UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

L'ESTIMATION DE POSE 6DOF POUR LES OBJETS INDUSTRIELS COMPLEXES DANS LE CADRE DE *BIN PICKING*

SCA730 : Activités de recherche et mémoire
Mémoire de maîtrise
En génie électrique

Supervisé par : Wael Suleiman
Réalisé par : Hajar El khamlichi

Sherbrooke, Québec, Canada
Mai 2023

MEMBRES DU JURY

Wael Suleiman

Directeur

Cao MinhTa

Rapporteur du mémoire

Codirecteur

Claudia Esteves

Évaluatrice

Évaluateur

REMERCIEMENTS

Ce travail de recherche est le fruit de plusieurs mois d'étude, d'analyse et de réalisation. Mes sincères remerciements s'adressent en premier lieu à mon **CRÉATEUR** qui m'a comblé de ses bienfaits. J'exprime ma gratitude à son don d'existence, de santé, d'analyse et de raison ainsi qu'à son précieux soutient tout au long de cette maîtrise.

Je remercie tout d'abord mon directeur de recherche Mr. **Wael Suleiman** de ses judicieux conseils qui ont alimenté ma réflexion. De même que son aide pour que ce travail se déroule dans les meilleures conditions.

J'exprime ma reconnaissance à tous mes professeurs de l'université de Sherbrooke ainsi qu'aux étudiants avec qui j'ai vécu de bons moments de partage de connaissances.

Je remercie aussi L'*Alliance de recherche numérique du Canada* de mettre à notre disposition des équipements technologiques de pointe afin de bonifier et de pousser plus loin notre recherche.

Finalement, j'aimerai bien passer un spécial remerciement à ma précieuse famille. Évidemment, mes parents Mr. **Mohammad El Khamlichi** et Mme. **Aicha El Motamid** et mon mari Mr. **Taoufik El Alj** pour leurs patience, accompagnement, soutien et encouragement tout au long de mes études. Je remercie également mes enfants pour leurs patience, motivation et pour tout beau moment de partage de connaissances.

RÉSUMÉ

Dans le domaine d'assemblage industriel, ce travail vise à apporter de la vision artificielle avancée à l'assemblage de la pièce à l'étude qui est considérée de nature complexe. Dans le cadre d'un problème de *Bin Picking*, où plusieurs pièces sont déposées de façon aléatoire dans un contenant, ce travail consiste à trouver une solution au bras robotisé pour détecter et estimer la 6DoF. En l'occurrence, une tentative d'essai avec PoseCNN a déjà été testée auparavant, mais cet algorithme n'a réussi à détecter aucune instance dans une scène réelle. Pour mener à bien ce projet, le présent travail vise à étudier plus en détails les raisons de ce problème et à proposer une solution de détection et d'estimation de pose 6DoF qui convient le mieux à la pièce à l'étude étant un objet industriel complexe. Évidemment, l'estimation de pose 6DoF dépend grandement de la détection d'objets. Les approches d'estimation de la 6DoF de l'état de l'art sont fortement basées sur les images RGB qui impliquent une détection 2D, ce qui néglige certains détails géométriques de l'objet. D'autre côté, certaines d'entre elles sont complétement dépendantes des nuages de points, ce qui impose une détection négligeant la couleur et les caractéristiques 2D de l'image RGB. Donc, l'idée est d'avoir une combinaison qui tire parti de la détection 2D et 3D pour extraire le plus de caractéristiques déterminantes possibles. Dans ce contexte, PVN3D fusionne les caractéristiques 2D et 3D et ajoute un réseau de neurones profond de vote 3D pour déterminer les *keypoints* 3D à la manière de VoteNet lancé par Facebook. De ce fait, la contribution de ce travail est d'intégrer la base de données T-LESS d'objets industriels à PVN3D. Comme résultat final, en exécutant PVN3D, l'objet T-LESS est bien reconnu et ses 8 *keypoints* ainsi que son centre sont à l'emplacement exacte. Pour conclure, cette perception avancée du bras robotisé permettra aux entreprises d'accélérer leurs assemblages de pièces et de multiplier ainsi leur productivité.

TABLE DES MATIÈRES

LISTE ES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

2D, 3D: 2 Dimensions, 3 Dimensions
6DoF: Six Degrees of Freedom
BD: Base de Données
CAD : Conception Assistée par Ordinateur (Computer Aided Design)
CNN: Réseau Neuronal Convolutif (Convolutional Neural Network)
COCO: Common Objects in Context
FPS : Image par Seconde (Frames Per Second)
GAN : Réseau Adverse génératif (Generative Adversarial Networks)
HPC: High-Performance Computing
ICP: Iterative Closest Point
LM : LineMOD
PnP: Perspective-n-Point
PPF: Point Pair Feature
RANSAC: RANdom SAmple Consensus
RGB: Red Green Blue
RGB-D: Red Green Blue-Depth
RNA: Réseau de Neurones Artificiels
RNAP : Réseau de Neurones Artificiels Profond
ROI: Région d'intérêt (Region of Interest)
SCP : Secure Copy Protocol
SLURM: Simple Linux Utility for Resource Management
SSH: Secure Shell Protocol
VFH: View Feature Histogram

1 INTRODUCTION

De nos jours, et dans le contexte de la migration vers l'industrie 4.0, les machines dotées d'intelligence pour réaliser des actions humaines sont de plus en plus omniprésentes. Ces machines, appelées « robots », disposent d'une autonomie leurs permettant de collaborer avec les humains. De ce fait, la robotique collaborative représente une des principales technologies caractérisant l'industrie du futur (Combe, 2016).

Plusieurs domaines de l'industrie dépondent de ce type de robots. Comme l'agroalimentaire, automobile et l'aéronautique.

Les robots collaboratifs ou « cobots » constituent des bons partenaires à l'humain dans un grand nombre d'activités industrielles. Surtout, dans le cas des tâches pénibles et répétitives comme par exemple « l'assemblage de pièces » (Moign, 2020).

Le présent travail, fait partie d'un projet d'assemblage automatique d'une pièce industrielle (voir Figure 1). Il est constitué principalement d'un bras robotisé (voir Figure 2) effectuant la sélection et la saisie automatique des pièces afin de les placer au bon endroit. Il s'agit, donc, d'un projet « *pick and place* » ou « *bin picking* ». Cela dit que le bras robotisé repose essentiellement sur une première étape qui est à la fois importante et déterminante : La perception du robot.

Figure 1.1 : Modèle CAD de la pièce

Figure 1.2 : Le bras robotisé

Dans ce contexte, ce projet tire parti de la vision artificielle et plus précisément l'apprentissage profond pour permettre au bras robotisé de détecter avec précision les pièces à l'étude en temps réel. Ces dernières sont placées de façon aléatoire à l'intérieur d'un bac dans un environnement encombré. Ce qui rend la détection plus complexe.

Un premier essai a déjà été réalisé, il visait à tester la pertinence du PoseCNN (Xiang et al., 2018) dans la détection de la pièce industriel à l'étude. Malheureusement, les résultats étaient insuffisants.

Les tests sur les images synthétiques étaient capables de détecter la pièce à l'étude, mais dans des conditions spéciales comme le cas où il n'y a pas d'occlusion ou de troncation. Ce qui est impossible dans le cas de pièces disposées en vrac dans un bac.

En ce qui concerne les tests avec les images réelles, les résultats était décevantes, car il ne détectait aucune instance de la pièce.

La question qui se pose est : « Comment peut-on réaliser une estimation 6DoF des objets industriels complexes pour les applications de *Bin Picking* ? ».

De ce fait, il s'avère essentielle de procéder autrement. Cette étude vise à explorer les approches disponibles et de proposer la solution la plus pertinente. En tenant compte de la nature de la pièce à l'étude qui est : métallique, symétrique, sans texture, d'une seule couleur et qui peut briller avec de la lumière, le défi que présente ce projet est qu'aucune approche libre de droit disponible présentement ne traite les pièces industrielles. Donc, étant donné que la pièce en question fait partie de ce type de pièces, il est important d'étudier la détection avec une base de

données purement dédié à l'usage industriel. D'un autre côté, l'apprentissage doit se faire avec un système de détection ayant une architecture spécialement appropriée aux objets ayant un niveau élevé d'occlusion.

Cette approche vise à trouver une solution de détection plus générale pour les objets industriels. Cette dernière pourra être utilisé spécifiquement pour la pièce à l'étude ou même avec d'autres objets ayant des caractéristiques semblables et qui nécessitent un grand niveau de précision. Dans ce contexte, le présent projet contribue par l'intégration de la base de données industrielles T-LESS à l'architecture PVN3D de l'estimation de pose 6DoF. Donc, d'étendre le projet PVN3D aux objets industriels.

Pour ce faire, ce mémoire sera divisé en 2 grandes parties. La première est dédiée à l'exploration et l'analyse de l'état de l'art afin de situer le projet par rapport aux dernières découvertes à ce sujet.

Après avoir situé la présente recherche par rapport aux autres du même domaine, vient la section essentielle de ce mémoire. Elle présente les étapes de réalisation ainsi que les résultats obtenus. De plus, cette partie sera finie par une conclusion qui couvre la contribution et les prochaines étapes du projet.

2 ÉTAT DE L'ART

2.1 Généralités

2.1.1 Perception du robot dans le cas du *Bin Picking*

La perception en robotique est le processus avec lequel le robot fait la relation entre les mesures enregistrées à partir des capteurs et sa représentation interne de l'environnement. Dans ce contexte, la perception à base de caméras ne représente pas la totalité de perception robotique mais plutôt une partie. Il y en a d'autres comme celle basée sur lidar et les capteurs tactiles (Russell & Norvig, 2021).

La perception par caméra fournie au robot des informations pour naviguer en évitant les obstacles et aussi pour manipuler des objets. Cette manipulation regroupe la localisation et le choix de l'objet, sa saisie et son déplacement. Évidemment, dans le cas ou tous ces objets à manipuler son disposés dans un bac, cette tâche est nommée '*Bin picking*' ou '*Pick and place*'.

Dans un problème de *Pick and Place*, pour qu'un robot aperçoit un objet, il commence par le reconnaitre pour ensuite le positionner dans l'espace. Cette reconnaissance ne veut pas dire savoir la nature de tous les objets qui l'entoure. Mais plutôt, avoir les caractéristiques (ou *features*) du ou des objets qui l'intéresse.

Pour interagir avec le monde extérieur, l'estimation de pose permet au robot de positionner un objet reconnu dans l'espace. Ce qui fait qu'un pipeline d'estimation de pose rapide et précis représente un grand défi de *Bin Picking* robotisé.

La plupart des solutions libres de droit traitent les objets ménagers. Ces derniers ont des caractéristiques bien différentes que les objets manipulés dans le domaine industriel.

Les objets dans le milieu industriel ont souvent la même couleur et la plupart sont composés de formes primitives (sphères, cylindres…). Ces caractéristiques qui sont très similaires empêchent les algorithmes courants de bien les reconnaitre.

2.1.2 Les robots *Bin picking* en industrie

Le ramassage robotisé (ou *Bin picking*) est un processus commun dans l'industrie moderne. C'est une forme d'automatisation intelligente que l'industrie cherche d'en profiter et d'améliorer. Il permet de saisir des objets, connus ou non, avec des poses aléatoires.

Les 3 composants d'un système *Bin Picking* :

- Un détecteur d'images ou caméra perpendiculaire aux objets, il envoie les données 2D ou 3D au processeur.

- Un processeur contenant le programme à exécuter. À partir des entrées provenant du détecteur, il calcule la position et l'orientation des objets, puis décide le point et la trajectoire de la saisie.
- Une arme articulée permettant de saisir l'objet et de le déplacer vers la position finale.

Pour effectuer une saisie basée sur la vision, la plupart des robots localisent l'objet, puis estime sa pose 6DoF. Cette localisation peut comprendre plusieurs techniques comme la détection d'objets, la segmentation sémantique ou la segmentation par instances (Guoguang et al., 2021). Récemment, plusieurs architectures d'estimation de pose incluent les deux éléments étant la localisation et l'estimation de pose 6DoF. C'est cette catégorie qui décrie l'architecture de base du présent projet.

Dans ce contexte, la plupart des approches de saisie robotique repose essentiellement sur la localisation de l'objet. Ce qui implique 3 situations différentes selon les besoins en termes d'information utile :

- Localisation sans classification : ceci représente le cas de saisie le plus simple, ou le robot ne s'intéresse pas à la catégorie de l'objet, mais seulement à la région où il se trouve. La figure suivante en montre un exemple :

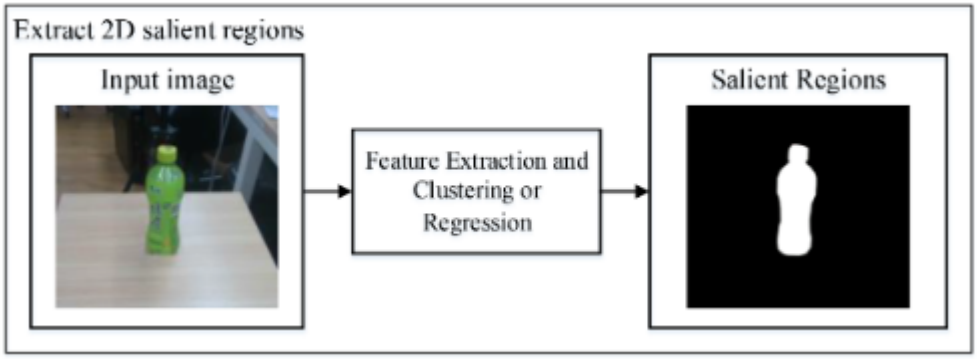

Figure 2.1(a) : Localisation d'objet en 2D sans classification(Guoguang et al., 2021)

- Détection d'objet : cette technique est utilisée lorsque le robot a besoin de l'emplacement de l'objet ainsi que de sa catégorie. Dans ce cas l'algorithme entoure l'objet par une boite et indique sa catégorie. Évidemment, cette boite peut être sous forme d'un rectangle dans une détection 2D (voir figure 2.1(b)) ou d'un parallélépipède rectangle en 3D.

Figure 2.1(b) : Exemple de détection d'objet en 2D

- Segmentation sémantique ou segmentation par instances : C'est une détection d'objet, mais, en comparaison avec la situation précédente, elle fournit encore plus de détails sur l'objet. Les détails, par rapport à son emplacement, sont sous forme d'identification de pixels ou de points définissant l'objet. De plus, concernant les détails liés à sa catégorie, une segmentation donne la même couleur aux objets du même type, alors que la segmentation par instance donne une couleur différente pour chaque instance (voir figure 2.1(c)).

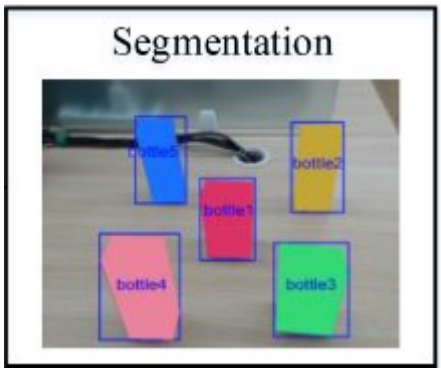

Figure 2.1(c) : Exemple de segmentation par instances en 2D

2.1.3 Apprentissage profond et l'6DoF

Récemment, l'apprentissage profond a dominé ce domaine de vision artificielle. Ses approches robustes et fiables permettent aux cobots d'être des bons collaborateurs aux humains.

Bien que l'humain interprète son environnement extérieur directement en 3D, un robot le fait différemment. La perception du robot a commencé en 2 dimensions. En l'occurrence, les résultats des recherches dans la détection 2D ont atteints des niveaux surprenants, mais les limitations liées à la restriction de 2 dimensions restent présentes. Pour pallier ces limitations, les recherches récentes se sont dirigées vers l'intégration de la troisième dimension. Cette intégration été réalisé selon les 2 approches suivantes:

- Une approche entièrement 3D qui est précise, mais gourmande en termes de calcul. Elle utilise assez souvent les nuages de points comme données d'entrée.
- Une approche qui tire parti de la rapidité et simplicité de la détection 2D ainsi que de la précision 3D. Donc, c'est un mixte des deux. Elle a comme entrée l'image RGB qui est en 2D. Elle utilise l'image de profondeur par la suite pour ajouter de la précision de la 3ème dimension.

Les méthodes traditionnelles de l'estimation de pose dépondent principalement des techniques de « Template Matching ». Elles sont sensibles aux changements d'apparence des objets surtout dans les environnements encombrés.

