W.u$ où $\tilde{u}$ est la reconstruction du vecteur u. Les matrices sont

donc déterminées lors de l'entraînement sur le training set X en minimisant l'erreur de reconstruction quadratique sur X :

$$T^*, W^* = arg\min_{T,W} \sum_{u \in X} \|u - \tilde{u}\|^2 = arg\min_{T,W} \sum_{u \in X} \|u - W.T.u\|^2$$

Lors de l'inférence, un vecteur représentant le masque $\tilde{v} \in \mathbb{R}^n$ est prédit, et le masque y est obtenu par l'opération $y = T^*.\tilde{v}$.

Les masques produits par MEInst paraissent de bonne qualité et adaptés à la présence de nombreuses instances dans l'image, bien que sur certains exemples proposés, les contours semblent un peu grossiers, en particulier pour les objets écharpés. Des exemples sur MS-COCO sont présentés figure 29.

FIGURE 10 – Exemples de masques produits par MEInst sur MS-COCO

Les résultats détaillés de l'étude menée sur le modèle MEInst sont présentés en annexe. Le modèle MEInst sera comparé aux autres modèles de segmentation d'instances étudiés dans le paragraphe suivant.

1.3.4 Comparaison des modèles de segmentation d'instances

Les trois modèles étudiés sont maintenant étudiés dans trois configurations différentes : leur configuration la plus performante, leur configuration la plus rapide, et leur configuration qui réalise le meilleur compromis performances/rapidité.

Les résultats des modèles sur le dataset MS-COCO dans leur configuration la plus performante sont présentés table 1 :

Modèle	Dataset	AP	FPS	Temps (ms)	AP^S	AP^M	AP^L	Aug	Backbone	GPU
BlendMask (detectron2) + deform conv	MS-COCO	41.3	- ($\approx$ 10)	105.0	22.7	44.1	**54.5**	O	ResNet-101+FPN	1080Ti
CenterMask	MS-COCO	**41.8**	**12.9**	**77**	**24.4**	**44.4**	54.3	-	VoVNetV2-99+FPN	Titan Xp
MEInst	MS-COCO	38.2	- ($\approx$ 10)	-	22.6	40.0	49.3	O	ResNeXt-101 + FPN + DCN	1080Ti

TABLE 1 – Comparaison des versions les plus performantes des modèles étudiés

La colonne *Aug.* indique si une augmentation de données est réalisée lors de l'entraînement du modèle.

Il apparaît que le modèle qui atteint les meilleures performances est le modèle CenterMask muni du backbone VoVNetv2-99 introduit dans le papier. Ce niveau de performances est atteint à 12.9 FPS (77 ms de temps d'inférence). Le modèle BlendMask muni du backbone ResNet-101 et implémenté sur Detectron2 avec des convolutions déformables présente des performances très proches avec un temps d'inférence supérieur (105 ms).

Les performances de MEInst sont inférieures de l'ordre de 3 points d'Average Precision par rapport à BlendMask et CenterMask, avec un temps d'inférence du même ordre.

La table 2 présente les résultats des modèles sur MS-COCO dans leur version la plus rapide :

Modèle	Dataset	AP	FPS	Temps (ms)	AP^S	AP^M	AP^L	Backbone	Résolution	GPU
BlendMask-RT	MS-COCO	**35.1**	-	36.0	-	-	-	ResNet-50+FPN	550x*	1080Ti
CenterMask-Lite	MS-COCO	26.7	**50.0**	**20**	9.0	27.0	40.9	MobileNetv2 + FPN	-	Titan Xp
MEInst	MS-COCO	23.9	28.2	-	23.9	42.4	24.1	ResNet-50+FPN	400x*	1080Ti

TABLE 2 – Comparaison des versions les plus rapides des modèles étudiés

Cette comparaison montre que la version temps-réel de CenterMask est celle

qui permet, de loin, la plus grande rapidité (50 FPS). Cette version est atteinte avec le backbone MobileNetv2. Cependant, cela se fait au prix d'un écart de performances important avec la version temps-réel de BlendMask (presque 10 points d'AP), qui tourne aux alentours de 28 FPS. Il convient de noter que les GPUs utilisés pour évaluer CenterMask et BlendMask diffèrent, ce qui pourrait induire des différences de rapidité et fausser la comparaison.

La table 3 présente les résultats des modèles sur MS-COCO dans leur version qui offre le meilleur compromis performances/rapidité :

Modèle	Dataset	AP	FPS	Temps (ms)	AP^S	AP^M	AP^L	Backbone	Résolution	GPU
BlendMask-RT	MS-COCO	36.8	-	47.6	-	-	-	ResNet-101+FPN	550x*	1080Ti
CenterMask-Lite	MS-COCO	**40.7**	**35.7**	**28**	22.4	43.2	53.5	VoVNetv2-39+FPN	-	Titan Xp
MEInst	MS-COCO	28.4	18.5	-	-	-	-	ResNet-50+FPN	600x*	1080Ti

TABLE 3 – Comparaison des versions des modèles étudiés offrant le meilleur compromis performances/rapidité

Cette comparaison montre que le meilleur compromis rapidité/performances est permis par le modèle CenterMask-Lite muni du backbone VoVNetv2-39 introduit par le papier. Parmi les modèles temps-réel, il obtient la meilleure Average Precision d'environ 4 points sur les deux autres, tout en étant le plus rapide (35.7 FPS). Les compromis proposés par MEInst et BlendMask ne dépassent pas les 22 FPS, ce qui est un écart important avec CenterMask.

Cependant, les performances de CenterMask sont mesurées sur un GPU Titan Xp alors que les deux autres modèles sont évalués sur un 1080Ti, ce qui peut affecter la mesure de rapidité et fausser la comparaison.

A l'aulne de cette étude, le modèle qui paraît le plus intéressant est donc CenterMask, puisqu'il présente les meilleures performances et le meilleur compromis performances/rapidité. Une interrogation subsiste quant à son évaluation sur un GPU 1080Ti, qui pourrait affecter la rapidité reportée dans le papier.

1.3.5 Autres modèles intéressants

— Fully Convolutional Networks for Panoptic Segmentation (CVPR21)

— PolarMask (CVPR20)
— Deep Snake (CVPR20)

1.4 Estimation de poses

Cette sous-section vise à proposer l'état de l'art des méthodes d'apprentissage profond d'estimation de poses.

1.4.1 FS-Net

Le modèle **FS-Net** a été introduit dans le papier [4], présenté à la conférence CVPR 2021. Le code relatif à son implémentation est disponible à : https://github.com/DC1991/FS_Net

Le modèle Fast Shape-based Network (FS-Net) vise à proposer une méthode d'estimation de poses 6D rapide à partir d'une image RGB-D basée sur la forme des objets. Le problème posé est un problème de *category level pose estimation* qui consiste à prédire la pose et l'échelle d'un objet relativement à une représentation canonique de la classe à laquelle appartient l'objet.

L'architecture de FS-Net, présentée figure 11, comporte trois parties principales :
— un module de détection d'objets 2D au sein de l'image, et d'estimation de leur bounding box. Pour cela, le modèle YOLOv3 est utilisé.
— un module de segmentation 3D et d'estimation de rotation, réalisé par réseau graphe de convolution 3D. La rotation est ensuite extraite de manière découplée par deux décodeurs à partir des features prédites par le graphe de convolution 3D.
— un module d'estimation de translation et de taille, réalisé par un réseau résiduel basé sur le modèle PointNet.

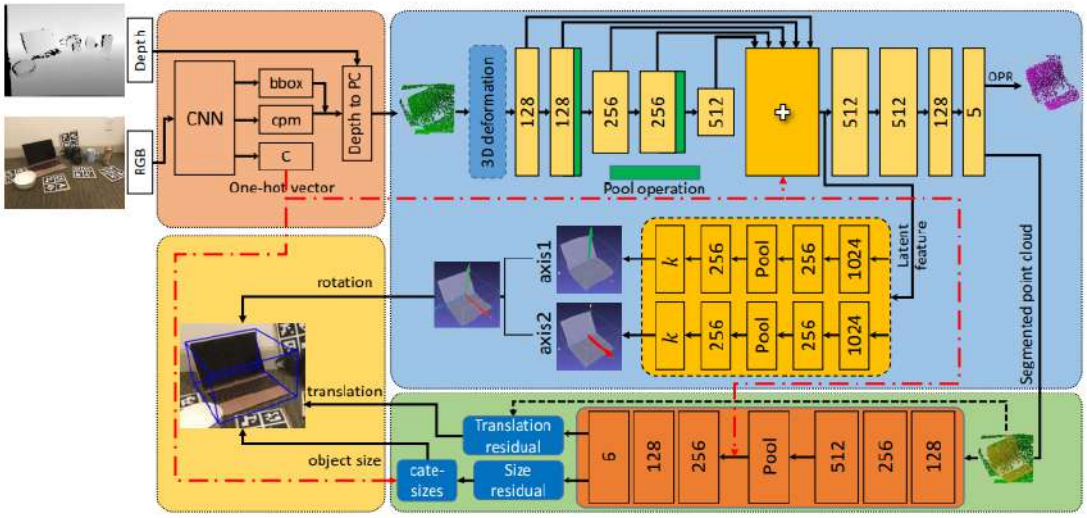

FIGURE 11 – Architecture générale du modèle FS-Net

Le module auto-encoder de convolution de graphe 3D (3DGC) consiste en un noyau de convolution de m vecteurs unitaires qui balayent un nuage de point en entrée du module. Pour chaque point du nuage, n vecteurs sont générés entre le point et chacun de ses n plus proches voisins dans le nuage. Le noyau de la convolution de graphe 3D s'applique alors aux n vecteurs générés en obtenant la somme des similarités cosinus entre les vecteurs du noyau et les n vecteurs générés.

