Vision industrielle et réseaux de neurones profonds: Application au dévracage de pièces plastiques industrielles

Julien Langlois

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE NANTES
COMUE **U**NIVERSITÉ **B**RETAGNE **L**OIRE

École Doctorale N°601
Mathématique et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

« Julien LANGLOIS »

« Vision industrielle et réseaux de neurones profonds »

« *Application au dévracage de pièces plastiques industrielles* »

Thèse présentée et soutenue à NANTES, LE 28 AOÛT 2019
Unité de recherche : LS2N UMR CNRS 6004 - IPI
Thèse N° : CIFRE 1460/2015

Rapporteurs avant soutenance :

M. Antoine TABBONE, Professeur, Université de Lorraine
M. Cédric DEMONCEAUX, Professeur, Université de Bourgogne

Composition du jury :

Présidente : Mme Véronique EGLIN, Professeure, INSA Lyon
Examinateur : M. Vincent LEPETIT, Professeur, Université de Bordeaux
Dir. de thèse : M.Christian VIARD-GAUDIN, Professeur, Université de Nantes
Co-dir. de thèse : M. Nicolas NORMAND, Professeur, Université de Nantes
Encad. de thèse : M. Harold MOUCHÈRE, Maître de conférences, HDR, Université de Nantes

Invitée :

Mme Morgane TANGUY, Docteure, Responsable R&D et innovation du groupe Wedo

AVANT-PROPOS

Ce travail a été réalisé en partie au sein de l'équipe Image-Perception-Interaction (IPI) dirigée par M. Nicolas NORMAND intégrée au pôle Signaux, Images, Ergonomie et Langues (SIEL). Ce pôle appartient au Laboratoire des Sciences du Numériques de Nantes (LS2N), Unité Mixte de Recherche du Centre National de la Recherche Scientifique (UMR CNRS) 6004 dirigé par M. Claude JARD. Cette thèse a également été accomplie au sein de l'entreprise MULTITUDE-TECHNOLOGIES membre du groupe WEDO par le biais d'une Convention Industrielle de Formation par la Recherche (CIFRE) financée en partie par l'Association Nationale de la Recherche et de la Technologie (ANRT).

REMERCIEMENTS

Je tiens à remercier Christian VIARD-GAUDIN, professeur de l'université de Nantes, pour avoir dirigé ma thèse. L'aboutissement de mes travaux est indéniablement le fruit de sa grande expérience et clairvoyance qu'il a su me transmettre. Je tiens à remercier Nicolas NORMAND, professeur de l'université de Nantes, pour avoir co-dirigé ma thèse. En alliant rigueur et passion, il m'a toujours brillamment aiguillé et conforté dans mes recherches tout en éveillant ma curiosité sur d'autres domaines scientifiques. Je remercie Harold MOUCHÈRE, professeur de l'université de Nantes, pour m'avoir encadré durant ces trois ans. Sa détermination permanente et son expertise m'ont fortement soutenu dans le déroulement de mes travaux ainsi que sur le plan personnel.

Je tiens à remercier Francis PERRIN, président du groupe Wedo, pour l'intuition et la curiosité scientifique dont il a su faire preuve afin de donner naissance à ce projet. Je le remercie également pour la confiance sans faille qu'il su m'accorder. Je remercie Hervé BAUDRON, référent technique machines spéciales de Multitude-Technologies, pour m'avoir accueilli, accompagné et soutenu dans l'entreprise durant les premières années. Je remercie Morgane TANGUY, responsable R&D et innovation du groupe Wedo, pour le support qu'elle m'a fourni durant le projet ainsi que son expérience de docteure qu'elle m'a partagé.

Je tiens à remercier Véronique EGLIN, professeure à l'INSA de Lyon d'avoir accepté de juger la qualité de mes recherches ainsi que Vincent LEPETIT, professeur de l'université de Bordeaux, dont les travaux furent une grande source d'inspiration. Je remercie Antoine TABBONE, professeur de l'université de Lorraine ainsi que Cédric DEMONCEAUX, professeur de l'université de Bourgogne, pour avoir rapporté en détail mon manuscrit. Je remercie également Benoît FURET, professeur de l'université de Nantes, pour avoir accepté de participer à mon comité de suivi de thèse.

Je tiens à remercier Myriam SERVIÈRES, maître de conférences HDR à l'école centrale de Nantes, pour m'avoir donné goût à la recherche et fortement accompagné

pour débuter ma thèse. Je remercie Jean-Pierre GUÉDON, professeur de l'université de Nantes, pour sa formidable générosité et les nombreux conseils pédagogiques qu'il m'a apporté.

Je remercie tous les membres de l'équipe IPI et anciens doctorants et post-doctorants (Romain, Yashas, Lukáš, Rémi, Mona, Vincent, Matthieu, Erwan, Toinon, Antoine, Ali, Patrick, Yoann, Suiyi, Jésus...). Ces trois années passées à vos côtés dans la bonne humeur et les mots fléchés seront à jamais un magnifique souvenir. Je remercie également les nombreux collègues du groupe Wedo (Sébastien, Sylvie, la compta, Sonia, Mathilde, Guillaume, Florent, Cyril, Vincent, JC, Philippe...) pour leur soutien, leurs encouragements et l'intérêt qu'ils ont porté à mes travaux. Ces travaux ont également été étoffés par des étudiants de Polytech Nantes qui ont su m'apporter leur aide et je les remercie.

Je souhaite remercier les membres de ma famille pour le soutien indéfectible dont ils ont su faire preuve durant ces trois ans. Votre présence à ma soutenance en est la plus belle preuve. Un grand merci à mes amis Centraliens et dentaires (Adumas, Alice, Raph, Fañch, Maylis, Buzz, JB, Julie, Lucie et j'en passe) pour toutes ces rencontres, soirées et votre joie de vivre contagieuse. Enfin, je remercie tous les amis du LossGaming (Ketchup, Exo, Sif, Medion, Boogie, Gala et tous les autres) pour les nombreuses parties de CS et de WoW les soirs et votre soutien constant envers Mayonnaise.

" Dans ton espace virtuel, tu apparais tellement réel ! "

Générique de Digimon, Bandai

SOMMAIRE

INTRODUCTION

Intelligence artificielle et industries

Ces travaux de thèse s'inscrivent dans un contexte d'intégration de l'intelligence artificielle (IA) à la production industrielle et plus particulièrement de l'apprentissage automatique ou "*machine learning*". L'apprentissage automatique s'intègre déjà dans des grands groupes en assistance à la production et à la prise de décisions développant ainsi des industries intelligentes dites "*smart industries*" ou industries 4.0. Cette intégration dans la production permet dans un premier temps une universalisation des outils de production. En effet, les algorithmes mis en jeu s'adaptent aisément à de nouvelles données et peuvent être utilisés dans des processus divers. L'intégration de l'IA réduit également les coûts de recherche et développement ainsi que ceux de production. Un algorithme adaptable limite la conception et fabrication de machines dédiées à un processus particulier et favorise donc une mise en production rapide. Grâce aux fortes capacités combinatoires qu'elle offre, certains composants onéreux en particulier en traitement du signal, peuvent être simulés. Un processus industriel peut alors être envisagé à bas coût tout en se libérant de certaines contraintes des fournisseurs de technologies (matériels propriétaires, interventions...). L'IA peut également être employée à des fins d'anticipation et de traitement des données issues des machines, offrant alors des possibilités de maintenance prédictive sur les installations d'une usine. Ceci assure une meilleure qualité de fabrication tout en limitant les éventuels arrêts de production, gage de confiance pour les clients de l'entreprise. L'IA intervient aussi en assistance à la production afin de guider les opérateurs dans leurs tâches en analysant leurs gestes et environnements. Cet intérêt s'est rapidement démontré en robotique à travers l'émergence des robots collaboratifs désormais représentés sous le terme de cobotique. Ils optimisent ainsi les cycles de production tout en améliorant les conditions de travail des opérateurs. Enfin, les applications d'IA basées sur un paradigme biologique ou du biomimétisme, offrent la possibilité d'automatiser des processus contraignant pour des opérateurs, tout en assurant une qualité et des cadences de production supérieures.

Le dévracage industriel

L'exemple du dévracage a été choisi pour illustrer l'intérêt d'une approche IA dans une application industrielle, il apparaît utile d'en décrire les principales caractéristiques.

Le processus

Il existe plusieurs étapes à la création d'une pièce en plastique injecté. La matière initialement sous forme granulée, est chauffée puis injectée dans un moule pressé. Après un refroidissement, la pièce dite non-finie, est éjectée du moule afin d'être placée dans un bac (Fig. 1). Dans la majorité des cas, la pièce finale n'est obtenue qu'après un assemblage de différentes pièces non-finies. Cette étape pouvant être réalisée dans une autre usine, les pièces à assembler doivent être au préalable chacune extraites de leur bac respectif. Cette étape est appelée **dévracage**. Une fois placées sur des empreintes, un processus industriel permet d'obtenir la pièce finale avec, par exemple, un soudage des parties (Fig. 2).

FIGURE 1 – Vue de dessus d'un bac de pièces plastiques (parties de boîte à air de moteur).

La tâche de dévracage est aujourd'hui réalisée manuellement par un opérateur de production. Puisque ce travail est répétitif et peu valorisant, les ressources humaines sont difficiles à trouver ce qui augmente les délais de production. De plus, des gestes répétés accroissent les risques de troubles musculo-squelettiques (TMS) et nécessitent donc des pauses régulières et des aménagements ergonomiques adaptés. Enfin, une

majorité des pièces produites sont des éléments de sécurité automobile et les mauvaises manipulations pouvant être liées à la fatigue de l'opérateur (chute, mauvais placement sur l'empreinte), ne permettent plus de respecter les contraintes de qualité et la pièce risque d'être envoyée au rebut. Afin de pallier ces défauts, le processus de dévracage peut être automatisé.

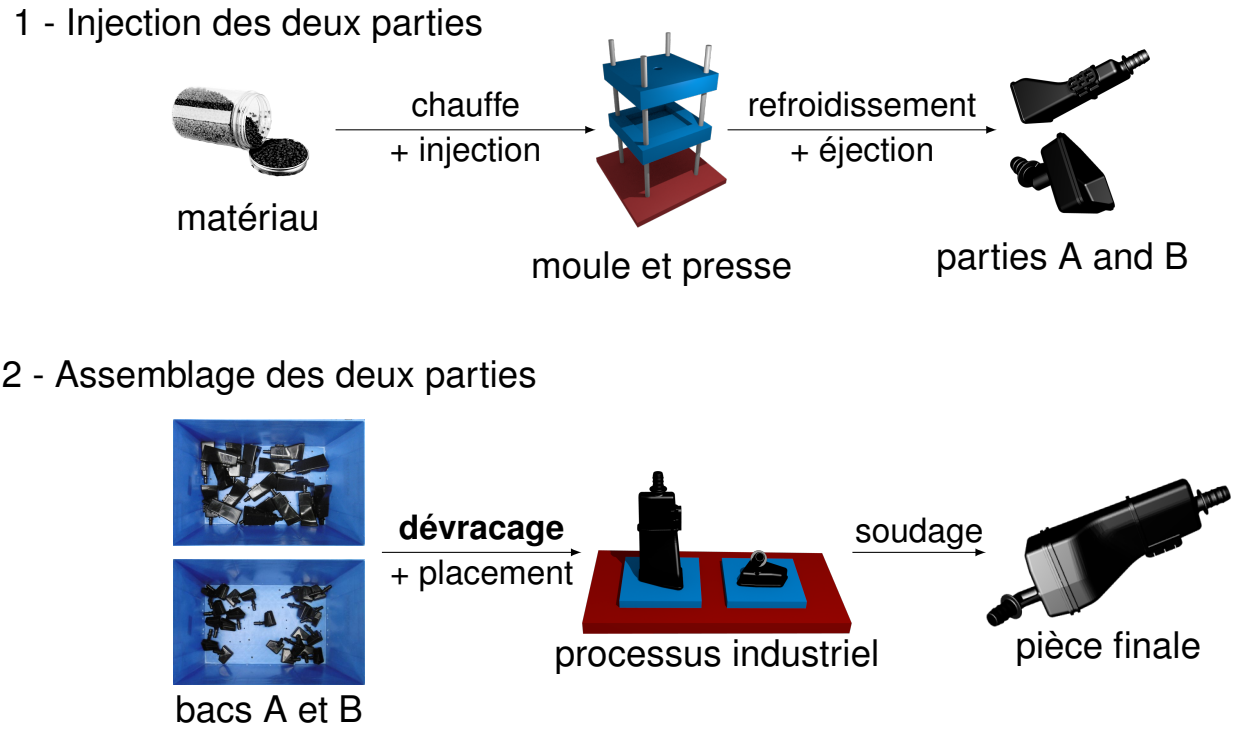


FIGURE 2 – Détails de l'injection et de l'assemblage par soudage d'une pièce en plastique injecté.

Automatisation du dévracage

Principes et critères

Il convient dans un premier temps, de distinguer deux types de dévracage automatisé. Le dévracage de tri consiste à séparer des pièces de natures différentes contenues dans un bac, afin de les placer dans des bacs respectifs. Dans ce cas, le problématique principale se concentre sur l'identification de la pièce et sa bonne préhension. Dans ce type de dévracage, l'estimation de pose est grossière et un préhenseur universel absorbant l'erreur de pose est employé. La dépose est réalisée dans un autre bac et ne nécessite donc pas de préhension rigide (la pièce peut éventuellement changer de pose durant la saisie). Le second type concerne le dévracage avec dépose. Dans ce cas, toutes les pièces dans le bac sont identiques. La pièce à saisir doit être déposée précisément sur une empreinte ce qui nécessite au préalable, une estimation de pose précise ainsi qu'une préhension rigide.

Un système de vision (2D ou 3D) est chargé de détecter les pièces dans le bac avant d'estimer leur position et orientation. En connaissant le modèle 3D de l'objet ainsi qu'une liste de zones de préhension autorisées, un bras de robot muni d'un préhenseur (pince, ventouse...) saisit une des pièces afin de la placer sur une empreinte d'un processus d'assemblage (Fig. 3).

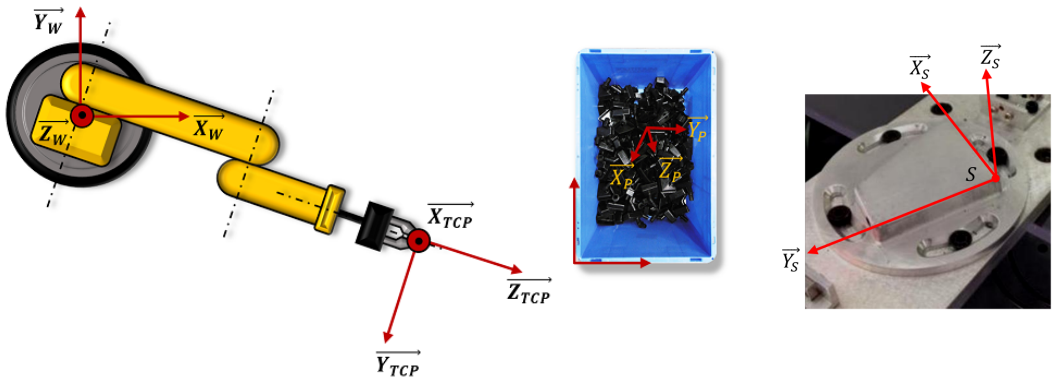


FIGURE 3 – Un bras de robot dans le repère du monde $(\overrightarrow{X_W}, \overrightarrow{Y_W}, \overrightarrow{Z_W})$ disposant d'un préhenseur $(\overrightarrow{X_{\text{TCP}}}, \overrightarrow{Y_{\text{TCP}}}, \overrightarrow{Z_{\text{TCP}}})$ devant saisir une pièce $(\overrightarrow{X_P}, \overrightarrow{Y_P}, \overrightarrow{Z_P})$ dans un bac puis la placer sur une empreinte $(\overrightarrow{X_S}, \overrightarrow{Y_S}, \overrightarrow{Z_S})$.

Ces travaux de thèse traitent du dévracage avec dépose et plus particulièrement, de la détection et de l'estimation de pose de pièces dans l'image d'un bac. Le module de vision proposé doit respecter certains critères de l'entreprise concernant le type de pièces à traiter, les temps d'acquisition ainsi que les performances attendues (Table 1).

TABLE 1 – Les différents critères que doit respecter le module de vision.

Données	**Critère**
Acquisition	module de vision déplaçable utilisation de technologies peu onéreuses distance de vue par rapport au haut du vrac > 50cm et < 250cm robuste aux éclairages
Pièces	pièces plastiques peu texturées couleurs sombres géométries complexes diamètre > 30mm
Traitement	estimation rapide < 100ms plusieurs pièces en parallèle critère d'accessibilité de la préhension
Précision	précision suffisante pour une préhension correcte (à définir en fonction du préhenseur) détection des erreurs

Verrous technologiques

Le problème du dévracage peut être simplifié en plaçant des pièces sur des surfaces planes ou en utilisant des cibles spécifiques sur les objets. Bien que l'emploi d'un convoyeur (tapis roulant) permette de limiter les rotations d'un objet et de fixer sa translation sur l'axe Z, cette solution est onéreuse et ne permet pas de réaliser des modules de taille réduite. De façon similaire, les solutions en peau vibrante (ou bol vibrant) permettent de désunir les pièces et de limiter les occlusions. Cependant, elles sont réservées à des pièces de faible volume. Enfin, les pièces étant éjectées des moules, il est complexe de leur appliquer des marquages (cibles) de façon automatisée. De plus, elles sont pendant un certain temps encore à température élevée et le

collage peut ne pas fonctionner. Il en va de même pour l'application d'une peinture qui n'est pas toujours autorisée sur les pièces par les constructeurs et peut ne pas tenir sur toutes les gammes de plastiques.

Face à ce constat, certains prestataires industriels proposent des solutions de dévracage avec dépose depuis des bacs. Les systèmes les plus courants comportent généralement un module d'acquisition 3D (stéréoscopie, infrarouge, lasers...) fournissant un nuage de points dense de la scène. Un second module permet à la fois de détecter les pièces dans le nuage et d'estimer leur pose selon le modèle 3D de l'objet. Une pièce est ensuite choisie puis saisie selon une liste pré-établie de zones de préhension par un bras de robot. La communication avec le robot tout comme la phase de dépose, sont souvent laissées à la charge du client. Si ces solutions sont classées au niveau 9 de *Technology Readiness Level* (TRL), elles soulèvent néanmoins de nombreux verrous technologiques (Table 2).

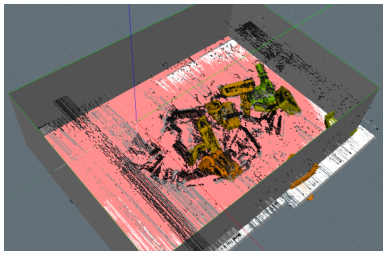
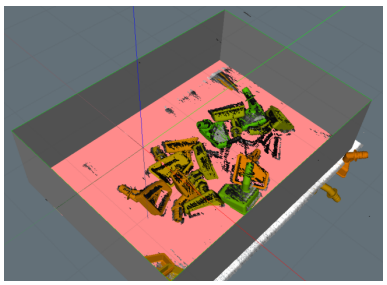
TABLE 2 – Verrous technologiques des solutions de dévracage.

Données	**Verrous**
Acquisition	capteurs onéreux emploi de lasers (sécurité de l'opérateur, gamme non tolérée en France) sensibilité aux réflexions lumineuses distance de travail avec le haut du bac < 150cm
Pièces	pièces plastiques noires et peu texturées mal détectées
Traitement	estimation lente > 5s peu de gestion du contrôle robot
Précision	déplacement moyen entre le nuage de point capturé et réel d'environ 2mm pas en adéquation avec les technologies et le prix

Test d'une solution industrielle

Les verrous technologiques (Table 2) s'expriment notamment à travers les tests d'une solution haut de gamme du marché sur des pièces complexes en plastique, réalisés au début de la thèse. Ce système possède un laser balayant le bac afin d'obtenir une nuage de points de la scène par stéréoscopie. Le modèle 3D de l'objet est alors classiquement recalé dans le nuage afin de détecter et d'estimer la pose d'un certain nombre de pièces. Cependant, cette technologie présente des défauts lorsque l'arrière-plan du bac n'absorbe pas la lumière (Table 3) : le nuage de points apparaît bruité et l'algorithme de recalage propose des poses de pièce non conformes en un temps long.

TABLE 3 – Impact de la couleur de fond du bac sur les performances en détection et estimation de pose de la solution industrielle testée.

Couleur du fond	**Nuage de points acquis**	**Pièces détectées**	**Détectées et correctement positionnées**	**Temps d'acquisition par pièce (ms)**
Orange		$6/14$	$5/6$	526
Noir		$12/14$	$12/12$	130

Lorsque le volume de l'objet est trop faible, toutes les pièces de la scène sont détectées mais les poses sont fortement erronées : l'algorithme de recalage ne dispose pas d'assez de points pour correctement placer le modèle 3D. Avec un plus grand volume, les poses sont correctes mais tous les objets ne sont pas détectés et le temps d'acquisition augmente car le nombre de points alloués à une pièce est important (Fig. 4).