Dans la littérature, les méthodes basées sur l'apprentissage profond de la pose 6DoF sont nombreuses et offrent de bonnes améliorations face aux limitations des méthodes classiques. Ils comprennent généralement deux phases. Le début se caractérise par une phase de détection des objets, nommée aussi reconnaissance ou identification des objets. Cette phase regroupe la classification, la localisation et la segmentation. Cette dernière peut être sémantique et/ou d'instances. Et puis, l'estimation de la pose 3D de translation et de rotation. La figure suivante extraite de l'article (Blank et al., 2019) montre une vue générale de ces phases selon les données d'entrée :

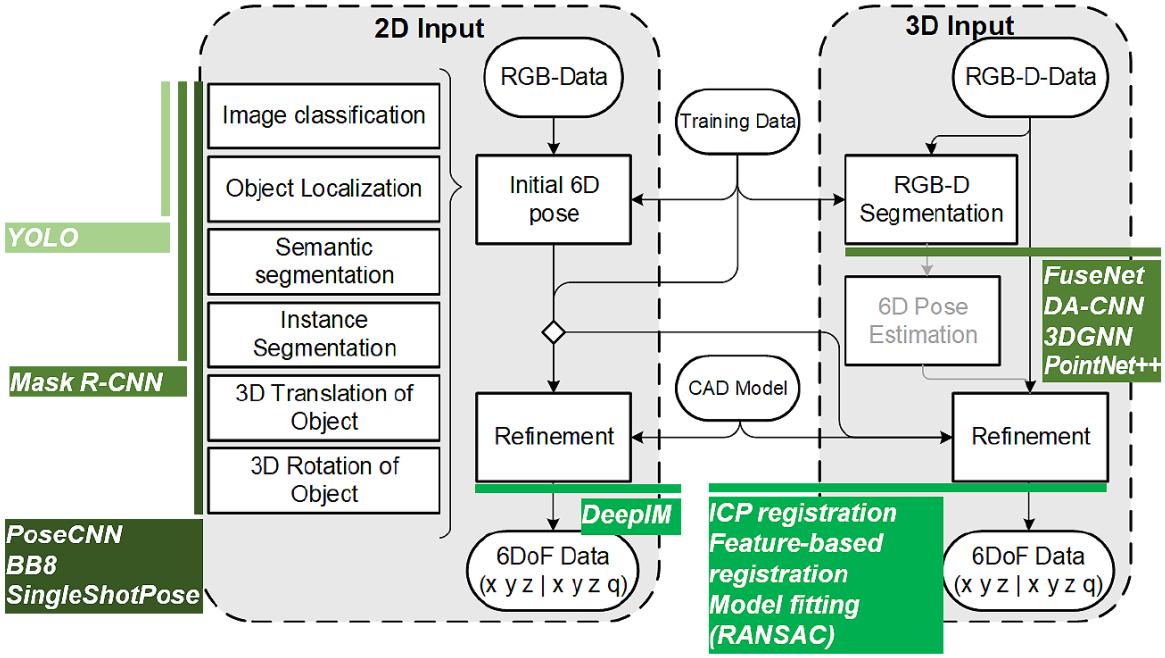

Figure 2.2 : Vue générale des étapes de l'estimation de 6DoF selon une approche par apprentissage profond

La classification est la tâche de prédire la classe de l'objet. À partir d'une image en entrée, l'algorithme aura comme sortie les étiquettes des objets présents. Pourtant, la localisation d'objets consiste à entourer les objets par une boîte. Ainsi, l'algorithme permettant de localiser les objets avec des boîtes et indiquer leurs étiquettes prend le nom d'algorithme de détection d'objets. Une tâche importante qui vient après, est la segmentation. Elle permet d'isoler pixel par pixel les objets identifiés. Cette segmentation peut être sémantique ou par instances.

Également, plusieurs articles dans la littérature utilisent le terme « Reconnaissance d'objets » qui est plus général et regroupe toutes les tâches d'identification d'objets dans une image (classification, localisation...).

La partie suivante explore plus en détails la détection d'objets.

2.2 Détection d'objets

2.2.1 Détection d'objets 2D

La détection d'objets est un domaine largement étudié. Ce qui rend la littérature riche en matière de ressources. Bien que les travaux proposés soient tellement diversifiés, les réseaux de neurones à convolution (CNNs) sont souvent omniprésents dans leurs architectures.

Les plateformes de détection d'objets se divisent en deux groupes :

- *One stage* : Ce type de détecteurs divise l'image d'entrée en cellules de grilles de taille fixe et parallélise la détection sur chaque cellule. Il est représenté le plus souvent par SSD (Liu et al., 2016) et YOLO (Redmon et al., 2016).
- *Multi-stages* : Le plus commun est le *Two-Stages*, comme la famille R-CNN (K. He et al., 2017; Ren et al., 2016) et R-FCN (Dai et al., 2016). Mask R-CNN par exemple priorisent la précision plus que le temps d'exécution. En premier lieu il génère les régions d'intérêts (ROIs), puis les classifie et les segmente. Chaque ROI a besoin de « *re-pooling features* », ce qui demande plus de temps de calcul et empêche, de ce fait, d'avoir une vitesse d'exécution temps réel. Récemment, certaines approches ont passé au *Multi-stage* en ayant recours à une forme en cascade (Cai & Vasconcelos, 2018).

2.2.2 La segmentation d'instances 2D en temps réel

La perception du robot lors d'une opération de *Bin Picking* est fortement liée à la contrainte de l'exécution en temps réel. Ce qui veut dire que le temps de réponse doit être inférieur à 30 FPS (Bolya et al., 2019).

Dans ce contexte, plusieurs recherches ont traité la détection d'objets et la segmentation sémantique (Dvornik et al., 2017; Paszke et al., 2016; Zhao et al., 2018). Cependant, il y a moins de recherches traitant la segmentation d'instances en temps réel (Bolya et al., 2019; Jetley et al., 2017; Uhrig et al., 2018; Xiong et al., 2019; Xu et al., 2019).

Les approches qui représentent l'état de l'art dans la segmentation d'instances comme Mask-RCNN (K. He et al., 2017) et FCIS (Y. Li et al., 2017) sont lents et n'offre pas une exécution en temps réelle. Ce qui explique le recours aux techniques de détection d'objets « one stage » pour avoir une exécution plus rapide.

Ces travaux sont basés sur une segmentation d'instances 2D. Cependant, il y a un grand manque dans la littérature pour ce qui est des approches 3D particulièrement dédiées à l'exécution en temps réel.

2.2.3 Détection d'objets 3D

Durant ce type de détection, chaque objet reconnu va être entouré d'une boîte 3D. Elle dépend grandement de la qualité des nuages de points collectionnés par le capteur. Bien que ces points offrent une géométrie précise, ils ont l'inconvénient d'être irréguliers. Pour pallier cette irrégularité, les détecteurs 3D courantes proposent plusieurs approches, en voici quelques-uns:

- Convertir les points de nuages en des images de vues d'oiseaux régulières (« *2D bird's eye view images* »), puis y appliquer un détecteur 2D (Zhou & Tuzel, 2018). Cette technique perd de la précision géométrique. De ce fait, elle est plus appropriée pour la détection dans un environnement extérieur. Toutefois, elle est moins pertinente dans le cas du *Bin Picking* qui manipule des pièces d'intérieur encombrées.
- Améliorer la détection 2D pour fournir une détection 3D. Cette technique repose sur la voxelisation des nuages de points en des grilles 3D qui sont réguliers, et applique par la suite un détecteur 3D CNN (Hou et al., 2019; S. Song & Xiao, 2016). Ces 3D convolutions sont coûteux, car ils demandent un niveau de calcul élevé.

- Combiner la détection 2D et la localisation 3D (Lahoud & Ghanem, 2017; Qi et al., 2018). Vu que cette approche repose en premier lieu sur une détection 2D, un objet qui n'est pas détecté en 2D va être complètement absent lors de la localisation.
- Ne pas dépondre d'une détection 2D et détecter les objets entièrement en 3D (Qi et al., 2019).

2.3 L'estimation de pose 6DoF

Dans le contexte des solutions industrielles avancées liée à la robotique collaborative, la 6DoF représente une des principaux problèmes à résoudre. Elle permet d'informer le robot des poses des objets. Elle représente la grande partie de sa perception ou sa manière de voir son environnement avec précision.

La 6DoF est la tâche d'estimer les coordonnées (x, y, z) ainsi que les angles de rotation selon un système de référence préétabli. Dans le cas de manipulation robotique, l'estimation de pose 6D présente plusieurs défis : la géométrie de l'objet utilisé, le temps de manipulation et la précision requise.

2.3.1 Approche par CNNs

Une grande partie des chercheurs dans le domaine de la vision par ordinateur trouvent que l'approche par apprentissage profond est prometteuse. Ainsi, de nombreux travaux sont apparus dans la littérature traitant la 6DoF en tirant partie des CNNs. Ils ont comme objectif de repousser les limites des méthodes classiques. Cette partie présente les principales techniques.

BB8 (Rad & Lepetit, 2017) est une technique *multi-stage* qui effectue la détection 3D des objets à partir des images RGB. Elle applique aux objets détectés un réseau de neurones à convolution CNN entrainé pour prédire la pose 3D.

SSD-6D (Kehl et al., 2017) est une différente technique. Elle s'appuie sur l'architecture SSD pour effectuer la détection 3D et estimer la 6DoF en un seul coup. Ces techniques (SSD-6D et BB8) nécessitent un raffinement supplémentaire pour améliorer la précision.

Comme SSD-6D, YOLO-6D (Tekin et al., 2018) est une architecture *single shot* qui repose sur YOLO. Aussi, PoseCNN (Xiang et al., 2018) est une autre technique pionnière basée sur VGG-16 qui effectue l'estimation de pose 6D. Ce réseau effectue 3 différentes tâches : segmentation sémantique, estimation de la translation 3D et la régression de la rotation 3D. Tous les travaux précédents sont entrainés avec des images réels.

DOPE (Tremblay et al., 2018) est constitué d'un réseau de neurones d'architecture simple VGG-19. Selon l'article, il a des performances semblables à PoseCNN. Toutefois, l'apprentissage se fait complètement avec des données synthétiques.

BB8, SSD-6D, PoseCNN et DOPE sont des méthodes holistiques. C-à-d qu'il estiment la localisation et la rotation 3D en un seul coup avec un seul réseau de neurones. Il existe aussi des méthodes basées sur les *keypoints* comme (Tekin et al., 2018). Elles estiment la pose en 2 étapes. Elles commencent par prédire les *keypoints* 2D de l'objet puis, calcule la pose avec l'algorithme PnP à partir de la correspondance 2D/3D. De plus, il y a les méthodes denses. Dans ce cas, chaque pixel produit une prédiction qui sera suivi d'un vote (Michel et al., 2017).

PvNet (Peng et al., 2019) est une architecture qui peut être considéré comme un hybride des deux types car elle utilise les *keypoints* et un vote pour avoir le résultat final. Elle exécute la segmentation sémantique et la prédiction du champ vectoriel. Elle est considérée, selon (C. Song et al., 2020), l'état de l'art des méthodes à base de *keypoints*.

Les techniques YOLO-6D, PvNet et DPOD donnent de meilleurs résultats sans raffinement. Toutefois, pour raffiner la 6DoF qui est déjà calculée, DeepIM [35] et ICP sont les choix les plus répondus. DeepIM est utilisable seulement avec les images de couleur alors que l'algorithme ICP est dédié au nuage de points. Ce post-traitement améliore considérablement la précision de la 6 DoF.

2.3.2 Approche par GANs

En principe, les modèles génératives à base d'auto-encodeur sont utilisés pour reconstruire une image en enlevant le bruit ou retrouver les parties manquantes. Cela dit, qu'un modèle génératif peut servir à préparer l'image d'entrée pour qu'elle soit le plus idéale possible pour la détection et l'estimation de pose. Pix2Pose [36] propose une estimation de pose à base d'un GAN pour récupérer les parties manquantes en cas d'occlusion.

2.3.3 Approche par nuages de points

Tel que vu dans les parties précédentes, les méthodes basées sur les données RGB-D effectuent assez souvent une détection 2D qui dépond des images RGB et intègre l'image de profondeur pour raffiner le résultat. Une question se pose : Puisque le monde réel est en 3D, pourquoi ne pas apprendre directement et efficacement en 3D ? Dernièrement, Facebook AI a lancé VoteNet (Qi et al., 2019) qui effectue une détection d'objets 3D bout-à-bout en ayant recours au vote profond des points. Certains chercheurs se sont inspirés de cette approche de détection 3D pour

tirer parti pleinement de la géométrie des objets dans leur estimation 6DoF. Dans ce contexte, PVN3D (Y. He et al., 2020) est une solution qui vient d'apparaitre dans la littérature et qui s'inspire de VoteNet.

PvNet vu dans la partie précédente gère bien les problèmes d'occlusion et troncation en proposant un réseau de vote pixel par pixel pour localiser en 2D les *keypoints*. Bien que ces *keypoints* 2D vise à minimiser l'erreur de projection 2D des objets, ces erreurs peuvent être significatives en monde réel 3D. PVN3D pousse encore plus loin cette approche pour supporter des *keypoints* 3D.

PVN3D propose un module de segmentation d'instances basé sur un réseau PointNet++ (Qi et al., 2017) . Ce dernier est un réseau de neurones profond bout-à-bout qui permet d'extraire les informations géométriques directement du nuage de points. PVN3D se distingue par extraire les caractéristiques 3D et 2D des images RBG-G puis ajoute un réseau de vote 3D pour faire la détection.

Récemment, PVN3D a sorti une nouvelle version qui est plus rapide et donne plus de précision nommé FFB6D (Y. He et al., 2021). C'est le même principe, toutefois il a amélioré la façon d'extraire les caractéristiques des images RGB-D.

Le tableau suivant montre quelques techniques et leurs façons d'estimer la pose 6DoF :

La technique de DoF	Estimation	À partir de
PoseCNN	RANSAC	Les résultats d'étiquetage sémantique et les résultats de la régression des coordonnées des objets 3D
PvNet, YOLOX(Zhou & Liu, 2022)	Algorithme PnP	*KeyPoints* 2D
PVN3D	Algorithme d'ajustement par moindres carrés	*KeyPoints* 3D

Tableau 2.1 : Techniques d'estimation 6DoF avec l'apprentissage profond

2.3.4 6DoF dans les applications industrielles de *Bin Picking*

Dans la littérature, dans le cadre du *Bin Picking*, plusieurs solutions pour la 6DoF ont rendu leurs codes à accès libre pour permettre aux chercheurs de les tester et pousser plus loin leurs

techniques. Toutefois, aucun d'entre elles n'est spécifiquement dédiée aux objets manipulés dans le milieu industriel.

Les techniques disponibles de la 6DoF éprouvent des difficultés à détecter les objets industriels surtout lorsqu'ils sont disposés de façon aléatoire dans un bac. C'est pour cette raison, plusieurs travaux proposent des solutions personnalisées. Cette partie en présente quelques exemples.

(Blank et al., 2019) propose une approche par plusieurs étapes pour les objets métalliques sans texture qui sont commun dans le milieu industriel. Selon l'article, parmi les solutions 6DoF disponibles, très peu qui sont partiellement convenables aux composants industriels, par exemple YOLO-6D. Cette approche combine une localisation d'objet basée sur l'apprentissage profond et une estimation de pose 6D qui repose sur « *Template Matching* ». Vu que ce dernier éprouve des difficultés à estimer correctement la pose dans un environnement où les objets sont souvent accumulés étroitement, elle implémente une pré-segmentation avec YOLOv3. Cette pré-segmentation permet de réduire le flux RGB-D pour que LineMOD exécute le *Template Matching* seulement sur les régions pré-segmentées. Pour améliorer la précision de l'estimation de pose, elle ajoute un raffinement par ICP.

Certains travaux créent préalablement une base de données de modèles 3D. À cette fin, il suffit d'extraire les caractéristiques à partir du modèle CAD ou des nuages de points capturés d'une scène réelle. L'estimation de la 6DoF se fait par un processus de correspondance (*matching*). (Yan et al., 2020) met l'accent sur la rapidité de la 6DoF. Il compare à base de votes les modèles pour estimer la 6DoF puis, l'améliore avec ICP. (Jiao et al., 2019) est basé sur *Template Matching*. Il utilise des fonctions pour choisir les k-items les plus hauts dans le bac en comparant leurs *templates* avec ceux sa la base de données. Il raffine ensuite la position détectée avec ICP. (D. Li et al., 2019) utilise VFH (*View Feature Histogram*) pour l'estimation de la 6DoF mais, d'une façon améliorée. Bien que ce descripteur soit fiable dans le cas de surface bruitée ou d'informations de profondeur manquantes, il peut y avoir des difficultés quand les objets sont placés symétriquement devant le point de vue. Pour résoudre ce problème, la totalité du nuage de point d'un objet est divisée en quatre parties et chacun d'entre elles caractérise cet objet. (M. Li & Hashimoto, 2018) a une approche rapide et robuste. Il calcule la 6DoF par l'algorithme PPF (*Point Pair feature*) mais d'une façon plus efficace. Seulement un peu de paires de points sont calculées pour améliorer l'efficacité. Il ajoute ensuite un raffinement ICP.

Ces travaux montrent que les méthodes classiques ajustées sont plus utilisées que les méthodes par apprentissage profond dans un milieu industriel. Cela peut être expliqué par le fait que dans une usine les objets manipulés sont bien limités. De plus, elle dispose souvent d'une base de données de tous leurs modèles 3D. Au contraire, un projet de recherche en milieu universitaire n'a pas nécessairement ce type de base de données. C'est pourquoi il est de monnaie courante que les chercheurs aillent recours aux bases de données d'apprentissage libre de droit pour accomplir leurs recherches. Ainsi, l'apprentissage profond offre au robot plus d'intelligence vis-à-vis la manipulation de nouveau objets.