La 3DGC apprend alors l'orientation des m vecteurs dans son noyau, et obtient de plus fortes valeurs lorsqu'il matche avec un pattern 3D défini par le point central et ses n plus proches voisins.

Un auto-encodeur basé sur la 3DGC est donc implémenté dans FS-Net (partie haute de la zone bleue, figure 11) pour extraire les features de forme du nuage de points pour la segmentation et l'estimation de rotation.

Deux décodeurs (partie basse de la zone bleue, figure 11) extraient la rotation de manière découplée. Ils prédisent deux vecteurs orthogonaux qui représentent la rotation de l'objet.

Le dernier module de FS-Net est composé d'un réseau PointNet qui prend le nuage de points segmenté de chaque objet en entrée (zone jaune, figure 11). Ce réseau performe à la fois une estimation du résidu entre la translation ground-truth et le point moyen du nuage de points segmenté, et une estimation du résidu entre la taille de l'objet et la taille moyenne des objets de sa classe.

Ce module permet d'estimer la taille et la translation de l'objet.

Des exemples de prédictions réalisées par FS-Net sur NOCS-REAL sont présentés figure 34 :

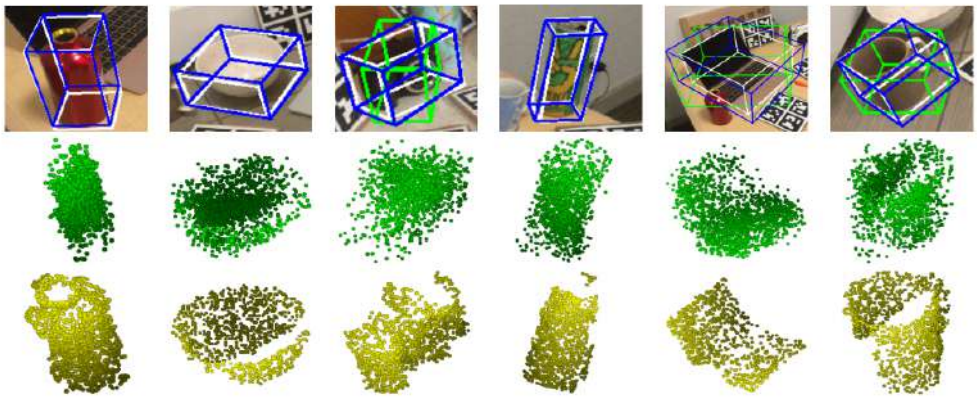

FIGURE 12 – Exemples de prédictions de FS-Net sur NOCS-REAL

La pose et la taille des objets sont bien prédites dans la plupart des cas, et la bounding box 3D prédite (en bleu) est bien alignée avec la ground-truth (en blanc).

Les résultats détaillés de l'étude menée sur le modèle FS-Net sont présentés en annexe. Le modèle FS-Net sera plus tard comparé aux autres modèles d'estimation de poses étudiés.

1.4.2 Single-stage 6D pose estimation

Le modèle étudié dans cette section a été introduit dans le papier [5], présenté à la conférence CVPR 2020. Le code relatif à son implémentation est disponible à : https://github.com/cvlab-epfl/single-stage-pose

Le modèle proposé dans ce papier vise à créer une architecture profonde et one-stage pour régresser la pose 6D des objets au sein d'une image RGB à partir de correspondances 2D-3D. Les objets sont supposés rigides, et un modèle 3D de chaque objet est supposé disponible.

La détection des objets et la production des correspondances entre une image en entrée et les points 3D du modèle 3D de l'objet (correspondance 2D-3D) est effectuée par l'architecture PVNet.

Ensuite, étant donnés n ensembles de m correspondances 2D-3D $\{p_i \leftrightarrow u_{i,m}\}_{1 \leq i \leq n, 1 \leq j \leq m}$ (appelés *clusters*), le papier propose de retrouver la rotation et la translation des objets par rapport à la caméra en effectuant une régression non-linéaire implémentée par un réseau profond.

Ce réseau composé de 3 modules principaux (cf figure 13) :

— un module de feature extraction locale.
— un module d'agrégation des features extraites.
— un module de prédiction des poses.

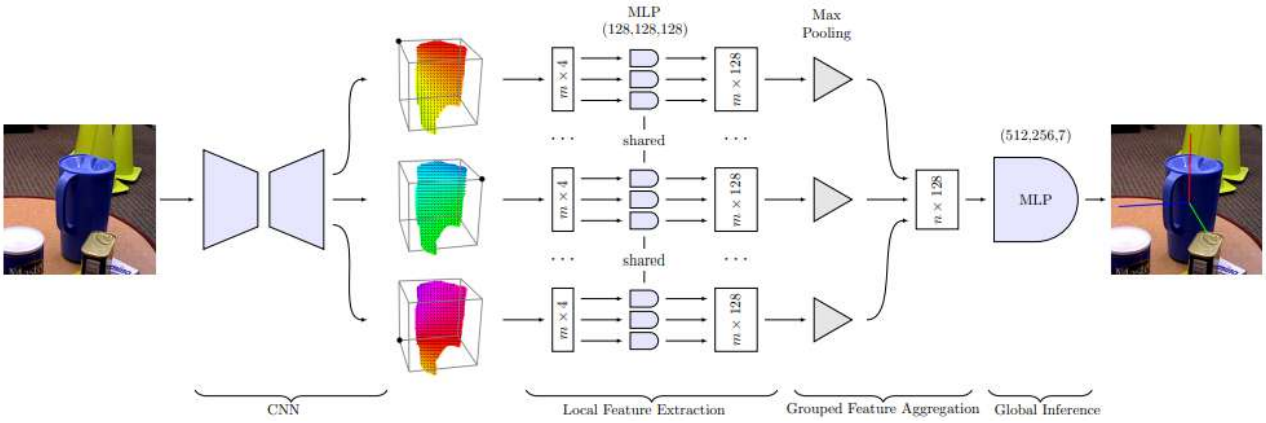

FIGURE 13 – Architecture du réseau de régression non-linéaire de la pose à partir des correspondances 2D-3D

Le module de **feature extraction locale** est constitué d'un réseau MLP à 3 couches qui est appliqué sur chacun des clusters et extrait des features locales pour chacune des correspondances. Pour chacun des clusters $i \in 1, ..., n$, on obtient pour chacune des $j \in 1, ..., m$ correspondances un feature vector $f_{i,j} \in \mathbb{R}^D$.

Le module d'**agrégation de features** consiste à appliquer pour chaque cluster $i \in 1, ..., n$ une opération de max pooling sur les features vectors obtenus $f_{i,j}, j \in 1, ..., m$ afin d'obtenir un feature vector $f_i \in \mathbb{R}^D$ pour le cluster i. Une opération de concaténation des features vectors ainsi obtenus pour chacun des cluster est ensuite réalisée. Un feature vector de dimension nD est ainsi obtenu en sortie du module.

Le module de **prédiction** consiste en un réseau MLP à 3 couches qui prend en entrée le vecteur de dimension nD en sortie du module précédent, et renvoie la pose finale avec le vecteur de translation et les angles de rotation sous forme de quaternion.

Des exemples de prédictions de poses 6D sur LINEMOD-Occluded sont présentés figure 37. La pose est visualisée comme la reprojection 2D du maillage 3D prédit pour chaque objet :

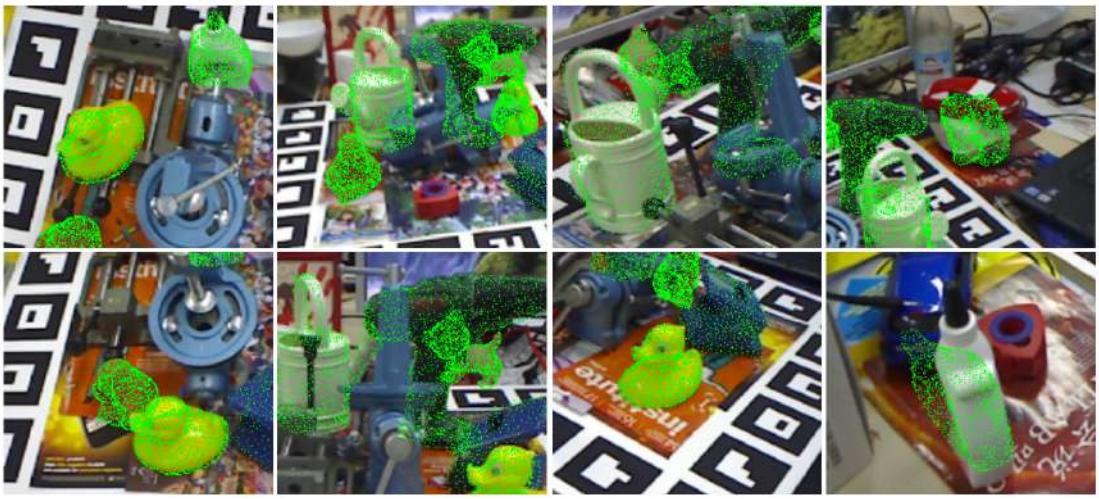

FIGURE 14 – Exemples de prédictions sur LINEMOD-Occluded

Les résultats montrent que les poses prédites sont dans l'ensemble bonnes, malgré les occlusions qui peuvent exister dans ces scènes. La dernière colonne présente des cas d'échecs qui peuvent être expliqués par des symétries subtiles ainsi que des occlusions.