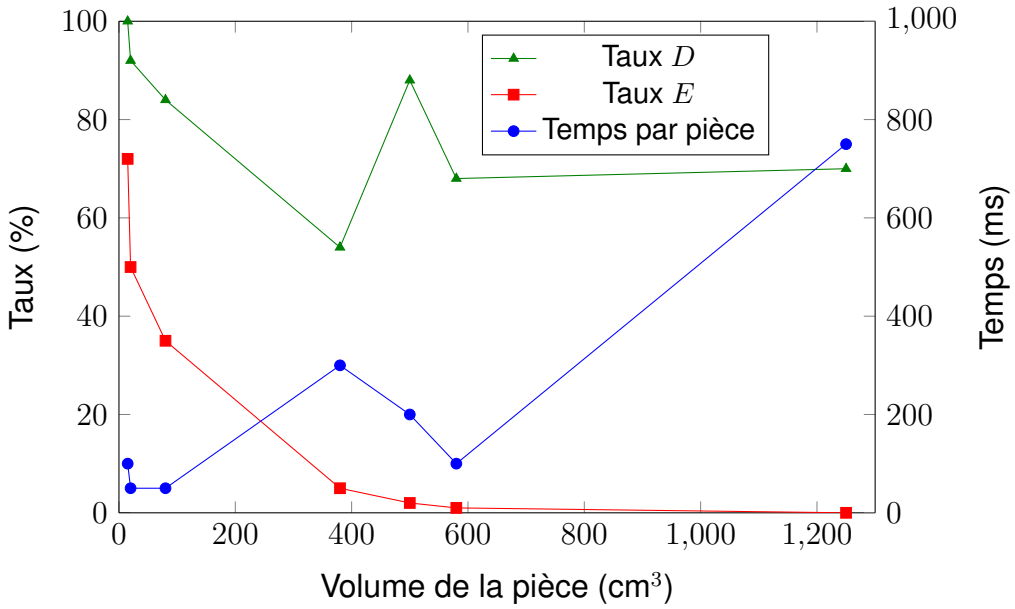


FIGURE 4 – Taux de détections des pièces D, de mauvaises estimations de pose des pièces détectées E et temps d'acquisition par pièce en fonction du volume de l'objet pour la solution industrielle testée (fonctionnant à 80cm du haut du vrac avec des paramètres optimaux pour chaque objet).

Pour une gamme de pièces plastiques standard évoluant de 350 à 1200 cm^3, le taux de détection moyen est de seulement 70% (pour une scène peu complexe et sans occlusions) mais les poses estimées sont toutes correctes. Cette solution industrielle ne permet pas de travailler à plus de 110 cm du haut du bac ce qui réduit considérablement l'espace des mouvements du robot.

La solution industrielle haut de gamme testée ne permet pas de répondre à tous les critères du module d'estimation de pose souhaité (Table 1). Dans ces travaux de thèse, nous proposons d'étudier l'apport de méthodes issues de l'intelligence artificielle pour construire un module de vision permettant de lever la totalité ou une partie de ces verrous technologiques.

Dévracage et intelligence artificielle

Choix technologiques

Dans le cadre du module de vision du dévracage, l'algorithme doit permettre de localiser (segmenter) un nombre prédéterminé de pièces dans la scène et d'estimer leur position et orientation associées. Le choix de la zone de préhension et le calcul de la trajectoire du robot sont réalisés par des algorithmes *ad hoc* et ne sont pas détaillés dans cette thèse. Le fonctionnement du module consiste dans un premier temps, à isoler les pièces du bac puis, à les découper afin de former un nombre déterminé d'instances de la pièce. Chaque instance est ensuite traitée pour estimer à la fois la position de la pièce dans le repère du bac et son orientation par rapport à un modèle 3D de référence. Contrairement au module industriel du commerce testé précédemment, les estimations de pose de ces pièces doivent être réalisées en un temps court. Les calculs de pose sont donc effectués en parallèle par des composants de calculs adaptés notamment en utilisant des cartes graphiques (*Graphics Processing Unit*, GPU). Dans le but d'employer des technologies peu onéreuses et de limiter les défauts liés aux scanners 3D (Table 2), ces travaux de thèse proposent de se limiter à une image 2D en couleurs (*Red Green Blue*, RGB) du haut du vrac comme donnée d'entrée pour le module de vision. L'algorithme va donc chercher à établir deux correspondances principales : une image de bac vers une ou plusieurs images de pièces segmentées et une image d'une pièce segmentée vers une position et une orientation.

Ces deux correspondances sont complexes puisqu'un bac peut contenir un grand nombre de pièces enchevêtrées chacune présentant des réflexions lumineuses différentes. Cependant, à partir d'exemples, l'apprentissage automatique doit permettre de construire automatiquement ces correspondances démontrant alors, la pertinence de l'intelligence artificielle pour ce processus. Afin de respecter les critères du module de vision (Table 1), l'algorithme doit être capable d'estimer une pose parmi toutes les orientations possibles d'une pièce mais il doit également être robuste aux différentes variations lumineuses que peut subir le matériau plastique. Pour cela, les exemples servant à l'apprentissage doivent être suffisamment hétérogènes afin de couvrir une large gamme de poses de pièce aux luminances variées. Néanmoins, construire un tel jeu de données est chronophage car l'apprentissage nécessite plusieurs centaines de milliers d'images pour fonctionner de façon optimale. De plus, il n'est pas simple

d'obtenir en pratique toutes les orientations d'une pièce dans une scène et les tables tournantes, parfois utilisées dans de tels cas, sont limitées aux positions d'équilibre de l'objet. Enfin, toute session de prises de vue est dépendante de l'éclairage de la pièce et peut ne pas correspondre aux futures conditions d'utilisation en usine. En effet, le poste de dévracage peut être placé à n'importe quel emplacement d'une usine, sans contrôle d'éclairage, aussi bien en production de jour comme de nuit.

Face à ce constat, ces travaux de thèse proposent d'employer un apprentissage basé sur des images de synthèse du modèle 3D de la pièce. Un rendu 3D permet d'envisager toutes les orientations possibles de l'objet tout en contrôlant à la fois l'éclairage de la scène virtuelle et les propriétés du matériau. Ces rendus peuvent être effectués par une carte graphique durant l'entraînement de l'algorithme ce qui ne nécessite pas de stockage de données au préalable. Les paramètres de scène sont modifiables dynamiquement et aléatoirement, ce qui permet à l'apprentissage automatique de constamment traiter des données différentes. Pour générer des bacs, un moteur de collision peut être injecté dans le rendu 3D afin de créer un vrac réaliste par l'intermédiaire d'un lancer de pièces. De plus, selon l'initialisation du lancer et le nombre de pièces souhaité, l'algorithme peut apprendre à traiter un vrac plus ou moins dense. Cependant, le défaut majeur de la génération d'images est le biais introduit entre les données synthétiques et les données réelles. En effet, certaines pièces possèdent des défauts d'aspects (lignes de soudure dans le moule, traces des éjecteurs et injecteurs...) et des gravures de traçabilité (logo, numéro de lot...) relativement difficiles à simuler. Il convient donc de bien maîtriser les paramètres de scène afin de fournir des images réalistes anticipant les futures conditions d'exploitation en usine mais aussi, de proposer un algorithme d'apprentissage absorbant ce biais introduit. L'utilisation d'images de synthèse pour l'apprentissage permet également de s'affranchir de la production *a priori* de la pièce. L'entraînement peut alors être réalisé en connaissant le type de plastique employé ainsi que les pigments de couleur. Ceci est un avantage lorsque les délais de réponses aux clients sont courts et que l'entreprise doit s'assurer que la pièce finale peut être produite dans les meilleures conditions.

Pour l'apprentissage automatique de la correspondance entre une image d'un objet et sa pose, des réseaux de neurones profonds sont employés [59] et notamment les réseaux à convolutions [60]. Ces réseaux sont particulièrement bien adaptés au traitement des images et de nombreux travaux scientifiques les emploient pour des tâches diverses (classification d'éléments dans des scènes naturelles [56], détection et identification de personnes dans des foules [48], localisation et détection de tumeurs dans des images médicales [76], reconnaissance d'écriture [100]...). En particulier cette thèse emploie des architectures de réseaux de neurones à convolutions récurrentes pour la recherche de pièces à segmenter, mais également des encodeurs-décodeurs [5] pour la transformation d'image.

État de l'art

L'état de l'art en estimation de pose d'objets est divisé en deux parties. La première traite des techniques de segmentation d'objets dans une scène (Ch. 3). Dans un premier temps, les travaux basés sur des analyses descendantes et ascendantes sont présentés. Puis, les différentes techniques cherchant à émettre des relations logiques entre les éléments de la scène dans l'image (compréhension de scène) sont introduites. Par la suite, les méthodes reposant sur des réseaux de neurones sont explicitées notamment l'utilisation de réseaux convolutifs, la proposition de région et les réseaux récurrents. La seconde partie de l'état de l'art traite de l'estimation de pose d'objets (Ch. 4). Les différents travaux employant des correspondances 2D-3D sont dans un premier temps détaillés (méthodes PnP). Puis, l'estimation de pose basée sur la mise en correspondance des caractéristiques locales dans l'image ainsi que la mise en correspondance de motifs ou de patrons d'objet est présentée. Enfin, les récentes techniques utilisant des réseaux de neurones sont explicitées.

Problématique

Cette introduction permet de mettre en avant les verrous technologiques présents dans une solution de dévracage industriel automatisé. À la vue des diverses contraintes (Table 1), les choix technologiques retenus pour élaborer un module de vision se concentrent sur l'apprentissage automatique et plus particulièrement l'utilisation de réseaux de neurones à convolutions. Afin d'anticiper les conditions hostiles d'un environnement industriel (éclairages non maîtrisés) et la versatilité des poses des pièces dans un bac, l'entraînement des réseaux sera réalisé en utilisant des images synthétiques du modèle 3D de l'objet.

Ces travaux tentent donc de répondre à la problématique suivante :

Peut-on construire un module de vision pour le dévracage industriel répondant aux critères de l'entreprise (Table. 1), en employant des réseaux de neurones entraînés uniquement sur des images synthétiques ?

Le chapitre 1 de ces travaux définit la structure d'un modèle 3D d'objet ainsi que les différentes transformations mathématiques applicables. Puis, une définition des gammes d'objets considérés basée sur leurs matériaux et géométries est détaillée.

Le chapitre 2 traite de la génération d'images par moteur de rendu 3D. La projection du modèle 3D dans le plan caméra ainsi que sa forme tensorielle sont présentées. Puis, les paramètres de scène employés pour obtenir un jeu de données hétérogène sont introduits. La génération de vrac par lancer de pièces et détection de collisions est détaillée en fin de chapitre.

Le chapitre 3 est consacré à la segmentation d'instances. Après une suppression de l'arrière-plan du bac basée sur un réseau encodeur-décodeur, un réseau récurrent est utilisé pour fournir une liste de pièces extraites de l'image du bac. Un modèle d'attention visuelle est présenté pour optimiser la recherche de pièces dans l'image.

Le chapitre 4 traite du pipeline d'estimation d'orientation de translation. Cette étape est constituée d'un réseau encodeur décodeur employant des couches denses et fournissant le nuage de points 3D de la pièce. Puis, deux réseaux travaillent en parallèle

pour fournir l'orientation de l'objet et sa translation dans la scène. Une étape de raffinement par recalage est ensuite proposée.

Le chapitre 5 est consacré au déploiement du module neuronal en industrie. Selon le plastique choisi pour produire la pièce, un algorithme est chargé de calculer les paramètres de rendus offrant le meilleur réalisme dans les images d'entraînement. L'implémentation des réseaux dans l'usine est basée sur une architecture contrôleur-serveur. Le choix de la préhension et le module de dépose sont également détaillés.

Enfin, les conclusions de ces travaux de thèse ainsi que les différentes perspectives pour à la fois améliorer l'estimation de pose et prolonger l'étude afin d'optimiser le pilotage du bras de robot, sont présentées.

PREMIÈRE PARTIE

Génération de données

CHAPITRE 1

DÉFINITION ET MANIPULATION DES MODÈLES 3D

Introduction

Ce premier chapitre permet d'appréhender la représentation et la manipulation de modèles 3D d'objets issus de la conception assistée par ordinateur (CAO) ainsi que de définir les objets traités dans les travaux de thèse.

Dans un premier temps, la structure en maillage triangulaire d'un modèle CAO d'un objet est introduite (Sec. 1.1). L'unification des sommets ainsi que la représentation par nuage de points sont détaillées. Pour manipuler cet objet, deux outils indispensables sont ensuite présentés : la rotation et la translation avec leurs topologies associées (Sec. 1.2). Différentes formulations d'une rotation sont introduites (angle-axe, matrice et quaternion) ainsi que la composition avec la translation. Enfin, les différentes gammes d'objets et jeux de données considérés dans les travaux sont présentés (Sec. 1.3). Une méthode d'estimation de complexité est proposée ainsi que le choix des objets à traiter selon leur géométrie et leurs matériaux.

1.1 Modèle CAO de l'objet

Pour générer les données d'apprentissage des réseaux, un objet est dans un premier temps placé dans une scène virtuelle. Cet objet est stocké numériquement sous la forme d'un modèle 3D pouvant être représenté par un maillage triangulaire ou un nuage de points. Ces deux formes permettent à la fois de réaliser des rendus réalistes (en calculant notamment les normales à chaque sommet) et plus tard, de construire des métriques quantifiant une erreur d'estimation de pose.

1.1.1 Maillage triangulaire

Le modèle 3D (ou modèle CAO) est représenté par un lot de N_s surfaces triangulaires $\{S_i\}_{i=1,\dots,N_s}$ (Fig. 1.1).

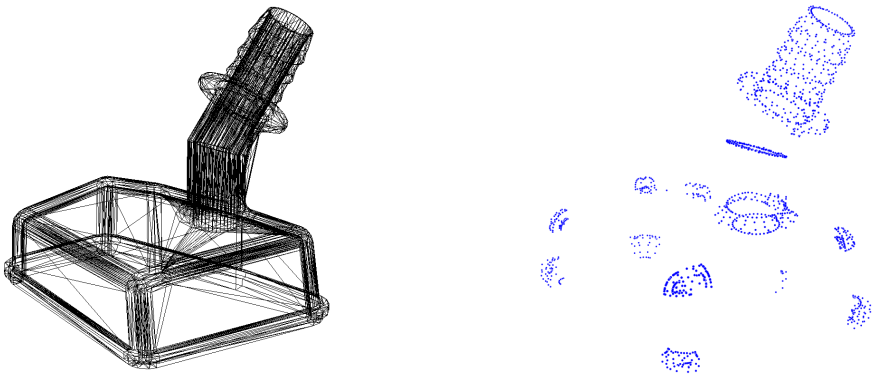


FIGURE 1.1 – Le modèle CAO représenté par un maillage triangulaire.

Chaque surface S_i est caractérisée par les coordonnées des 3 sommets (ou vertex) formant le triangle (prises par rapport à une origine) $(v_{i_1}, v_{i_2}, v_{i_3})$, ainsi que sa normale $\vec{n_i}$ (Fig. 1.2).

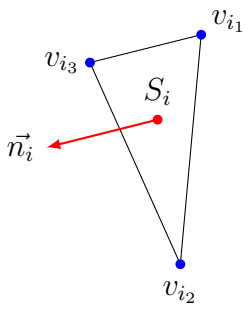
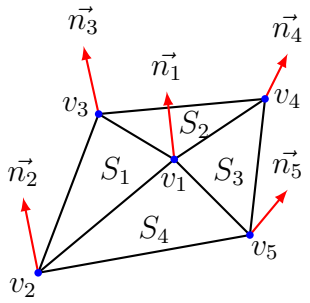


FIGURE 1.2 – Représentation d'une surface triangulaire S_i composée de ses sommets $(v_{i_1}, v_{i_2}, v_{i_3})$ et de sa normale $\vec{n_i}$.

1.1.2 Unification et normales aux sommets

Pour effectuer des rendus graphiques (notamment avec OpenGL), il convient de travailler avec des vertex à la place des surfaces. Dans un premier temps, les sommets sont indexés puis unifiés dans le modèle CAO afin qu'un sommet puisse être attribué à plusieurs surfaces voisines (Fig. 1.3). La normale à un vertex est ensuite calculée en moyennant les normales des triangles auxquels il appartient. Ces N_v sommets avec leur normale forment une collection $\{V_j\}_{j=1,\dots,N_v}$ et une surface S_i, est donc définie par un triplet d'indices choisis parmi les N_v :

$$\begin{aligned} \{V_j\}_{j=1,\dots,N_v} \quad &\text{avec} \quad V_j = (v_j, \vec{n_j}) \\ \{S_i\}_{i=1,\dots,N_s} \quad &\text{avec} \quad S_i = (a,b,c) \in [[1, N_v]]^3 \end{aligned} \tag{1.1}$$

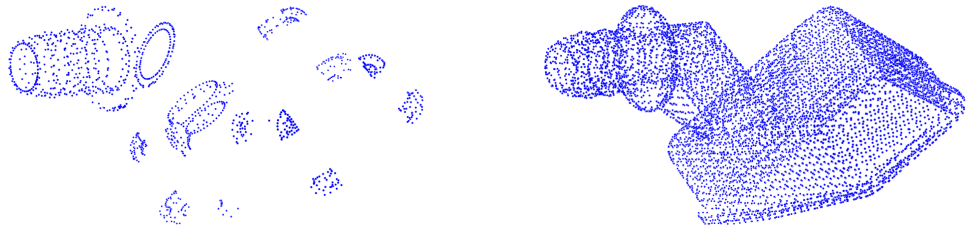


$$\begin{aligned} S_1 &= (1,2,3) \\ S_2 &= (1,3,4) \\ S_3 &= (1,4,5) \\ S_4 &= (1,2,5) \end{aligned}$$

FIGURE 1.3 – Représentation de 4 surfaces triangulaires S_i avec les vertex associés. Le vertex numéro 1 est utilisé dans 4 surfaces.

1.1.3 Nuage de points

Dans certains cas, les surfaces ne sont pas utilisées et le modèle est considéré comme un simple nuage de points avec pour chacun la normale associée. Cependant, utiliser les vertex pour former le nuage n'est pas toujours une méthode pertinente car une majorité des modèles d'objets proviennent d'éditeurs CAO représentant les surfaces de façon éparse : un seul triangle peut couvrir une grande surface plane (Fig. 1.4). Pour obtenir un nuage de points couvrant de façon homogène la surface du solide, il convient de densifier le modèle par un algorithme d'échantillonnage étant donnée la distance souhaitée entre deux points voisins (algorithme *Marching Cubes* [67] par exemple).



Nuage par sommets Nuage par échantillonnage

FIGURE 1.4 – Modèle CAO en tant que nuage de points par vertex (gauche) et par échantillonnage uniforme (droite).

1.2 Manipulations du modèle

La manipulation du modèle CAO dans la scène se décompose en deux opérations : une rotation puis une translation. Le choix de la représentation de la rotation est primordial pour assurer non seulement une couverture angulaire de tout l'espace des orientations de l'objet (pas de blocage de Cardan) mais également une bonne intégration dans un réseau de neurones et un calcul d'erreur d'estimation de pose pertinent. Afin de simplifier la transformation d'un grand nombre de sommets dans le modèle, ces deux opérations peuvent être combinées en un tenseur facilement utilisable au sein d'un moteur de rendus dans une carte graphique.

1.2.1 Représentations de la rotation

Dans un espace 3D (l'espace Euclidien $\mathbb{R}^3$ ici), une succession de 3 rotations (en général axées sur la base orthonormée de l'espace) permet de couvrir toutes les orientations possibles d'un solide dans un repère (angles d'Euler). Chaque rotation peut modifier le repère précédent (rotations extrinsèques E) ou garder l'initial (intrinsèques I). Ainsi, pour une rotation sur $\vec{Z}$ puis sur le nouvel axe $\vec{X}$ et enfin sur le nouvel axe $\vec{Z}$, on note qu'il s'agit d'une rotation de convention $E - Z - X - Z$.