2.5 Données d'apprentissage

2.5.1 Données insuffisantes

Comme tous les projets d'apprentissage profond, ceux de l'estimation 6DoF dépendent grandement des données d'apprentissage.

(Blank et al., 2019) utilise YoloV3 pour détecter les objets. Ce dernier est pré-entrainé avec la base de données COCO. Bien évidemment, cette BD rassemble des images contenant des scènes de tous les jours avec les objets communs placés dans leurs contextes naturels. Ce qui est vraiment loin des objets manipulés dans le milieu industriel. Comme solution à ce problème, (Blank et al., 2019) a construit, dessus les poids entrainés avec la base de données COCO, un entrainement additionnel. Cet entrainement cible les dernières couches de l'architecture du réseau et comprend 250 images annotées de leurs propres objets à manipuler.

Selon (Chollet, 2018, p. 143), cette approche est communément utilisée et très efficace en apprentissage profond. Entrainer un réseau qui manque d'une grande quantité de données d'apprentissage avec une BD suffisamment large et générale lui offre des connaissances génériques qui améliore grandement son fonctionnement.

Une autre technique qui permet d'étendre artificiellement la base de données existante est « *Data Augmentation* ». Cette technique effectue des transformations (tourner, couper, zoomer, décaler…) et produit des images à aspect crédible. Les modèles exposés à plus d'aspects de données généralise mieux et évite, de ce fait, un problème courant qui est l'«*Overfitting* » (Chollet, 2018, p. 138).

Plusieurs travaux récents traite un des grands défis des données synthétiques : « *the reality gap* ». Annoter des grandes quantités de données du monde réel est un processus long et fastidieux surtout pour les données 3D. Généralement, sans paramétrage additionnel (*fine-tuning*), un

réseau entrainé avec des données synthétiques ne peut pas bien généraliser durant les tests avec les données réelles. (Tremblay et al., 2018) traite ce problème dans le contexte de pose 6DoF. Il propose de combiner « *Domain Randomization* » qui consiste à générer aléatoirement les données de façon non-réaliste et « *Photorealistic Data* » qui fait pareil mais, de manière réaliste.

2.5.2 Base de données disponibles

Bien que les base de données d'apprentissage utilisées pour la détection d'objets sont nombreuses, il s'avère intéressant de présenter celles les plus répondues dans l'estimation de pose 6DoF.

BOP (*BOP: Benchmark for 6D Object Pose Estimation*, s. d.; Sundermeyer et al., 2023) est une compétition internationale permettant d'attirer les différentes solutions de l'estimation de pose 6DoF en leur offrant une méthode qui facilite l'entrainement de leurs architectures. Il propose 12 ensembles de données de format standard qui seront prête à être utilisées pour tester leurs solutions. Ces bases de données inclues les éléments suivants :

- Modèles 3D des objets
- Données de test et d'entrainement
- Images RGB-D étiquetées avec les poses 6DoF *ground-truth.*
- Les paramètres intrinsèques de la caméra

Evidemment, certaines versions de ces BDs nécessite une étape de prétraitement selon les besoins des architectures de réseaux de neurones sur lesquels vont être appliqués.

Parmi ses BDs, il y a LineMOD et YCB-video (*BOP: Benchmark for 6D Object Pose Estimation*, s. d.) qui content des objets de différentes catégories. Ils sont largement utilisés par des projets d'estimation de pose 6DoF comme PoseCNN.

LineMOD (voir Figure 2.3) est une référence standard pour la 6DoF. Elle contient des images de scènes avec des objets ménagers. Ces objets sont disposés de façon aléatoire avec différentes conditions d'éclairage. De plus, cette référence possède deux sous-ensembles; *Occlusion LineMOD* contient des images où plusieurs objets faisant objet d'occlusion sont annotés. Ces scènes posent un bon défi pour l'estimation de pose. L'autre sous-ensemble est *Truncation LineMOD*, il vise la précision vis-à-vis les objets coupés. Seulement de 40% à 60% de l'objet

reste sur l'image. En général, ces deux sous-ensembles de LineMOD sont utilisés particulièrement pour les tests.

Figure 2.3 : Exemples d'objets LineMOD

YCB-Video (voir Figure 2.4) est une référence dans la manipulation robotique, elle contient des images RGB-D des objets ménagers avec leurs masques de segmentation ainsi que les informations de calibration qui y sont associés. Cette BD ajoute aussi les modèles 3D.

Figure 2.4 : Exemple d'objets YCB-video

BOP propose aussi des BD ayant des objets manipulés dans le milieu industriel, il y en a présentement juste 2 : T-LESS (Hodan et al., 2017) et ITOOD (Drost et al., 2017).

ITODD (voir Figure 2.5) est réalisée à partir d'un détecteur Gray-D. Ses objets sont tous métalliques. Les poses 'ground truth' sont seulement disponible publiquement pour les images de validation mais, pas pour celles de tests.

Figure 2.5 : Exemple de scène ITODD

T-LESS regroupe des scènes réelles de 30 objets industrielles qui ne sont pas métalliques mais, de couleur unie. La figure 2.6 en montre un aperçu :

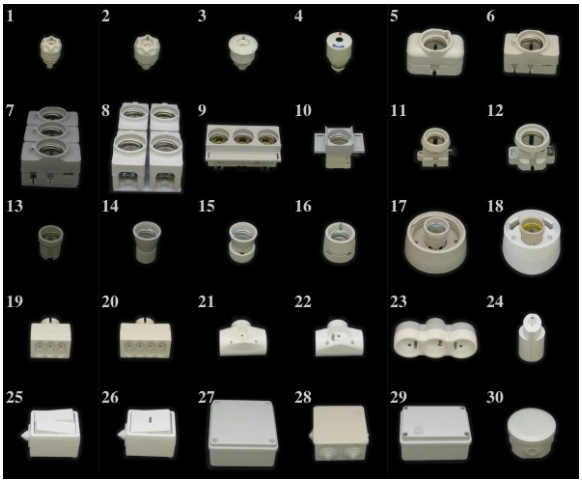

Figure 2.6 : Exemple d'objets T-LESS

Les images sont réalisées par des caméra RGB-D. De ce fait, elle comporte des images RGB ainsi des images de profondeur.

T-LESS est présentement la seul BD ayant des images d'entrainement de scènes réelles, avec des objets industriels, et qui son capturées par des caméra RGB-D.

Il existe aussi une BD faite à partir de T-LESS pour des raisons de compétition appelé PBR (*BOP: Benchmark for 6D Object Pose Estimation*, s. d.). Cet ensemble de données est destiné pour entrainement des réseaux de neurones profonds. En l'occurrence, ses scènes différentes avec beaucoup d'occlusions et de variations de luminosité permettent au réseau d'apprendre plus de *patterns*, donc, d'être plus fiable. En voici un aperçu :

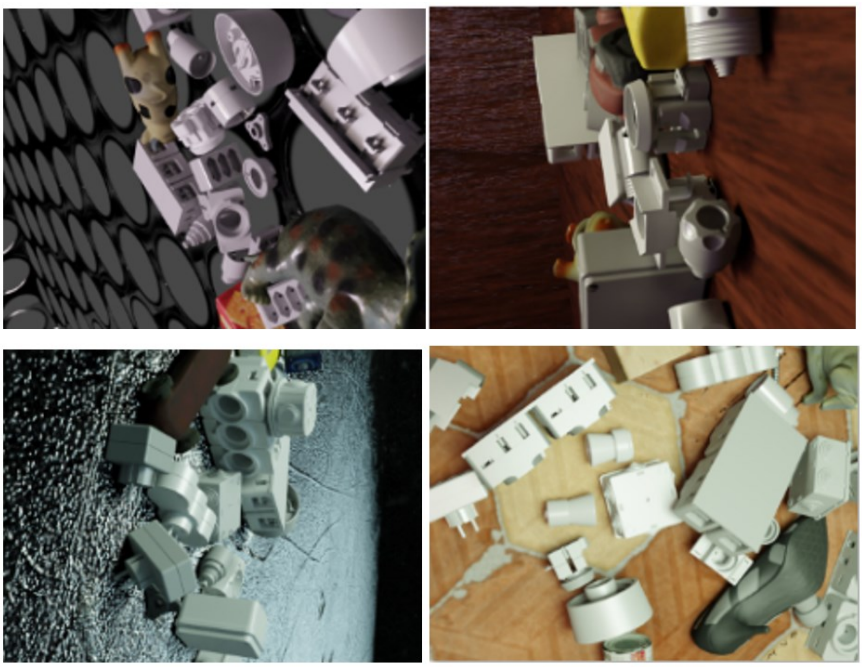

Figure 2.7 : Exemple de scènes BPR

En dehors de BOP, la base de données « Siléane » (*Dataset*, s. d.) utilisée dans l'article est publique. Elle comprend à la fois des scènes simulées et du monde réel annotées. Elle utilise 3 objets industriels de l'ensemble de données T-LESS (voir Figure 2.8(a)). L'autre appelé 'gear' est une simulation. La figure 2.8(b) montre un exemple de ses scènes.

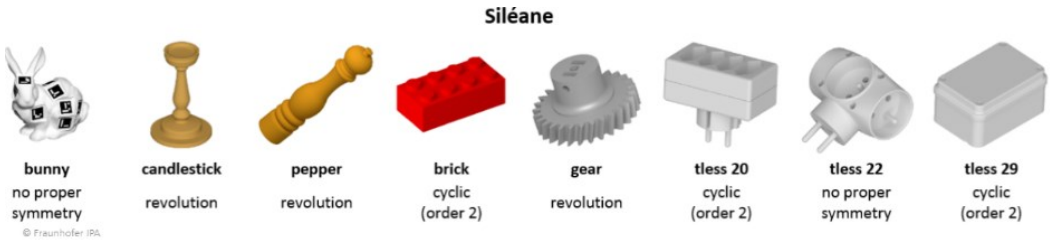

Figure 2.8(a) : Objets de la base de données Siléane

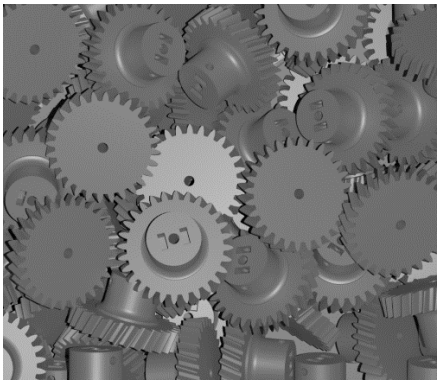

Figure 2.8(b) : Exemple de scène Siléane

Concernant les BDs 3D liées au *Bin Picking*, il y a « *Dex-Net Mesh v1.1* », « Dex-Net 2.0 HDF5 » et « *KIT ObjectModels Web Database* » qui sont mentionnées dans le projet Dex-Net (Mahler et al., 2017) et sont disponibles dans (*Dex-Net by BerkeleyAutomation*, s. d.). Dans ce projet, les données 3D et 2D sont capturés automatiquement, à partir de différents points de vue, grâce une structure spéciale contenant un numériseur 3D (voir figure 2.9(a)). Un exemple des objets utilisés en entrainement est présenté dans la figure 2.9(b).

Figure 2.9(a): Structure dotée d'un numériseur 3D

Figure 2.9(b) : Les modèle les 3D des objets Dex-Net

3 RÉALISATION DU PROJET : CADRE THÉORIQUE

3.1 6DoF

6DoF est la tâche d'estimer la transformation qui permet de déplacer un objet à partir de son système de coordonnées vers le système de coordonnées de la caméra. Elle consiste à déterminer la position 3D de l'objet qui est la translation 3D ainsi que son orientation 3D qui est la rotation 3D.

C'est un élément important de différent problèmes réels comme : la manipulation et la saisie aléatoire robotique, conduite autonome, réalité augmentée.

Dans un contexte de *Bin picking*, le robot a besoin de trouver la 6DoF pour pouvoir localiser et saisir l'objet dans son environnement de proximité. De plus, cela lui permet d'interagir de façon sécuritaire côté à côté avec les humains.

Il existe plusieurs architectures pour calculer une estimation de la pose 6Dof :

1- Avec les méthodes traditionnelles : '*modele matching*' ou '*features matching'*

Consiste à faire une correspondance entre l'image 2D et le modèle de l'objet. Puisque la définition des caractéristiques pour effectuer cette correspondance se fait manuellement cette approche n'est pas efficace en cas de changement de luminosité ou d'environnement avec un haut niveau d'occlusion.

2- Avec réseau de neurones profond

Elle estime la pose 6DoF en effectuant une régression de rotation et de translation directement avec RNAP. Cette méthode a des limitations vues que le RNAP donne une mauvaise généralisation. Ceci est due à la non-linéarité des données d'entrainement.

3- Avec RNAP et les points-clés 2D

Dans ce cas, la pose 6DoF est calculée généralement par un algorithme comme PnP en effectuant une projection de l'espace 2D à 3D. Cette méthode donne des résultats meilleurs que les deux précédentes. Toutefois, elle ne convient pas aux objets sans texture. Due à la projection, des informations géométriques de l'objet à détecter sont partiellement perdues. En plus, cette

projection fait en sorte que des erreurs qui sont petites en espace 2D vont être plus grandes en 3D ainsi que plusieurs points seront superposés en 3D.

4- Avec RNAP et les points-clés 3D

Cette approche utilise comme capteur une caméra RGB-D qui transmet des informations 3D directement sans avoir besoin d'appliquer une projection. Dans ce cas, l'image de profondeur combinée à l'image RGB peut être converti à son équivalente représentation en 3D comme les nuages de points. Ce qui préserve toute l'information géométrique liée à l'objet et donne de meilleurs résultats avec les objets sans textures ainsi qu'avec la variation de luminosité.

Quant aux algorithmes de calcul la pose 6DoF, on trouve :

- **RANSAC** :

C'est un algorithme permettant de réaliser un vote de pixels pour déterminer les points-clés. Dans certains projets d'estimation de pose 6DoF, cette étape de d'utilisation de RANSAC est suivie par l'algorithme PNP pour effectuer l'estimation de pose 6DoF. Dans PoseCNN, *pre-emptive RANSAC* estime la pose 6DoF à partir des coordonnées 3D résultant de la régression.

- **PnP** :

C'est un algorithme largement utilisé dans le domaine de la vision par ordinateur pour estimer la pose 6DoF. Il fait référence à la relation entre l'objet et la caméra. Pour cela, il combine les coordonnées d'image obtenues à partir de la caméra avec les coordonnées de n points caractéristiques connus dans le système de coordonnées. Ce problème peut se résoudre avec des méthodes linéaires et non-linéaire. Toutefois, cette dernière méthode est considérée plus efficace, précise et stable.

Récemment, l'algorithme PNP est amplement étudié et appliqué dans le domaine de la robotique autonome, positionnement de robots, reconnaissance et suivi de cible, réalité augmentée et SLAM. (Qiao et al., 2023).

- **Ajustement par moindres carrés** :

C'est une façon de calculer la pose 6DoF. Elle est utilisée par PVN3D à la dernière étape de son architecture. Cette méthode permet de calculer la rotation R et la translation T à partir de l'ensemble de points -clés dans le système de coordonné de la caméra ainsi que les points

correspondants dans le système de coordonnées de l'objet. L'objectif est de minimiser : $L_{\text{least-squares}} = \sum_{j=1}^{M} ||kp_j - (R \cdot kp_j^{'} + t)||^2$, avec M le nombre de points-clés pour un objet.

3.2 Termes et algorithmes

Quaternion

Quaternion est un nombre hyper-complexe. C'est une extension des nombres complexes. Au lieu d'avoir une seule partie imaginaire *bi*, un quaternion possède trois composante i, j, et k. De ce fait, il s'écrit comme suit : $q=q1+q_2i+q_3j+q_4k$.

Avec un quaternion unitaire, on dispose de 3 degrés de liberté. Pour appliquer une rotation à un vecteur, il suffit de multiplier par un quaternion.

Cette représentation est largement utilisée en robotique, et précisément en *Bin Picking*, pour représenter la rotation 3D du bras robotisé mobile aussi bien pour l'orientation des objets lors de l'estimation de pose 6DoF.

L'algorithme ICP (*Iterative- Closest Point*)

C'est un algorithme largement utilisé en robotique et en vision par ordinateur.

C'est un algorithme qui sert à minimiser la différence entre deux nuages de points soit 2D ou 3D. Il répète un processus jusqu'à convergence. Il associe chaque point du premier nuage de points au point le plus proche du second nuage de point (voir Figure 3.1). Ensuite, il calcule une transformation sous forme d'une translation et/ou d'une rotation pour minimiser la distance entre les points. Ce processus est répété jusqu'à l'atteinte de la distance totale minimale.