Les résultats détaillés de l'étude menée sur ce modèle sont présentés en annexe. Ce modèle sera plus tard comparé aux autres modèles d'estimation de poses étudiés.

1.4.3 G2L-Net

Le modèle **G2L-Net** a été introduit dans le papier [6], présenté à la conférence CVPR 2020. Le code relatif à son implémentation est disponible à : https:

//github.com/DC1991/G2L_Net

G2L-Net vise à estimer la pose 6D d'objets présents dans une image RGB-D. Pour cela, son architecture est composée de 3 parties principales (cf figure 15) :

— un **module de localisation** globale des objets qui détecte les objets et extrait les nuages de points par détection 2D.

— un **module de localisation des translations** qui effectue une segmentation 3D et une prédiction de la translation.

— un **module de localisation des rotations** pour prédire la rotation des objets.

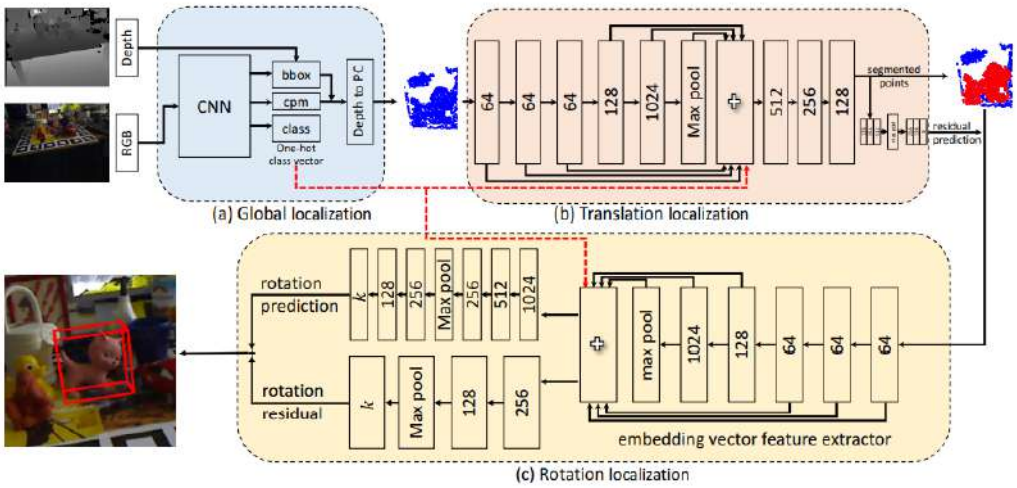

FIGURE 15 – Architecture générale du modèle G2L-Net

Le premier module de **localisation globale** des objets est composé du détecteur YOLO-v3 qui permet de prédire les bounding boxes d'objets détectés dans l'image, puis une sphère 3D est déterminée autour de chaque objet prédit. Cela permet de ne considérer que le nuage de points à l'intérieur de la sphère pour chaque objet. Le rayon de la sphère est le diamètre de l'objet détecté.

Le module de **localisation des translations** est composé de 2 réseaux PointNet entraînés à effectuer une segmentation 3D sur chacun des nuages de points des objets détectés pour ne conserver que les points appartenant aux objets. De

plus, ces réseaux renvoient la distance résiduelle entre la valeur moyenne des points segmentés appartenant à l'objet, $\tilde{T}$, et la translation de l'objet, T. Cette distance résiduelle est ensuite utilisée pour calculer la translation de l'objet.

L'architecture du module de **localisation de la rotation** est présentée sur la figure 16 :

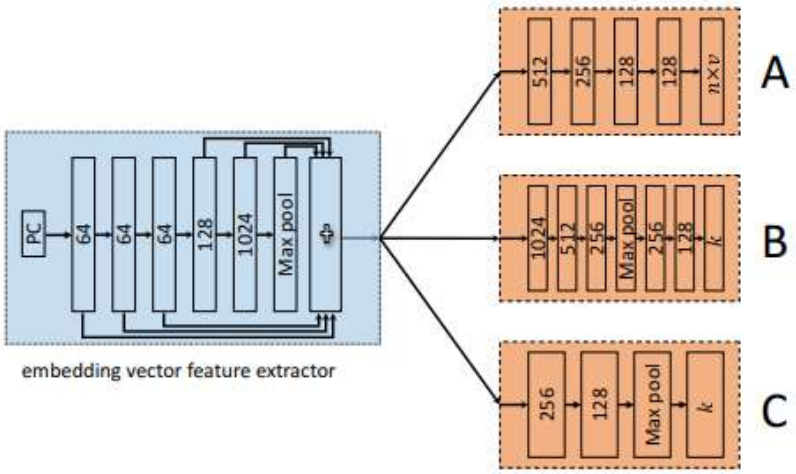

FIGURE 16 – Architecture du module de prédiction de la rotation de G2L-Net

Ce module transfère le nuage de points de chacun des objets détectés dans un espace aux coordonnées locales canoniques. Le bloc A vise à prédire, pour chaque point du nuage, un vecteur unitaire orienté dans la direction d'un keypoint défini comme l'un des coins de la bounding box 3D de l'objet détecté. Ce bloc est constitué d'un réseau MLP à 5 couches et n'est pas déployé pendant la phase d'inférence. Sa fonction de coût est définie comme étant l'erreur quadratique moyenne entre les vecteurs directionnels prédits et les vecteurs directionnels ground-truth.

Le bloc B est ensuite utilisé pour intégrer les features prédits à chaque point pour prédire la rotation de l'objet avec une fonction de coût Mean Square Error entre la rotation prédite et la rotation ground-truth. En sortie de ce bloc se trouvent les keypoints de position 3D prédits. Pendant l'inférence, la matrice de rotation est calculée depuis la position des keypoints prédits par l'algorithme de Kabsch.

Le résidu de rotation est calculé dans le bloc C, qui prend en entrée les key-

points de position 3D prédits, et estime le résidu de la rotation avec la ground-truth.

Des exemples de prédictions réalisées par G2L-Net sur LINEMOD sont présentées figure 39 :

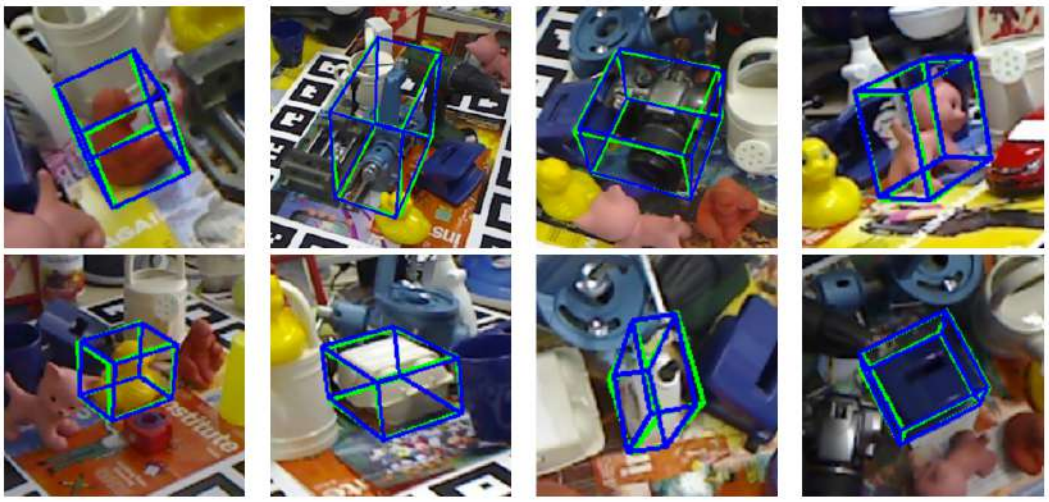

FIGURE 17 – Exemples de prédictions de G2L-Net sur LINEMOD

La pose des objets est bien prédite dans tous les cas présentés, la bounding box 3D prédite (en bleu) étant bien alignée avec la ground-truth (en vert) dans la plupart des cas.

Les résultats détaillés de l'étude menée sur le modèle G2L-Net sont présentés en annexe. Dans le paragraphe suivant, le modèle G2L-Net sera comparé aux autres modèles d'estimation de poses étudiés.

1.4.4 Comparaison des modèles d'estimation de poses

Les modèles sont maintenant comparés entre eux dans la table 4 :

Modèle	Dataset	Input	ADD(-S) (%)	FPS	Temps (ms)	GPU
FS-Net	LINEMOD	RGB-D	97.6	20	-	1080Ti
Single stage 6D	LINEMOD-Occluded	RGB	43.3 (?)	45	22	1080Ti
G2L-Net	LINEMOD	RGB-D	98.7	23	44	1080Ti

TABLE 4 – Comparaison des modèles étudiés

Pour les données RGB-D, le modèle G2L-Net obtient des performances légèrement supérieures à FS-Net sur le dataset LINEMOD selon la métrique ADD(-S). Il est également légèrement plus rapide, atteignant 23 FPS contre 20 FPS pour FS-Net.
Cependant, le problème d'estimation de pose category-level traité par le modèle FS-Net a des applications plus intéressantes qu'un modèle qui réalise uniquement de l'estimation de pose instance-level. En effet, compte tenu de la diversité d'objets susceptibles d'être présents dans les scènes du projet Chiron, encoder les features instance par instance semble trop coûteux. Il vaudrait mieux s'orienter sur des modèles qui encodent les features au niveau des classes et traitent le problème de la différence de taille/couleur des objets au sein des classes.