Toute rotation ou composition de rotations sur $\mathbb{R}^3$ peut être décrite par une rotation équivalente autours d'un vecteur unitaire $\vec{u}$ selon un angle θ (théorème des pôles Eulériens). Sous cette forme, la rotation d'un vecteur $\vec{v} \in \mathbb{R}^3$ en $\vec{v'}$ est obtenue par la formule de Rodrigues (démonstration en annexe B) où . désigne le produit scalaire sur $\mathbb{R}^3$ et $\wedge$ le produit vectoriel :

$$\vec{v'} = \vec{v}\cos(\theta) + (\vec{u} \wedge \vec{v})\sin(\theta) + \vec{u}(\vec{u}.\vec{v})(1 - \cos(\theta)) \tag{1.2}$$

Également, toute rotation ou composition de rotations sur $\mathbb{R}^3$ peut être décrite par une matrice $R \in \mathbb{R}^{3\times3}$. Cette matrice est orthogonale $R^T R = I$ et son déterminant vaut 1. L'ensemble des matrices respectant ces critères appartiennent au groupe spécial orthogonal $SO(3)$ également nommé groupe des rotations 3D, topologiquement identique à la 3-sphère. Lorsque l'on effectue une rotation de θ autours d'un vecteur unitaire $\vec{u} = (u_x, u_y, u_z)^T \in \mathbb{R}^3$, la matrice de rotation R est obtenue par l'équation 1.3 (démonstration en annexe B). Tout vecteur de l'espace $\vec{v} \in \mathbb{R}^3$ devient $\vec{v'}$ par la rotation R selon la transformation $\vec{v'} = R\vec{v}$.

$$\begin{gathered} c = \cos(\theta), s = \sin(\theta) \\ R = \begin{pmatrix} u_x^2(1-c)+c & u_x u_y(1-c) - u_z s & u_x u_z(1-c) + u_y s \\ u_x u_y(1-c) + u_z s & u_y^2(1-c)+c & u_y u_z(1-c) - u_x s \\ u_x u_z(1-c) - u_y s & u_y u_z(1-c) + u_x s & u_z^2(1-c)+c \end{pmatrix} \end{gathered} \tag{1.3}$$

Enfin, toute rotation ou composition de rotation peut être décrite par un quaternion unitaire aussi appelé *versor*. Un quaternion est un nombre hypercomplexe construit

sur 3 imaginaires purs i, j, k respectant les relations dites "quaternionniques" :

$$\begin{gathered} i^2 = j^2 = k^2 = ijk = -1 \\ ij = k \quad ji = -k \quad jk = i \quad kj = -i \quad ki = j \quad ik = -j \end{gathered} \tag{1.4}$$

Tout quaternion q de composantes $(a, b, c, d) \in \mathbb{R}^4$ peut se représenter en tant que :

- nombre hypercomplexe $q = a + bi + cj + dk$
- vecteur de l'espace $(1, i, j, k)$: $q = (a, b, c, d)$
- scalaire-vecteur $q = a + \vec{v}$ avec $\vec{v} = (b, c, d)$ un quaternion imaginaire pur

On distingue trois opérateurs avec les quaternions : le produit de Hamilton (multiplication standard), le produit scalaire $.$ et le produit vectoriel $\wedge$. Le produit de Hamilton entre deux quaternions $q_1 = a_1 + \vec{u_1}$ et $q_2 = a_2 + \vec{u_2}$ s'écrit (démonstration en annexe A) :

$$q_1 q_2 = (a_1 + \vec{u_1})(a_2 + \vec{u_2}) = a_1 a_2 - \vec{u_1}.\vec{u_2} + a_1\vec{u_2} + a_2\vec{u_1} + \vec{u_1} \wedge \vec{u_2} \tag{1.5}$$

Les quaternions unitaires ($||q||^2 = a^2 + b^2 + c^2 + d^2 = 1$) forment un groupe de Lie nommé $Sp(1)$ topologiquement identique à la 3-sphère (comme $SO(3)$). Tout quaternion unitaire peut s'écrire sous la forme $q = e^{\theta u} = \cos(\theta) + \sin(\theta)\vec{u}$ avec $\vec{u}$ un quaternion imaginaire pur unitaire. Soit $\vec{v} \in \mathbb{R}^3$ et $q = \cos(\theta) + \sin(\theta)\vec{u}$, la relation $\vec{v'} = q\vec{v}q^{-1}$ correspond à la formule de Rodrigues (Éq. 1.2) pour un angle 2θ (démonstration en annexe B) :

$$\vec{v'} = q\vec{v}q^{-1} = \cos(2\theta)\vec{v} + \sin(2\theta)(\vec{u} \wedge \vec{v}) + \vec{u}(\vec{u}.\vec{v})(1 - \cos(2\theta)) \tag{1.6}$$

Si l'on réalise les mêmes calculs pour $q = -\cos(\theta) - \sin(\theta)u$, le résultat pour $\vec{v'}$ reste inchangé. Ceci signifie que toute représentation axe-angle possède deux représentations par des quaternions : q et $-q$. Ceci est primordial lorsque l'on souhaite établir une métrique sur $Sp(1)$.

On montre ainsi que ce groupe est étroitement lié à $SO(3)$ puisque tout quaternion unitaire $q = \cos(\theta/2) + \sin(\theta/2)\vec{u}$ réalise une rotation de θ autour de $\vec{u}$. Le passage d'un quaternion unitaire $q = a + bi + cj + dk$ à une matrice de rotation R est alors trivial (démonstration en annexe B) :

$$R = \begin{pmatrix} 1 - 2c^2 - 2d^2 & 2bc - 2ad & 2ac + 2bd \\ 2ad + 2bc & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\ 2bd - 2ac & 2ab + 2cd & 1 - 2b^2 - 2c^2 \end{pmatrix} \tag{1.7}$$

1.2.2 Rotations et translations

La translation selon le vecteur $\vec{t} = (t_x, t_y, t_z)^T \in \mathbb{R}^3$ d'un point $P = (P_x, P_y, P_z, 1)^T \in \mathbb{R}^4$ de la scène en $P' = (P'_x, P'_y, P'_z, 1)^T \in \mathbb{R}^4$ peut être écrite sous forme matricielle en utilisant les coordonnées homogènes. Cette forme autorise l'insertion d'une rotation en remplaçant l'identité de la partie supérieure gauche par une matrice $R \in SO(3)$ afin obtenir la composition d'une rotation **puis** d'une translation notée $\left(R, \vec{t}\right)$:

$$P' = \left(R, \vec{t}\right) P = \begin{pmatrix} R & \vec{t} \\ 0^T & 1 \end{pmatrix} P = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} P \tag{1.8}$$

L'inverse de cette transformation notée $(R, \vec{t})^{-1}$ est obtenue par :

$$(R, \vec{t})^{-1} = \begin{pmatrix} R^{-1} & -R^{-1}\vec{t} \\ 0^T & 1 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{21} & R_{31} & -R_{11}t_x - R_{21}t_y - R_{31}t_z \\ R_{12} & R_{22} & R_{32} & -R_{12}t_x - R_{22}t_y - R_{32}t_z \\ R_{13} & R_{23} & R_{33} & -R_{13}t_y - R_{23}t_y - R_{33}t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.9}$$

Une matrice définie selon $\left(R, \vec{t}\right) = \begin{pmatrix} R & \vec{t} \\ 0^T & 1 \end{pmatrix}$ avec $R \in SO(3)$ et $\vec{t} \in \mathbb{R}^3$ appartient au groupe spécial Euclidien de dimension 3 noté $SE(3)$. **La pose d'un objet dans l'espace étant définie par une translation et une rotation, elle sera alors représentée dans la suite des travaux par une matrice de** $SE(3)$.

1.3 Gammes d'objets considérés

La méthode d'estimation de pose proposée dans ces travaux nécessite d'employer des pièces ayant un matériau uniforme et faiblement texturé ainsi que des géométries peu symétriques. Une méthode de calcul de complexité du modèle CAO est proposée permettant de corréler l'erreur d'estimation de pose avec la distribution des courbures de l'objet. Une grande variété de courbures facilite l'extraction de caractéristiques par un réseau de neurones et des objets d'apparences trop simples ne sont pas forcément éligibles à une telle technique. Deux jeux de données sont introduits pour évaluer la méthode d'estimation de pose. Le premier, bien connu dans la littérature scientifique, comporte des objets disposés sur une table. Il permet de comparer les performances de l'estimation de pose proposée dans ces travaux avec celles de l'état de l'art. Le second est construit à l'aide d'une pièce de l'entreprise dont la demande en dévracage automatisé est la plus forte.

1.3.1 Géométrie

Matériaux et symétries

Les géométries des pièces industrielles sont variées et peuvent posséder des symétries radiales en particulier dans le cas des géométries tubulaires (Fig. 1.5). Dans ce cas précis, certaines métriques angulaires peuvent être modifiées pour prendre en compte l'invariance en pose sur un angle. Cependant, l'entreprise n'ayant pas de besoin de dévracage de pièces à symétrie radiale, cette notion n'est pas abordée dans ces travaux.

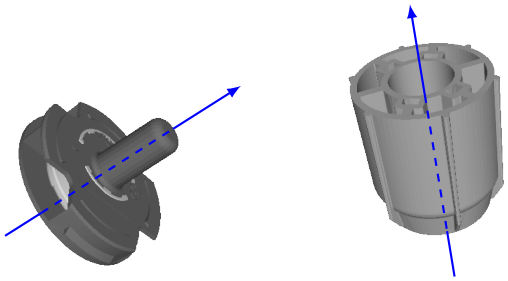


FIGURE 1.5 – Pièces à géométrie tubulaire possédant une symétrie radiale (pièces de flotteur de réservoir).

Complexité

Les réseaux de neurones construisent des caractéristiques propres à une pièce pour inférer correctement son orientation et peuvent donc éprouver des difficultés à traiter des géométries peu complexes. Afin d'être corrélée avec l'estimation de pose, la complexité d'un modèle CAO peut être évaluée en utilisant la distribution des courbures de chaque vertex **??**. Soient v_i un vertex du modèle et ses voisins v_j pris avec une connectivité en 1-anneau, le défaut angulaire du vertex est obtenu en sommant les angles α_j formés par deux vertex voisins successifs (Fig. 1.6).

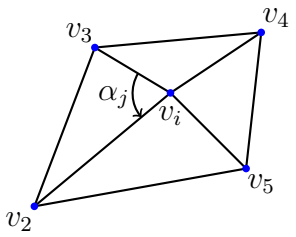


FIGURE 1.6 – Vertex v_i avec la connectivité en 1-anneau formant les angles α_j.

La courbure à chaque vertex v_i, notée κ_i, est obtenue en divisant le défaut angulaire par la somme du tiers de l'aire des triangles voisins A_i (théorème de Gauss-Bonnet) :

$$\kappa_i = \frac{3}{A_i}\left(2\pi - \sum_{j=0}^{n-1}\alpha_j\right) \tag{1.10}$$

La densité de probabilité des courbures p, est calculée à travers une estimation par noyau gaussien avec une bande passante h, sur l'ensemble des N vertex :

$$p(x) = \frac{1}{Nh\sqrt{2\pi}}\sum_{i=1}^{N} e^{-\frac{(x-\kappa_i)^2}{2h^2}} \tag{1.11}$$

En particulier, la bande passante h peut être fixée de façon optimale par [94] :

$$C_1 = \int e^{-x^2}dx, \quad C_2 = \int x^2 e^{-x^2/2}dx, \quad h = \left(\frac{243C_1}{35C_2^2N}\right)^{1/5}\sigma \tag{1.12}$$

La complexité du modèle CAO est alors :

$$\mathcal{C} = -\sum p(x) log_N(p(x)) \tag{1.13}$$

1.3.2 Jeux de données

LINEMOD

Le jeu de données LINEMOD [41] est un standard pour l'évaluation d'algorithmes d'estimation de pose. Ce jeu est composé d'images RGB-D (images avec profondeur) d'une scène comportant des objets placés sur un table. Seule la pose d'un objet (placé au centre de la table) est connue. Isoler l'objet de la scène ne nécessite alors pas de segmenter plusieurs instances. Chaque objet (une dizaine au total) est accompagné de son modèle CAO. Ceci permet d'employer les techniques présentées dans ces travaux afin de générer des vues pour l'apprentissage des translations et rotations par réseaux de neurones. En revanche, les objets étant posés sur une surface plane, toutes les rotations ne peuvent être représentées dans la scène ce qui limite son intérêt pour évaluer une solution de dévracage. De plus, certains objets possèdent des textures et ne peuvent par conséquent, pas être traités dans le cadre des travaux de thèse. Ainsi, dans le jeu de données, seuls les modèles suivants sont considérés : APE, DUCK, CAN, CAT (Fig. 1.7). La complexité des objets est étudiée d'après l'équation 1.13 (Table 1.1).

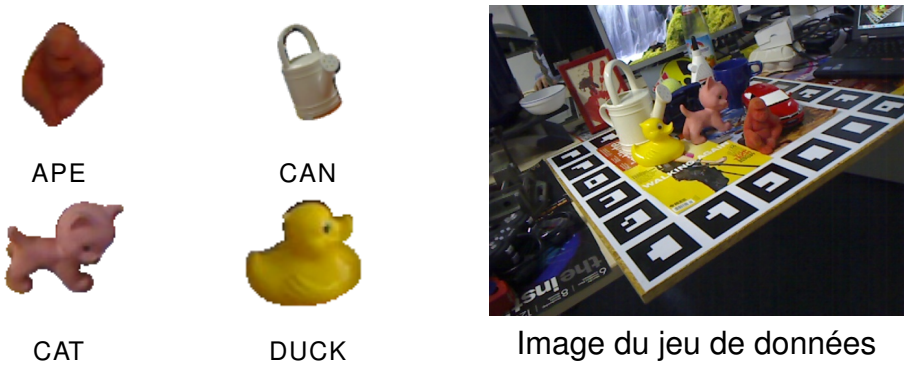


FIGURE 1.7 – Objets APE, CAN, CAT, DUCK de LINEMOD (gauche) avec une image du jeu de données (droit).

TABLE 1.1 – Complexité entropique des modèles de LINEMOD.

APE	CAN	CAT	DUCK
0.2	0.5	0.3	0.15

MULTITUDE

Afin d'évaluer le module de vision proposé pour le dévracage, une pièce plastique de l'entreprise répondant aux critères d'utilisation du module de vision (Table 1) est considérée. Cette pièce fait partie d'une boîte à air servant à récupérer les éventuels résidus d'huile d'un moteur de voiture tout en évitant une surpression dans le carter. Cette pièce nommée BREATHER (Fig. 1.8), est de couleur noire avec une texture brillante. Elles est alors difficilement utilisable dans des solutions de vision standard par laser et sans contrôle de luminosité. Il s'agit donc d'une parfaite candidate à l'entraînement et l'évaluation des réseaux de neurones pour l'estimation de pose. D'après l'équation 1.13, sa complexité vaut environ 0.6.

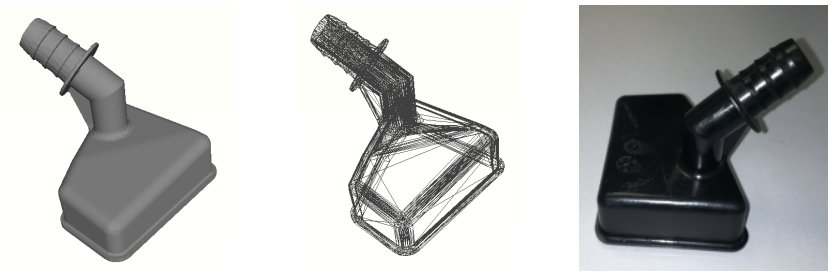


FIGURE 1.8 – Le modèle CAO de BREATHER (gauche) avec une vue en fils de fer (centre) et une image réelle (droite).

Synthèse et conclusion du chapitre

Ce chapitre introduit la représentation d'un modèle CAO par maillage triangulaire. Pour être utilisés dans un rendu 3D, les sommets du modèle sont indexés puis unifiés et leurs normales sont calculées. Ceci permet de diminuer la taille du modèle en mémoire, de faciliter l'injection des coordonnées dans OpenGL et d'utiliser un modèle d'ombrage pour colorier les pixels. Le modèle peut également être représenté par un nuage de points obtenu par un échantillonnage homogène du maillage. Ceci est important lorsque deux modèles 3D doivent être comparés dans un algorithme de recalage. Puis, les différentes transformations qu'il est possible d'appliquer au modèle sont décrites. La représentation d'une rotation axe-angle sous la forme d'un quaternion est détaillée avec l'algèbre associée pour le manipuler. Enfin, le choix des objets à traiter selon leur nature géométrique et leurs matériaux est explicité. Une méthode d'estimation de complexité d'un modèle CAO basée sur la distribution des courbures et proposée. Les réseaux de neurones se nourrissant de caractéristiques géométriques particulières pour apprendre l'orientation d'un pièce, cette complexité peut être mise en corrélation avec les performances en estimation de pose. Deux jeux de données utilisés dans cette thèse sont présentés : LINEMOD et MULTITUDE. Le jeu de données LINEMOD est partiellement traité pour ne considérer que les objets sans texture. Le jeu MULTITUDE est composé d'une pièce plastique industrielle de l'entreprise. Cette pièce est également sans texture mais possède des géométries plus complexes que LINEMOD ainsi qu'une plus large gamme de poses. Ce jeu de données est donc naturellement plus représentatif d'une situation de dévracage que LINEMOD.

La manipulation d'un modèle est une étape essentielle pour la génération de données et de prises de vues par OpenGL. Ces opérations sont détaillées dans le chapitre suivant (Ch. 2).

CHAPITRE 2

CRÉATION D'IMAGES

Introduction

Ce second chapitre détaille le processus de génération de données virtuelles et d'obtention de données réelles servant respectivement à entraîner et tester les réseaux de neurones.

Dans un premier temps, les différentes méthodes employées pour réaliser des rendus 3D sont présentées (Sec. 2.1). La projection d'un modèle dans le plan caméra est formalisée afin d'introduire les paramètres intrinsèques de la caméra. Les pixels obtenus sont ensuite coloriés selon une lumière incidente et différents paramètres du matériau. Afin de produire un nombre suffisant de vues en un temps court, la génération d'images est intégrée à un pipeline graphique sous forme de calculs tensoriels. Les procédés permettant de choisir des orientations sur $SO(3)$ ainsi que générer un vrac de pièces sont détaillées (Sec. 2.2). Enfin, afin d'évaluer les réseaux de neurones, une méthode estimant une matrice d'homographie sur des images est introduite afin de générer un jeu de données réelles (Sec. 2.3).

2.1 Rendu 3D

Afin de générer des prises de vue d'un modèle CAO dans une scène, deux opérations sont nécessaires : la projection dans un plan image et la coloration des pixels. La première est issue d'un modèle mathématique construit sur les paramètres intrinsèques d'une caméra. La seconde cherche à appliquer une couleur aux pixels précédemment calculés en utilisant la position d'une source lumineuse dans la scène et son angle d'incidence sur la surface de l'objet. Pour générer un grand nombre d'images en un temps réduit, l'opération de projection peut être entièrement mise sous forme tensorielle et groupée avec l'opération de coloriage au sein d'un programme exécuté dans la carte graphique.

2.1.1 Projection

Toute surface s du modèle CAO peut être projetée dans le plan image d'une caméra selon une matrice de projection. Par des simples lois optiques (Fig. 2.1), un point de l'espace P_W dans le repère du monde $(\vec{X}, \vec{Y}, \vec{Z})$ est vu à travers une lentille de focale f puis projeté en P_C sur une grille de photorécepteurs selon l'équation 2.1 dans le repère du capteur $(\vec{x}, \vec{y}, \vec{z})$ (Fig. 2.1).

$$P_C = (P_{C_x}, P_{C_y}) = \left(-f \frac{P_{W_x}}{P_{W_z}}, f \frac{P_{W_y}}{P_{W_z}} \right) \tag{2.1}$$

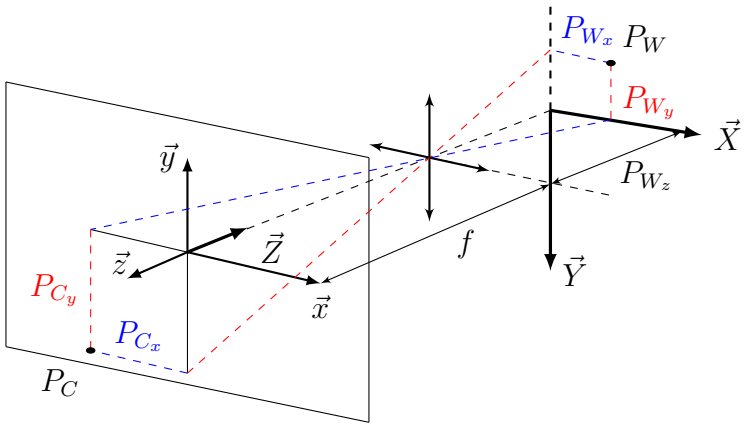


FIGURE 2.1 – Projection en P_C d'un point de l'espace $P_W = (P_{W_x}, P_{W_y}, P_{W_z})$.

La grille de photorécepteurs de taille (s_w, s_h) va contenir les informations lumineuses donnant naissance aux pixels de l'image étant donnée sa taille (I_w, I_h). Ce passage d'informations continues à discrètes est appelé rastérisation ou matricialisation de l'image. Un moyen numérique simple d'effectuer cette opération est de premièrement normaliser l'information sur le capteur $(P_{C_x}, P_{C_y}) \in [0,1]^2$ puis d'appliquer une mise à l'échelle en prenant les parties entières du résultat pour obtenir un pixel p dans le repère $(\vec{u}, \vec{v})$ (Fig. 2.2) :

$$p = (p_u, p_v) = \left(\lfloor -f \frac{I_w P_{W_x}}{s_w P_{W_z}} + \frac{I_w}{2} \rfloor, \lfloor f \frac{I_h P_{W_y}}{s_h P_{W_z}} + \frac{I_h}{2} \rfloor \right) \in [[0, I_w] \times [[0, I_h]] \tag{2.2}$$

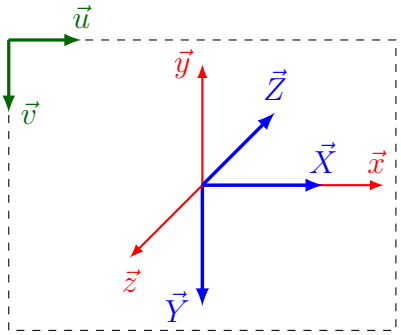


FIGURE 2.2 – Différenciations entre le repère du monde $(\vec{X}, \vec{Y}, \vec{Z})$, du capteur $(\vec{x}, \vec{y}, \vec{z})$ et de l'image $(\vec{u}, \vec{v})$.