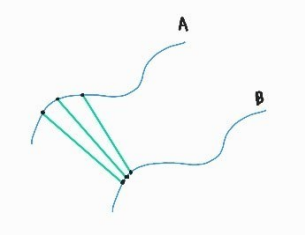

Figure 3.1 : ICP associe chaque point de A à B

Dans le cadre de l'estimation de la pose 6DoF, il est utilisé pour raffiner les poses trouvées.

FPS (*Farthest Point Sampling*)

Utilisé pour choisir les points-clés 3D sur le modèle 3D de l'objet. Dans PVN3D, 8 points-clés ainsi que le centre ont été choisis.

4 RÉALISATION DU PROJET : ANALYSE ET CONCEPTION

4.1 Situation du présent projet par rapport au travail précédent

Dans le même objectif global de trouver une solution permettant la détection d'objets industriels complexes avec l'estimation de pose 6DoF, le travail précédent se résume dans l'essai d'intégrer directement la pièce à l'étude à PoseCNN. À partir d'un modèle CAD 3D libre de droit, un objet physique a été imprimé. Ayant l'hypothèse qu'une BD sous forme d'images synthétiques formées à partir de l'objet physique numérisé pourrait être efficace pour la détection dans un environnement réel. Malheureusement, se limiter à des données synthétiques seulement avec PoseCNN ne permettait pas la détection de la pièce en question. Dans le présent projet, en conséquence, plusieurs points ont été mises en question :

- PoseCNN représentait-il un bon choix d'algorithme de détection pour la pièce à l'étude ?
- Est-il faisable d'intégrer à l'algorithme d'apprentissage une BD ayant des données issues de scènes réelles ? Comment faire ?

En analysant et en répondant à ces questions, les grandes lignes du présent projet ont été définies. La section suivant en montre un aperçu.

4.2 Objectifs du présent projet

Donner une solution complète qui augmente la précision de la détection d'objets en industrie est très demandant en termes de ressources informatiques ainsi qu'on terme du temps accordé à cette fin. En l'occurrence, ce sujet incluant son analyse, sa conception et sa réalisation, est bien adapté à un projet de doctorat en termes de complexité. C'est pour cette raison le présent projet de maitrise constitue un bon début pour :

- Découvrir des possibilités de projets disponibles pouvant être un point de départ pour cette solution.
- Trouver comment utiliser ce projet de départ et quelles sont les modifications à apporter pour qu'il soit adapter aux objectifs définis. Analyser la faisabilité et la pertinence de cette approche.

- Concevoir un environnement de développement approprié avec les outils pertinents.
- Mettre en pratique cette conception, puis analyser la faisabilité et la pertinence de l'architecture.
- Mettre en pratique le projet de départ et étudier les modifications à planifier.
- Réaliser les modifications qui vont permettre de déterminer la pertinence de cette approche.

Dans ce contexte, comment l'architecture et la base de données du départ ont été choisis ?

4.3 Pourquoi PVN3D ?

Le point de départ dans cette analyse, est de démystifier les détails de détection d'objets et de l'estimation de pose 6DoF dans PoseCNN. Étant donné que ce projet réalisé par NVIDIA représente un excellent choix de détection d'objet, qu'est ce qui le rend moins efficace lors des tests avec la pièce à l'étude ?

Pour répondre à cette question, il fallait analyser en détail son architecture pour avoir une idée de quel type d'objet est-il destiné.

4.3.1 Analyse de PoseCNN

PoseCNN commence par une détection d'objet en premier lieu, puis estime en conséquence la pose 6DoF. Pour avoir la 6DoF, il estime la translation 3D de l'objet en localisant son centre sur l'image et en prédisant sa distance par rapport à la caméra. La rotation 3D est estimée en régressant à une représentation quaternion.

Pour ce faire, PoseCNN propose une architecture à 3 étapes (voir Figure 4.1). Il commence par étiqueter chaque pixel en l'associant à un objet spécifique. Dans ce contexte, cette estimation du centre introduit la notion de vote vis-à-vis les l'ensemble des pixels appartenant à un objet. À partir des pixels de chaque objet, il localise les coordonnées 2D du centre puis estime la distance du centre par rapport à la caméra. Ensuite, étant donné que la matrice intrinsèque de la caméra est déjà connue, ces informations 2D concernant le centre de l'objet vont permettre à PoseCNN de trouver sa translation 3D (T). Finalement, la rotation 3D (R) sera estimée à partir de la régression des caractéristiques à l'intérieur des boîtes entourant chaque objet à une représentation quaternion.

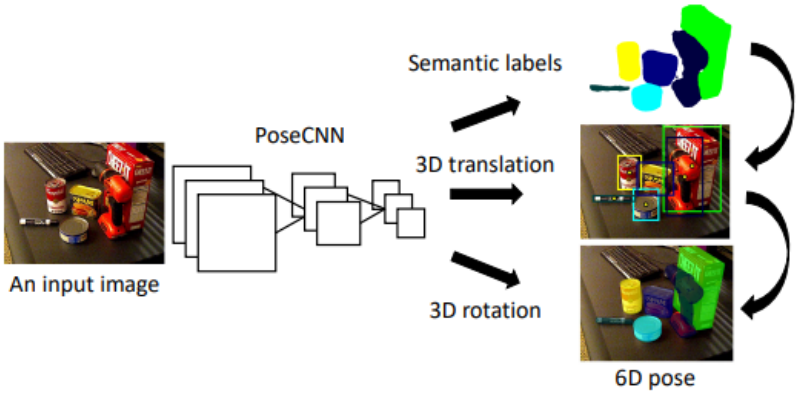

Figure 4.1 : Étapes de détection d'objets de PoseCNN

Cette technique peut être appliquée à des objets avec ou sans texture. De plus, elle est efficace dans un certain niveau d'occlusion vu que le vote effectué sur les pixels se fait même quand des parties de l'objet sont cachées. Toutefois, détecter des objets symétriques en constitue encore un défi, car plusieurs orientations de l'objet peuvent générer une même observation. Pour résoudre ce problème, une nouvelle fonction de perte (*loss function*) a été ajoutée. Cette fonction mesure le décalage entre chaque point du modèle estimé et le point le plus proche du modèle *ground-truth*. C'est à peu près semblable à l'algorithme ICP.

Concernant l'estimation de la translation 3D, la façon naïve de le faire est d'effectuer une régression des caractéristiques de l'image vers T. Cette méthode ne couvre pas le cas de plusieurs instances du même objet dans une image. Donc, pour que PoseCNN peut être appliqué dans le cas de plusieurs instances, il a introduit une autre façon de faire. Elle se résume dans l'estimation de la translation 3D à partir des coordonnées 2D du centre de l'objet. La 3eme dimension est la distance du centre par rapport à la caméra. À partir des annotations sémantiques des images RGB et des résultats de la régression, l'estimation de pose est calculée avec l'algorithme RANSAC.

En l'occurrence, l'estimation de pose 6DoF est testée sur LineMOD et Occluded-LineMOD. Cependant, la détection par instances est testée avec YCB-vidéo. Si l'image de profondeur est disponible, cette pose peut être raffinée par la suite avec l'algorithme ICP.

Les 3 BDs utiliées dans PoseCNN ne traitent pas le problème de *Bin Picking* où les objets sont déposés aléatoirement dans un contenant. Cependant, il détecte des objets mis sur une table avec certains niveaux d'occlusion. Sa précision diminue avec les scènes complexes où l'on dispose d'un environnement ayant des objets difficilement reconnaissables comme de *Bin picking* ayant

beaucoup d'occlusion. D'un autre côté, la détection dans PoseCNN est basée sur les pixels appartenant à l'objet et qui apparaisse dans l'image 2D. Plus ce nombre de pixels est grand, plus la détection est meilleure, et vice versa. De ce fait, plus il y a occlusion moins l'objet est reconnu.

Pour voir plus en détail le mécanisme de détection dans PoseCNN (voir Figure 4.2), PoseCNN localise le centre ***c*** et estime sa profondeur *Tz*. Avec l'équation $\begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} f_x \frac{T_x}{T_z} + p_x \\ f_y \frac{T_y}{T_z} + p_y \end{bmatrix}$ il effectue une projection et trouve *Tx* et *Ty*. T représente l'estimation de la translation 3D de la pose 6 DoF.

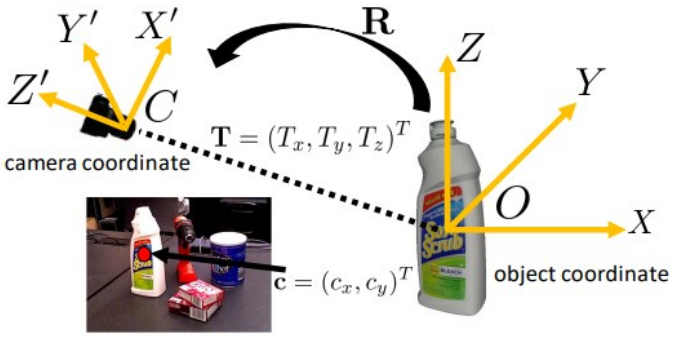

Figure 4.2 : L'estimation de la translation T avec PoseCNN

Cela montre que la détection dans PoseCNN se fait complétement en 2D. Donc les objets qui ne sont pas reconnus en 2D ne seront plus localisés en 3D. Donc, l'estimation de pose 6DoF dépend complétement de la détection 2D, alors la profondeur qui est la 3ème dimension ne sera plus utile pour raffiner les résultats de l'estimation de pose 6DoF des objets non détectés en 2D. Concernant les modèles 3D des objets, ils sont utilisés pour faire le rendu des objets ainsi que pour des raisons de comparaison. Donc, ces modèles 3D ne sont pas utilisés en apprentissage pour extraire des caractéristiques 3D pouvant améliorer la détection. Pour conclure, limiter la pose 6DoF à une détection 2D diminue la fiabilité des résultats surtout pour les objets complexes se trouvant dans un environnement complexe lui aussi.

L'idée maintenant est : comment améliorer cette perception ?

Ceci amène la réflexion vers la détection 3D. Ce type de détection a pour but de localiser et reconnaitre les objets dans une scène 3D. Elle utilise les nuages de points comme entrée. En l'occurrence, Facebook fait sortir VoteNet qui traite spécifiquement cet objectif.

4.3.2 Analyse de VoteNet

VoteNet est le premier système de détection 3D de bout en bout qui ne dépend plus des détecteurs 2D. C'est une façon d'utiliser les informations géométriques incluse dans les nuages de points pour la détection sans avoir besoin des images RGB. Ceci donne plus robustesse vis-à-vis les problèmes de variation de luminosité.

Concernant l'architecture de VoteNet, elle est basée sur PointNet++ pour convertir les nuages de points à des structures régulières. PointNet++ est un algorithme de classification et de segmentation sémantique qui est utiliser cette fois-ci pour la détection 3D. Il dispose d'une architecture séquentielle. Au début, VoteNet extrait les caractéristiques des nuages de points, puis effectue un vote pour avoir une estimation du centre. Ensuite, il associe les points à leurs centres pour permettre la détection (voir Figure 4.3).

VoteNet propose de doter son réseau de neuronnes d'un mécanisme de vote 3D qui est similaire au vote de Hough. Le but est de générer des points plus proches du centre. Ces points vont par la suite être groupé pour générer des boites 3D autour des objets détectés.

Concernant les objets entrainés, cette architecture a été testée avec 2 BDs: SUN RGB et ScanNet.

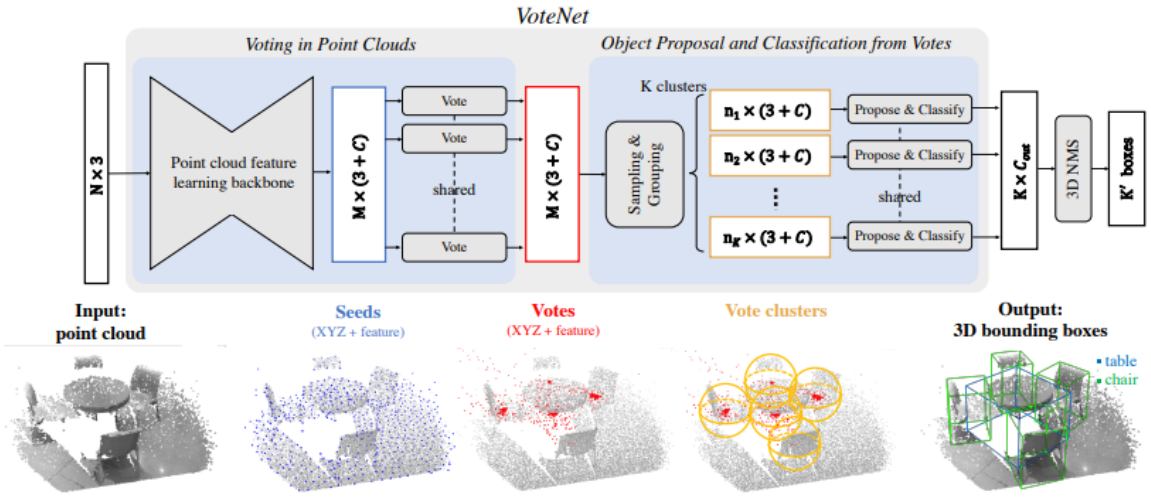

Figure 4.3 : Étapes de détection dans VoteNet

Donc, VoteNet propose précisément une détection d'objet basée sur des informations purement géométriques de la scène en cours. Il n'estime pas la pose 6DoF. Cette architecture, donne une

idée de comment peut-on améliorer la perception du robot dans un environnement qui ne donne pas d'importance aux couleurs des objets. C'est semblable au cas des objets industriels.
La question qui se pose : pourquoi ne pas combiner les 2 types de détections pour tier partie d'une part des images RGB et ce qu'elles comportent d'informations 2D. Et d'autre part, des nuages de points avec tous ce qu'ils contiennent d'informations géométriques ?
Ceci a permis de diriger cette analyse vers PVN3D.

4.3.3 Analyse de PVN3D

Présentement, PVN3D est le seul projet libre de droit qui offre cette possibilité de combiner la détection 2D et 3D pour avoir les avantages des deux. De plus, il fait l'estimation de la pose 6DoF qui est essentielle dans le cadre de ce projet. De plus, il offre les meilleurs résultats avec LineMOD et YCB-video par rapport aux autres projets d'estimation de pose 6DoF.
Quant à l'architecture de PVN3D, la figure 4.4 en montre un aperçu.

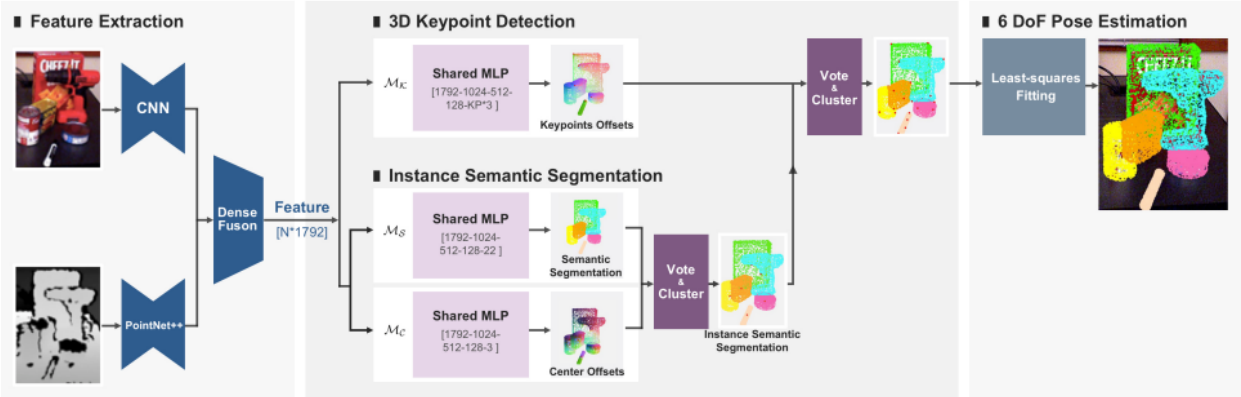

Figure 4.4 : Architecture PVN3D

Premièrement, PVN3D reçoit comme entrée l'image RGB et l'image de profondeur. Il s'inspire de DenseFusion (Wang et al., 2019) pour fusionner les caractéristiques de couleurs liées aux images RGB ainsi que les caractéristiques géométriques liées aux nuages de points. Donc, fusionner les deux pour avoir plus d'informations par rapport aux objets. La figure 4.5 illustre comment DenseFusion extrait les deux types d'information. Ensuite, PVN3D applique un réseau de vote 3D pour détecter les points-clés 3D. Finalement, il estime la pose 6DoF (voir Figure 4.4) avec ajustement par moindres carrés.