Pour cela, le modèle FS-Net paraît le plus intéressant à développer, d'autant plus que ses performances et sa rapidité sur un dataset d'estimation de poses instance-level sont proches de celles de G2L-Net.
Le modèle Single stage 6D est lui difficilement comparable aux deux autres, du fait de l'incertitude autour de la métrique utilisée pour l'évaluer, et du fait qu'il soit évalué sur une variante de LINEMOD. Cependant, sa rapidité très élevée (45 FPS) est intéressante, au moins comme baseline, et il pourrait donc être également intéressant à implémenter.

1.4.5 Autres modèles intéressants

— DPOD (ICCV 2019)
— MoreFusion [7] (CVPR20)
— Multi-path encoder (CVPR20)

2 Implémentation

A la suite de l'étude de l'état de l'art, j'ai choisi d'implémenter le modèle FS-Net pour les raisons évoquées dans la sous-section précédente. Pour cela, je me suis basé sur l'implémentation des auteurs, en utilisant le dataset NOCS - Real.

Les résultats se sont avérés très difficiles à reproduire. Tout d'abord, les auteurs demandent d'effectuer une phase de pré-traitement des données grâce à un

code donné dans le GitHub d'une autre méthode, Shape Prior Deformation (https://github.com/mentian/object-deformnet).
Ce code est composé de deux fichiers (*shape_data.py* et *pose_data.py*) à appliquer au dataset NOCS. Il s'est avéré que la structure du dataset NOCS tel qu'il est possible de le télécharger ne correspondait pas au code fourni dans ces deux fichiers.
Certains fichiers du dataset étaient erronnés et ne permettaient pas de faire tourner cette phase du pre-processing. D'autres incohérences dans le code bloquaient également le fonctionnement, ce qui a été long à débloquer de par la taille du dataset et la longueur du code.

Suite à cette première phase de pré-traitement du dataset NOCS, l'implémentation de FS-Net possède une phase de pré-traitement qui lui est propre. Cette phase de pré-traitement se situe dans le fichier *gen_pts.py*. Cette phase de pré-traitement a posé beaucoup de difficultés car elle ne fonctionne pas en l'état.
La structure du dataset tel qu'utilisé par le code suggère que les auteurs ont utilisé une version différente de NOCS, qui semble introuvable. Outre l'arborescence qui diffère, certains fichiers sont également manquants. Il a donc fallu passer par une compréhension exhaustive de cette phase de pré-processing afin de pouvoir déduire la structure de NOCS que les auteurs avaient utilisé et identifier les fichiers manquants.

Il s'agit d'une tâche très longue, qui aura occupé la deuxième moitié de mon stage. Elle n'est à ce jour pas terminée, mais je pense être sur le point de la finir.
La première étape a été de reconstituer l'arborescence exacte du dataset NOCS tel qu'utilisé par FS-Net, et d'identifier les fichiers manquants.
Les premiers fichiers manquants qui sont apparus sont des fichiers *.ply* qui contiennent les modèles 3D d'objets. Mes recherches ont montré que je n'étais pas le seul à m'interroger sur l'absence de ces fichiers, les auteurs n'indiquant pas qu'il fallait les générer.
Les autres fichiers manquants sont des fichiers *.txt* dont le rôle est difficile à comprendre à partir du code seul. J'ai fini par comprendre (et cela a été confirmé par la communauté qui a cherché à reproduire ces résultats) que l'absence de ces fichiers venait d'un code de pré-traitement incomplet dans FS-Net.

Après de longues recherches, il m'a été possible de générer les fichiers *.ply*. Certaines fonctions dans *gen_pts.py* semblent permettre de générer les fichiers *.txt* manquants, ce qui constitue la dernière étape avant d'avoir intégralement reconstitué le dataset utilisé. Une fois cette étape effectuée, il sera possible

d'entraîner le modèle et répliquer les résultats.

En conclusion de cette partie d'implémentation, le modèle choisi s'est avéré un mauvais choix dans la mesure où sa reproductibilité est très mauvaise. Cela m'a fait perdre beaucoup de temps et m'a empêché d'avancer pour obtenir des résultats intéressants, mais je continue à travailler dessus et espère proposer une implémentation fonctionnelle prochainement.

Références

[1] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. Blendmask : Top-down meets bottom-up for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[2] Youngwan Lee and Jongyoul Park. Centermask : Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] Rufeng Zhang, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan. Mask encoding for single shot instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin Shen, and Ales Leonardis. Fs-net : Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1581–1590, June 2021.

[5] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, and Ales Leonardis. G2l-net : Global to local network for real-time 6d pose estimation with embedding vector features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[7] Kentaro Wada, Edgar Sucar, Stephen James, Daniel Lenton, and Andrew J. Davison. Morefusion : Multi-object reasoning for 6d pose estimation from volumetric fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Annexe

Détails sur les métriques utilisées et les datasets

Métriques pour la segmentation d'instances

Cette section s'attache à décrire les métriques classiquement utilisées pour évaluer les modèles de segmentation d'instances. Une métrique importante est l'**Average Precision** . Elle est déterminée par l'aire sous la courbe $precision = f(recall)$:

$$AP = \int_0^1 f(r) d_r$$

La fonction f est parfois lissée ou interpolée pour minimiser l'impact des "rebonds" que peut présenter la courbe precision-recall, du fait de la non-monotonie de la précision au cours de l'entraînement.

Des variantes de l'Average Precision existent telles que l'AP^{50} et l'AP^{75}, qui sont l'Average Precision respectivement calculée avec un seuil d'Intersection over Union fixé à 0.5 et 0.75 pour conserver les bonnes prédictions. De même, il existe des variantes de l'AP pour la taille des objets : l'AP^{small}, l'AP^{medium} et l'AP^{large} qui sont calculées pour des objets dont l'aire dans l'image couvre respectivement moins de 32 pixels carrés, entre 32 et 96 pixels carrés, et plus de 96 pixels carrés.

Métriques pour l'estimation de poses

Une métrique couramment utilisée en estimation de poses est la métrique **ADD**, définie pour un modèle 3D M de rotation R et de translation T par rapport à la caméra. Elle est calculée par la distance moyenne des points $x \in M$ à leur version transformée :

$$\frac{1}{|M|} \sum_{x \in M} \left\| (R.x + T) - (\tilde{R}.x + \tilde{T}) \right\|$$

où $\tilde{R}$ et $\tilde{T}$ sont la rotation et translation estimées.
Pour les objets symétriques, une variante de l'ADD appelée ADD-S est souvent utilisée, où la distance moyenne est calculée par rapport au point le plus proche :

$$\frac{1}{|M|} \sum_{x_1 \in M} \min_{x_2 \in M} \left\| (R.x_1 + T) - (\tilde{R}.x_2 + \tilde{T}) \right\|$$

Pour définir une métrique de précision à partir de l'ADD, il faut également définir un seuil d'ADD au-delà duquel la pose prédite est considérée comme fausse. Ce seuil est couramment défini comme étant une fraction du diamètre de l'objet, souvent 10%.
Arguant du fait que l'utilisation d'un seuil fixe pour l'ADD ne permet pas de savoir comment le modèle performe sur les prédictions fausses par rapport à ce seuil, le papier introduisant le modèle PoseCNN propose une autre version de l'ADD, l'ADD-AUC (ADD calculée par l'aire sous la courbe).
Elle consiste à faire varier le seuil de distance utilisé pour calculer l'ADD, et à tracer la courbe seuil-précision, où la précision est calculée à partir de la métrique ADD et du seuil. La métrique ADD-AUC est ensuite obtenue en calculant l'aire sous la courbe seuil-précision.

Une autre métrique parfois utilisée est la méthode $n°m\ cm$ où chaque prédiction avec une erreur de rotation de moins de $n°$ et moins de m cm est jugée valide.

Datasets pour la segmentation d'instances

Le dataset le plus classique utilisé en détection d'objets et segmentation est le dataset **MS-COCO**, qui contient 328k images. Ses annotations contiennent notamment des bounding boxes pour 80 classes d'objets et des annotations de segmentation pixel par pixel pour 91 classes.
Le split de 2017 contient 118k images pour l'entraînement, 5k images pour la validation et 41k images pour le test. Sur la figure 18 sont présentés des exemples d'annotations de masques de segmentation contenus dans COCO.

FIGURE 18 – Exemples de masques de segmentation dans le dataset MS-COCO

Datasets pour l'estimation de poses

Le dataset **Linemod** est régulièrement utilisé en estimation de poses. Il s'agit d'un dataset RGB-D qui contient 15 séquences vidéo de 15 objets ayant peu de texture, présentés dans un environnement encombré. Chaque séquence contient plus de 1100 images avec différents points de vue sur les objets. Les images dans chaque séquence contiennent plusieurs objets, mais seul un objet possède des annotations sur sa pose 6D, sa classe et sa bounding box. L'image 19 présente des exemples de ces scènes annotées.

FIGURE 19 – Exemples de scènes annotées dans Linemod

Une amélioration du dataset nommée **Linemod - Occluded** a été proposée. Cette amélioration contient, pour une des 15 scènes (*Benchvise*), des annotations de poses de 8 autres objets au sein de la scène. Dans certaines des images de la scène, les objets sont sévèrement occlus.

Un autre dataset fréquemment utilisé en estimation de poses est le dataset **YCB Video** (YCB-V), qui contient les poses 6D de 21 objets issus du dataset YCB, observés dans 92 vidéos (133 827 images). Chacun des objets possède un modèle 3D. La figure 20 présente un exemple de scène dans YCB-V, et sa reconstitution 3D à partir des annotations de pose 6D.