Ce modèle n'est cependant pas toujours compatible avec un capteur sur le marché car il ne tient pas compte de la forme rectangulaire des photorécepteurs est du décentrage possible entre la lentille et le centre du capteur. De plus, les proportions du capteurs peuvent ne pas correspondre avec celles de l'image souhaitées. Ceci nécessite donc de couper une partie du capteur pour limiter les effets de zoom. Ce modèle est alors souvent modifié pour employer des focales (f_X, f_Y) ainsi qu'un centre d'image (c_x, c_y) tous deux exprimés en pixels :

$$p = \left(\lfloor -f_X \frac{P_{W_x}}{P_{W_z}} + c_x \rfloor, \lfloor f_Y \frac{P_{W_y}}{P_{W_z}} + c_y \rfloor \right) \tag{2.3}$$

2.1.2 Ombrage de Phong

Le modèle d'ombrage de Phong permet de calculer empiriquement la couleur des pixels issus de la rastérisation. Ce modèle de coloration est défini d'après trois composantes :

- **Ambiante** : intensité renvoyée en tout point de l'espace
- **Diffuse** : intensité renvoyée selon l'angle d'une source lumineuse
- **Spéculaire** : intensité renvoyée selon l'angle d'une source lumineuse et le point d'observation

La scène possède une quantité de luminosité ambiante i_a et une source lumineuse d'une quantité de luminosité diffuse i_d et spéculaire i_s. Il est également possible de compléter ce modèle en considérant des paramètres similaires pour le matériau d'un objet (m_a, m_d, m_s) en ajoutant un paramètre de brillance m_b contrôlant la taille des saillances lumineuses. L'intensité lumineuse I d'une information sur une surface de normale $\vec{n}$, vue selon $\vec{v}$ avec une lumière placée en $\vec{l}$ (Fig. 2.3) est obtenue selon l'équation 2.4.

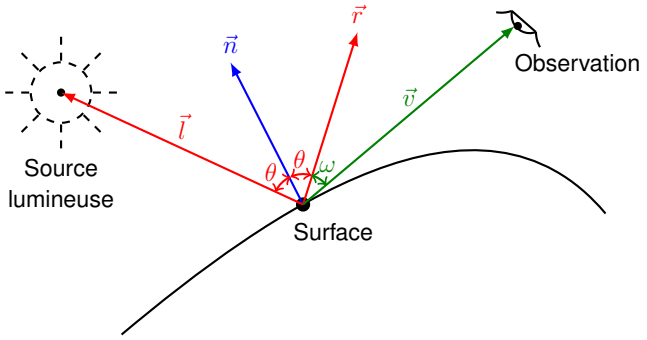


FIGURE 2.3 – Lumière placée en $\vec{l}$, réfléchie en $\vec{r}$ par la normale $\vec{n}$ et observée selon $\vec{v}$.

$$\begin{aligned} I_a &= i_a m_a \\ I_d &= i_d m_d (\vec{l}.\vec{n}) = i_d m_d \cos(\theta) \\ I_s &= i_s m_s (\vec{r}.\vec{v})^{m_b} = i_s m_s \cos^{m_b}(\omega) \\ I &= I_a + I_d + I_s \end{aligned} \tag{2.4}$$

La variation des paramètres de l'ombrage de Phong permet de générer des saillances lumineuses plus ou moins fortes sur la pièce (Fig. 2.4).

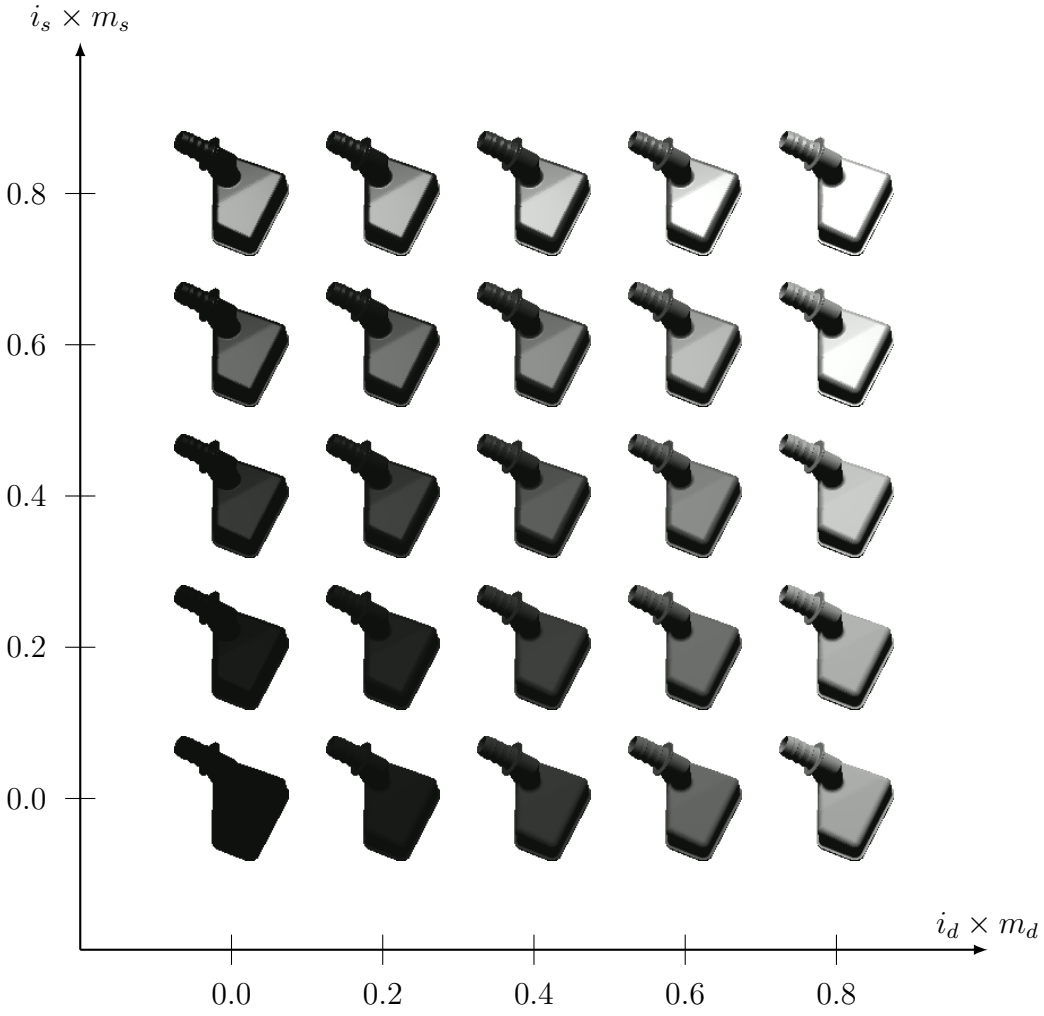


FIGURE 2.4 – Rendus d'une pièce selon la quantité diffuse $i_d \times m_d$ et spéculaire $i_s \times m_s$.

Pour réaliser un rendu de pièces d'une même matière, les composantes du matériau (m_a, m_d, m_s) peuvent être omises en ne considérant uniquement que des composantes d'une lumière équivalente $i_d \equiv i_d \times m_d$.

2.1.3 Intégration OpenGL

La génération d'images en OpenGL repose sur deux programmes exécutés successivement par la carte graphique appelés *shaders* : le *vertex shader* et le *fragment shader*. Le *vertex shader* place les vertex du modèle dans la scène et réalise la projection dans le repère de la caméra. La rastérisation est ensuite effectuée avant de passer dans le *fragment shader* se chargeant de la coloration des pixels selon un modèle de luminosité.

Lors d'un rendu, toutes les surfaces de la pièce ne sont pas visibles. OpenGL se charge alors au travers d'un test sur Z appelé *depth-test*, de supprimer les informations cachées de la géométrie de l'objet. Ce processus nécessite de garder en mémoire la distance à la caméra P_{W_z} de chaque point. Afin d'homogénéiser les données projetées, les coordonnées sont normalisées sur $[-1, 1]$ avant la rastérisation afin de former un cube OpenGL (3 coordonnées sur x, y et z). Toute information non contenue dans le cube n'est pas gardée en mémoire.

Normalisation des projections

La normalisation des projections $\{P_C\} \rightarrow \{\bar{P_C}\}$ s'effectue en utilisant la taille de l'image souhaitée (I_w, I_h) :

$$\bar{P_C} = \left(-\frac{2f_X P_{W_x}}{I_w P_{W_z}} + \frac{2c_x}{I_w} - 1, \frac{2f_Y P_{W_y}}{I_h P_{W_z}} + \frac{2c_y}{I_h} - 1 \right) \in [-1, 1]^2 \tag{2.5}$$

La profondeur P_{W_z} est normalisée sur $[-1, 1]$ selon un P_{W_z} proche, z_{near}, et éloigné, z_{far}, avec une loi affine sur $\frac{1}{P_{W_z}}$. Il est important de noter que le gradient de la profondeur normalisée et plus important pour P_{W_z} proche de z_{near} (Fig. 2.5). Le *depth-test* sera donc moins performant pour des informations éloignées.

$$\bar{P_{W_z}} = -\frac{1}{P_{W_z}} \times \frac{2z_{\text{near}} z_{\text{far}}}{z_{\text{far}} - z_{\text{near}}} + \frac{z_{\text{near}} + z_{\text{far}}}{z_{\text{far}} - z_{\text{near}}} \tag{2.6}$$

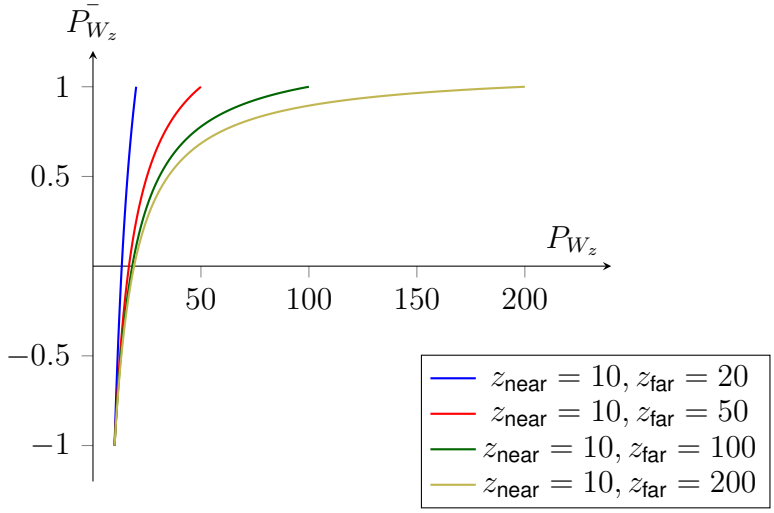


FIGURE 2.5 – Valeurs de profondeurs normalisées $\bar{P_{W_z}}$ en fonction de la profondeur P_{W_z} et des intervalles $[z_{\text{near}}, z_{\text{far}}]$.

Forme matricielle

Le principe du vertex shader repose sur de simples opérations matricielles optimisées dans le processeur graphique. En utilisant les coordonnées homogènes, la projection définie dans l'équation 2.5 ainsi que la normalisation de la profondeur définie dans l'équation 2.6 peuvent être combinées sous une forme matricielle H explicitée dans l'équation 2.7. Notons que cette équation fait intervenir un signe de proportionnalité entre les points du modèle et les projections associées du fait des coordonnées homogènes (la dernière composante de la projection devant être 1, il convient de normaliser par $-P_{W_z}$). **Par la suite, un signe d'égalité sera utilisé.**

$$\begin{pmatrix} \bar{P_{C_x}} \\ \bar{P_{C_y}} \\ \bar{P_{W_z}} \\ 1 \end{pmatrix} \propto H \begin{pmatrix} P_{W_x} \\ P_{W_y} \\ P_{W_z} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \bar{P_{C_x}} \\ \bar{P_{C_y}} \\ \bar{P_{W_z}} \\ 1 \end{pmatrix} \propto \begin{pmatrix} \frac{2f_X}{I_w} & 0 & 1 - \frac{2c_x}{I_w} & 0 \\ 0 & -\frac{2f_Y}{I_h} & 1 - \frac{2c_y}{I_h} & 0 \\ 0 & 0 & -\frac{z_{\text{near}} + z_{\text{far}}}{z_{\text{far}} - z_{\text{near}}} & \frac{2z_{\text{near}} z_{\text{far}}}{z_{\text{far}} - z_{\text{near}}} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} P_{W_x} \\ P_{W_y} \\ P_{W_z} \\ 1 \end{pmatrix} \tag{2.7}$$

Le rendu par OpenGL permet non seulement d'obtenir l'image RGB de l'objet selon un modèle de luminosité, mais également d'obtenir l'image de profondeur normalisée (valeur de $\bar{P_{W_z}}$ pour chaque pixel). Pour obtenir une bonne précision dans cette image, les paramètres z_{near} et z_{far} sont réglés pour englober tout l'objet dans la scène. Avant la projection, le modèle est placé et orienté dans la scène à l'aide d'une matrice $(R, \vec{t}) \in SE(3)$ (Éq.1.8). Ainsi, pour un objet de rayon maximal r_{obj}, on utilise $z_{\text{near}} = t_z - r_{\text{obj}}$ et $z_{\text{far}} = t_z + r_{\text{obj}}$. La matrice de projection H (Éq. 2.7) est alors utilisée dans la définition de la fonction de projection $\Pi : \mathbb{R}^3 \to [-1, 1]^3$ comportant les paramètres (t_z, r_{obj}) :

$$\begin{aligned}\Pi(P_{W_x}, P_{W_y}, P_{W_z}; t_z, r_{\text{obj}}) \triangleq & \begin{pmatrix} \frac{2f_X}{I_w} & 0 & 1 - \frac{2c_x}{I_w} & 0 \\ 0 & -\frac{2f_Y}{I_h} & 1 - \frac{2c_y}{I_h} & 0 \\ 0 & 0 & -\frac{t_z}{r_{\text{obj}}} & \frac{t_z^2 - r_{\text{obj}}^2}{r_{\text{obj}}} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} P_{W_x} \\ P_{W_y} \\ P_{W_z} \\ 1 \end{pmatrix} \\ & \begin{pmatrix} \bar{P_{C_x}} \\ \bar{P_{C_y}} \\ \bar{P_{W_z}} \\ 1 \end{pmatrix} = \Pi(P_{W_x}, P_{W_y}, P_{W_z}; t_z, r_{\text{obj}})\end{aligned} \tag{2.8}$$

Il est possible d'inverser la fonction de projection Π afin de rétro-projeter à la fois les coordonnées normalisées des pixels et la profondeur normalisée dans la scène. Cependant, Π étant dépendante de t_Z, Π^{-1} n'est obtenue que si cette translation est connue. La fonction de rétro-projection $\Pi^{-1} : [-1, 1]^3 \to \mathbb{R}^3$ est définie par :

$$\begin{aligned}\Pi^{-1}(\bar{P_{C_x}}, \bar{P_{c_y}}, \bar{P_{W_z}}; t_z; r_{\text{obj}}) \triangleq & \begin{pmatrix} -\frac{t_z^2 - r_{\text{obj}}^2}{r_{\text{obj}}} \frac{I_w}{2f_x} & 0 & 0 & \frac{t_z^2 - r_{\text{obj}}^2}{r_{\text{obj}}} \left(1 - \frac{2c_x}{I_w}\right) \frac{I_w}{2f_x} \\ 0 & \frac{t_z^2 - r_{\text{obj}}^2}{r_{\text{obj}}} \frac{I_h}{2f_y} & 0 & \frac{t_z^2 - r_{\text{obj}}^2}{r_{\text{obj}}} \left(1 - \frac{2c_y}{I_h}\right) \frac{I_h}{2f_y} \\ 0 & 0 & 0 & \frac{t_z^2 - r_{\text{obj}}^2}{r_{\text{obj}}} \\ 0 & 0 & 1 & \frac{t_z}{r_{\text{obj}}} \end{pmatrix} \begin{pmatrix} \bar{P_{C_x}} \\ \bar{P_{C_y}} \\ \bar{P_{W_z}} \\ 1 \end{pmatrix} \\ & \begin{pmatrix} P_{W_x} \\ P_{W_y} \\ P_{W_z} \\ 1 \end{pmatrix} = \Pi^{-1}(\bar{P_{C_x}}, \bar{P_{c_y}}, \bar{P_{W_z}}; t_z; r_{\text{obj}})\end{aligned} \tag{2.9}$$

2.2 Jeu d'images virtuelles

Les différents modèles mathématiques présentés permettent de générer une image d'un objet dans une scène comportant une luminosité réglable. Pour construire un jeu d'images virtuelles servant à l'entraînement des différents réseaux de neurones pour l'estimation de pose, il convient dans un premier temps de choisir les différentes poses appliquées à l'objet. Deux méthodes sont présentées pour couvrir les poses d'un objet vu sur une table (espace $SO(2)$ pour LINEMOD) et couvrir l'ensemble des rotations spatiales (espace $SO(3)$ pour MULTITUDE). Pour les réseaux de segmentation, une méthode basée sur une simulation d'un lancer de pièces dans un bac est introduite. Elle permet de réaliser un vrac plus ou moins complexe comportant un nombre variable de pièces. Différentes techniques d'incrustation d'arrière-plans et d'augmentation de données sont employées pour parfaire le jeu de données.

2.2.1 Génération de pose

Tirage homogène sur $SO(2)$

L'orientation d'une pièce dans la scène peut être représentée par un triplet Eulérien (ϕ, θ, ψ) en convention $E-Z-X-Z$. Les deux premiers angles orientent la vue sur une sphère avec $(\phi, \theta) \in [-\pi, \pi] \times [0, \pi]$ tandis que $\psi \in [-\pi, \pi]$ définit un angle de rotation du plan de la caméra. Afin de générer des vues couvrant toutes les orientations possibles d'une pièce, échantillonner régulièrement les intervalles des angles ϕ et θ n'est pas une méthode souhaitable car la couverture angulaire n'est pas homogène au niveau des pôles de la sphère (Fig. 2.6). Si les subdivisions d'icosaèdres (géodes) permettent de répartir de façon homogène des points sur une sphère, elles sont néanmoins limitées à un nombre de points précis N pour chaque subdivision i de l'icosaèdre original (ayant 12 sommets) avec $N_i = 10 \times 4^i + 2$. Une technique plus complète consiste à utiliser une sphère de Fibonacci [28] (Annexe D) étant donnés un nombre de points N et le nombre d'or Φ :

$$\begin{gathered} \forall i \in \{0, ..., N\}, \\ y_i = \frac{2i + 1 - N}{N} \quad x_i = \sqrt{1 - y_i^2} \times \cos\left(2\pi i \left(2 - \Phi\right)\right) \quad z_i = \sqrt{1 - y_i^2} \times \sin\left(2\pi i \left(2 - \Phi\right)\right) \\ \phi_i = \text{atan2}(y_i, x_i) \quad \theta_i = \text{atan2}(\sqrt{x_i^2 + y_i^2}, z_i) \end{gathered} \tag{2.10}$$

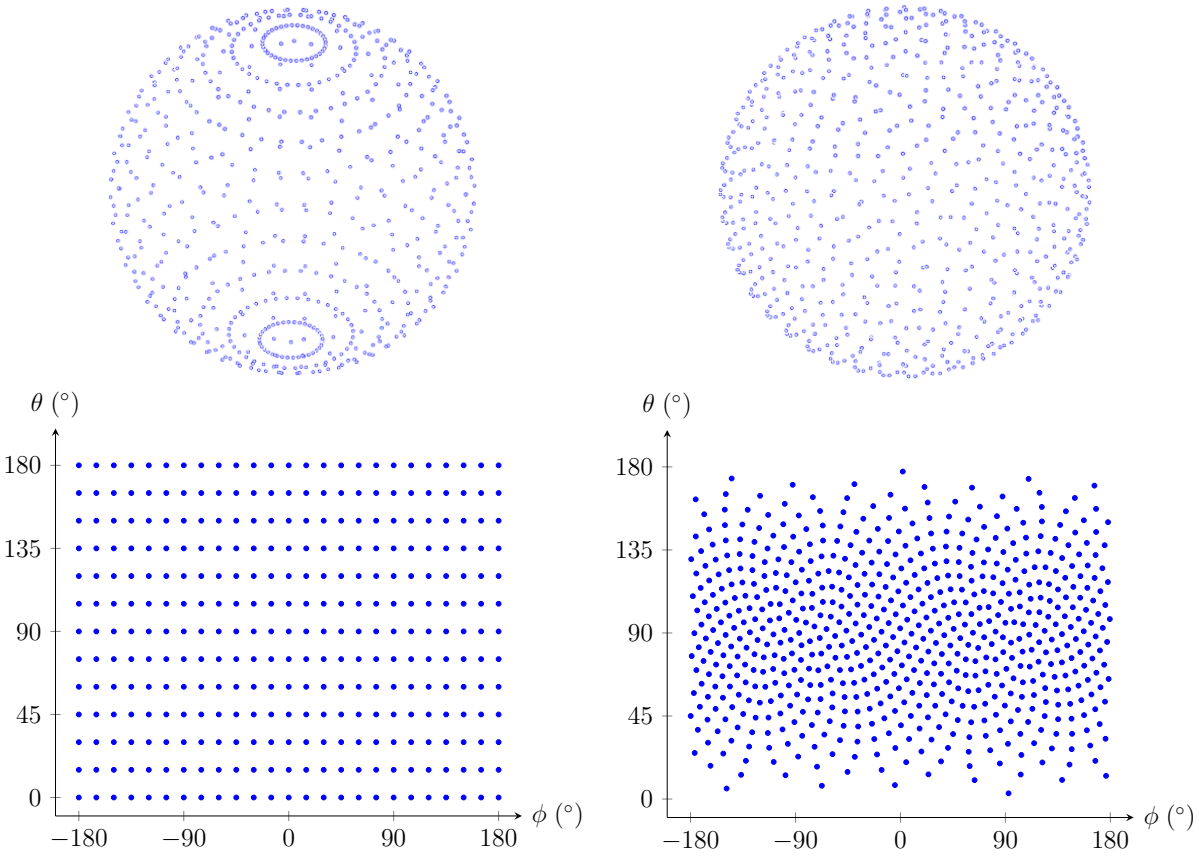


FIGURE 2.6 – Échantillonnage d'une sphère par intervalle angulaire (gauche) et par Fibonacci (droit) avec 650 points (Éq. 2.10).