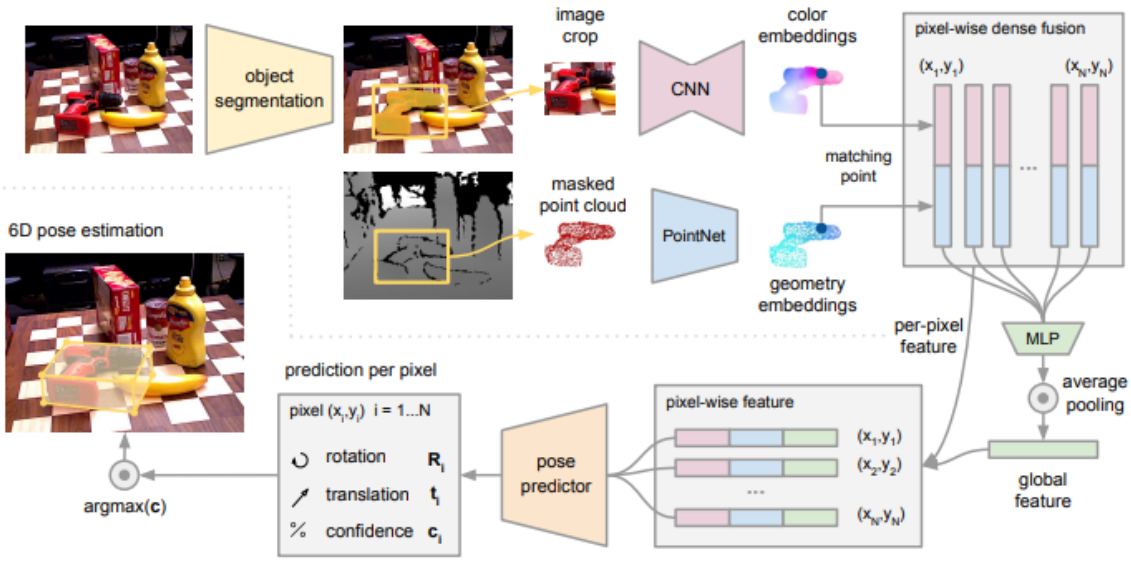

Figure 4.5 : DenseFusion fusionne les caractéristiques géométriques et de couleurs de l'objet

Comme VoteNet, PVN3D utilise PointNet++ pour extraire les caractéristiques des nuages de points. De plus, il propose un réseau de neurone de vote 3D pour choisir les points-clés 3D. De cette façon, il n'a pas besoin de faire une projection 2D à 3D. Soit directement par régression ou même en localisant le centre de l'objet et sa distance par rapport à la caméra qui est la technique utilisée par PoseCNN. Ce qui lui permet d'être approprié pour les objets sans textures qui se trouve dans un environnement à luminosité variable comme les objets industriels.

Comme PoseCNN, PVN3D offre la détection sémantique et par instances. Cette dernière est essentielle pour reconnaître un objet parmi plusieurs pareils. C'est exactement le contexte du présent projet qui est de permettre au bras robotisé de détecter la pièce à l'étude parmi plusieurs dans un contenant.

Ce qui est intéressant aussi dans PVN3D est l'estimation de pose 6DoF évidemment car, dans un problème de *Bin Picking*, la détection d'objets ne permet pas au bras robotisé d'apercevoir avec précision son environnement. Il est donc essentiel d'avoir cette information de pose pour lui permettre d'avoir une idée de comment l'objet est positionné dans l'espace. De cette façon, il peut le manipuler sécuritairement.

De plus, prendre comme entrée l'image RGB donne la possibilité de tirer davantage de ses caractéristiques comme les couleurs de l'objets. De toute évidence, un objet industriel est généralement métallique ou d'une seule couleur. Bien que la couleur ne présente pas une

caractéristique déterminante pour ces objets si on les compare avec les détails géométriques, il reste une information qui a son importance lors de la détection.

Selon son article, PVN3D donne des résultats significativement meilleurs même avec les objets symétriques. Le tableau 4.1 montre une comparaison entre PVN3D, PoseCNN et DenseFusion avec l'ensemble de données YCB-video. C'est pareil en ajoutant l'étape ICP pour raffiner les résultats de la pose 6DoF.

	Without Iterative Refinement						With Iterative Refinement					
	PoseCNN[52]		DF(per-pixel)[50]		PVN3D		PoseCNN+ICP[52]		DF(iterative)[50]		PVN3D+ICP	
	ADDS	ADD(S)	ADDS	ADD(S)	ADDS	ADD(S)	ADDS	ADD(S)	ADDS	ADD(S)	ADDS	ADD(S)
002_master_chef_can	83.9	50.2	95.3	70.7	96.0	**80.5**	95.8	68.1	**96.4**	73.2	95.2	79.3
003_cracker_box	76.9	53.1	92.5	86.9	**96.1**	**94.8**	92.7	83.4	95.8	94.1	94.4	91.5
004_sugar_box	84.2	68.4	95.1	90.8	97.4	96.3	**98.2**	**97.1**	97.6	96.5	97.9	96.9
005_tomato_soup_can	81.0	66.2	93.8	84.7	**96.2**	88.5	94.5	81.8	94.5	85.5	95.9	**89.0**
006_mustard_bottle	90.4	81.0	95.8	90.9	97.5	96.2	**98.6**	**98.0**	97.3	94.7	98.3	97.9
007_tuna_fish_can	88.0	70.7	95.7	79.6	96.0	89.3	**97.1**	83.9	**97.1**	81.9	96.7	**90.7**
008_pudding_box	79.1	62.7	94.3	89.3	97.1	95.7	97.9	96.6	96.0	93.3	**98.2**	**97.1**
009_gelatin_box	87.2	75.2	97.2	95.8	97.7	96.1	98.8	98.1	98.0	96.7	**98.8**	**98.3**
010_potted_meat_can	78.5	59.5	89.3	79.6	93.3	**88.6**	92.7	83.5	90.7	83.6	**93.8**	87.9
011_banana	86.0	72.3	90.0	76.7	96.6	93.7	97.1	91.9	96.2	83.3	**98.2**	**96.0**
019_pitcher_base	77.0	53.3	93.6	87.1	97.4	96.5	**97.8**	96.9	97.5	96.9	97.6	**96.9**
021_bleach_cleanser	71.6	50.3	94.4	87.5	96.0	93.2	96.9	92.5	95.9	89.9	**97.2**	**95.9**
024_bowl	69.6	69.6	86.0	86.0	90.2	90.2	81.0	81.0	89.5	89.5	**92.8**	**92.8**
025_mug	78.2	58.5	95.3	83.8	97.6	95.4	94.9	81.1	96.7	88.9	**97.7**	**96.0**
035_power_drill	72.7	55.3	92.1	83.7	96.7	95.1	**98.2**	**97.7**	96.0	92.7	97.1	95.7
036_wood_block	64.3	64.3	89.5	89.5	90.4	90.4	87.6	87.6	**92.8**	**92.8**	91.1	91.1
037_scissors	56.9	35.8	90.1	77.4	**96.7**	**92.7**	91.7	78.4	92.0	77.9	95.0	87.2
040_large_marker	71.7	58.3	95.1	89.1	96.7	91.8	97.2	85.3	97.6	**93.0**	**98.1**	91.6
051_large_clamp	50.2	50.2	71.5	71.5	93.6	93.6	75.2	75.2	72.5	72.5	**95.6**	**95.6**
052_extra_large_clamp	44.1	44.1	70.2	70.2	88.4	88.4	64.4	64.4	69.9	69.9	**90.5**	**90.5**
061_foam_brick	88.0	88.0	92.2	92.2	96.8	96.8	97.2	97.2	92.0	92.0	**98.2**	**98.2**
ALL	75.8	59.9	91.2	82.9	95.5	91.8	93.0	85.4	93.2	86.1	**96.1**	**92.3**

Tableau 4.1 : Comparaison de PVN3D avec PoseCNN et DenseFusion avec et sans raffinement (les objets symétriques sont en gras)

Afin de montrer l'efficacité de PVN3D de façon plus générale, le tableau suivant (4.2) compare PVN3D avec plusieurs projets faisant l'estimation de pose 6DoF.

Les architectures basées sur une détection 2D comme PoseCNN, PVNet et CDPN ont comme entrée l'image RGB seulement. Donc, certains détails géométriques des objets manquent.

DenseFusion combine les caractéristiques 2D et 3D en entrée mais, ne fait pas un vote 3D comme PVN3D et VoteNet.

SSD-6D et 'Implicit ICP' ont comme entrée l'image RGB et utilise la profondeur pour ajuster les résultats de pose 6DoF.

	RGB			RGBD					
	PoseCNN DeepIM [26, 52]	PVNet [37]	CDPN [27]	Implicit ICP[45]	SSD-6D ICP[22]	Point-Fusion[50]	DF(per-pixel)[50]	DF(ite-rative)[50]	PVN3D
ape	77.0	43.6	64.4	20.6	65.0	70.4	79.5	92.3	**97.3**
benchvise	97.5	**99.9**	97.8	64.3	80.0	80.7	84.2	93.2	99.7
camera	93.5	86.9	91.7	63.2	78.0	60.8	76.5	94.4	**99.6**
can	96.5	95.5	95.9	76.1	86.0	61.1	86.6	93.1	**99.5**
cat	82.1	79.3	83.8	72.0	70.0	79.1	88.8	96.5	**99.8**
driller	95.0	96.4	96.2	41.6	73.0	47.3	77.7	87.0	**99.3**
duck	77.7	52.6	66.8	32.4	66.0	63.0	76.3	92.3	**98.2**
eggbox	97.1	99.2	99.7	98.6	**100.0**	99.9	99.9	99.8	99.8
glue	99.4	95.7	99.6	96.4	**100.0**	99.3	99.4	**100.0**	**100.0**
holepuncher	52.8	82.0	85.8	49.9	49.0	71.8	79.0	92.1	**99.9**
iron	98.3	98.9	97.9	63.1	78.0	83.2	92.1	97.0	**99.7**
lamp	97.5	99.3	97.9	91.7	73.0	62.3	92.3	95.3	**99.8**
phone	87.7	92.4	90.8	71.0	79.0	78.8	88.0	92.8	**99.5**
ALL	88.6	86.3	89.9	64.7	79.0	73.7	86.2	94.3	**99.4**

Figure 4.2 : Comparaison de PVN3D avec plusieurs architectures d'estimation 6DoF

De ce fait, après cet analyse détaillée, PVN3D représente un point de départ promoteur pour effectuer une estimation de pose 6DoF pour les objets industriels complexes.

La section suivante analyse plus en détail les données d'apprentissage qui seront utilisées dans la solution.

4.4 Pourquoi T-LESS?

4.4.1 Analyse par rapport au travail précédent

L'approche adoptée par le travail précédent avec PoseCNN consistait à construire une base de données d'un seul objet qui est : la pièce à l'étude. Cette BD regroupe des images synthétiques d'apprentissage ainsi que leurs masques et leurs fichiers de paramètres. Au moment des tests avec les images réelles, le résultat était décevant. L'algorithme n'était pas capable de détecter la pièce en question. Ceci a été expliqué par le manque des données réelles pour l'entrainement d'un côté. D'un autre côté, les images de profondeur prisent par la caméra RGB-D diffèrent de façon significative d'eux utilisées lors de l'entrainement. De ce fait, la prochaine étape de ce projet était d'entrainer le réseau de neurones avec des images RGB-D réelles ayant leurs masques correspondants ainsi que leurs fichiers de paramètres.

Dans le cadre du présent travail, bien que le choix de remplacer PoseCNN par PVN3D représente une bonne amélioration de l'algorithme de détection, l'adoption de telle base de données influencera significativement les résultats. De ce fait, l'objectif est d'avoir des données d'entrainement réelles mais, comment faire ?

Le choix de construire une nouvelle BD d'entrainement à partir des images RGB-D prisent par la caméra ne requière pas seulement l'étiquetage mais aussi les informations *ground truth* pour

les objets. Ce processus n'est pas seulement fastidieux mais aussi sujet à des erreurs vu qu'on doit le faire manuellement. Sans oublier le but du projet qui vise principalement évaluer le comportement de PVN3D vis -à-vis les objets industriels. De ce fait, la création de cette BD n'est pas priorisée pour le moment et va ralentir l'atteinte des objectifs. Donc, ce choix a été rejeté dans le cadre de ce projet de maîtrise mais, pourra être réalisé si PVN3D démontre une bonne détection avec les objets industriels.

Puisque la pièce à l'étude est un objet industriel, pourquoi ne pas intégrer à PVN3D une BD qui regroupe des objets de ce type et qui, en même temps, formée d'images RGB-D issues de scènes réelles ? En analysant l'état de l'art, T-LESS répond spécifiquement à ce besoin. De cette façon, T-LESS a été choisie comme ensemble de données d'apprentissage au lieu de celui essayé avant avec la pièce à l'étude. Maintenant, T-LESS représente-t-elle un bon choix par rapport à PVN3D ?

4.4.2 Analyse par rapport à PVN3D

Durant l'analyse du choix de l'algorithme PVN3D, il a été indiqué que ce dernier été testé avec deux base de données : LineMOD et YCB-video. Ces deux BD regroupent des objets d'usage quotidien seulement. Le fait que ces objets sont très différents les uns des autres en termes de forme, de couleur, de taille et aussi de texture les rend plus ou moins facilement reconnaissable. Ces objets ne sont pas considérés complexes malgré l'existence de certaines symétries avec certaines poses. Contrairement à la pièce à l'étude, qui est de nature métallique, sans texture, d'une seule couleur et qui peut briller selon la luminosité. De plus, elle a plusieurs poses ayant des symétries qui la rend difficilement reconnaissable, comme la vue du côté droit, du côté gauche, par en haut ou aussi par en bas. Ces caractéristiques la rendent complexe en termes de forme, de couleur et de texture. De ce fait, les caractéristiques des objets des deux BDs LineMOD et YCB-video sont loin de celles de la pièce à l'étude.

En l'occurrence, la pièce en question fait partie de la catégorie des objets industriels. Ces derniers sont généralement d'une seule couleur, et de formes très similaires les unes des autres. De plus, ils sont sans texture et comporte beaucoup de pose de symétrie. C'est exactement le cas de la BD T-LESS. Donc, T-LESS en termes de la nature de ces objets combiné avec PVN3D offre une meilleure solution pour l'objectif qui est la détection des objets complexes industriels. Maintenant, il reste à vérifier si l'intégration de T-LESS est faisable dans le cadre de ce projet de maîtrise.

5 RÉALISATION DU PROJET : IMPLÉMENTATION

5.1 Entrainement et détection avec LineMOD

Comme c'est déjà indiqué, PVN3D fait la détection sur la base de données LineMOD qui regroupe des objets utilisés dans la vie courante. Puisque le présent rapport vise la détection des objets industriels, il est important de commencer par une analyse de l'exécution sur LineMOD avant d'amorcer l'intégration de la nouvelle BD. Pour des raisons de fiabilité et de précision, PVN3D ne va pas être testé directement avec le modèle pré-entrainé. Il est fortement probable que l'auteur de PVN3D utilise une carte graphique plus performante de celle adoptée pour le projet. Donc, il est judicieux de refaire l'entrainement pour qu'on n'aille pas un grand gap de performance entre LineMOD et T-LESS lié aux ressources utilisées.

Figure 5.1 : Exemple d'image de couleur, de profondeur et de masque pour l'objet caméra

Étant donné que chaque objet comporte environ 1235 images RGB (697 MO) et un nombre pareil pour leurs masques et images de profondeur (voir la figure 5.1). Cela montre que cette base de données est assez petite pour faire de l'apprentissage profond. Pour entrainer PVN3D avec LineMOD, la première étape était d'étendre artificiellement la base de données existante. De ce fait, le projet *Raster Triangle* (*GitHub - ethnhe/raster_triangle: A simple renderer with z-buffer for synthesis data generating.*, s. d.) a été utilisé. Ce dernier a permis de générer de nouvelles images formées en combinant les objets LineMOD (un objet à la fois) avec un arrière-plan de la base de données SUN2012pascalformat. Donc, le résultat était environ 69998 fichiers (172 GO) pour chaque objet. La figure 5.2 en montre un exemple de ces images ainsi que son masque associé.

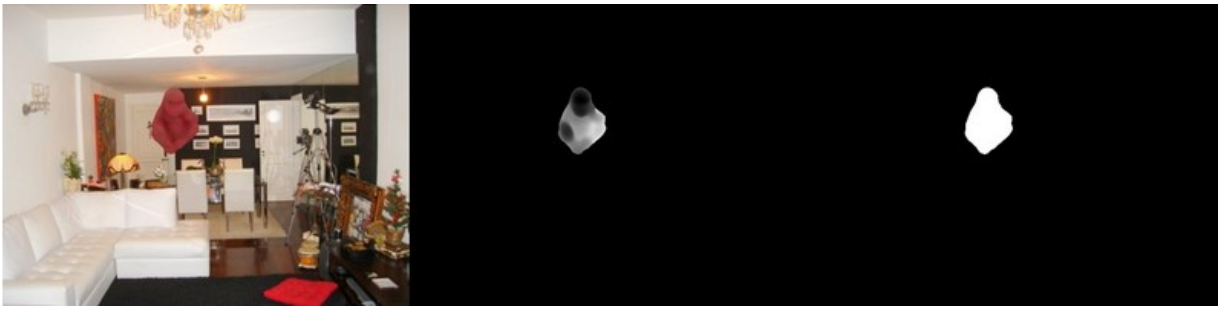

Figure 5.2 : Exemple d'images générées en positionnant des objets LineMOD sur un arrière-plan SUN2012pascalformat

Le projet *Raster Triangle* permettait aussi de générer des images contenant plusieurs objets sur la même scène. Le résultat était environ 9584 fichiers (24 GO) pour chaque objet. En voici un exemple (voir Figure 5.3).