FIGURE 20 – Exemples de scènes dans YCB-V (gauche) et reconstitution 3D à partir des annotations de pose (droite)

Enfin, le dataset **T-LESS** est également régulièrement utilisé. Il est composé de 30 objets dont les formes et couleurs sont assez semblables. Trois capteurs sont utilisés pour capturer les images : deux capteurs RGB-D et un capteur RGB. Le training set est composé de 38k images par capteur où les objets sont seuls devant un fond noir. Le test set est composé de 10k images par capteur qui proviennent de 20 scènes de vue d'une table du dessus où sont entreprosés les objets.
Deux modèles 3D sont disponibles pour chaque objet. La figure 21 présente des exemples d'images d'entraînement et de test du dataset T-LESS.

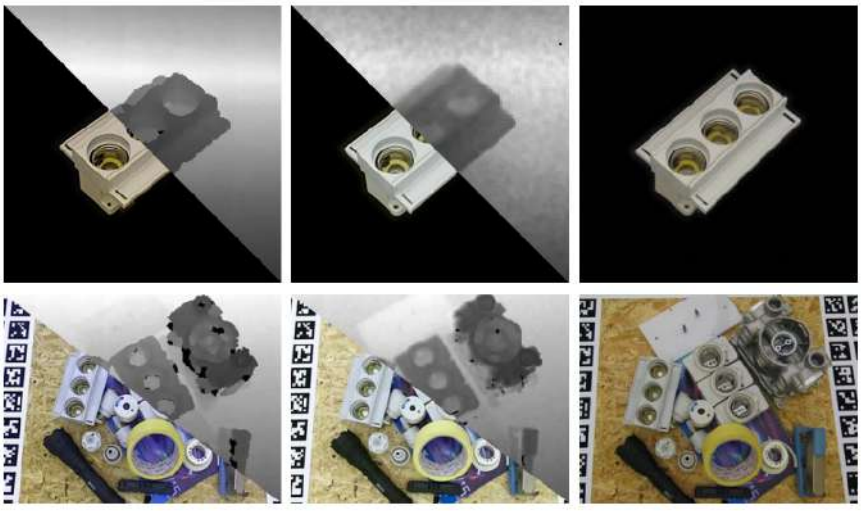

FIGURE 21 – Images du training set (haut) et du test set (bas) pour les 3 capteurs

Le dataset **NOCS** est un dataset utilisé pour les problèmes d'estimation de poses *category-level* : il s'agit de prédire la pose et l'échelle d'un objet relativement à une représentation canonique de la classe à laquelle appartient l'objet.

Ce dataset est composé de scènes de réalité "mixte" et de scènes entièrement réelles. Dans les scènes mixtes, des objets synthétiques sont disposés au sein d'une scène réelle de manière à respecter le contexte de la scène. Les scènes entièrement réelles sont souvent regroupées sous l'appelation NOCS - REAL. Ces scènes contiennent plus de 5 objets pour simuler des conditions réelles d'obstructions.
6 catégories d'objets sont présentes dans le dataset : bouteille, bol, caméra, canette, PC, et tasse. NOCS - REAL est composé de 8k images RGB-D issues de 18 scènes différentes. Dans le training et le test set de NOCS - REAL, 3 instances d'objets sont présentes pour chaque catégorie. Chaque instance présente un maillage 3D obtenu en utilisant un algorithme de reconstruction 3D appliqué aux images RGB-D.

La figure 22 présente des exemples de scènes du dataset NOCS-Real.

FIGURE 22 – Exemples de scènes dans NOCS Real

Détails des résultats des modèles de segmentation d'instances

BlendMask

Le modèle BlendMask est étudié sur le dataset MS-COCO et est comparé à d'autres modèles classiques tels que Mask R-CNN ou TensorMask. Les mesures de temps d'inférence sont réalisées sur un GPU 1080Ti. Les résultats sont présentés figure 23 :

Method	Backbone	Epochs	Aug.	Time (ms)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [12]	R-50	12		97.0	34.6	56.5	36.6	15.4	36.3	49.7
Mask R-CNN*		72	✓	97+	36.8	**59.2**	39.3	17.1	38.7	52.1
TensorMask [7]		72	✓	400+	35.5	57.3	37.4	16.6	37.0	49.1
BlendMask		12		**78.5**	34.3	55.4	36.6	14.9	36.4	48.9
BlendMask		36	✓	**78.5**	**37.0**	58.9	**39.7**	**17.3**	**39.4**	**52.5**
Mask R-CNN	R-101	12		118.1	36.2	58.6	38.4	16.4	38.4	52.1
Mask R-CNN*		36	✓	118+	38.3	61.2	40.8	18.2	40.6	54.1
TensorMask		72	✓	400+	37.3	59.5	39.5	17.5	39.3	51.6
SOLO [24]		72	✓	-	37.8	59.5	40.4	16.4	40.6	54.2
+deform convs [24]		72	✓	-	40.4	62.7	43.3	17.6	43.3	58.9
BlendMask		36	✓	101.8	38.4	60.7	41.3	18.2	41.5	53.3
BlendMask*		36	✓	**94.1**	39.6	61.6	42.6	22.4	42.2	51.4
+deform convs (interval = 3)		60	✓	105.0	**41.3**	**63.1**	**44.6**	**22.7**	**44.1**	**54.5**

FIGURE 23 – Résultats comparatifs de BlendMask sur MS-COCO

L'étude réalisée montre que BlendMask obtient de meilleurs performances que Mask R-CNN et TensorMask avec les backbones ResNet-50 et ResNet-101. De plus, le temps d'inférence reporté pour BlendMask avec ces deux backbones en fait le modèle le plus rapide des modèles comparés.
Cependant, ces temps d'inférence restent élevés et ne permettent pas d'envisager des applications en temps réel de ce modèle. En effet, avec le backbone ResNet-101, cela correspond à une rapidité reportée sur le GitHub de 11 FPS pour BlendMask* (implémenté sur Detectron2) et de 10 FPS pour la meilleure version (celle avec les convolutions déformables).

Une version allégée du modèle appelée BlendMask-RT est donc également proposée afin d'obtenir un compromis performances-rapidité qui permette des applications en temps réel. Cette version est comparée à un autre modèle temps-réel de référence, YOLACT, sur le dataset MS-COCO . Les temps d'inférence mesurés sont obtenus avec un GPU 1080Ti. La figure 24 présente les résultats de ce modèle temps-réel :

Method	Backbone	NMS	Resolution	Time (ms)	AP^{bb}	AP	AP_{50}	AP_{75}
YOLACT	R-101	Fast	550 × 550	**34.2**	32.5	29.8	48.3	31.3
YOLACT		Fast	700 × 700	46.7	33.4	30.9	49.8	32.5
BlendMask-RT		Batched	550 × ∗	47.6	**41.6**	**36.8**	**61.2**	**42.4**
Mask R-CNN	R-50	Batched	550 × ∗	63.4	39.1	**35.3**	**56.5**	**37.6**
BlendMask-RT				**36.0**	**39.3**	35.1	55.5	37.1

FIGURE 24 – Résultats comparatifs de BlendMask - real time sur MS-COCO

Sur le backbone ResNet-101, BlendMask-RT présente de meilleures performances que YOLACT. YOLACT est présenté comme plus rapide, mais il utilise un algorithme de Fast-Non Maximum Suppression qui sacrifie les performances pour la vitesse (12ms d'accélération rapportées par les auteurs grâce à cet algorithme). Les auteurs ont utilisé une batched-NMS pour BlendMask-RT, plus performante mais moins rapide. En prenant en compte cette différence, les temps d'inférence de BlendMask-RT et YOLACT sur MS-COCO avec un backbone ResNet-101 sont du même ordre : environ 47ms par image.
Avec le backbone ResNet-50, BlendMask-RT atteint un temps d'inférence de 36 ms tout en conservant de meilleures performances que YOLACT muni du ResNet-101, et sans utiliser de Fast NMS. Cela prouve l'efficacité de BlendMask-RT.

Une étude qualitative montre également que les masques de BlendMask sont plus précis que ceux produits par Mask R-CNN. Un exemple est présenté figure 25

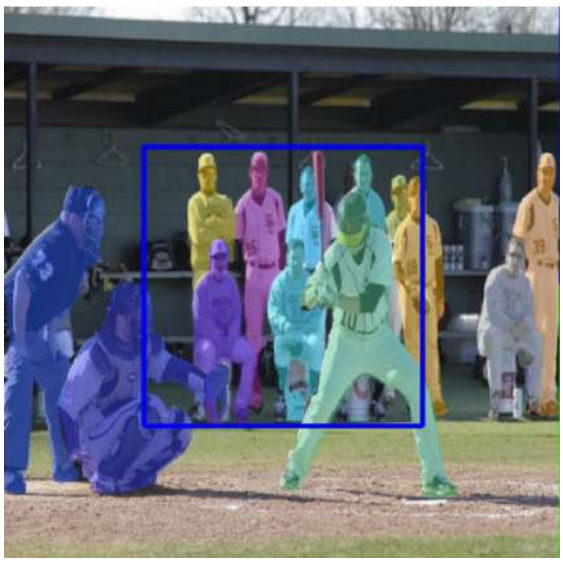

FIGURE 25 – Prédiction de masques par BlendMask sur MS-COCO

Une application en temps réel de BlendMask sur des scènes de films est présentée ici, et témoigne de la capacité du modèle à être efficace en temps réel en produisant des masques précis. De plus, cette vidéo montre que BlendMask est effectivement capable d'effectuer une segmentation d'instances en identifiant les objets.