La distance géodésique moyenne entre deux points proches sur une 2-sphère échantillonnées avec N points, $G(N)$, peut être approximée par (Annexe D) (Fig. 2.7) :

$$G\left(N\right) \approx \frac{3.36}{\sqrt{N}} \tag{2.11}$$

Tirage uniforme sur $SO(3)$

Une autre approche pour la génération de pose consiste à réaliser un tirage uniforme directement sur l'espace des rotations $SO(3)$ [4]. En initialisant des points sur $\mathbb{R}^3$ selon une séquence de Halton, il est possible d'obtenir un nombre souhaité de matrices de rotation couvrant de façon quasi-homogène l'espace $SO(3)$ (assimilé à une 3-sphère) [9]. Soit N le nombre de rotations souhaitées, les point $(x_i, y_i, z_i)_{\{i=1,\dots,N\}}$ sont donnés par la séquence de van der Corput (VDC) en base 2 et 3 tels que :

$$(x_i, yi, zi) = (\text{VDC}(3, i), \text{VDC}(2, i), i) \tag{2.12}$$

Les matrices de rotation résultantes R_i sont alors définies par :

$$V_i = \frac{1}{\sqrt{N}} \begin{pmatrix} \cos(2\pi y_i)\sqrt{z_i} \\ \sin(2\pi y_i)\sqrt{z_i} \\ \sqrt{N - z_i} \end{pmatrix} \quad R_i = -\left(I - 2V_i V_i^T\right) \begin{pmatrix} \cos(2\pi x_i) & \sin(2\pi x_i) & 0 \\ -\sin(2\pi x_i) & \cos(2\pi x_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

Pour $N > 1500$, le tirage est quasi-homogène et la distance géodésique moyenne entre deux points proches sur une 3-sphère échantillonnée avec N points, $G(N)$, peut être approximée par (Annexe D) (Fig. 2.7) :

$$G(N) \approx \frac{3.47}{\sqrt[3]{N}} \tag{2.14}$$

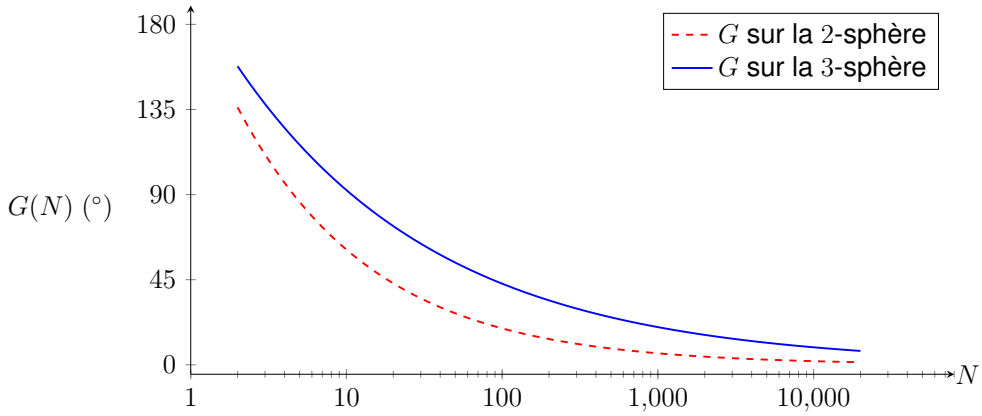


FIGURE 2.7 – Distance géodésique selon N points sur la 2-sphère et 3-sphère.

Processus de création d'images

La première stratégie consiste à obtenir différentes orientations selon la sphère de Fibonacci (Éq. 2.10), afin de simuler un jeu de données dont les images sont obtenues par une caméra tournant autours d'un objet fixe (LINEMOD). Le nombre de points est aléatoirement choisi dans un intervalle géodésique (Éq. 2.11). Ceci permet de s'assurer que les deux premiers angles de rotation (ϕ, θ) sont répartis de façon homogène et ne se recoupent pas entre les différents tirages. La troisième angle ψ peut être choisi sur un intervalle restreint de quelques degrés pour simuler une éventuelle rotation du plan caméra durant la prise d'images. Le quaternion q est calculé à l'aide du triplet (ϕ, θ, ψ) en utilisant la convention $E - Z - Y - Z$. La translation vérité de la pièce dans la scène, $\vec{t}$, est aléatoirement tirée afin que l'objet soit contenu dans l'image.
La seconde stratégie permet d'obtenir des poses générées par un tirage direct sur $SO(3)$ selon un intervalle géodésique (Éq.2.14), afin de simuler un jeu de données dont les orientations de l'objet sont diverses (MULTITUDE). Les matrices de rotations obtenues sont ensuite transformées en quaternion (Éq. B.5). La translation vérité de la pièce dans la scène, $\vec{t}$, est aléatoirement tirée selon les dimensions du bac.
Dans les deux stratégies, des paramètres de l'ombrage de Phong sont aléatoirement choisis pour chaque pose. Ils sont par la suite intégrés au *fragment shader* d'OpenGL pour obtenir l'image finale I (Fig. 2.8).

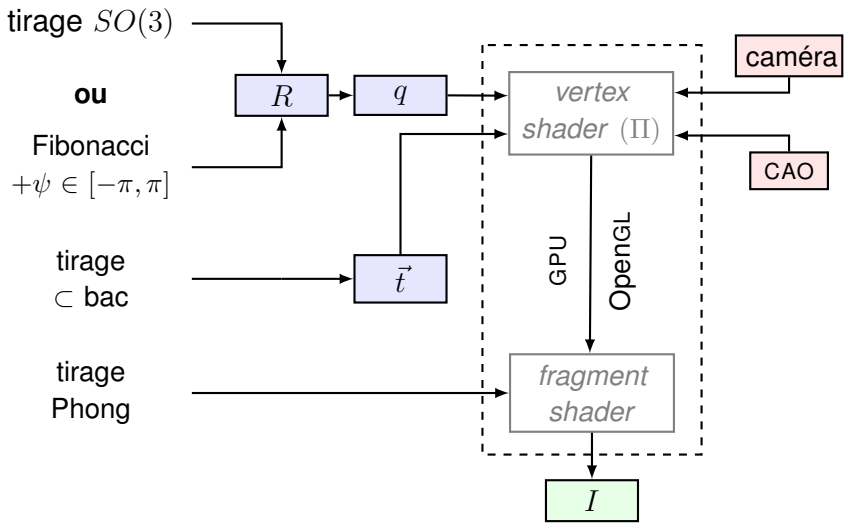


FIGURE 2.8 – Génération d'une image I selon une pose $(R, \vec{t})$ par OpenGL.

2.2.2 Génération de vrac

Il est possible de placer aléatoirement plusieurs modèles CAO dans la scène et d'injecter un arrière-plan afin de générer des images de bacs. Cependant, il est complexe de vérifier que les modèles ne se chevauchent pas et le rendu final apparaît peu réaliste. La génération de vrac est donc réalisée après un jeter de pièces dans une scène virtuelle à l'aide d'un moteur de collision (*Bullet*). Le bac est représenté par un modèle CAO simple, dont les propriétés physiques sont déterminées par des données URDF (inertie, masse, zones de collision...). Pour la zone de collision des pièces, l'enveloppe convexe du modèle CAO est utilisée.

Il n'est pas souhaitable d'initialiser les pièces au même point puisque si des modèles sont déjà imbriqués lors du lancement de la simulation, la gestion de collision n'est plus possible et les pièces sont expulsées. Pour éviter ce phénomène, les N modèles sont placés aléatoirement sur N étages espacés de façon homogène sur l'axe Z. Les boîtes englobantes de la pièce à l'étage i est à l'étage $i-1$ sont comparées. Si aucune intersection n'est détectée, la pièce de l'étage i est descendue à l'étage $i-1$ (Fig. 2.10). Le placement d'une pièce en (x, y) sur un étage est contrôlée selon une distribution de probabilité permettant de générer des vracs plus ou moins complexes. Soit $\vec{t_i}$ la position de chacune des N pièces du bac, la complexité du vrac peut être définie d'après l'entropie de la distribution obtenue par un estimateur gaussien de bande-passante $H \in \mathbb{R}^{3\times3}$ [91] :

$$
\begin{aligned}
H &= \left(\frac{4}{5N}\right)^{1/7} \begin{pmatrix} \hat{\sigma}_{tx} & 0 & 0 \\ 0 & \hat{\sigma}_{ty} & 0 \\ 0 & 0 & \hat{\sigma}_{tz} \end{pmatrix} \\
\hat{p}(\vec{t}) &= \frac{1}{N\sqrt{(2\pi)^3|H|}} \sum_{i=1}^{N} e^{-\frac{1}{2}(\vec{t}-\vec{t_i})^T H^{-1}(\vec{t}-\vec{t_i})} \\
\mathcal{C} &= -\sum_{i=1}^{N} \hat{p}(\vec{t_i}) \log\left(\hat{p}(\vec{t_i})\right)
\end{aligned}
\tag{2.15}
$$

L'arrêt de la simulation est programmé lorsque plus aucune pièce n'est en mouvement et que la moyenne des centroïdes des objets est contenue dans le bac (Fig. 2.10). Le lot de N poses $(R, \vec{t})_i$ (Fig. 2.9) est ensuite transmis à OpenGL (Fig. 2.8) afin de réaliser un rendu de plusieurs objets. L'image RGB de la scène est obtenu par un ombrage de Phong avec des paramètres aléatoires. Le masque binaire du premier-plan M est

obtenu par le canal alpha de l'image de sortie. Enfin, les masques binaires de chaque instance $\{M_i\}_{i=1,\dots,N}$ comportant des occlusions, sont extraits par coloriage binaire séquentiel de chaque pièce : la pièce considérée est à 1, les autres à 0.

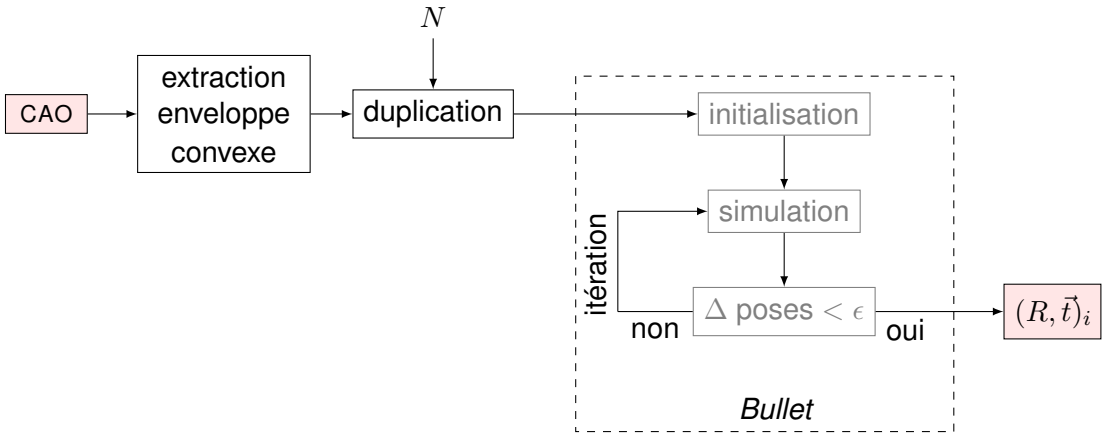


FIGURE 2.9 – Génération d'un vrac de N pièces donnant un lot de N transformations $(R, \vec{t})_i$.

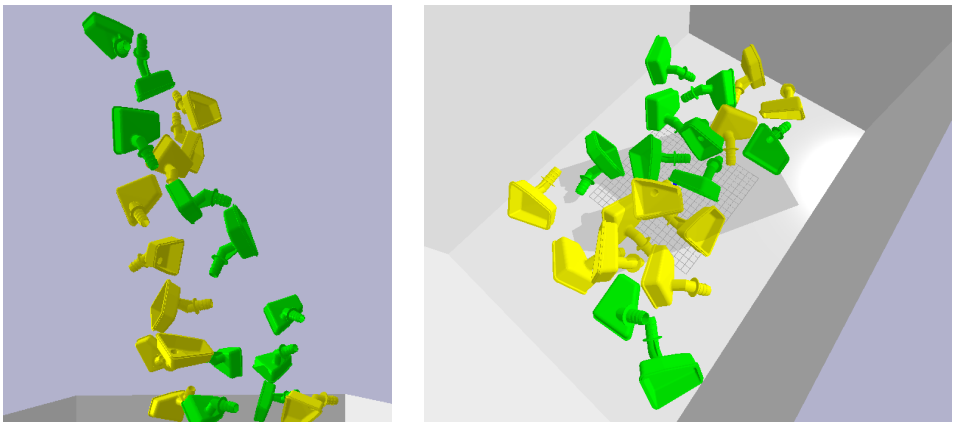


FIGURE 2.10 – Initialisation de la simulation par étages (gauche) et arrêt de la simulation de collisions du moteur *Bullet* (droite) (les couleurs des pièces sont non informatives).

2.2.3 Augmentation des données

Une fois le vrac généré, un arrière-plan est inséré parmi des photos de bacs en plastique. Différents contrastes et rotations de $180°$ sont appliqués afin d'augmenter la variabilité du fond. La génération de données étant réalisée sur un GPU dédié aux calculs, elle nécessite l'émulation d'un écran pour assurer le bon fonctionnement d'OpenGL. Cependant, cette technique ne permet pas de réaliser des rendus à plusieurs échelles et l'anti-crénelage (*anti-aliasing*) n'est donc pas disponible. La frontière entre la pièce et l'arrière-plan apparaît alors grossière (Fig. 2.11). Un flou Gaussien est donc appliqué sur le canal alpha du rendu (Fig. 2.11) afin d'incruster un fond de façon réaliste (Fig. 2.12).

FIGURE 2.11 – Images OpenGL brute (gauche) et avec contours adoucis (droite).

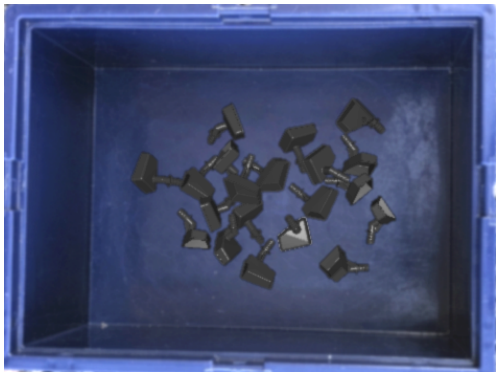


FIGURE 2.12 – Rendu d'un bac de pièces par OpenGL après simulation de collisions.

Différentes techniques de détérioration d'images sont ensuite appliquées afin d'augmenter les données mais aussi de diminuer le biais virtuel-réel : bruitage, flou et variations de gamma.

2.3 Jeu d'images réelles

Les réseaux de neurones étant entraînés sur des images synthétiques, il convient de tester leurs performances sur des images réelles. Si LINEMOD comporte des données réelles annotées, ce n'est pas le cas pour MULTITUDE. Deux jeux de données sont alors construits : un premier pour la segmentation d'instances et un second pour l'estimation de pose d'une pièce isolée. La création d'un jeu d'images d'une pièce isolée annotée avec sa translation et son orientation dans la scène fait intervenir une technique d'estimation d'homographie par cibles.

2.3.1 Vracs de pièces

Pour créer un jeu de données réelles comportant des pièces en vrac, le masque binaire du premier plan est obtenu en annotant manuellement les images. Le processus étant chronophage, seulement 10 images sont disponibles comportant un nombre variable de pièces (25 au maximum). Pour augmenter les données, différents contrastes et luminosités sont appliqués ainsi que des rotations de $180°$ du plan image (Fig. 2.13).

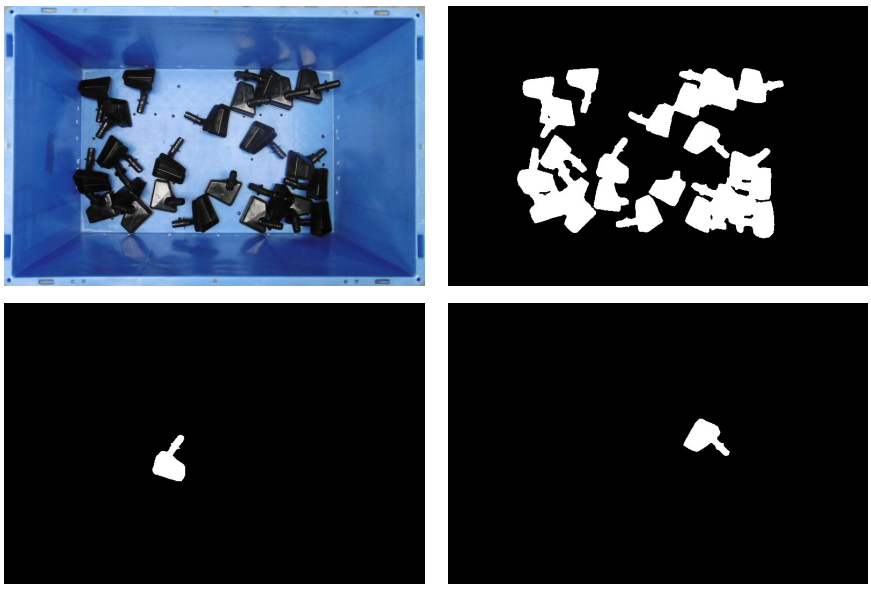


FIGURE 2.13 – Image de bac (en haut à gauche) et masque du premier-plan (en haut à droite) avec deux exemples d'instances (en bas).

2.3.2 Prises de vue

Afin d'évaluer quantitativement la méthode d'estimation de pose, un jeu d'images réelles de pièces industrielles est construit. Une pièce est placée au centre d'une scène comportant 4 cibles circulaires coplanaires et autours de laquelle, une caméra enregistre régulièrement des prises de vue. Un algorithme de détection de tâches (*blobs*) délimitant des zones connexes, permet dans un premier temps d'isoler les cibles dans l'image. Les cibles circulaires ayant des rayons croissants, leur identification dans les *blobs* est par la suite simplement basée sur leur aire respective dans l'image. En utilisant les coordonnées des cibles dans la scène, la matrice d'homographie est finalement calculée puis convertie en pose $(R, \vec{t}) \in SE(3)$ (Fig. 2.14).

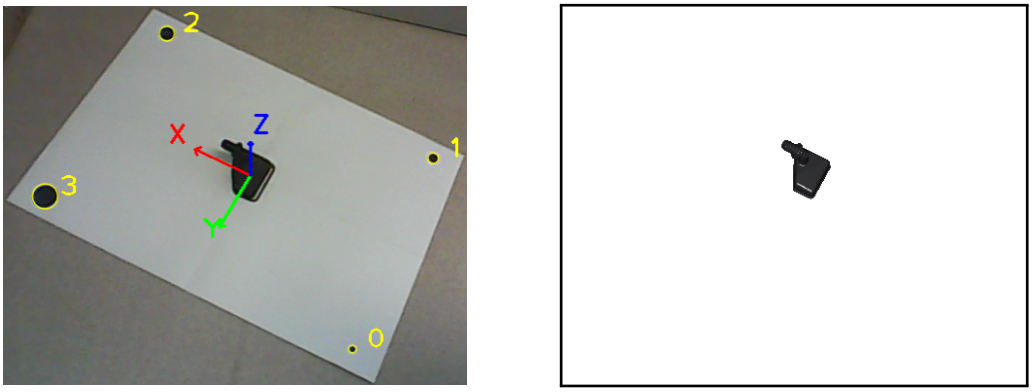


FIGURE 2.14 – Détection des cibles en *blob* avec homographie (gauche) et prise de vue OpenGL correspondante (droite).