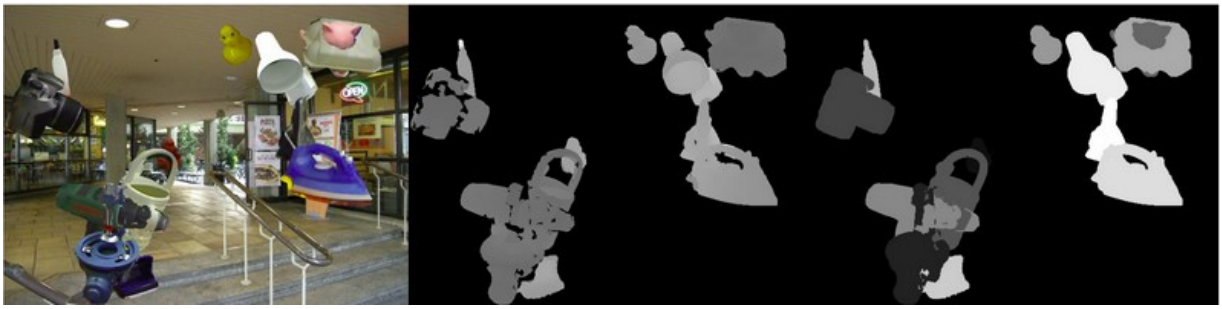

Figure 5.3 : Exemple d'images générées avec plusieurs objets LineMOD

Lors de l'entrainement du PVN3D sur la carte graphique « NVIDIA GeForce RTX 2070S 8 GO» la mémoire était insuffisante. Une des solutions était de réduire la taille de la 'batch'. Cette dernière était bien réussie mais, elle a pris une dizaine d'heures d'entrainement. Comme pistes de solution, il y avait la possibilité d'adapter *Raster Triangle* à nos besoins en générant moins de données. Ou encore, exécuter une version réduite de PVN3D en réduisant le nombre de paramètres d'entrée. Ce sont des techniques sont utiles, mais le besoin du présent projet est de tester PVN3D dans sa forme originale et ne pas contourner ce type de problèmes.

5.2 Intégration de T-LESS à PVN3D

Cette intégration commence par effectuer un prétraitement à TLESS. Donc, il est essentiel de d'analyser plus en détail ses composants. Dans ce contexte, T-LESS est une base de données qui regroupe des images liées à une trentaine d'objets industriels qui sont sans texture, n'ont pas de couleur discriminant où certains objets sont des parties d'autres. Ces images sont réalisées à

partir de 3 types de caméra. Dans ce projet, puisque l'entrainement et les tests se font par objets, un à la fois, il n'était pas nécessaire de commencer avec tous les objets d'un coup. C'est pour cette raison que T-LESS a été utilisée en partie. Seulement les images et les fichiers de paramètres liés aux objets de 0 à 6 et prisent par la caméra *Primesense* ont été retenues.

T-LESS contient des données d'entrainement et de tests. Les images des données d'entrainement sont seulement sous la forme d'un objet dans une scène avec différentes poses (voir Figure 5.4). Ces images sont utiles pour faire apprendre au réseau les différentes poses, cependant elles ne seront pas le bon moyen pour entrainer les divers cas d'occlusion et de troncation.

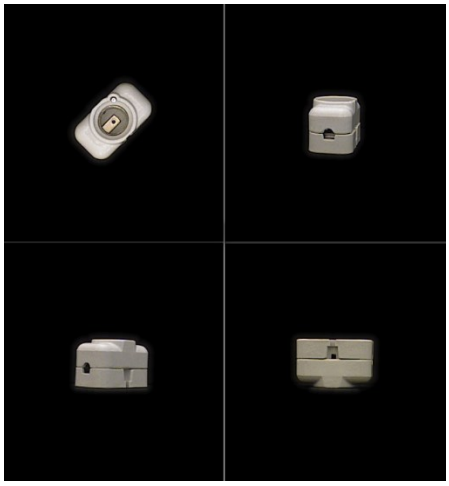

Figure 5.4 : Exemples des données d'entrainement de T-LESS (objet 5[1])

Les données de tests de leurs côtés sont organisées selon des scènes. Chacune d'entre elles contient un ensemble d'objets fixe qui sert à produire des images de différentes poses dans la même scène. Les figures 5.5 montre des images de deux scènes différentes.

[1] 5 est l'identificateur de cet objet dans la base de données T-LESS

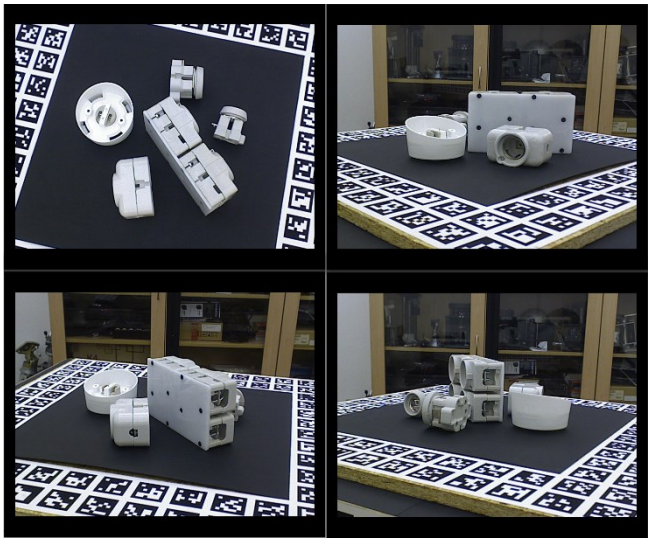

Figure 5.5(a) : exemple d'images liées à une scène contenant 5 objets

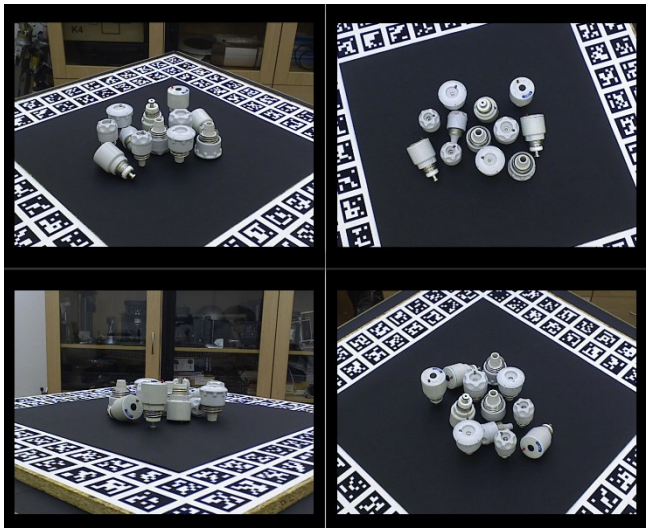

Figure 5.5(b) : exemple d'images liées à une scène contenant 12 objets

Dans le cas du présent projet, vu les divers cas d'occlusion et de troncation (voir Figure 5.6), ces données de tests sont les plus pertinentes pour entrainer le réseau PVN3D.

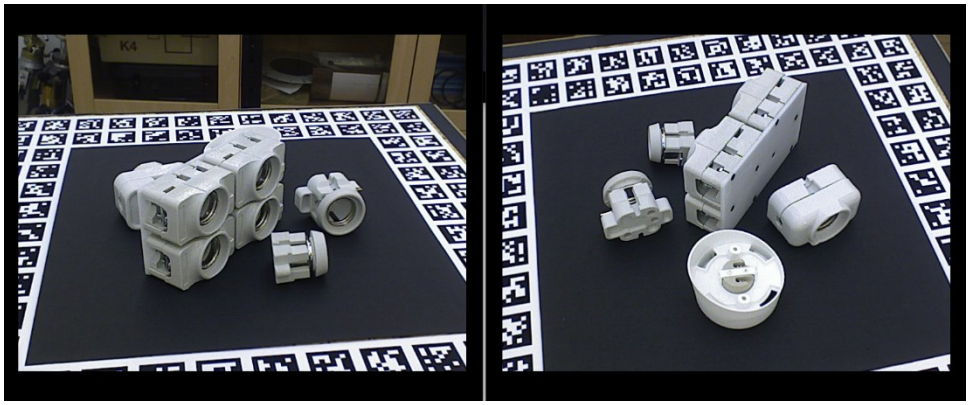

Figure 5.6 : Exemple de cas d'occlusion de l'objet 5[2]

Dans le but d'intégrer T-LESS à PVN3D, cette étape de réalisation de projet permet de procurer à T-LESS un format qui est semblable à LineMOD. De ce fait, un prétraitement de cet ensemble de données est primordial. Pour ce faire, l'intégration de T-LESS à PVN3D comporte deux étapes. La première est de modifier Cette BD pour avoir un format acceptable par l'architecture de PVN3D. Deuxièmement, apporter des modifications au code PVN3D pour ajouter cet ensemble de données.

Avant tout, pour télécharger T-LESS, il fallait l'installer à partir de T-LESS Toolkit (Hodan, 2016/2022) pour télécharger les scènes désirées.

Étant donné que les images T-LESS disponibles dans la version courante sont organisées selon des scènes, donc un objet peut se manifester dans plusieurs scènes. De ce fait, il fallait réorganiser la base de données selon les objets où toutes les scènes et les poses d'un objet doivent être mises ensembles. Puis associer à chaque image RGB son image de profondeur et son masque. Autrement dit, il fallait changer la structure de T-LESS, au lieu d'avoir des sous-ensembles de données pour chaque scène, il fallait les rendre pour chaque objet. En l'occurrence, vu le nombre élevé d'images (environ 9000 fichiers), il serait fastidieux et quasiment impossible de faire cette réorganisation à la main. Donc, cette étape est faite en programmant ses changements en langage Python.

Bien évidemment, organiser la base de données selon les objets implique beaucoup plus de changements dans les fichiers de paramètres contenant les informations de poses des objets ainsi que celles de la caméra. Dans cette étape il fallait refaire ces fichiers en respectant

[2] L'objet 5 est illustré dans la figure 5.4

soigneusement le format de ceux de LineMOD pour pouvoir utiliser PVN3D. Cette régénération des fichiers de paramètres a été implémenté en langage Python.

Concernant la deuxième étape de l'intégration de T-LESS, étant donné que le code de PVN3D intègre 2 ensembles de données LineMOD et YCB-video pour chaque étape de son architecture il fallait y ajouter la BD T-LESS. De cette façon, il peut fonctionner avec les trois. Autrement dit, l'adaptation de PVN3D à de nouvelles BDs consiste à les intégrer dans chaque étape de son fonctionnement.

6 RÉALISATION DU PROJET : ENVIRONNEMENT ET RESSOURCES UTILISÉES

6.1 Environnement logiciel et Carte graphique

Ce projet dispose d'une caméra de type « RealSense D435 » qui sera installée sur le bras robotisé. Il permettra de capturer les images RGB-D des objets du bac durant l'opération de *Bin Picking*. Selon (Yan et al., 2020), cette caméra RGB-D est un dispositif permettant d'avoir l'information sur un niveau commerciale ou même industriel. Elle capture l'apparence et la géométrie de la scène. De plus, elle peut déduire la pose des objets peu texturés même en cas du mal éclairage. Aussi, elle est efficace pour construire une base de données d'objets dans des poses aléatoires.

« RealSense D435 » est considérée une solution à faible coût qui est idéale pour le développement ainsi que pour la création rapide de produits robotisés (*Depth Camera D435*, s. d.).

L'outil logiciel « RealSense SDK 2.0 », qui vient avec le produit, permet de visualiser et de capturer les images de profondeur et les nuages de points.

Concernant la carte graphique, le poste de travail sera doté d'une « NVIDIA GeForce RTX 2070S » qui est particulièrement dédiée à l'intelligence artificielle haute performance.

La solution sera implémentée dans un environnement Linux (Ubuntu 18.04) en Python avec le l'infrastructure logicielle Pytorch. De plus, elle utilisera CUDA 10.2 pour effectuer l'exécution sur la carte graphique « NVIDIA GeForce RTX 2070S 8G».

6.2 Choix de ressources

Le fait que ce projet entraine de grandes quantités d'images avec de l'apprentissage profond le rend gourmand en termes de ressources surtout la mémoire. De ce fait, la carte graphique « NVIDIA GeForce RTX 2070S 8G» n'est pas suffisante vue quelle envoie une erreur de dépassement de mémoire. Donc, il fallait chercher d'autres ressources.

L'Université de Sherbrooke dispose d'une plateforme de calcul puissante nommé 'Mammouth'. Cette dernière offrait seulement la possibilité de calcul avec CPUs.

Vue qu'aucune GPU n'était mise à disposition, la possibilité d'utiliser un serveur Amazon AWS a été étudiée. Le choix s'orientait vers une instance EC2 qui propose plusieurs instances GPU. Aucune instance GPU n'était offerte durant la période gratuite. L'instance *p2.xlarge* ayant une carte graphique Tesla 61GB est amplement suffisante. Cette dernière avait une tarification de 0.9$ par heure.

Il y avait aussi un serveur de calcul dédié aux membres de 3IT et qui était en train d'être mis en fonction.

En l'occurrence, Calcul Canada offrait une plateforme nationale de calcul informatique de pointe, maintenant elle est connue sous le nom de 'l'Alliance de recherche numérique du Canada'. Le présent projet a opté pour cette option vue qu'elle est gratuite pour les chercheurs.

6.3 Calcul Canada

L'*Alliance de recherche numérique du Canada* donne accès à plusieurs grappes de calcul utilisant des cartes graphiques comme Belouga, Cerdar et Graham. La grappe Cedar, la plus utilisée dans ce projet, offre les générations de cartes Tesla suivantes :

- 4 x NVIDIA P100 (12G)
- 4 x NVIDIA P100 (16G)
- 4 x NVIDIA V100 (32G)

Cette grappe de calcul permet un accès via le protocole SSH, pour pouvoir travailler dans l'espace personnel ou du groupe réservé du nœud de calcul. Pour transférer des données d'un ordinateur vers le nœud ou vice versa il est possible d'utiliser la commande '*SCP*' (*Secure Copy Protocol*). Ce type de transfert était quand même long (environ 8h pour 9G). Récemment, des nœuds de transfert ont été mise à disposition pour ce type de tâches afin de libérer les nœuds de calcul.

Pour exécuter le code PVN3D, il fallait commencer par installer les dépendances de l'environnement de développement. Le nœud de calcul permet deux types d'installations :

- Installation des modules : avec la commande *:$module load cuda*
- Installation des packages : *$pip install pytorch*

Puisque les packages et les versions disponibles sont limités, il y avait des dépendances de PVN3D qu'on ne pouvait pas installer directement dans le nœud de calcul. D'un autre côté, toute installation de dépendance en dehors de celles déjà fournies déclenche une erreur de privilège d'utilisation. Évidemment, une installation nécessite les droits *root* pour être

accomplie. Toutefois, les utilisateur des nœuds de calcul n'ont pas ces privilèges, ils ne sont que de simples utilisateurs. De ce fait, l'utilisation des nœuds de calculs avec des logiciels déjà mis à disposition rond la tâche de se bénéficier de ces équipements technologiques de pointe plus pratique. Cependant, dans le cas des programmes complexes ayant des dépendances manquantes l'utilisateur est mis devant de nombreux défis d'installation.

Comme solution à ce problème, le choix a été orienté vers la conteneurisation. Le choix le plus courant est Docker. Toutefois, les grappes de calculs n'utilise pas ce dernier pour des raisons de sécurité vu que dans un conteneur docker toutes les commandes sont faites en tant que *root*. De ce fait, ce HPC (High-Performance Computing) proposent S*ingularity* qui, quant à elle, permet d'exécuter des commandes en tant que simple utilisateur. Donc, l'idée est d'avoir une image *Singularity* avec toutes les dépendance pré-installées et de l'exécuter par la suite dans le nœud de calcul. Il y avait deux façons de faire qui ont été réalisées :

- Installer S*ingularity* en tant que *root* pour avoir les droits du super utilisateur. Puis créer une image *Singularity* avec le code et les dépendances pré-installées.
- Installer docker et créer une image contenant le code et toutes les dépendances. Par la suite, convertir cette image Docker en une image *Singularity*.

À cette étape, l'image *Singularity* ainsi que la base de donnnées LineMOD ont été transférés vers un *login node* de la grappe Cedar. Pour exécuter le code sur un nœud de calcul, à chaque fois, il faut préalablement lancer une *job* avec SLURM pour se connecter à un nœud de calcul ayant une carte graphique et le réserver pour une durée précise. C'est à ce moment-ci que l'image *Singularity* du code a été exécutée. Plusieurs flag était utils:

- *--nv* : avec *Singularity run* pour permettre à Singularity d'utiliser le pilote GPU de NVIDIA.
- *-Y* ou *–X* : lors d'une connection *SSH* pour pouvoir afficher les images.
- *--x11* : lors du lancement de la *job* SLURM avec *Salloc* pour afficher les images RGB sur le nœud de calcul.