CenterMask

Le modèle CenterMask est évalué sur le dataset MS-COCO, en utilisant un GPU Titan Xp avec CUDA 10.0. Le modèle est dans un premier temps comparé à d'autres méthodes classiques présentant de bonnes performances mais de bas FPS : Mask R-CNN, TensorMask et RetinaMask. La figure 26 présente ces résultats :

Method	Backbone	epochs	AP^{mask}	AP^{mask}_S	AP^{mask}_M	AP^{mask}_L	AP^{box}	AP^{box}_S	AP^{box}_M	AP^{box}_L	Time	FPS	GPU
Mask R-CNN, *ours*	R-101-FPN	24	37.9	**18.1**	40.3	53.3	42.2	24.9	45.2	52.7	94	10.6	V100
ShapeMask [17]	R-101-FPN	N/A	37.4	16.1	40.1	53.8	42.0	24.3	45.2	53.1	125	8.0	V100
TensorMask [5]	R-101-FPN	72	37.1	17.4	39.1	51.6	-	-	-	-	380	2.6	V100
RetinaMask [7]	R-101-FPN	24	34.7	14.3	36.7	50.5	41.4	23.0	44.5	53.0	98	10.2	V100
CenterMask	R-101-FPN	24	**38.3**	17.7	**40.8**	**54.5**	**43.1**	**25.2**	**46.1**	**54.4**	**72**	**13.9**	V100
CenterMask*	R-101-FPN	36	39.8	21.7	42.5	52.0	44.0	25.8	46.8	54.9	66	15.2	V100
Mask R-CNN, *ours*	X-101-FPN	36	39.3	19.8	41.4	55.0	44.1	27.0	46.7	54.6	165	6.1	V100
CenterMask	X-101-FPN	36	39.6	19.7	42.0	55.2	44.6	27.1	47.2	55.2	123	8.1	V100
CenterMask	V-99-FPN	36	**40.6**	**20.1**	**42.8**	**57.0**	**45.8**	**27.8**	**48.3**	**57.6**	**84**	**11.9**	V100
CenterMask*	V-99-FPN	36	41.8	24.4	44.4	54.3	46.5	28.7	48.9	57.2	77	12.9	V100
YOLACT-400 [1]	R-101-FPN	48	24.9	5.0	25.3	**45.0**	28.4	10.7	28.9	**43.1**	22	45.5	Xp
CenterMask-Lite	M-v2-FPN	48	**26.7**	**9.0**	**27.0**	40.9	**30.2**	**14.2**	**31.9**	40.9	**20**	**50.0**	Xp
YOLACT-550 [1]	R-50-FPN	48	28.2	9.2	29.3	44.8	30.3	14.0	31.2	43.0	23	43.5	Xp
CenterMask-Lite	V-19-FPN	48	**32.4**	**13.6**	**33.8**	**47.2**	**35.9**	**19.6**	**38.0**	**45.9**	**23**	**43.5**	Xp
YOLACT-550 [1]	R-101-FPN	48	29.8	9.9	31.3	47.7	31.0	14.4	31.8	43.7	30	33.3	Xp
YOLACT-700 [1]	R-101-FPN	48	31.2	12.1	33.3	47.1	33.7	16.8	35.6	45.7	42	23.8	Xp
CenterMask-Lite	R-50-FPN	48	32.9	12.9	34.7	48.7	36.7	18.7	39.4	48.2	29	34.5	Xp
CenterMask-Lite	V-39-FPN	48	**36.3**	**15.6**	**38.1**	**53.1**	**40.7**	**22.4**	**43.2**	**53.5**	**28**	**35.7**	Xp

FIGURE 26 – Résultats comparatifs des modèles CenterMask et CenterMask-Lite sur MS-COCO

Il est possible d'observer qu'avec le même backbone (ResNet-101 + FPN), CenterMask obtient de meilleures performances que les autres modèles en étant plus rapide (13.9 fps). L'implémentation reproduite sur l'outil Detectron2 (CenterMask*) propose même des performances légèrement supérieures, et une inférence légèrement plus rapide (15.2 FPS). Cependant cette rapidité ne permet pas de faire de segmentation en temps réel.

CenterMask-Lite est donc ensuite comparé à la meilleure méthode de segmentation en temps réel à l'époque de la sortie du papier, YOLACT (muni du backbone ResNet-101 + FPN). CenterMask-Lite est utilisé avec différents backbones dont MobileNetv2 et VoVNetV2, suivis d'un FPN. Les résultats (figure 26) montrent que CenterMask-Lite obtient de meilleurs performances que YOLACT tout en étant plus rapide. Avec le backbone MobileNetv2, CenterMask-Lite atteint une rapidité de 50 FPS, bien que ses performances soient nettement inférieures à celles de CenterMask.
Le meilleur compromis est réalisé avec le backbone proposé dans le papier, VoVNetv2, qui permet des performances qui s'approchent de celles de TensorMask

et ShapeMask tout en restant inférieures, à une vitesse de 35.7 FPS.

CenterMask-Lite peut donc obtenir de bonnes performances en temps réel (>30 fps), ce qui est intéressant pour le projet Chiron. La figure 27 récapitule le rapport performances/rapidité des différents modèles sur le dataset MS-COCO, pour un GPU Titan Xp avec CUDA 10.0.

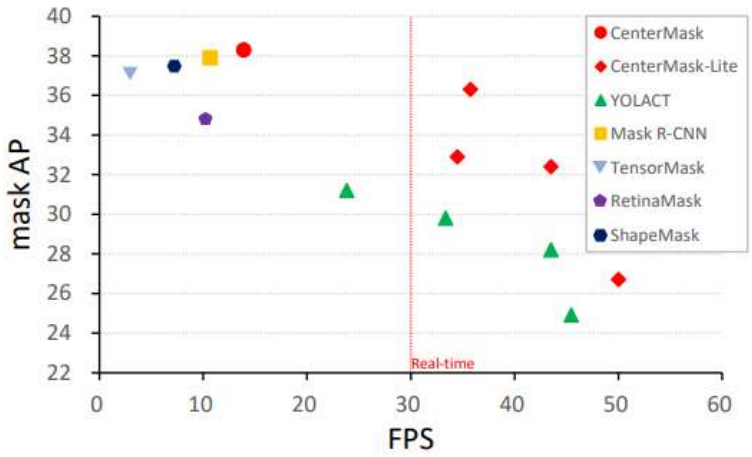

FIGURE 27 – Comparaison performances-rapidité pour CenterMask et CenterMask-Lite sur MS-COCO

Enfin, la qualité des masques produits par CenterMask paraît bonne sur les exemples fournis dans le papier. Le modèle semble capable de bien segmenter les objets même lorsqu'ils sont présents en grand nombre dans la scène, ainsi qu'il est possible de le voir figure 28.

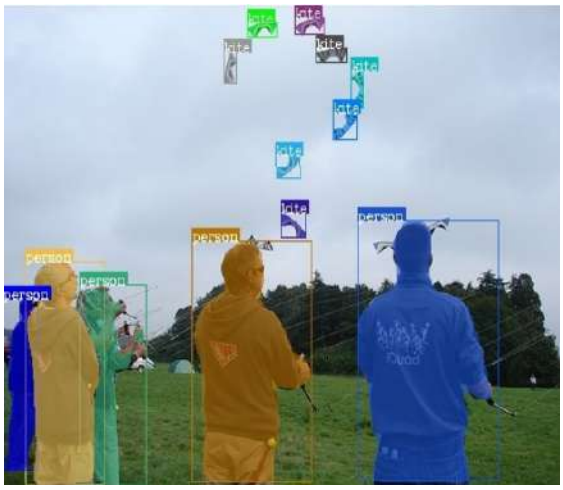

FIGURE 28 – Exemple de masques produits par CenterMask sur MS-COCO

MEInst

Le modèle MEInst est comparé avec d'autres modèles one-stage et two-stage sur le dataset MS-COCO. Les temps d'inférence sont mesurés sur un GPU GTX 1080Ti. Les résultats sont reportés figure 29 :

Method	Backbone	epochs	aug.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage									
MNC [9]	ResNet-101-C4	12	–	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [17]	ResNet-101-C5-dilated	12	–	29.2	49.5	–	7.1	31.3	50.0
Mask R-CNN [13]	ResNeXt-101-FPN	12	–	37.1	60.0	39.4	16.9	39.9	**53.5**
One-stage									
ExtremeNet [35]	Hourglass-104	100	✓	18.9	44.5	13.7	10.4	20.4	28.3
TensorMask [7]	ResNet-101-FPN	72	✓	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT [3]	ResNet-101-FPN	48	✓	31.2	50.6	32.8	12.1	33.3	47.1
PolarMask [32]	ResNet-101-FPN	12	–	30.4	51.9	31.0	13.4	32.4	42.8
PolarMask [32]	ResNeXt-101-FPN	12	–	32.9	55.4	33.8	15.5	35.1	46.3
MEInst	ResNet-101-FPN	12	–	33.0	56.4	34.0	15.2	35.3	46.3
MEInst	ResNeXt-101-FPN	12	–	35.5	59.7	36.7	17.5	38.0	49.0
MEInst	ResNet-101-FPN-DCN	12	–	34.9	58.8	36.0	16.3	37.0	49.6
MEInst	ResNeXt-101-FPN-DCN	12	–	36.8	61.6	38.4	18.1	39.2	51.8
MEInst	ResNet-101-FPN	36	✓	33.9	56.2	35.4	19.8	36.1	42.3
MEInst	ResNeXt-101-FPN	36	✓	36.9	60.5	38.9	21.8	39.0	46.7
MEInst	ResNeXt-101-FPN-DCN	36	✓	**38.2**	**61.7**	**40.4**	**22.6**	**40.0**	49.3

FIGURE 29 – Résultats comparatifs du modèle MEInst sur MS-COCO

Ces résultats montrent qu'avec le même backbone (ResNet-101), MEInst obtient de meilleures performances que toutes les architectures one-stage avec lesquelles le modèle est comparé, à l'exception de TensorMask. Cependant, TensorMask

possède un temps d'inférence important et un coût de calcul en entraînement également important, ce n'est pas un modèle real-time.
Bien que le temps d'inférence ou les FPS du modèle ne soient pas indiquées sur ce tableau, les résultats de rapidité présentés ci-dessous (et sur le GitHub) tendent à indiquer que MEInst avec le backbone ResNet-101 pourrait tourner aux alentours de 10 FPS. Ceci serait bien supérieur aux 2.6 FPS reportés pour TensorMask dans l'étude de CenterMask, figure 26.
De plus, avec le même backbone ResNext-101, MEInst obtient des performances s'approchant de celles de Mask R-CNN. Les performances reportées pour MEInst sont meilleures que pour les deux autres méthodes two-stage comparées (MNC et FCIS). Cela permettant ainsi de se rapprocher de l'objectif de combler l'écart entre modèles one-stage et modèles two-stage.