Soient des cibles de coordonnées $\{C_i\}$ dans la scène (avec $C_{i_z} = 0$) et $\{P_i\}$ dans l'image, l'homographie H est définie par la relation :

$$\begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix} = H \begin{pmatrix} C_x \\ C_y \\ 0 \\ 1 \end{pmatrix} \tag{2.16}$$

La résolution du système homogène suivant permet d'obtenir la forme vectorielle de la matrice H notée $\mathrm{vec}(H)$:

$$\begin{pmatrix} -C_{1_x} & -C_{1_y} & -1 & 0 & 0 & 0 & P_{1_x}C_{1_x} & P_{1_x}C_{1_y} & P_{1_x} \\ 0 & 0 & 0 & -C_{1_x} & -C_{1_y} & -1 & P_{1_y}C_{1_x} & P_{1_y}C_{1_y} & P_{1_y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -C_{4_x} & -C_{4_y} & -1 & 0 & 0 & 0 & P_{4_x}C_{4_x} & P_{4_x}C_{4_y} & P_{4_x} \\ 0 & 0 & 0 & -C_{4_x} & -C_{4_y} & -1 & P_{4_y}C_{4_x} & P_{4_y}C_{4_y} & P_{4_y} \end{pmatrix} \begin{pmatrix} H_{11} \\ H_{12} \\ \vdots \\ H_{32} \\ H_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

$$M.\mathrm{vec}(H) = 0 \tag{2.17}$$

La résolution de l'équation 2.17 repose sur la décomposition en valeurs singulières (SVD) de $M = U\Sigma V^T$ avec U et V les vecteurs singuliers à gauche et à droite et Σ, la matrice diagonale contenant les valeurs singulières. La solution approchée $\tilde{H}$, est le vecteur $\vec{v_k}$ associé à la plus faible valeur singulière σ_k. Cette valeur singulière donne une estimation de l'erreur entre H et $\tilde{H}$ (en cas de valeur nulle, l'homographie estimée est parfaite). L'homographie étant la composition d'une translation $\vec{t}$, d'une matrice de rotation R, puis d'une matrice de projection A tel que $H = A(R, \vec{t})$, la décomposition de H s'obtient ainsi :

$$\begin{gathered} H = \begin{pmatrix} \vec{h_1} & \vec{h_2} & \vec{h_3} \end{pmatrix} \quad \lambda = \frac{1}{||A^{-1}\vec{h_1}||_2} \\ R = \begin{pmatrix} \lambda A^{-1}\vec{h_1} & \lambda A^{-1}\vec{h_2} & \left(\lambda A^{-1}\vec{h_1}\right) \wedge \left(\lambda A^{-1}\vec{h_2}\right) \end{pmatrix} \\ \vec{t} = \lambda A^{-1}\vec{h_3} \end{gathered} \tag{2.18}$$

Cependant, la matrice de rotation trouvée n'est pas toujours orthogonale ($R \notin SO(3)$). Il convient donc d'utiliser une décomposition polaire basée sur une SVD telle que $R = U\Sigma V^T$ pour obtenir $R = UV^T$. En réalisant une prise de vue en OpenGL avec un modèle CAO placé selon $\left(R, \vec{t}\right)$, le masque binaire de la pièce est estimé puis appliqué à l'image de la scène afin d'éliminer l'arrière-plan (Fig.2.14).

Synthèse et conclusion du chapitre

Ce chapitre présente les différentes méthodes mises en œuvre pour générer un jeu de données virtuelles et réelles. La projection dans un plan image par des coordonnées homogènes permet dans un premier temps d'obtenir les pixels de l'objet vu selon une pose de $SE(3)$. La coloration par ombrage de Phong permet d'attribuer une couleur à chaque pixel en utilisant une lumière définie par sa position dans la scène mais aussi par des composantes ambiantes, spéculaires et diffuses. Les paramètres du matériau permettent de générer des saillances lumineuses plus ou moins importantes dans l'image finale et ainsi, de garantir une grande variabilité dans le jeu de données. Le choix des poses peut être effectué par un tirage homogène sur $SO(2)$ par une sphère de Fibonacci lorsque l'objet est placé sur une table (LINEMOD) ou bien directement par un tirage uniforme sur $SO(3)$ (MULTITUDE). La création d'un vrac est rendue possible par une simulation d'un lancer de pièces utilisant une gestion des collisions. Enfin, la création d'un jeu de données réelles pour la segmentation est issu d'une annotation manuelle d'images prises dans des conditions variables de luminosité. Pour l'estimation de pose, une technique estimant la matrice d'homographie à partir de cibles dans une image est employée permettant de couvrir une partie de l'espace $SE(3)$.

DEUXIÈME PARTIE

Méthodes d'estimation de pose

CHAPITRE 3

SEGMENTATION DE PIÈCES

Introduction

Avant d'estimer la pose d'une pièce, chaque objet doit être isolé des autres dans l'image. Ce chapitre propose des méthodes permettant de traiter en partie ce problème dit de segmentation.

Dans un premier temps, après un état de l'art scientifique sur les différentes techniques de segmentation (Sec. 3.1), un premier réseau de type encodeur-décodeur permettant la suppression de l'arrière-plan de la scène est introduit (Sec. 3.2). Ce réseau est capable d'isoler les pièces du fond du bac mais également de générer une carte angulaire des pièces lorsque des occlusions sont présentes dans la scène. Pour isoler les pièces entre elles, un réseau de neurones récurrent est construit afin de produire une quantité souhaitée de masques binaires représentant chacun le découpage d'un objet (Sec. 3.3). Ce réseau permet également de garder en mémoire la liste des instances déjà produites afin de limiter les doublons. Plusieurs variantes de cette architecture sont proposées notamment pour nettoyer les masques produits ou encore se focaliser sur une partie de la scène avant la segmentation à l'aide d'un module d'attention visuelle.

3.1 État de l'art

La segmentation sémantique et d'instances est un problème considérablement traité dans la littérature en particulier ces dernières années suite à l'émergence des véhicules autonomes. La segmentation sémantique consiste à dissocier des informations dans une scène selon leur nature (arbre, humain...). La segmentation d'instances réalise un découpage dans une scène des instances d'une même nature (isoler chaque arbre, chaque humain...). Cet état de l'art décrit les méthodes principalement employées pour la segmentation : les analyses ascendantes et descendantes, la compréhension de scène et plus récemment, les réseaux de neurones.

3.1.1 Analyses ascendantes et descendantes

L'analyse ascendante cherche des détails locaux dans l'image pour remonter progressivement au contexte global afin de produire des segments homogènes formant une segmentation. En utilisant une représentation à plusieurs échelles [89], il est possible de mesurer des contrastes ou des frontières pour produire une carte de segmentation sans classification [90]. Des indices visuels basés sur des textures ou des intensités peuvent être utilisés dans un mélange d'experts (*mixture of experts*) pour du partitionnement de graphe produisant une segmentation hiérarchique [2]. L'utilisation de graphes pour la segmentation est également possible à travers des coupes de graphes [99][12][13].

3.1.2 Compréhension de scènes

L'utilisation d'informations contextuelles dans une image permet d'établir des relations logiques dans la scène et de faciliter ainsi la localisation d'objets [98]. Par exemple, dans une scène urbaine, une voiture se situe le plus souvent sur la route et apparaît plus large dans l'image qu'un piéton. De nombreuses techniques proposent de modéliser mathématiquement ces relations pour la localisation d'objets dans des images. Ces relations peuvent être établies par l'intermédiaire de champs de Markov aléatoires (MRF), en utilisant notamment les interactions entre les classes [57]. Également, l'utilisation de champs aléatoires conditionnels (CRF) permet de classifier les pixels d'une image en utilisant le contexte du voisinage [86][54].

3.1.3 Réseaux de neurones

Réseaux entièrement convolutifs

Les réseaux de neurones ont fortement contribué à l'amélioration des performances en segmentation notamment à travers l'utilisation de convolutions (CNN). Les dernières couches denses du réseaux (FC) peuvent être transformées pour produire une carte de probabilité d'appartenance à une classe [66][31]. Cependant, les couches de *pooling* du réseau tendent à produire une carte très grossière. Des interpolations bilinéaires ainsi que des CRF peuvent alors être appliqués en sortie de réseau pour affiner la découpe [15][21]. Il est possible de transformer ces CRF en réseaux de neurones récurrents pour obtenir un pipeline plus homogène [106]. En utilisant des convolutions dilatées, des informations à plusieurs échelles peuvent être prises en compte sans opérer sur des images à basse résolution [104]. Une autre stratégie baptisée *Segnet*, consiste à réaliser la segmentation à la même échelle que l'image source à travers un réseau encodeur-décodeur utilisant des couches de convolution après chaque modification de la taille d'image [5]. Si la segmentation d'instances demeure plus complexe pour un simple CNN, la méthode PFN (*Proposal-Free Network*) [64] utilise les caractéristiques apprises par le réseau de segmentation sémantique afin d'inférer le nombre d'instances d'une classe dans l'image puis, de les découper au sein du masque sémantique.

Propositions de régions

Afin d'optimiser la segmentation, la méthode R-CNN pré-sélectionne des régions de l'image selon des caractéristiques issues d'un CNN puis, les classifie par un séparateur à vaste marge (SVM) [25]. Une autre variante d'initialisation consiste à extraire des régions depuis des segmentations à plusieurs échelles par la suite groupées en explorant leur espace combinatoire [3]. Ces régions de l'image ainsi que leur segmentation peuvent être placées dans deux CNN puis unifiées dans un SVM pour prédire une carte de segmentation selon une suppression des non-maxima [35] (SDS). Cette méthode peut être améliorée en ajoutant une mise en correspondance avec un patron d'objet robuste aux occlusions [16], ou en utilisant des *hypercolonnes* contenant les informations de toutes les couches de convolution d'un pixel de l'image de sortie [34]. Les CNN étant contraints de travailler avec des tailles d'images fixes, la stratégie de SPP-*Net* consiste en un *pooling* pyramidal permettant aux réseaux de s'adapter à différentes dimensions des régions pré-extraites [38]. Cependant, SPP-*Net* et R-CNN

reposent sur un pipeline large impactant fortement la vitesse d'inférence. Des améliorations sont proposées sous le nom de *Fast* R-CNN [24] utilisant une architecture *end-to-end* puis, *Faster* R-CNN [81] employant un nouveau réseau proposant des régions d'intérêts dans l'image (*Region Proposal Network*, RPN) et enfin, *Mask* R-CNN [37] ajoutant une branche permettant de prédire le masque binaire de l'objet en plus de sa boîte englobante.

Récurrences

Lorsque les instances sont relativement proches entre-elles, des techniques utilisant des réseaux de neurones à mémoire court-long terme (LSTM) sont envisagées une première fois dans [93]. Les caractéristiques sont extraites par le réseau GoogLeNet [95] puis un LSTM détecte les fenêtres délimitant dans ce cas des visages humains dans une scène dense. Des travaux récents emploient les LSTM pour de la segmentation d'instances par pixel [83], nourris avec les caractéristiques d'un CNN. Ces LSTM sont capables de travailler avec des images 2D en gardant en mémoire les zones déjà observées. Une fonction de coût invariante aux permutations d'instances est construite par un algorithme Hongrois [77]. Les couches de *pooling* offrant une segmentation grossière, des CRF peuvent être employés pour raffiner les masques. En travaillant en RGB-D, [88] place les canaux de l'image 3D en entrée d'un CNN pour une extraction de caractéristiques. La couche de profondeur est transformée en 3 couches produisant une image RGB selon la distance du pixel à la table de support de l'objet, avant d'être transmise à un second CNN. Les caractéristiques de ces deux images sont fusionnées par la suite par un premier SVM pour une segmentation sémantique puis dans un second pour une segmentation d'instances. De façon similaire, [33] transforme la couche de profondeur en un modèle RGB dans lequel chaque canal correspond à une caractéristique spatiale : disparités horizontales, hauteur au sol et angle avec la gravité. Deux CNN sont employés en parallèle et la fusion de caractéristiques est effectuée par un SVM réalisant à la fois une segmentation sémantique et d'instances.

3.1.4 Positionnement de nos travaux

Les travaux de cette thèse pour la segmentation d'instances reposent sur un réseau encodeur-décodeur pour la segmentation sémantiques [5], puis un réseau récurrent de type convLSTM [93] pour délimiter les instances. Plusieurs implémentation du réseau récurrent sont proposées [83][80] afin de tenter de répondre à la problématique.

3.2 Segmentation sémantique

Une étape préalable et cruciale pour l'estimation de pose consiste à éliminer l'arrière-plan de la scène et ne garder que les objets d'intérêts. Cette segmentation sémantique peut être réalisée au moyen d'un réseau de neurones inférant un masque binaire séparant l'arrière-plan du premier-plan. Si plusieurs instances sont présentes dans la scène, ce masque peut être complété par une carte angulaire des pièces servant d'initialisation au découpage des différentes instances de l'objet. Des tests sont conduits sur LINEMOD comportant un seul objet d'intérêt dans la scène et sur MULTITUDE permettant d'éliminer efficacement le fond du bac.

3.2.1 Architecture

La première étape de la segmentation est la suppression de l'arrière-plan de la scène. L'image d'entrée RGB $I \in [0,1]^{h \times w \times 3}$ est envoyée dans un réseau de neurones à convolutions de type encodeur-décodeur [5] afin d'obtenir le masque binaire des pièces $M \in \{0,1\}^{h \times w}$. Cette architecture consiste une phase d'encodage utilisant une série de convolutions avec une réduction de la taille de l'image par 2 (*pooling*) puis, une phase de décodage employant des convolutions transposées (déconvolutions) avec un pas (*stride*) de 2 pour doubler la taille de l'image. En fin de réseau, la taille de l'image produite est identique à celle d'entrée. Le réseau utilise des sauts de connexion (*skip connections*) [36] permettant aux couches de déconvolution, d'utiliser une partie du contexte des couches de convolution, afin de produire une image fortement dépendante de celle d'entrée (Fig 3.1).

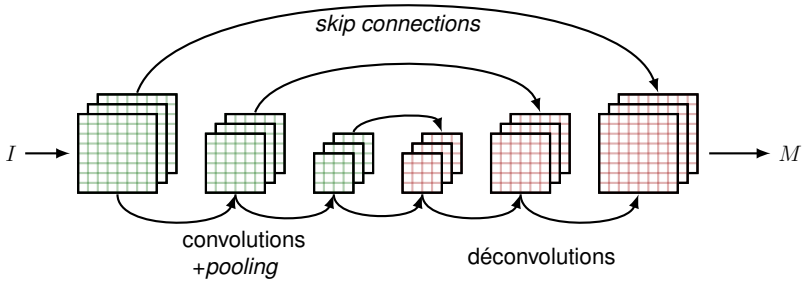


FIGURE 3.1 – Réseau encodeur-décodeur pour la suppression de l'arrière-plan.

Ces réseaux encodeurs-décodeurs ont récemment montré leurs capacités dans la littérature notamment pour la segmentation [5] ou la transformation d'images [107]. De plus, l'utilisation de convolutions ne requiert que peu de ressources en mémoire et l'apprentissage du réseau par rétro-propagation de l'erreur est rapide. Toutes les non-linéarités des couches sont des fonctions non saturantes (*ReLU*) et la première couche de chaque groupe de convolutions et déconvolutions utilisent des normalisations par *batch* [45] (Table.3.1).

TABLE 3.1 – Détails des couches du réseau de segmentation sémantique. Les couches en caractères gras utilisent des normalisations par *batch*.

Encodage		Décodage	
couche	taille	couche	taille
conv$_{11}$	$3, 32$	**deconv**$_{11}$	$3, 256, 2$
conv$_{12}$	$3, 64$	*skip*$_1$	conv$_{53}$+deconv$_{11}$
pooling$_1$	2	deconv$_{12}$	$3, 128, 1$
conv$_{21}$	$3, 64$	**deconv**$_{21}$	$3, 128, 2$
conv$_{22}$	$3, 96$	*skip*$_2$	conv$_{47}$+deconv$_{21}$
pooling$_2$	2	deconv$_{22}$	$3, 128, 1$
conv$_{31}$	$3, 96$	**deconv**$_{31}$	$3, 96, 2$
conv$_{32}$	$3, 128$	*skip*$_3$	conv$_{31}$+deconv$_{31}$
pooling$_3$	2	deconv$_{32}$	$3, 96, 1$
conv$_{41}$	$3, 128$	**deconv**$_{41}$	$3, 64, 2$
conv$_{42}$	$3, 128$	deconv$_{42}$	$3, 64, 1$
↓		**deconv**$_{51}$	$3, 32, 2$
conv$_{47}$	$3, 128$	deconv$_{52}$	$3, 1, 1$
conv$_{48}$	$3, 256$	*skip*$_4$	entrée+deconv$_{52}$
pooling$_4$	2	deconv$_{61}$	$3, 2, 1$
conv$_{51}$	$3, 256$		
conv$_{52}$	$3, 256$		
conv$_{53}$	$3, 256$		
conv$_{54}$	$3, 512$		
pooling$_5$	2		

Détails :	
conv	Couche de convolution
deconv	Couche de déconvolution
skip	Saut de connexion
pooling	Diminution de la taille de l'image
taille	Taille des filtres, nombre de filtres, pas

3.2.2 Inférence

Masque binaire

Lorsque les pièces sont correctement séparées dans l'image (pas d'occlusions), un simple masque binaire suffit à correctement préparer la segmentation d'instances (par détection de contours). Le réseau prédit donc une image de la même dimension que la source avec un seul canal. La fonction de coût du réseau est alors une simple MSE (erreur quadratique moyenne) entre le masque vérité $\hat{M}$ et la prédiction M.

Cartes angulaires

Lorsque des occlusions sont présentes, le réseau précédent peut être modifié pour produire une carte angulaire des pièces. Le centroïde de chaque instance est calculé en tant que centre du contours de la pièce sans occlusions. Depuis ce centroïde, 8 masques binaires couvrant la pièce tous les $45°$ sont extraits (Fig. 3.2). La dernière couche du réseau (Fig. 3.1) infère alors 9 canaux correspondant aux 8 angles et à l'arrière-plan de la scène. La somme des masques donnant une image composée de valeurs unitaires, il est possible d'utiliser une non-linéarité *softmax* en sortie, afin d'apprendre le réseau avec une fonction de coût en entropie croisée. Chacune des pièces étant labellisées tous les $45°$ depuis leur centre, le réseau de segmentation peut par la suite davantage se concentrer sur une instance et ses contours.

FIGURE 3.2 – Image synthétique de bac (gauche) et carte angulaire des pièces (droite).

3.2.3 Performances

Le réseau de segmentation utilise une descente de gradient *Adam* [52] et tous les paramètres sont initialisés selon une distribution uniforme de Glorot [26]. Le réseau est implémenté en Theano Lasagne et entraîné durant 1000 epochs. Les poids donnant la plus faible erreur sur un jeu de validation sont conservés. Pour évaluer le réseau de segmentation sémantique, la moyenne des intersection sur l'union (IOU) entre prédiction et vérité est employée (MIOU) ainsi que l'IOU 0.9 quantifiant la proportion d'exemples dont l'IOU est supérieure à 0.9. Pour l'inférence des cartes angulaires, l'IOU est utilisée sur chaque intervalle angulaire (AIOU).

LINEMOD

Le jeu de données de LINEMOD contient des séquences RGB d'une scène comportant un objet d'intérêt parmi plusieurs. Le réseau de segmentation doit alors éliminer l'arrière-plan de la scène mais également supprimer les autres objets. Cependant, le jeu de données ne dispose pas de masque binaire sémantique délimitant la pièce dans l'image. Pour obtenir le masque vérité de chaque image de la séquence, la pose vérité $(R, \vec{t})$ est utilisée avec le modèle CAO de l'objet et les paramètres intrinsèques de la caméra dans le pipeline OpenGL, afin de générer les pixels attribués à la pièce. Pour chaque objet, 15% des données sont utilisées pour l'entraînement, 15% pour la validation et le reste pour les tests. Les performances sont présentées dans le tableau 3.2.

TABLE 3.2 – Erreurs en segmentation sur LINEMOD.

Métrique	Objet			
	APE	**CAN**	**CAT**	**DUCK**
MIOU	0.94	0.90	0.93	0.94
IOU 0.9	0.93	0.60	0.93	0.94

Le réseau de segmentation permet d'éliminer efficacement l'arrière-plan de la scène. Dans l'image, d'autres objets possédant un matériau similaire (e.g. la voiture rouge et le singe rouge, le gobelet jaune et le canard jaune) ne sont pas détectés. Ceci montre que le réseau ne se base pas uniquement sur la couleur pour délimiter un

objet (Fig. 3.3). Lorsque l'objet présente des trous (e.g. anse de l'arrosoir) le réseau produit bien un masque troué permettant d'éliminer l'arrière plan au sein du masque. Cependant, ces résultats sont difficilement comparables avec la littérature puisque le jeu de données LINEMOD ne comporte pas de masques vérités.