Il s'avère important de mentionner que les deux images Singularity et Docker réalisées fonctionnaient parfaitement dans un poste de travail ordinaire et l'entrainement de PVN3D était considérablement long mais arrivait à s'exécuter avec succès sur celui-ci. Toutefois, dans un nœud de calcul il fallait réinstaller PointNet++ pour résoudre des erreurs du *cuda kernel*. Ce qui mène à son tour au problème des privilèges *root* pour effectuer l'installation.

Pour conclure, l'utilisation d'une grappe de calcul d'un HPC représente un excellent choix pour l'entrainement des réseaux de neurones profonds sur des cartes graphiques puissantes, surtout pour les programme volumineux et gourmands en termes de mémoires. Le présent projet, représente une bonne exploration des défis rencontrées lors de cette utilisation.

7 RÉALISATION DU PROJET : RÉSULTATS ET DISCUSSION

7.1 Entrainement avec LineMOD

Figure 7.1 : Exemple du résultat de la détection PVN3D avec l'objet « Gorille »

Diminuer considérablement la *batch* ne vise pas la précision de détection mais juste découvrir si d'un côté l'apprentissage va finir correctement, et de l'autre côté estimer les besoins de PVN3D en termes de ressources. De ce fait, cette configuration ne donne pas une bonne courbe d'apprentissage, mais plutôt avec d'importantes fluctuations. Donc, cet hyperparamètre étant la *batch* doit être ajusté par la suite pour éviter les fluctuations. L'apprentissage a été terminé, toutefois due au manque d'espace mémoire dans la carte graphique NVIDIA GeForce RTX 2070S, le réseau de neurones profond n'a pas abouti à une solution optimale.

De ce fait, visuellement on remarque que PVN3D à bien détecté l'objet rouge (Gorille) sur l'image 7.1, mais pas nécessairement dans la majorité des cas. Dans l'attente d'avoir les ressources pertinentes pour exécuter PVN3D, on s'est contenté par une évaluation visuelle de différentes images. Dans ce contexte, le but de cette partie ce n'est pas valider la précision de la détection et comparer avec les résultats de l'article mais plutôt un essai pour étudier la pertinence de la solution et ce dont elle a besoin en termes de ressources.

Donc, comme conclusion, pour faire l'entrainement de PVN3D dans sa forme originale, sans diminuer ses paramètres ni l'entrainer avec moins de données, il est essentiel de l'exécuter sur une carte graphique ayant plus de 8G en mémoire.

7.2 Intégration de T-LESS

Dans le cadre de l'intégration de T-LESS, et vu la grande quantité de données à modifier, plusieurs modules disponibles publiquement offrent un soutient par rapport à plusieurs aspects liés à l'estimation de pose 6DoF. Donc, il fallait y appliquer les changements nécessaires et les adapter à ses besoins.

Afin de s'assurer que les poses *groud-truth* sont bel et bien correspondantes au objets affichés, Sixd_toolkit (Hodan, 2016/2023) a été utilisé pour les visualiser. Ceci a permi de valider que chaque objet est bien localisé. Donc, c'est un bon indice pour commencer le pré-traitement sur T-LESS. La figure suivante donne un exemple des images générées.

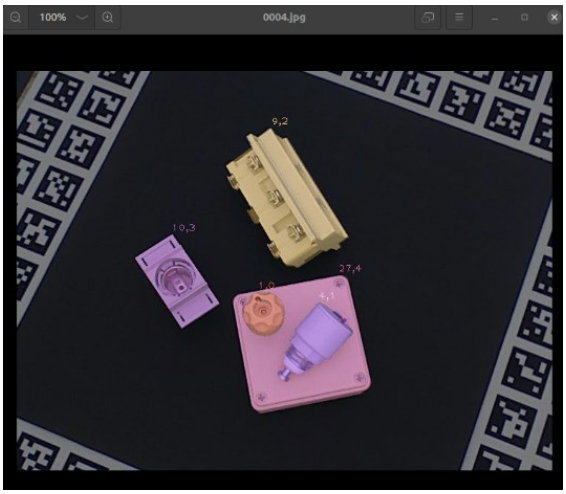

Figure 7.2 : Exemple de visualisation de poses *ground-truth* pour T-LESS

Pour chaque image il fallait générer les masques des objets qui y appartient en apportant des modifications à Sixd_toolkit. Les masques générés sont enregistrés à l'emplacement adéquat. Donc, chacun d'entre eux est associé au bon objet et à la bonne image RGB. La figure 7.3 montre un exemple de masques générés pour T-LESS. La partie suivante traite la génération des fichiers de paramètres.

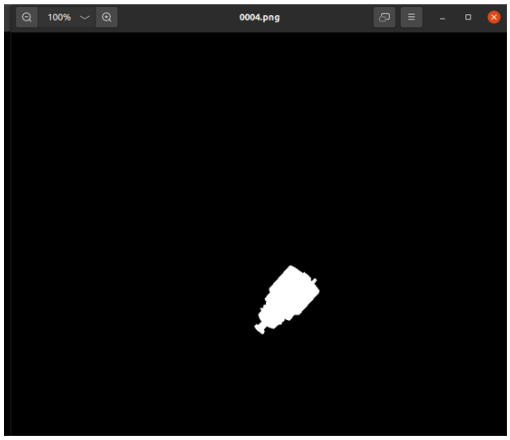

Figure 7.3 : Exemple de masques générés pour les objets T-LESS

PVN3D utilise 8 points-clés (*keypoints*) ainsi que le centre pour chaque objet. Le projet Clean_pvnet (*GitHub - zju3dv/clean-pvnet: Code for « PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation » CVPR 2019 oral*, s. d.) dispose de certains outils importants parmi lesquels la génération de ces points. Il applique l'algorithme FPS sur le modèle 3D, qui est un fichier *.ply*, de l'objet pour les définir (voir Figure 7.4(a)). De plus, d'autres informations ont été générées : le rayon, le centre de l'objet et les 8 points de la boîte entourant l'objet (*bounding box*) lors de la détection. Les figures 7.4(b) et 7.4(c) montrent un aperçu.

pvn3d/datasets/tless/tless_obj_kps/01/obj_01_fps.txt

```
1  +-0.0088708 -0.008226 0.0162393
2  +0.0035927 0.0035615 -0.0305659
3  +0.012555299 0.0111471005 0.0003977
4  +0.0136514 0.0055168 0.0293014
5  +-0.0121603 0.0034409 -0.008264599
6  +0.0083965 -0.0136483 -0.0009816
7  +-0.0050793 0.0167133 0.0173034
8  +0.011338101 -0.013330401 0.020100001
```

Figure 7.4(a) : Exemple des 8 *keypoints* générés par l'algorithme FPS

pvn3d/datasets/tless/tless_obj_kps/01/obj_01_corners.txt

```
1  +-0.0174958 -0.0174958 -0.0306
2  +-0.0174958 -0.0174958 0.0306
3  +-0.0174958 0.0174958 -0.0306
4  +-0.0174958 0.0174958 0.0306
5  +0.0174958 -0.0174958 -0.0306
6  +0.0174958 -0.0174958 0.0306
7  +0.0174958 0.0174958 -0.0306
8  +0.0174958 0.0174958 0.0306
```

Figure 7.4(b) : Exemples de 8 coins du *boundingbox* générés

pvn3d/datasets/tless/tless_obj_kps/01/obj_01_radius.txt

```
1  +0.039351820945739746
```

pvn3d/datasets/tless/tless_obj_kps/02/obj_02_center.txt

```
1  +0.0 0.0 0.0
```

Figure 7.4(c) : Exemple de génération du rayon et du centre

Quant aux fichiers de paramètres, ils ont été générés en adaptant Sixd_toolkit. Le fichier *info.yml* et *gt.yml* contiennent les paramètres liés à la caméra et à la position *ground-truth* respectivement (voir annexe A et B). Les fichiers train.txt pour l'entrainement ainsi que test.txt ont été réalisés (voir Annexe C et D). Le fichier test.txt va être utilisé pour valider l'intégration de T-LESS par la suite.

Tous les résultats précédents sont considérés partiels. Le résultat final est celui qui est obtenu après avoir inclus cette base de données résultante Preprocessed T-LESS dans les différents composants de PVN3D. La matrice intrinsèque de la caméra Primesense introduite dans PVN3D est :

```
'tless': np.array([[1075.65091572, 0.,         641.068883438],
                   [0.,         1073.90347929,  507.72159802],
                   [0.,         0.,             1.]]),
```

.

Comme résultat final, l'exécution du script qui test l'intégration de *Preprocessed T-LESS* à PVN3D a bien réussi. On remarque que l'objet 5 est bien été détecté. De plus, les 8 points-clés en rouge ainsi que le centre en bleu sont exactement dans la bonne place. La figure suivante illustre ce résultat :

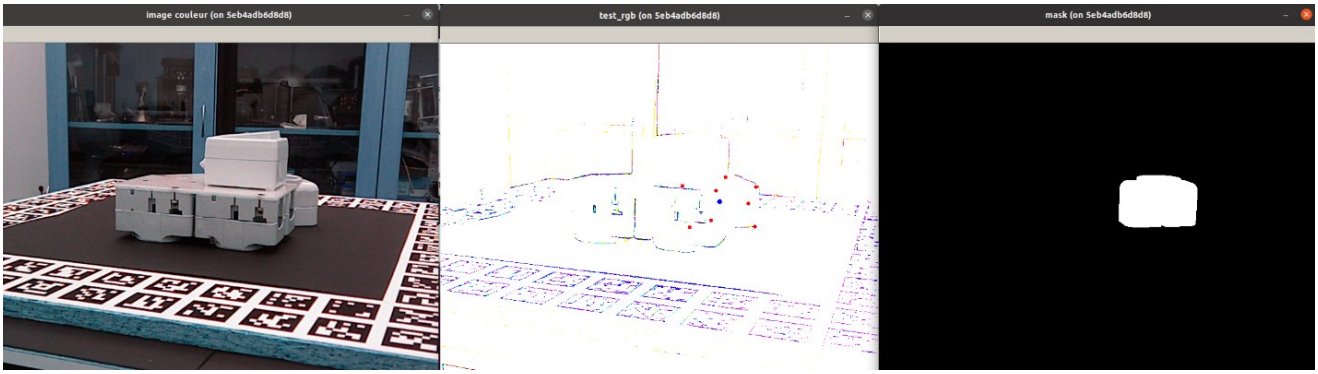

Figure 7.5 : Aperçu du résultat final de l'intégration de *Preprocessed T-LESS* à PVN3D

7.3 Discussion et étapes suivantes du projet

Le présent projet contribue du côté conception et implémentation dans l'objectif d'améliorer l'estimation de pose 6DoF pour les objets industriels complexes et en conséquence pour la pièce à l'étude. Concernant son analyse du problème ainsi que sa vision critique vis-à-vis les différentes architectures de l'état de l'art, il contribue par une solution prometteuse qui est d'ajouter la précision de détection de PVN3D aux objets industriels de l'ensemble de données T-LESS. Pratiquement, ce travail contribue par la tâche déterminante et primordiale étant de réaliser un prétraitement à la base de données T-LESS et d'intégrer cet ensemble de données résultant à l'implémentation de PVN3D.

Selon l'entrainement réalisé de PVN3D avec LineMOD et l'utilisation de la grappe de calcul Cedar, il est essentiel de se procurer des ressources nécessaires pour entrainer PVN3D. Le cas échéant, il faut entrainer PVN3D avec *Preprocessed T-LESS* et analyser les résultats obtenus. Cela aura pour objectif de tester la précision de cette architecture vis-à-vis les objets industriels complexes. Dans le cas où ça donne les résultats attendus, il sera le temps d'intégrer la pièce à l'étude à PVN3D. Cette étape peut se faire de 2 façons :

- Intégrer la pièce à l'étude à T-LESS, comme la démarche suivie durant le travail précédent qui l'on intégré à LineMOD.
- Construire un nouvel ensemble de données dédiées à la pièce en question ou même avec d'autres objets si c'est possible. Puis y adapter PVN3D. Il serait possible de tirer parti des outils déjà vus avec *Preprocessed T-LESS*.

Afin de conserver l'avantage du modèle pré-entrainé avec T-LESS, il est aussi possible de faire du *transfer learning* pour l'objet à l'étude.

Pour que PVN3D apprenne plus de *patterns* avec T-LESS, il y a aussi la possibilité de l'entrainer avec la base de données PBR (*BOP: Benchmark for 6D Object Pose Estimation*, s. d.). Toutefois, cette BD nécessite un prétraitement supplémentaire pour être utilisé avec PVN3D. Pour ce faire, il faut absolument changer sa structure de fichiers. Contrairement à T-LESS, cette BD n'est pas organisée selon des scènes. De plus, chaque image dispose des images de masques pour chaque objet T-LESS qui y appartient. Autrement dit, les étapes de prétraitement effectuées à la version courante de T-LESS ne seront pas toutes pertinentes pour PBR. Toutefois, certains outils et fichiers implémentés à cette fin pourront être utile en y apportant des modifications.

PVN3D pour le moment avec T-LESS, pourra faire la détection d'une seule instance d'un objet à la fois. De ce fait, pour lui permettre de détecter plusieurs instances du même objet sur une image, il faut faire les changements par rapport à YCB-video.

Un autre point important est que *Preprocessed T-LESS* convient aussi à FFB6D, il serait intéressant de le tester cette nouvelle amélioration de PVN3D. De cette façon le bras robotisé aura plus de rapidité et de précision. Étant donné que l'algorithme d'estimation de pose 6DoF sera exécuté en temps réel sur le robot, sa vitesse de perception de son environnement est un élément critique et essentiel pour être à jour des changements pouvant avoir lieu dans les scènes capturées. Ce qui procure aussi plus de sécurité à la manipulation robotique.

Bien que FFB6D constitue une version améliorée de PVN3D, ce dernier reste un excellent choix par rapport aux autres algorithmes d'estimation de pose 6DoF. On peut aussi se contenter de PVN3D car il est de nature modulaire. Ceci est un point intéressant vu qu'après les tests avec *Preprocessed T-LESS* il y aura la possibilité de modifier et d'améliorer son architecture selon les besoins.

8 CONCLUSION

Ce travail explore les principales approches d'estimation de pose 6DoF avec l'apprentissage profond dans les applications de *Bin picking*. Cette étude couvre d'un côté celles utilisées dans un contexte général avec les objets de tous les jours, et d'un autre côté celles spécifiquement dédiés aux composants industriels. Vu que l'estimation de pose 6DoF repose grandement sur la détection d'objets, et que les caméras RGB-D sont de plus en plus répandues, les techniques de détection 2D ainsi que 3D ont été analysées.

De plus, ce projet représente aussi une étude traitant la relation indéniable et fondamentale entre l'architecture de la détection et la nature de l'objet à détecter. Ceci explique clairement comment une détection peut s'avérer fiable alors qu'elle ne l'est pas. Comment la symétrie, la forme, la couleur et la texture de l'objet ainsi que la variation de luminosité et le niveau d'occlusion présent dans la scène influence amplement le choix de l'architecture de détection. Cette réflexion structurée donne un aperçu des problèmes rencontrés durant le travail précédent avec PoseCNN et la pièce à l'étude, et en même temps dessine les pistes de la solution à envisager.

Dans ce contexte, à la suite de l'analyse de l'état de l'art ainsi que de la problématique, la solution sera basée sur un travail récent PVN3D qui, pour la première fois, extrait les caractéristiques 2D et 3D et utilise un réseau profond de vote 3D pour choisir les points-clés 3D (*3D keypoints*). Cette architecture couvre les différents aspects concernant un objet complexe. Toutefois, PVN3D est entrainé avec des données de tous les jours, ce qui est loin de notre contexte. En ce sens, après une analyse de l'état de l'art ainsi que des objectifs du projet, ce travail contribue par étendre cette approche d'estimation 6DoF aux objets industriels complexes. Autrement dit, il réalise ainsi l'intégration de la base de données industriels T-LESS à PVN3D. De ce fait, C'est cette combinaison qui fait de la solution proposée une solution prometteuse.

En ce qui a trait à l'impact du projet, cette perception avancée du robot permettra aux entreprises et particulièrement aux entreprises aéronautiques d'accélérer leur assemblage de pièces et de multiplier ainsi leur productivité.