La figure 30 présente les résultats de MEInst avec le backbone ResNet-50 sur le dataset MS-COCO ainsi que sa rapidité pour différentes résolutions en entrée du modèle. Les mesures sont réalisées sur un GPU GTX 1080Ti.

Scale	Method	AP	AP_{50}	AP_{75}	FPS
416	ESE-Seg [33]	21.6	48.7	22.4	**38.5**
400	MEInst	23.9	42.4	24.1	28.2
600	MEInst	28.4	49.3	28.8	18.5
800	MEInst	**30.3**	**53.0**	**31.1**	12.8

FIGURE 30 – Etude de la rapidité de MEInst sur MS-COCO avec le backbone ResNet-50 sur un GPU GTX 1080Ti

Avec une résolution 400×400, MEInst tourne à 28.2 fps, permettant d'envisager des applications real-time. Cependant, il faut noter que ses performances sont significativement moins bonnes que celles présentées figure 29. La résolution 600×600 réalise un compromis entre performances et rapidité, en présentant des performances qui s'approchent des meilleures performances de MEInst présentées figure 29, tout en tournant à 18.5 FPS, ce qui paraît faible mais pas incompatible pour des applications en temps réel.

Les masques produits par MEInst paraissent de bonne qualité et adaptés à la présence de nombreuses instances dans l'image, bien que sur certains exemples proposés, les contours semblent un peu grossiers, en particulier pour les objets écharpés. Des exemples sur MS-COCO sont présentés figure 29.

FIGURE 31 – Exemples de masques produits par MEInst sur MS-COCO

Détails des résultats des modèles d'estimation de poses

FS-Net

FS-Net est dans un premier temps comparé aux autres modèles d'estimation de poses *instance level* de l'état de l'art sur le dataset LINEMOD. Les temps d'inférence mesurés sont réalisés sur un GPU 1080Ti. Les résultats sont présentés figure 32 pour la précision dérivée de la métrique ADD-S (le seuil utilisé n'est pas précisé) :

Method	Input	ADD-(S)	Speed(FPS)
PVNet [24]	RGB	86.3%	25
CDPN [17]	RGB	89.9%	33
DPOD [44]	RGB	95.2%	33
G2L-Net [6]	RGBD	98.7%	23
Densefusion[40]	RGBD	94.3%	16
PVN3D [10]	RGBD	99.4%	5
Ours	RGBD	97.6%	20

FIGURE 32 – Résultats comparatifs de FS-Net en estimation de poses instance-level sur LINEMOD

Ces résultats montrent que le modèle proposé offre un bon compromis performances/rapidité par rapport aux méthodes de l'état de l'art. La méthode tourne en 20 fps, ce qui pourrait permettre des applications en temps réel pour le projet Chiron.
Il est à noter que le modèle est comparé avec des modèles d'estimation de poses sur des problèmes d'*instance level*. FS-Net est lui un modèle d'estimation de poses pour un problème de *category level*. Ainsi, ses bonnes performances par rapport à l'état de l'art sur un problème instance level montrent que le

modèle est capable d'extraire des features à la fois au niveau des instances et des catégories.

FS-Net est également comparé à d'autres modèles de l'état de l'art sur la tâche d'estimation de pose category-level. Pour cela, le dataset NOCS-REAL est utilisé, ainsi que la métrique $IoU_X, X \in \{25, 50, 75\}$ qui est la précision à partir de l'IoU pour de la détection d'objets 3D avec différents seuils. Les résultats sont présentés figure 33 :

Method	IoU_{25}	IoU_{50}	IoU_{75}	5°5cm	10°5 cm	10°10 cm	Speed(FPS)
NOCS [41]	84.9%	80.5%	30.1%(69.5%)	9.5 %(40.9%)	26.7%	26.7%	5
CASS [4]	84.2%	77.7%	-	23.5 %	58.0%	58.3%	-
Shape-Prior [35]	83.4%	77.3%	53.2%(83.1%)	21.4%(59.0%)	54.1%	-	4
6-PACK [39]	94.2%	-	-	**33.3 %**	-	-	10
Ours	**95.1%**	**92.2%**	**63.5%**(85.17%)	28.2 %(62.01%)	**60.8%**	**64.6%**	**20**

FIGURE 33 – Résultats comparatifs de FS-Net en estimation de poses category-level sur NOCS-REAL

FS-Net obtient les meilleurs résultats sur la plupart des métriques bien que 6-Pack ait une meilleure précision sur la métrique 5°5 *cm*. Cependant, FS-Net est au moins deux fois plus rapide que les autres modèles en tournant à 20 FPS, ce qui permet d'envisager des applications en temps réel.

Enfin, des exemples de prédictions réalisées par FS-Net sur NOCS-REAL sont présentés figure 34 :

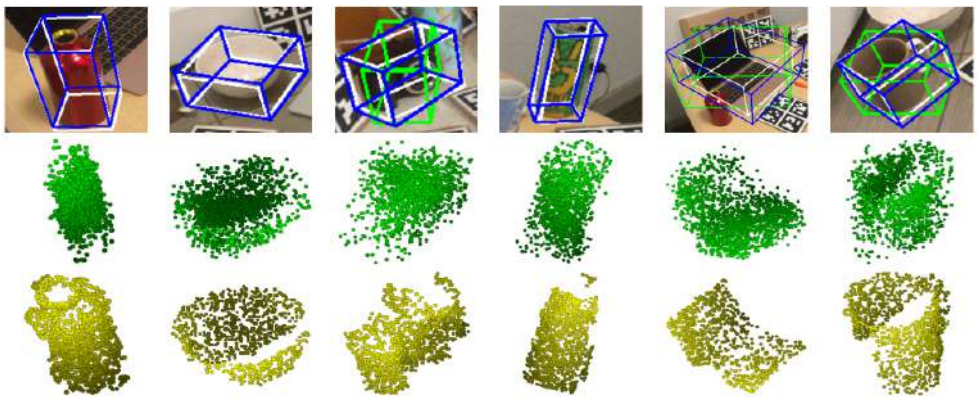

FIGURE 34 – Exemples de prédictions de FS-Net sur NOCS-REAL

La pose et la taille des objets sont bien prédites dans la plupart des cas, et la

bounding box 3D prédite (en bleu) est bien alignée avec la ground-truth (en blanc).

Single-stage 6D pose estimation

Le modèle est notamment comparé à d'autres modèles de l'état de l'art basés sur de l'extraction de correspondances 2D-3D, SegDriven et PVNet, sur le dataset LINEMOD-Occluded.

Les métriques utilisées ne sont pas très claires : il est dit que l'ADD est utilisée avec un seuil à 10% du diamètre du modèle pour déterminer si une prédiction est juste ou pas. Cependant, les résultats reportés figure 35 sont très faibles par rapport à ceux présentés pour les autres méthodes avec cette métrique de précision.
Une autre métrique utilisée est l'erreur de reprojection 2D des points du modèle 3D, dénotée REP. Cette métrique s'accompagne également d'un seuil fixé à 5 pixels pour déterminer si une prédiction est juste ou pas, les résultats présentent donc la précision qui dérive de cette métrique.

Les résultats sur LINEMOD-Occluded sont présentés sur la figure 35 :

	ADD-0.1d				REP-5px			
	PoseCNN	SegDriven	PVNet	**Ours**	PoseCNN	SegDriven	PVNet	**Ours**
Ape	9.6	12.1	15.8	**19.2**	34.6	59.1	69.1	**70.3**
Can	45.2	39.9	63.3	**65.1**	15.1	59.8	**86.1**	85.2
Cat	0.9	8.2	16.7	**18.9**	10.4	46.9	65.1	**67.2**
Driller	41.4	45.2	65.7	**69.0**	7.4	59.0	**73.1**	71.8
Duck	19.6	17.2	25.2	**25.3**	31.8	42.6	61.4	**63.6**
Eggbox*	22.0	22.1	50.2	**52.0**	1.9	11.9	8.4	**12.7**
Glue*	38.5	35.8	49.6	**51.4**	13.8	16.5	55.4	**56.5**
Holepun.	22.1	36.0	39.7	**45.6**	23.1	63.6	69.8	**71.0**
Average	24.9	27.0	40.8	**43.3**	17.2	44.9	61.1	**62.3**

FIGURE 35 – Résultats comparatifs du modèle proposé sur LINEMOD-Occluded

Ces résultats montrent que pour la métrique ADD-0.1d, le modèle proposé est plus performant que les modèles de l'état de l'art pour tous les objets du dataset. Il améliore de 3.5 % (s'il s'agit bien de la précision dérivée de l'ADD) les performances de PVNet. Sous la métrique d'erreur de reprojection, REP, le modèle proposé performe mieux que les modèles auquel il est comparé sur la

plupart des objets.
Néanmoins, ces résultats sont difficilement comparables avec ceux des autres modèles étudiés dans notre état de l'art du fait du flou qui entoure la métrique utilisée.