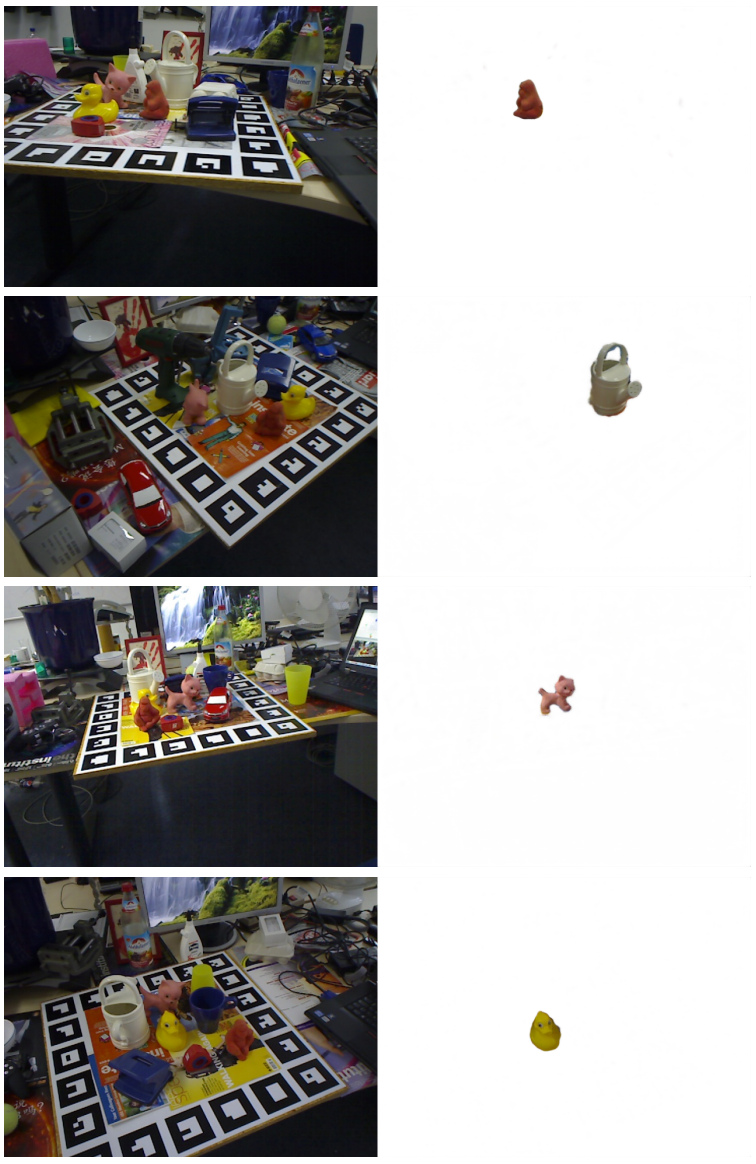


FIGURE 3.3 – Exemples de segmentation sur LINEMOD pour les différents objets.

MULTITUDE

Pour le jeu de données MULTITUDE, les images générées de bac sont utilisées pour l'entraînement. À chaque epoch, 20 images de bacs sont générées avec un nombre de pièces aléatoirement tiré entre 5 et 30. Pour chaque image générée, les paramètres de luminosité de la scène sont tirés aléatoirement. La complexité du bac est croissante durant l'entraînement permettant au réseau de comprendre dans un premier temps la géométrie d'une pièce et par la suite, de délimiter correctement les pièces de l'arrière-plan. Les tests se concentrent sur le réseau de segmentation sémantique mais aussi sur l'inférence de la carte angulaire. Les performances sont présentées dans le tableau 3.3.

TABLE 3.3 – Erreurs en inférence sémantique et de cartes angulaires sur MULTITUDE.

Inférence	Métrique	Bac
Sémantique	mIOU	0.87
Angulaire	maIOU	0.30

Le réseau encodeur-décodeur inférant le masque du premier-plan présente des performances correctes bien que certains masques soient mal inférés notamment lorsque la pièce est vue de dessous (Fig. 3.4). L'inférence de carte angulaires est performante pour des bacs de faible complexité. Lorsque la complexité augmente, les pièces isolées sont inférées sans difficulté tandis que les cartes angulaires au sein de lots de pièce tendent à être bruitées (Fig. 3.5). Cependant, dans la majorité des cas, au moins une pièce dans le lot dispose d'une carte angulaire correcte.

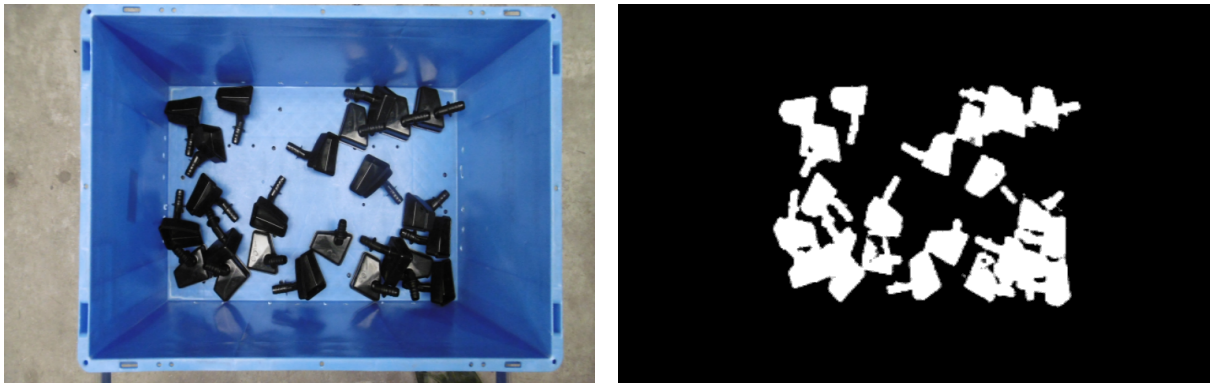


FIGURE 3.4 – Exemple d'inférence de masque binaire du premier-plan (droite) sur une image réelle de bac (gauche).

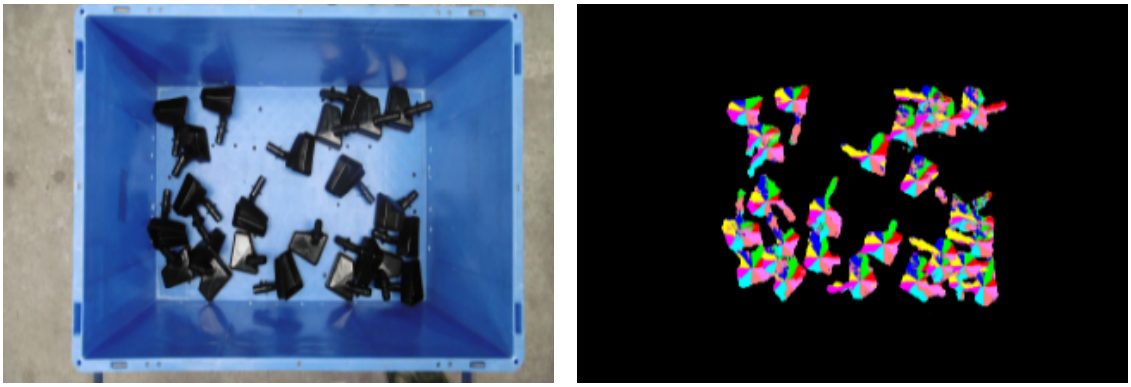


FIGURE 3.5 – Inférence de cartes angulaires (droite) depuis une image réelle (gauche).

3.3 Segmentation d'instances

Une fois l'arrière-plan de la scène éliminé, les instances doivent être découpées entre elles. Lorsque les pièces sont espacées et qu'aucune occlusion n'est présente dans l'image, une simple détection de contours permet une segmentation d'instances efficace. Dans le cas contraire, un réseau de neurones récurrent permet de proposer séquentiellement des instances en tenant compte des précédentes afin de ne pas produire de doublons. Différentes architectures sont mises en œuvre à la lumière de l'état de l'art et testées sur le jeu de données MULTITUDE.

3.3.1 Réseaux récurrents

Architecture

Pour segmenter un nombre donné d'instances dans une image, un réseau récurrent est employé. Cette récurrence est assurée par une cellule *long short-term memory* (LSTM) utilisant des convolutions (convLSTM) [103]. Chaque temporalité de la cellule produit une proposition de masque (en tant qu'état caché), M_i, découpant une instance dans la scène. L'état caché M_i est propagé à la temporalité suivante avec l'image source sans arrière-plan I', afin de tenir compte de la segmentation précédente (Fig. 3.6). Les convolutions sont réglées pour que l'état caché possède la même taille que l'image source I'.

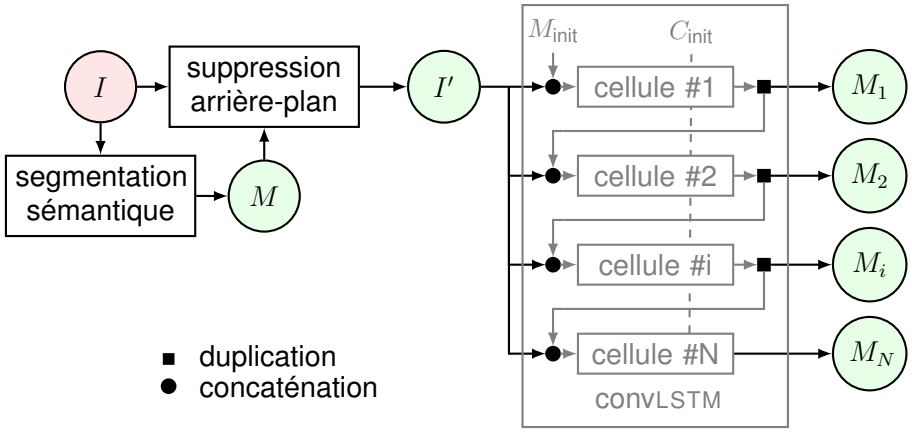


FIGURE 3.6 – Proposition d'instances M_i à l'aide d'un convLSTM.

Fonction de coût

La difficulté de la segmentation d'instances repose également sur la fonction de coût du réseau de neurones. Lorsqu'une instance est proposée, elle doit être attribuée à une des instances de la vérité afin d'être évaluée. Cette attribution est issue de la maximisation des intersection sur l'union (IOU) entre les prédictions M_i et les vérités $\hat{M}_j$ (Fig. 3.7). L'IOU de masques non-binaires $\in [0, 1]$ peut être définie par une version relaxée [55] telle que :

$$\text{IOU}\left(M_i, \hat{M}_j\right) = \frac{\langle M_i, \hat{M}_j \rangle}{||M_i||_1 + ||\hat{M}_j||_1 - \langle M_i, \hat{M}_j \rangle} \tag{3.1}$$

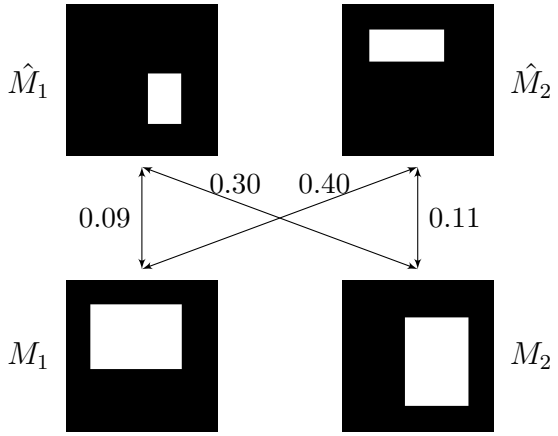


FIGURE 3.7 – Graphe des IOUs entre les masques prédits et vérités.

Cette définition permet de définir une métrique valide en tant que $1 - \text{IoU}$ [53]. Afin que le réseau apprenne à arrêter de segmenter lorsque toutes les pièces de la scène on été segmentées, le nombre de prédiction N est fixé selon le nombre d'instances vérité $\hat{N}$ tel que $N = \hat{N} + 2$. La matrice des IOU, notée $\overline{\text{IOU}}$ est alors :

$$\overline{\text{IOU}} = \begin{pmatrix} \text{IOU}\left(M_0, \hat{M}_0\right) & \dots & \text{IOU}\left(M_0, \hat{M}_{\hat{N}}\right) \\ \vdots & \text{IOU}\left(M_i, \hat{M}_j\right) & \vdots \\ \text{IOU}\left(M_N, \hat{M}_0\right) & \dots & \text{IOU}\left(M_N, \hat{M}_{\hat{N}}\right) \end{pmatrix} \tag{3.2}$$

L'attribution entre les prédictions M_i et les vérités $\hat{M}_j$ selon le maximum d'IOU, est réalisée par l'intermédiaire d'un algorithme Hongrois telle qu'une instance prédite peut être attribuée au maximum à une instance vérité et inversement. Soit $\delta_{i,j} \in \{0,1\}^{N,\hat{N}}$ l'attribution de la prédiction i avec la vérité j, les propriétés suivantes sont respectées :

$$\begin{aligned} \forall i \in [1, N], \quad \sum_{j=1}^{\hat{N}} \delta\left(i,j\right) \leq 1 \\ \forall j \in [1, \hat{N}], \quad \sum_{i=1}^{N} \delta\left(i,j\right) \leq 1 \end{aligned} \tag{3.3}$$

La fonction de coût peut alors s'écrire comme la somme des IOU attribuées :

$$\begin{aligned} \mathcal{L}_S &= -\langle \bar{\text{IOU}}, \delta \rangle \\ &= -Tr\left(\bar{\text{IOU}}\delta^T\right) \\ &= -\sum_{i=1}^{N}\sum_{j=1}^{\hat{N}} \text{IOU}\left(M_i, \hat{M}_j\right)\delta\left(i,j\right) \end{aligned} \tag{3.4}$$

Performances

Ce réseau est implémenté en Theano Lasagne et entraîné sur quelques images de vrac virtuels pour tester l'architecture. Cependant, le *framework* Theano n'autorise par la production d'un nombre variable de masques dans une cellule récurrente (compilation du réseau au préalable) ainsi, d'autres tests ont été conduits en Pytorch.

En entraînant le réseau avec une image unique ne comportant que peu d'instances, la fonction de coût diminue très faiblement. Le réseau arrive difficilement à isoler une instance et l'optimisation est effectuée globalement sur toute l'image (Fig. 3.8). Une interprétation possible est la difficulté d'activation en sortie du convLSTM produisant un masque aux valeurs faibles ne permettant pas un calcul pertinent de l'IOU. L'attribution Hongroise oscille régulièrement et le réseau n'arrive pas à cibler une pièce en particulier. Pour contrer cet effet, des travaux [83] proposent de compléter la sortie des réseaux récurrents par une inhibition spatiale servant à nettoyer et correctement seuiller le masque de sortie.

FIGURE 3.8 – Inférence des 6 premiers masques par le convLSTM après 500 epochs.

3.3.2 Réseaux récurrents avec inhibition

Les propositions d'instances du convLSTM peuvent être fortement bruitées et des travaux proposent donc de nettoyer les masques à l'aide d'un module d'inhibition [83]. Ce module permet également d'attribuer un score de confiance variant de 0 à 1 à l'instance segmentée. Le réseau propose des instances de confiance décroissante et lorsque cette dernière passe sous un seuil, l'itération du réseau récurrent est stoppée et l'image ne contient plus de pièces à segmenter. Dans les travaux originaux [83], le réseau récurrent considère chaque état caché de la cellule du convLSTM comme une proposition d'instance comportant un nombre spécifié de canaux (caractéristiques). La couche de convolution du module d'inhibition permet d'aplatir ces caractéristiques et de former un masque de segmentation. Le passage dans un log*softmax* prépare ce masque à un seuillage réalisé au moyen d'un biais appris, afin de produire le masque final de l'instance M_i. Le score de confiance s_i est issu d'une simple couche dense travaillant sur l'état caché, dont la sortie est passée dans une sigmoïde (Fig. 3.9).

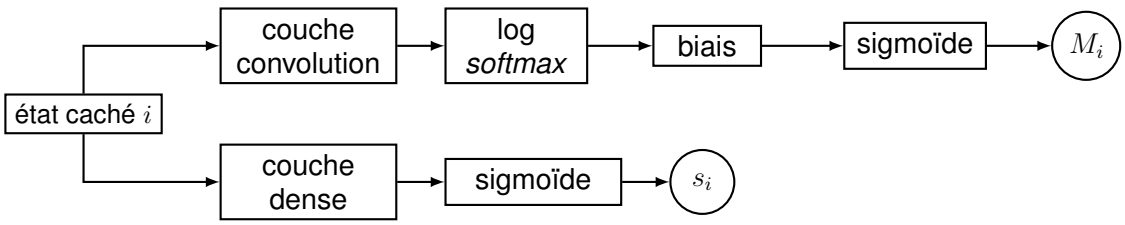


FIGURE 3.9 – Module d'inhibition d'un état caché du convLSTM produisant le masque binaire d'une instance M_i et son score de confiance associé s_i [83].

Pour apprendre ces branches et notamment le score de confiance, la fonction de coût basée sur l'attribution des IoU (Éq.3.4) est modifiée pour ajouter un terme d'entropie croisée binaire pénalisant le réseau lorsque le nombre d'instances prédites N est supérieur au nombre d'instances dans la vérité $\hat{N}$:

$$\begin{aligned} c_i &= [i \leq \hat{N}] \in \{0, 1\} \quad \text{(crochet d'Iverson)} \\ \mathcal{L}_s &= -\sum_{i=1}^{N} c_i \log(s_i) + (1 - c_i) \log(1 - s_i) \end{aligned} \tag{3.5}$$

Performances

À l'image du réseau précédent, cette architecture a été implémentée en Theano Lasagne et également en Lua Torch. Le score de confiance est très rapidement appris par la couche dense du module d'inhibition. En revanche, la production d'instances est fortement bruitée. L'état caché de la cellule du LSTM reste peu activé et l'activation en log+*softmax* produit des valeurs fortement négatives que le biais ne peut pas compenser. La sigmoïde propose alors des activations très faibles et en dépit d'une normalisation, les instances ne sont pas correctement segmentées (Fig. 3.10). Il semble que le LSTM éprouve des difficultés à travailler sur la scène entière et ne produit pas des zones ciblées dans l'image. Pour résoudre ce problème, un module d'attention visuelle est proposé dans la littérature afin de ne segmenter qu'une partie de l'image.

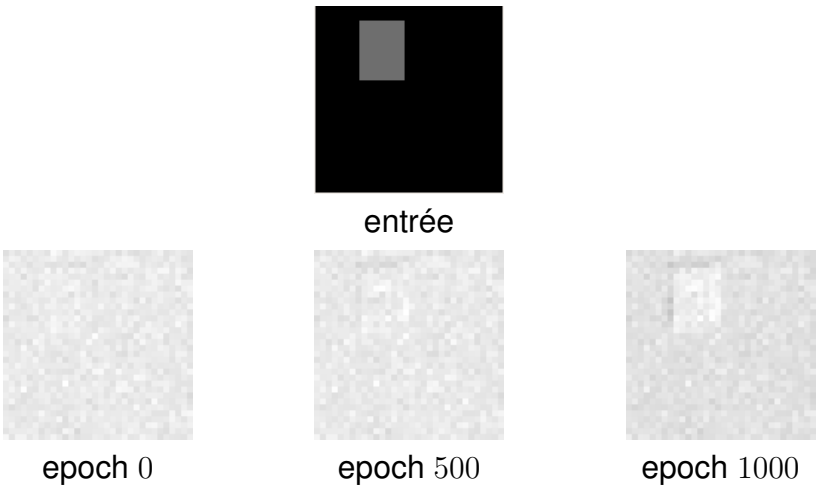
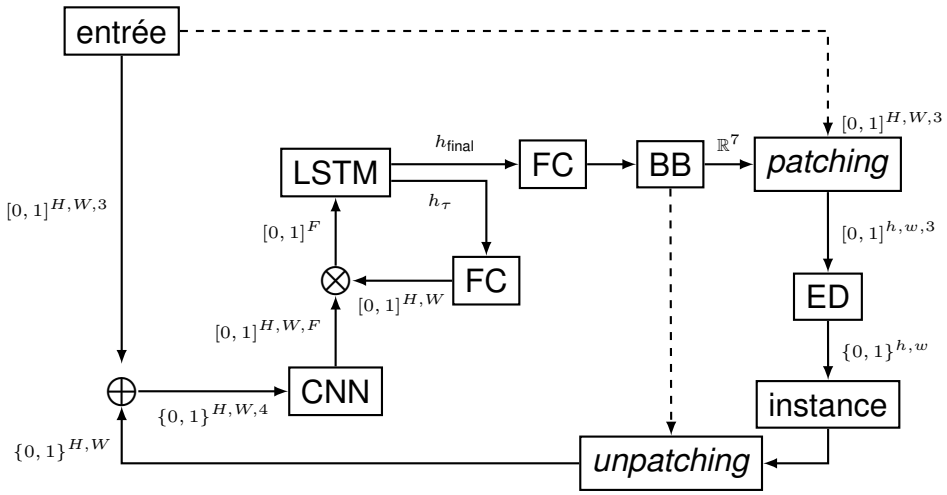


FIGURE 3.10 – Faible inférence après inhibition pour une seule instance.