ANNEXES

Annexe A: Info.yml

```
+0:
+  cam_K: [1075.65091572, 0.00000000, 372.06888344, 0.00000000, 1073.90347929, 255.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+1:
+  cam_K: [1075.65091572, 0.00000000, 372.06888344, 0.00000000, 1073.90347929, 255.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+2:
+  cam_K: [1075.65091572, 0.00000000, 373.06888344, 0.00000000, 1073.90347929, 256.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+3:
+  cam_K: [1075.65091572, 0.00000000, 373.06888344, 0.00000000, 1073.90347929, 256.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+4:
+  cam_K: [1075.65091572, 0.00000000, 374.06888344, 0.00000000, 1073.90347929, 256.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+5:
+  cam_K: [1075.65091572, 0.00000000, 374.06888344, 0.00000000, 1073.90347929, 256.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+6:
+  cam_K: [1075.65091572, 0.00000000, 374.06888344, 0.00000000, 1073.90347929, 257.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+7:
+  cam_K: [1075.65091572, 0.00000000, 375.06888344, 0.00000000, 1073.90347929, 257.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+8:
+  cam_K: [1075.65091572, 0.00000000, 375.06888344, 0.00000000, 1073.90347929, 257.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+9:
+  cam_K: [1075.65091572, 0.00000000, 376.06888344, 0.00000000, 1073.90347929, 258.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
+10:
+  cam_K: [1075.65091572, 0.00000000, 376.06888344, 0.00000000, 1073.90347929, 258.72159802, 0.00000000, 0.00000000, 1.00000000]
+  depth_scale: 0.10000000
```

Annexe B: gt.yml

```
+0:
+  cam_R_m2c: [0.34211295, -0.85906128, 0.38075265, 0.09968990, -0.36973420, -0.92377449, 0.93435572, 0.35399215, -0.04085136]
+  cam_t_m2c: [42.58652183, 60.46795431, 698.08398428]
+  obj_bb: [406, 293, 78, 115]
+  obj_id: 4
+1:
+  cam_R_m2c: [0.35548347, -0.81630812, 0.45527137, 0.12936309, -0.43943028, -0.88891314, 0.92568718, 0.37488923, -0.05061034]
+  cam_t_m2c: [36.21120270, 64.35236549, 696.97540963]
+  obj_bb: [392, 299, 85, 113]
+  obj_id: 4
+2:
+  cam_R_m2c: [0.36650453, -0.76712652, 0.52648956, 0.16024757, -0.50536370, -0.84789677, 0.91651241, 0.39512635, -0.06228847]
+  cam_t_m2c: [29.45151226, 67.69315961, 695.90169118]
+  obj_bb: [379, 305, 91, 112]
+  obj_id: 4
+3:
+  cam_R_m2c: [0.37456697, -0.71228462, 0.59359042, 0.19187330, -0.56679372, -0.80120553, 0.90712994, 0.41399954, -0.07563349]
+  cam_t_m2c: [22.42334605, 70.43457654, 695.03276810]
+  obj_bb: [365, 309, 96, 110]
+  obj_id: 4
+4:
+  cam_R_m2c: [0.37975685, -0.65201306, 0.65624942, 0.22410784, -0.62342027, -0.74908108, 0.89753022, 0.43153970, -0.09062663]
+  cam_t_m2c: [15.15501400, 72.58603203, 694.32807076]
+  obj_bb: [351, 313, 101, 107]
+  obj_id: 4
+5:
+  cam_R_m2c: [0.38174975, -0.58680321, 0.71409311, 0.25677508, -0.67485783, -0.69183285, 0.88788126, 0.44746850, -0.10695034]
+  cam_t_m2c: [7.77557079, 74.11104423, 693.80837936]
+  obj_bb: [337, 317, 105, 102]
+  obj_id: 4
+6:
+  cam_R_m2c: [0.38108590, -0.51707818, 0.76642296, 0.28944435, -0.72058050, -0.63006869, 0.87806353, 0.46194664, -0.12493808]
+  cam_t_m2c: [0.22456111, 75.01024322, 693.43299121]
+  obj_bb: [322, 321, 109, 97]
+  obj_id: 4
+7:
+  cam_R_m2c: [0.37729031, -0.44352788, 0.81297940, 0.32197163, -0.76027482, -0.56419642, 0.86832440, 0.47462204, -0.14404067]
+  cam_t_m2c: [-7.36471883, 75.28164331, 693.19960440]
+  obj_bb: [310, 323, 110, 92]
+  obj_id: 4
+8:
+  cam_R_m2c: [0.37053261, -0.36663574, 0.85339576, 0.35400419, -0.79370004, -0.49469302, 0.85871235, 0.48540526, -0.16430093]
+  cam_t_m2c: [-14.93682642, 74.91816151, 693.22315527]
+  obj_bb: [296, 324, 113, 86]
+  obj_id: 4
+9:
+  cam_R_m2c: [0.36091849, -0.28690054, 0.88737073, 0.38535265, -0.82059764, -0.42204577, 0.84925879, 0.49427443, -0.18561131]
+  cam_t_m2c: [-22.46652357, 73.91864925, 693.44487795]
+  obj_bb: [284, 326, 114, 80]
+  obj_id: 4
+10:
+  cam_R_m2c: [0.34845218, -0.20490892, 0.91465538, 0.41601160, -0.84062859, -0.34681106, 0.83994966, 0.50135404, -0.20767456]
+  cam_t_m2c: [-29.88322670, 72.27272493, 693.76115657]
+  obj_bb: [271, 326, 116, 75]
```

Annexe C: Extrait de train.txt

```
0000
0006
0012
0018
0024
0030
0036
0042
0048
0054
0060
0066
0072
0078
0084
0090
0096
0102
0108
0114
0120
0126
0132
0138
0144
0150
0156
0162
0168
0174
0180
0186
0192
0198
0204
0210
0216
0222
0228
0234
0240
```

Annexe D : Extrait de test.txt

```
0001
0002
0003
0004
0005
0007
0008
0009
0010
0011
0013
0014
0015
0016
0017
0019
0020
0021
0022
0023
0025
0026
0027
0028
0029
0031
0032
0033
0034
0035
0037
0038
0039
0040
0041
0043
```

RÉFÉRENCES

Blank, A., Hiller, M., Zhang, S., Leser, A., Metzner, M., Lieret, M., Thielecke, J., & Franke, J. (2019). 6DoF Pose-Estimation Pipeline for Texture-less Industrial Components in Bin Picking Applications. *2019 European Conference on Mobile Robots (ECMR)*, 1-7. https://doi.org/10.1109/ECMR.2019.8870920

Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J. (2019). YOLACT : Real-Time Instance Segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9156-9165. https://doi.org/10.1109/ICCV.2019.00925

BOP: Benchmark for 6D Object Pose Estimation. (s. d.). Consulté 3 mai 2023, à l'adresse https://bop.felk.cvut.cz/challenges/

Cai, Z., & Vasconcelos, N. (2018). Cascade R-CNN : Delving Into High Quality Object Detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6154-6162. https://doi.org/10.1109/CVPR.2018.00644

Chollet, F. (2018). *Deep learning with Python* (Bibliothèque des sciences et de génie QA 76.73 P98C46 2018). Manning.

Combe, M. (2016). *L'usine du futur : Vers l'avènement de la robotique*. Techniques de l'ingénieur; Techniques de l'ingénieur.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN : Object Detection via Region-based Fully Convolutional Networks. *arXiv:1605.06409 [cs]*. http://arxiv.org/abs/1605.06409

Dataset. (s. d.). Fraunhofer Institute for Manufacturing Engineering and Automation IPA. Consulté 13 juin 2020, à l'adresse https://www.fraunhofer.de/en/dataset.html

Depth Camera D435. (s. d.). Intel® RealSense™ Depth and Tracking Cameras. Consulté 11 juin 2020, à l'adresse https://www.intelrealsense.com/depth-camera-d435/

Dex-Net by BerkeleyAutomation. (s. d.). Consulté 8 mai 2023, à l'adresse https://berkeleyautomation.github.io/dex-net/

Drost, B., Ulrich, M., Bergmann, P., Hartinger, P., & Steger, C. (2017). Introducing MVTec ITODD — A Dataset for 3D Object Recognition in Industry. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2200-2208. https://doi.org/10.1109/ICCVW.2017.257

Dvornik, N., Shmelkov, K., Mairal, J., & Schmid, C. (2017). BlitzNet : A Real-Time Deep Network for Scene Understanding. *2017 IEEE International Conference on Computer Vision (ICCV)*, 4174-4182. https://doi.org/10.1109/ICCV.2017.447

GitHub—Ethnhe/raster_triangle : A simple renderer with z-buffer for synthesis data generating. (s. d.). Consulté 9 mai 2023, à l'adresse https://github.com/ethnhe/raster_triangle

GitHub—Zju3dv/clean-pvnet : Code for « PVNet : Pixel-wise Voting Network for 6DoF Pose Estimation » CVPR 2019 oral. (s. d.). Consulté 8 mai 2023, à l'adresse https://github.com/zju3dv/clean-pvnet

Guoguang, D., Link to external site, this link will open in a new window, Wang, K., Shiguo, L., & Kaiyong, Z. (2021). Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers : A review. *The Artificial Intelligence Review*, *54*(3), 1677-1734. https://doi.org/10.1007/s10462-020-09888-5

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980-2988. https://doi.org/10.1109/ICCV.2017.322

He, Y., Huang, H., Fan, H., Chen, Q., & Sun, J. (2021). *FFB6D : A Full Flow Bidirectional Fusion Network for 6D Pose Estimation* (arXiv:2103.02242). arXiv. http://arxiv.org/abs/2103.02242

He, Y., Sun, W., Huang, H., Liu, J., Fan, H., & Sun, J. (2020). PVN3D : A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation. *arXiv:1911.04231 [cs]*. http://arxiv.org/abs/1911.04231

Hodan, T. (2022). *T-LESS Toolkit* [Python]. https://github.com/thodan/t-less_toolkit (Original work published 2016)

Hodan, T. (2023). *SIXD Toolkit* [Python]. https://github.com/thodan/sixd_toolkit/blob/96bb268e1fb5ebd82ca1b8d352e3263561ba6f5c/doc/sixd_2017_datasets_format.md (Original work published 2016)

Hodan, T., Haluza, P., Obdrzalek, S., Matas, J., Lourakis, M., & Zabulis, X. (2017). T-LESS : An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 880-888. https://doi.org/10.1109/WACV.2017.103

Hou, J., Dai, A., & Nießner, M. (2019). 3D-SIS : 3D Semantic Instance Segmentation of RGB-D Scans. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4416-4425. https://doi.org/10.1109/CVPR.2019.00455

Jetley, S., Sapienza, M., Golodetz, S., & Torr, P. H. S. (2017). Straight to Shapes : Real-Time Detection of Encoded Shapes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4207-4216. https://doi.org/10.1109/CVPR.2017.448

Jiao, T., Xia, Y., Gao, X., Chen, Y., & Zhao, Q. (2019). 6DoF Pose Estimation for Intricately-Shaped Object with Prior Knowledge for Robotic Picking. *2019 3rd International Symposium on Autonomous Systems (ISAS)*, 199-204. https://doi.org/10.1109/ISASS.2019.8757758

Kehl, W., Manhardt, F., Tombari, F., Ilic, S., & Navab, N. (2017). SSD-6D : Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. *2017 IEEE International Conference on Computer Vision (ICCV)*, 1530-1538. https://doi.org/10.1109/ICCV.2017.169

Lahoud, J., & Ghanem, B. (2017). 2D-Driven 3D Object Detection in RGB-D Images. *2017 IEEE International Conference on Computer Vision (ICCV)*, 4632-4640. https://doi.org/10.1109/ICCV.2017.495

Li, D., Liu, N., Guo, Y., Wang, X., & Xu, J. (2019). 3D object recognition and pose estimation for random bin-picking using Partition Viewpoint Feature Histograms. *Pattern Recognition Letters*, *128*, 148-154. https://doi.org/10.1016/j.patrec.2019.08.016

Li, M., & Hashimoto, K. (2018). Fast and Robust Pose Estimation Algorithm for Bin Picking Using Point Pair Feature. *2018 24th International Conference on Pattern Recognition (ICPR)*, 1604-1609. https://doi.org/10.1109/ICPR.2018.8545432

Li, Y., Qi, H., Dai, J., Ji, X., & Wei, Y. (2017). Fully Convolutional Instance-Aware Semantic Segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4438-4446. https://doi.org/10.1109/CVPR.2017.472

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD : Single Shot MultiBox Detector. *arXiv:1512.02325 [cs]*, *9905*, 21-37. https://doi.org/10.1007/978-3-319-46448-0_2

Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A., & Goldberg, K. (2017). Dex-Net 2.0 : Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. *ArXiv:1703.09312 [Cs]*. http://arxiv.org/abs/1703.09312

Michel, F., Kirillov, A., Brachmann, E., Krull, A., Gumhold, S., Savchynskyy, B., & Rother, C. (2017). Global Hypothesis Generation for 6D Object Pose Estimation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 115-124. https://doi.org/10.1109/CVPR.2017.20

Moign, A. (2020). *Robotique collaborative industrielle : Veille sectorielle et technologique*. Techniques de l'ingénieur; Techniques de l'ingénieur.

Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). ENet : A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv:1606.02147 [cs]*. http://arxiv.org/abs/1606.02147

Peng, S., Liu, Y., Huang, Q., Zhou, X., & Bao, H. (2019). PVNet : Pixel-Wise Voting Network for 6DoF Pose Estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4556-4565. https://doi.org/10.1109/CVPR.2019.00469

Qi, C. R., Litany, O., He, K., & Guibas, L. (2019). Deep Hough Voting for 3D Object Detection in Point Clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9276-9285. https://doi.org/10.1109/ICCV.2019.00937

Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum PointNets for 3D Object Detection from RGB-D Data. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 918-927. https://doi.org/10.1109/CVPR.2018.00102

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++ : Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv:1706.02413 [cs]*. http://arxiv.org/abs/1706.02413

Qiao, R., Xu, G., Wang, P., Cheng, Y., Link to external site, this link will open in a new window, & Dong, W. (2023). An Accurate, Efficient, and Stable Perspective-n-Point Algorithm in 3D Space. *Applied Sciences*, *13*(2), 1111. https://doi.org/10.3390/app13021111

Rad, M., & Lepetit, V. (2017). BB8 : A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. *2017 IEEE International Conference on Computer Vision (ICCV)*, 3848-3856. https://doi.org/10.1109/ICCV.2017.413

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once : Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*. http://arxiv.org/abs/1506.01497

Russell, S. J. (Stuart J., & Norvig, P. (2021). *Artificial intelligence : A modern approach* (Fourth edition). Pearson.

Song, C., Song, J., & Huang, Q. (2020). HybridPose : 6D Object Pose Estimation under Hybrid Representations. *arXiv:2001.01869 [cs]*. http://arxiv.org/abs/2001.01869

Song, S., & Xiao, J. (2016). Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 808-816. https://doi.org/10.1109/CVPR.2016.94

Sundermeyer, M., Hodan, T., Labbe, Y., Wang, G., Brachmann, E., Drost, B., Rother, C., & Matas, J. (2023). *BOP Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects* (arXiv:2302.13075). arXiv. http://arxiv.org/abs/2302.13075

Tekin, B., Sinha, S. N., & Fua, P. (2018). Real-Time Seamless Single Shot 6D Object Pose Prediction. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 292-301. https://doi.org/10.1109/CVPR.2018.00038

Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., & Birchfield, S. (2018). Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. *ArXiv:1809.10790 [Cs]*. http://arxiv.org/abs/1809.10790

Uhrig, J., Rehder, E., Fröhlich, B., Franke, U., & Brox, T. (2018). Box2Pix : Single-Shot Instance Segmentation by Assigning Pixels to Object Boxes. *2018 IEEE Intelligent Vehicles Symposium (IV)*, 292-299. https://doi.org/10.1109/IVS.2018.8500621

Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., & Savarese, S. (2019). *DenseFusion : 6D Object Pose Estimation by Iterative Dense Fusion* (arXiv:1901.04780). arXiv. http://arxiv.org/abs/1901.04780

Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2018). PoseCNN : A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv:1711.00199 [cs]*. http://arxiv.org/abs/1711.00199

Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., & Urtasun, R. (2019). UPSNet : A Unified Panoptic Segmentation Network. *arXiv:1901.03784 [cs]*. http://arxiv.org/abs/1901.03784

Xu, W., Wang, H., Qi, F., & Lu, C. (2019). Explicit Shape Encoding for Real-Time Instance Segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 5167-5176. https://doi.org/10.1109/ICCV.2019.00527

Yan, W., Xu, Z., Zhou, X., Su, Q., Li, S., & Wu, H. (2020). Fast Object Pose Estimation Using Adaptive Threshold for Bin-Picking. *IEEE Access*, *8*, 63055-63064. https://doi.org/10.1109/ACCESS.2020.2983173

Zhao, H., Qi, X., Shen, X., Shi, J., & Jia, J. (2018). ICNet for Real-Time Semantic Segmentation on High-Resolution Images. Dans V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Éds.), *Computer Vision – ECCV 2018* (p. 418-434). Springer International Publishing. https://doi.org/10.1007/978-3-030-01219-9_25

Zhou, Y., & Liu, S. (2022). Object Pose Estimation Based on Improved YOLOX Algorithm. *2022 IEEE 11th Data Driven Control and Learning Systems Conference (DDCLS)*, 699-705. https://doi.org/10.1109/DDCLS55054.2022.9858495

Zhou, Y., & Tuzel, O. (2018). VoxelNet : End-to-End Learning for Point Cloud Based 3D Object Detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4490-4499. https://doi.org/10.1109/CVPR.2018.00472