La rapidité du modèle est également étudiée sur le dataset LINEMOD-Occluded sur un GPU GTX 1080Ti. L'extraction de correspondances est réalisée par la même architecture que celle de PVNet (figure **??**). Les résultats sont présentés sur la figure 36, en millisecondes :

	correspondence extraction	fusion	total time	FPS
PoseCNN	-	-	>250	<4
SegDriven	30	20	50	20
PVNet	**14**	26	40	25
Ours	**14**	**8**	**22**	**45**

FIGURE 36 – Comparaison de la rapidité (en ms) du modèle proposé sur LINEMOD-Occluded avec un GPU GTX 1080Ti

Ces résultats montrent que le modèle est beaucoup plus rapide que tous les autres modèles avec lesquels il est comparé. Cette rapidité s'explique par le fait que le modèle ne nécessite pas d'étape de fusion basée sur l'algorithme RANSAC, et est donc beaucoup plus rapide que pour les autres méthodes proposées. Le modèle proposé tourne à 45 FPS tandis que PVNet tourne à 25 FPS. Cette rapidité est très importante et permet des applications en temps réel pour le projet Chiron.

Enfin, des exemples de prédictions de poses 6D sur LINEMOD-Occluded sont présentés figure 37. La pose est visualisée comme la reprojection 2D du maillage 3D prédit pour chaque objet :

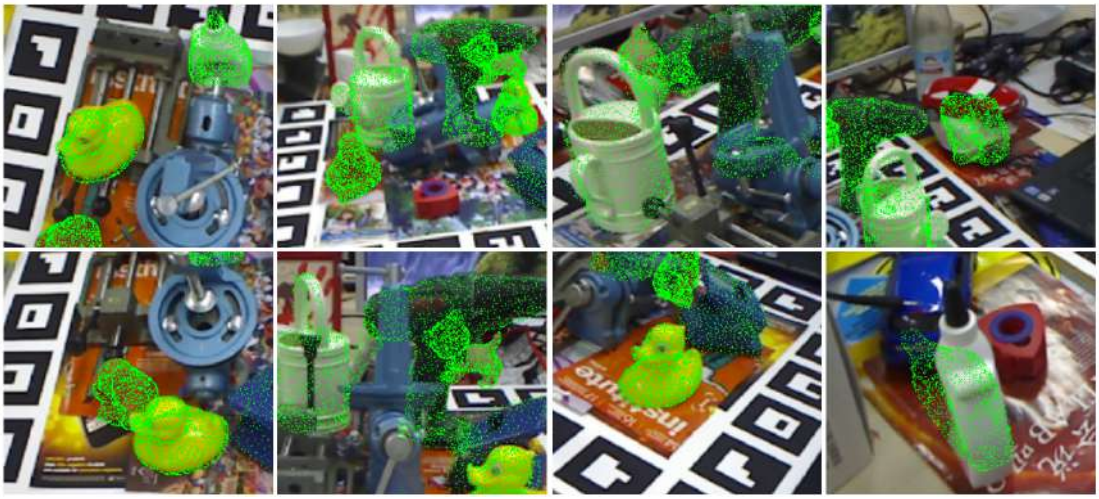

FIGURE 37 – Exemples de prédictions sur LINEMOD-Occluded

Les résultats montrent que les poses prédites sont dans l'ensemble bonnes, malgré les occlusions qui peuvent exister dans ces scènes. La dernière colonne présente des cas d'échecs qui peuvent être expliqués par des symétries subtiles ainsi que des occlusions.

G2L-Net

G2L-Net est comparé à d'autres modèles de l'état de l'art utilisant des données RGB et RGB-D sur le dataset LINEMOD. La métrique utilisée est la précision calculée grâce à la métrique ADD (ADD-S pour les objets symétriques tels que l'oeuf). Les mesures de temps d'inférence sont réalisées sur un GPU GTX 1080Ti. Les résultats sont présentés figure 38 :

Method	PVNet [28]	PoseCNN + DeepIM [42, 19]	DPOD [45]	Frustum-P [29]	Hinterstoisser [12]	DenseFusion [40]	Ours
Input	RGB	RGB	RGB	RGB+Depth	Depth	RGB+Depth	RGB+Depth
Refinement	×	✓	✓(×)	×	✓	✓(×)	×
Ape	43.6%	77.0%	87.7% (53.3%)	85.5%	**98.5%**	92.3% (79.5%)	96.8%
Bench Vise	**99.9%**	97.5%	98.5% (95.3%)	93.2%	99.0 %	93.2%(84.2%)	96.1%
Camera	86.9%	93.5	96.0% (90.4%)	90.0%	**99.3%**	94.4%(76.5%)	98.2%
Can	95.5%	96.5%	99.7% (94.1%)	91.4%	**98.7%**	93.1%(86.6%)	98.0%
Cat	79.3%	82.1%	94.7% (60.4%)	96.5%	**99.9%**	96.5%(88.8%)	99.2%
Driller	96.4%	95.0%	98.8% (97.7%)	96.8%	93.4%	87.0%(77.7%)	**99.8%**
Duck	52.6%	77.7%	86.3% (66.0%)	82.9%	**98.2%**	92.3%(76.3%)	97.7%
Egg Box	99.2%	97.1%	99.9% (99.7%)	99.9%	98.8%	99.8%(99.9%)	**100%**
Glue	95.7%	99.4%	96.8% (93.8%)	99.2%	75.4%	**100%** (99.4%)	**100%**
Hole Puncher	81.9%	52.8%	86.9% (65.8%)	92.2%	98.1%	92.1%(79.0%)	**99.0%**
Iron	98.9%	98.3%	**100%** (99.8%)	93.7%	98.3%	97.0% (92.1%)	99.3%
Lamp	99.3%	97.5%	96.8% (88.1%)	98.2%	96.0%	95.3%(92.3%)	**99.5%**
Phone	92.4%	87.7%	94.7% (74.2%)	94.2%	98.6%	92.8%(88.0%)	**98.9%**
Speed(FPS)	25	5	33(40)	12	8	16(20)	23
Average	86.3%	88.6%	95.2% (83.0%)	93.4%	96.3 %	94.3 %(86.2%)	**98.7 %**

FIGURE 38 – Résultats comparatifs de G2L-Net sur LINEMOD

Ces résultats montrent que G2L-Net possède les meilleures performances en général, ayant la précision moyenne la plus élevée avec un gain substantiel par rapport aux autres méthodes. Parmi les modèles qui utilisent des données RGB-D en entrée, G2L-Net est le modèle le plus rapide en tournant à 23 FPS. DPOD et PVNet sont cependant plus rapides, mais utilisent en entrée uniquement des données RGB. En revanche, les performances de PVNet sont beaucoup moins bonnes que celles de G2L-Net, tandis que celles de DPOD avec une phase de refinement sont un peu moins bonnes que celles de G2L-Net.
La rapidité de 23 FPS avec les meilleures performances de l'état de l'art permettent cependant d'envisager des applications en temps réel, bien qu'une application nécessitant un plus grand nombre de FPS pourrait mener à choisir DPOD en dépit de performances un peu moins bonnes.

Enfin, des exemples de prédictions réalisées par G2L-Net sur LINEMOD sont présentées figure 39 :

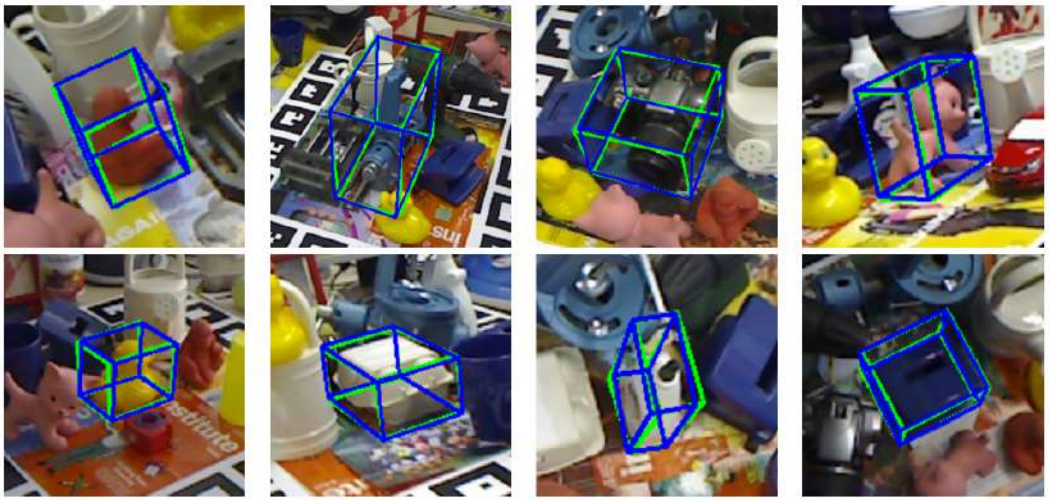

FIGURE 39 – Exemples de prédictions de G2L-Net sur LINEMOD

La pose des objets est bien prédite dans tous les cas présentés, la bounding box 3D prédite (en bleu) étant bien alignée avec la ground-truth (en vert) dans la plupart des cas.

LIRIS
36, Avenue Guy de Collongue
UMR 5205
69134 Écully