3.3.3 Réseaux récurrents avec attention visuelle

La segmentation d'instances sur l'image entière de la scène paraît trop complexe pour le précédent réseau. Des travaux [80] proposent donc de travailler sur des zones locales de l'image pour segmenter une instance. Une zone est proposée par un module d'attention visuelle construit sur un réseau récurrent LSTM sous la forme de coordonnées d'une boîte englobante. Un réseau encodeur-décodeur segmente cette zone de l'image pour obtenir un masque binaire d'une instance. L'architecture du réseau se décompose alors en trois parties. La première étape consiste à partir d'une image $I \in [0,1]^{H,W,3}$ à construire un lot de F caractéristiques $\in [0,1]^{H,W,F}$ par un réseau convolutif. Ces caractéristiques sont ensuite sommées $\in [0,1]^F$ puis placées dans un réseau récurrent LSTM. Chaque état caché à la temporalité τ est modifié par une couche dense produisant un masque d'attention multipliant l'entrée de la prochaine temporalité. La temporalité finale est ensuite transformée par des couches denses formant ainsi les coordonnées de la boîte englobante de la zone d'intérêt. Ces coordonnées nourrissent un module de *patching* générant une image de taille réduite de la zone d'intérêt par la suite segmentée dans un réseau un encodeur-décodeur.



ED : réseau encodeur décodeur
BB : boîte englobante
$\bigoplus$: concaténation
$\bigotimes$: multiplication et somme

FIGURE 3.11 – Réseau de segmentation d'instances par attention visuelle.

Pour que le gradient puisse être propagé du masque jusqu'à l'entrée du LSTM, le module de *patching* doit être une opération dérivable. Pour cela, une méthode Gaussienne est employée [32]. Soient une image source I de taille (H, W) et un objet ayant une boîte englobante de taille (b_h, b_w) centrée sur (b_x, b_y), le patch gaussien P de taille (h, w) est obtenu par (respectivement pour F_y) :

$$\begin{aligned} \sigma_x &= \log(W) - \log(w) \\ \forall (i,j) &\in [[0, W]] \times [[0, w]], \\ \mu_x^j &= b_x + (b_w + 1)\left(\frac{2j+1}{2w} - \frac{1}{2}\right) \\ F_x^{i,j} &= \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(i-\mu_x^j)^2}{2\sigma_x^2}} \\ P &= F_y^T I F_x \end{aligned} \tag{3.6}$$

Cette opération est quasi-inversible $I \approx F_y P F_x^T$ puisque le patch P ne contient pas l'arrière-plan original de I. Une fois le masque de l'instance obtenu, ce dernier est ajouté à l'image source de la première étape afin de tenir compte des zones déjà traitées (Fig. 3.11) dans le module d'attention visuelle.

Performances

Ce réseau est particulièrement complexe à implémenter sur le *framework* Theano Lasagne. Le module d'attention visuelle doit proposer des zones de segmentation selon les segmentations précédentes afin de limiter les doublons. La segmentation d'un patch nécessite l'extraction correcte d'une zone d'attention visuelle. Ces deux modules sont donc couplés et ne peuvent pas être appris en même temps au début de l'entraînement. Ainsi, le module d'attention est dans un premier temps appris en utilisant la vérité de segmentation. Le réseau encodeur-décodeur est ensuite appris en utilisant les coordonnées vérités des boîtes englobantes des instances. Enfin, les deux réseaux sont couplés en injectant l'inférence de l'attention visuelle dans l'encodeur-décodeur et l'inférence des masques de segmentation dans le module d'attention visuelle.

Cependant, le réseau récurrent chargé de proposer des boîtes englobantes diverge à un certain point (Fig. 3.12). La boîte d'une instance est dans un premier temps localisée au voisinage de l'instance puis, la fonction de coût tend vers son maximum et la

boîte est déplacée aux extrémités de l'image. Malgré les différents tests conduits et la variation des hyperparamètres du réseau, cette divergence se produit systématiquement. Il est possible que le *framework* Theano Lasagne soit peu fonctionnel lorsque la sortie d'un LSTM est bouclée avec son entrée. Le comportement en mémoire est instable et le gradient semble se déconnecter après une centaine d'epochs.

epoch 200, coût 0.8094

epoch 220, coût 0.999

FIGURE 3.12 – Divergence du coût du réseau de segmentation d'instances avec attention visuelle (boîtes englobantes montrées ici à droite) entre l'epoch 200 et 220.

3.3.4 Algorithme *ad-hoc*

L'emploi de réseaux de neurones permet d'obtenir en un temps court les masques binaires des instances. Cependant, les différentes architectures proposées ne fournissent pas les résultats attendus y compris dans des scènes peu denses. L'utilisation d'un algorithme spécifique aux cartes angulaires peut permettre de contrer ce défaut en dépit d'un temps de calcul plus élevé. Les pièces étant coloriées depuis leur centroïde, le motif de coloriage au centre est spécifique à une instance. Un algorithme de détection de ce motif permettrait par la suite de suivre les échantillons angulaires afin de construire le masque binaire de segmentation de chaque pièce.

Synthèse et conclusion du chapitre

Ce chapitre propose plusieurs techniques permettant de découper des pièces dans l'image d'une scène. Pour combler le manque d'images d'entraînement nécessaires aux réseaux de neurones, les données sont générées selon une simulation de collisions d'un nombre prédéterminé de pièces. En utilisant une grande variété d'éclairages et d'arrière-plans, un réseau encodeur-décodeur permet d'inférer correctement le premier plan de la scène et d'éliminer le fond du bac. Pour le jeu de données LINEMOD, l'utilisation d'images réelles est nécessaire puisque la scène est trop complexe pour être simulée. Cette segmentation sémantique peut également être réalisée par l'inférence de cartes angulaires permettant de colorier une pièce tous les $45°$ depuis son centroïde. Pour la segmentation d'instance résultante, différentes architectures sont proposées reposant sur l'utilisation d'un réseau récurrent convolutif (convLSTM). La fonction de coût est construite par attribution des instances prédites avec les instances vérités selon l'intersection sur l'union et un algorithme Hongrois. La prédiction des masques d'instances en sortie du réseau n'est cependant pas optimale car trop bruitée pour correctement calculer cette fonction de coût. En utilisant un module d'inhibition pour nettoyer le masque, la sortie est trop faiblement activée et le masque est également bruité. Une autre approche utilisant un module d'attention visuelle est alors introduite permettant de segmenter une pièce depuis une zone réduite de l'image. Cette architecture possède deux récurrences ainsi que deux réseaux couplés ce qui la rend difficile à implémenter. Les tests conduits sur le *framework* Theano Lasagne montrent une subite divergence de la fonction de coût durant l'entraînement sur des images simples. Malgré les différents coefficients d'apprentissage utilisés ainsi que les différentes initialisations des poids, ce réseau n'est malheureusement pas fonctionnel. Ce réseau se montre pertinent dans d'autres problèmes de segmentation d'instances dans la littérature et le choix du *framework* est primordial pour assurer son fonctionnement. Des librairies plus modulaires et évolutives sont à envisager pour des travaux futurs. Cependant, lorsque les pièces sont correctement espacées dans la scène (sur un convoyeur), la segmentation sémantique suffit à correctement délimiter les instances dans l'image.

CHAPITRE 4

ESTIMATION DE POSE

Introduction

La pose d'un objet dans un espace 3D est définie comme la combinaison d'une orientation (matrice $R \in SO(3)$ ou quaternion $q \in Sp(1)$) et d'une translation $\vec{t} = (t_x, t_y, t_z)$ par rapport à une origine. Cette transformation $(R, \vec{t}) \in SE(3)$ (Éq. 1.8) peut être appliquée à une caméra (en laissant l'objet fixe) ou un objet (en laissant la caméra fixe). Ces deux points de vue seront confondus puisqu'ils fournissent des orientations identiques à une inversion près $(R, \vec{t})^{-1}$ (Éq. 1.9). La méthode proposée dans ce chapitre pour l'estimation de la pose d'un objet dans une scène, repose sur l'utilisation de trois réseaux de neurones profonds.

Dans ce chapitre, l'objet est supposé correctement segmenté dans l'image source de la scène et représenté sous la forme d'un patch.

Après un rappel de l'état de l'art en estimation de pose, la première étape de la méthode proposée est détaillée. Cette étape consiste en un réseau de neurones encodeur-décodeur utilisant l'image segmentée de la pièce afin d'estimer la distance à la caméra de chaque pixel (Sec. 4.3). Puis un second réseau utilise cette profondeur avec les coordonnées normalisées des pixels dans l'image de la scène, afin d'inférer la rotation de la pièce sous la forme d'un quaternion unitaire (Sec. 4.4). En parallèle, un troisième réseau utilise les mêmes modalités d'entrée pour évaluer la translation du centroïde de l'objet sur Z. Enfin, en utilisant l'image de profondeur, l'orientation de l'objet et sa translation sur Z, un algorithme de recalage itératif est proposé pour à la fois obtenir les translations sur X et Y et affiner l'orientation et la translation sur Z (Sec. 4.5).

4.1 État de l'art

L'état de l'art en estimation de pose est principalement fondé sur des méthodes de mise en correspondance (*matching*) de données avec un modèle ainsi que sur des optimisations d'une fonction de coût. La mise en correspondance peut être basée sur des points particuliers de la scène ou bien des caractéristiques pertinemment choisies pouvant prendre la forme de descripteurs locaux ou bien de modèles. Leur choix étant primordial et souvent délicat pour des formes complexes, l'apprentissage automatique et en particulier les réseaux de neurones profonds ont permis de s'en affranchir et d'améliorer les performances en estimation de pose d'objets ces dernières années.

4.1.1 Points à points

Lorsque que des points 3D $\{P_W\}$ de la scène sont mis en correspondance avec leur projection dans le plan image 2D $\{P_C\}$, il est possible d'estimer la pose de la caméra. Estimer la pose de la caméra en utilisant n points de correspondance est un problème dit PnP (*Perspective-n-Points*). Les premiers travaux [82] font état d'un minimum de 6 correspondances *a priori*, pour identifier les termes de la matrice liant les points de la scène à leur projection homogène. Différents algorithmes itératifs basés sur une méthode de Newton sont également proposés pour l'identification de la matrice [70][105] à partir d'une pose initiale. Cette pose initiale peut cependant être précalculée en approximant la projection perspective par une projection orthographique pondérée [18], affinée par une méthode itérative nommée POSIT. Souffrant cependant d'un temps de calcul important, l'algorithme EPnP présenté dans [61] propose une solution en $O(n)$ du problème PnP pour au moins 4 points de correspondance. Le problème considère que chacun des n points de correspondance peuvent être exprimés comme la somme pondérée de 4 points virtuels devenant alors les inconnus du problème. Lorsque les correspondances ne sont pas connues a priori, il est possible de lier deux jeux de points par une transformation affine puis une matrice d'attribution, en utilisant une fonction *softmax* (assignation *softassign*) [27]. Plus tard, POSIT et *softassign* ont été combinées pour former *Soft*POSIT [17]. Ces correspondances *a priori* peuvent également être obtenues par une estimation grossière de la position de la caméra, notamment par raffinement d'hypothèses initialement exprimées par des modèles Gaussiens dans la méthode *blind*PnP [71].

4.1.2 Caractéristiques locales

Plutôt que d'employer des correspondances entre des points 3D et 2D pour l'estimation de pose, il est possible d'utiliser des caractéristiques locales. Ces caractéristiques peuvent être issues d'un ensemble de points d'intérêts, décrivant un objet ou une forme indépendamment de certaines transformations (rotations, déformations, échelles...). Elles peuvent être automatiquement extraites du modèle CAO d'un objet dans un cadre 2D [11] ou 3D par stéréoscopie [74]. D'autres méthodes peuvent également utiliser des indices visuels de la pièce dans l'image (ellipses) [79] pour traiter des pièces industrielles. Néanmoins ces pièces sont contraintes à posséder des caractéristiques singulières en plus d'être déposées sur une surface plane pour limiter les variations et déformations selon l'axe vertical. Il est donc plus judicieux d'employer une gamme de descripteurs couvrant une large plage de points de vue. Issus de l'étude des différences de Gaussiennes, les descripteurs locaux SIFT [69] sont invariants en échelle et luminosité. En utilisant leurs localisations dans l'image, il est possible de trouver la position de la caméra associée par une variante de la recherche du plus proche voisin dans un arbre k-d [30]. Cependant, la méthode souffre d'un délai d'extraction des SIFT important lorsque les scènes sont complexes. Les descripteurs SURF [8] fondés sur les réponses d'ondelettes de Haar de régions de l'image, sont conçus pour offrir de meilleures performances en temps de calcul, tout en respectant les mêmes invariances que les SIFT malgré leur nombre en général inférieur [7]. En revanche, ils apparaissent moins robustes aux rotations et déformations que les SIFT et offrent donc une qualité de reconnaissance de pose potentiellement inférieure [97]. Composés d'une part de descripteurs FAST [84] et d'autre part, de descripteurs BRIEF [14], les descripteurs ORB [85] sont plus rapidement extraits que les SURF. Cependant ils ne proposent pas non plus d'améliorations significatives de la reconnaissance de pose [49]. D'autres descripteurs intéressants et plus récents peuvent être cités tels que FREAK [1] ou BRISK [62]. Globalement, les descripteurs locaux sont peu adaptés aux pièces industrielles puisqu'ils sont fortement dépendants des textures des objets.

4.1.3 Modèles d'objets

Un modèle d'objet est défini comme une partie d'image ou une représentation générique d'un objet servant de modèle de comparaison. Son utilisation permet le traitement de caractéristiques complexes que les descripteurs locaux ne peuvent exprimer

et est donc souvent adapté aux pièces peu texturées. La comparaison d'une entrée avec un modèle est dans un premier temps abordée à travers une distance de chanfrein [6] ou par l'utilisation de cartes de distances [22]. Les modèles peuvent également être appris au préalable par un classifieur de Ferns [75] et être utilisés dans une carte de distance [44]. L'un des défauts majeurs des cartes de distances est l'obligation d'extraction de contours au préalable. Cette étape reste sensible aux fortes variations de luminosité, au bruit et au flou. Face à ce constat, les directions principales discrétisées du gradient de l'image sont extraites pour être mises en correspondances avec celles d'un modèle. La pose d'un objet est ensuite estimée par minimisation d'une fonction d'énergie [39] (méthode LINE). Considérant des régions de l'image selon une grille, la fonction d'énergie est adaptée pour être robuste aux petites déformations et translations. Une technique similaire est employée dans [72]. La mise en correspondance avec un patron reste cependant sensible aux scènes complexes dotées d'arrière-plans chargés. Bien qu'elle puisse traiter des pièces industrielles, son utilisation dans un environnement hostile conduit à envisager une dimension supplémentaire de profondeur pour mieux différencier les données. L'émergence des caméras RGB-D à faible coût a permis de tirer profit de modèles multimodaux. Basée sur LINE, LINEMOD [41] complète les modèles avec les normales aux surfaces du canal de profondeur. Cette technique est ensuite améliorée en tenant compte des modèles 3D des objets [40]. Pour rendre la méthode robuste aux occlusions, elle peut être intégrée aux forêts de Hough [96]. Une autre technique introduite dans [43] met en correspondance une image avec des patrons générés par différents points de vue d'un objet. Ces algorithmes demandent de lourds temps de traitement lorsque la pose doit être précisément estimée (nombre important de patrons par objet). De plus, ces patrons doivent pour la majorité être fabriqués à la main et leurs caractéristiques extraites par des algorithmes complexes.

4.1.4 Réseaux de neurones

Les réseaux de neurones à convolutions (CNN) peuvent être utilisés afin d'extraire des caractéristiques dépendantes de l'orientation d'une pièce dans l'image. Par la suite, avec une méthode des k plus proches voisins, il est possible de classifier la pose de l'objet selon ces caractéristiques [101]. S'appuyant sur [40], l'apprentissage est également réalisé sur des données synthétiques issues de différentes poses de caméra, réparties uniformément sur une sphère. La technique utilise des triplets formés d'une

paire de poses semblables et dissimilaires mais également d'un terme forçant la similarité de caractéristiques entre deux poses semblables. Cette similarité peut aussi être forcée par une architecture de réseaux siamoise [19]. Employant un CNN travaillant sur des images RGB-D, une nouvelle fonction de perte permet pour deux poses proches, de forcer la similarité entre la distance Euclidienne sur les caractéristiques et celle sur les poses estimées. Cette technique est également employée pour la gestion de fortes occlusions. En utilisant un CNN pré-entraîné pour la segmentation, la pose d'un objet peut également être classifiée par des séparateurs à vaste marge (SVM) en convertissant le canal de profondeur de l'image en caractéristiques RGB au préalable [88]. La correspondance entre une image RGB et l'orientation d'un objet reste néanmoins complexe. En construisant un réseau sur l'architecture *GoogLeNet* [95], la position de la caméra peut être obtenue dans des scènes extérieures [51]. D'autres travaux adaptent la méthode SSD [65] pour obtenir les orientations d'un objet puis les classifier [50]. Cependant, l'utilisation d'un seul réseau (*end-to-end*) pour l'estimation de pose n'est pas toujours la meilleure stratégie. En employant plusieurs réseaux permettant de travailler sur des régions d'intérêts, la pose de plusieurs objets dans l'image peut être obtenue en rétro-projetant les coordonnées des pixels dans la scène [102]. Une autre stratégie baptisée BB8 consiste à rétro-projeter la boîte englobante 3D de l'objet après une phase de segmentation [78]. La méthode *iPose* [46] infère les coordonnées des pixels de l'objet dans l'image par un réseau de neurones encodeur-décodeur afin d'estimer l'orientation de l'objet par un algorithme d'hypothèses RANSAC. Une méthode similaire nommée *DeepIM* [63] atteint aujourd'hui l'état de l'art en estimation de pose sur LINEMOD. Cependant, les images d'entraînement sont générées en se basant sur les distributions et luminosités du jeu d'images réelles, ce qui ne permet pas d'utiliser la technique sur des nouvelles prises de vue.

4.1.5 Positionnement de nos travaux

La méthode d'estimation de pose de nos travaux de thèse repose sur l'utilisation de données synthétiques aux luminosités multiples [42]. Plutôt que d'utiliser une image RGB-D [19][88], la profondeur est inférée par un réseau de neurones encodeur-décodeur [47]. Les orientations et translations sont obtenues par deux autres réseaux [63] puis raffinées par un algorithme ICP nourrit par la rétro-projection des coordonnées des pixels de l'objet dans l'image [46].

4.2 Modalités des réseaux

L'estimation de la pose d'un objet contenu dans une image repose sur trois réseaux de neurones profonds pouvant chacun opérer sur trois modalités différentes (Fig. 4.1). La première modalité est le patch RGB de la pièce précédemment segmentée noté P_I. Les pixels de la pièce sont dans l'intervalle $[0, 1]^3$ tandis que l'arrière-plan est placé à -1. La seconde représente les coordonnées normalisées des pixels du patch dans l'image de la scène, présentées dans l'équation 2.5. Pour simplifier les notations dans ce chapitre, ces coordonnées initialement notées $\bar{P_C}$, seront à présent décrites par P_C. Enfin, la troisième modalité est la carte de profondeur du patch notée P_D, contenant la distance à la caméra de chaque pixel. Ce sont les valeurs $\bar{P_{W_z}} \in [-1, 1]$ présentées dans l'équation 2.6, transposées dans l'intervalle $[0, 1]$ (0 étant la profondeur maximale z_{far} et 1 la minimale z_{near}) telles que $P_D = \frac{1}{2}(-\bar{P_{W_z}} + 1)$.

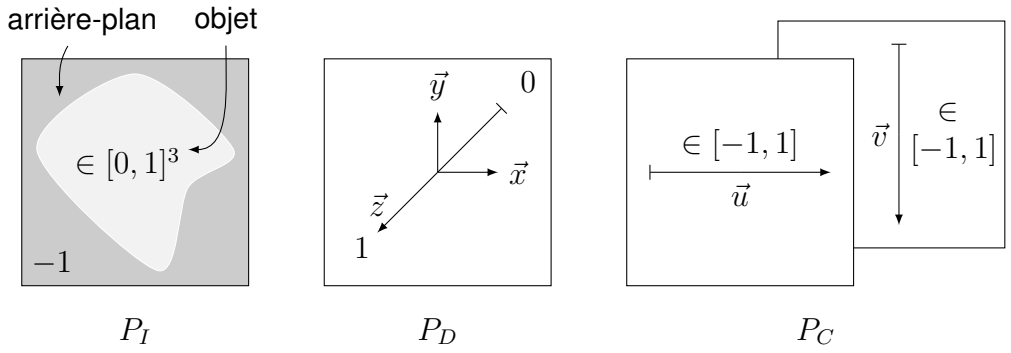


FIGURE 4.1 – Les différentes modalités des réseaux de neurones. De gauche à droite : le patch RGB P_I, la profondeur P_D et les coordonnées normalisées des pixels P_C.

4.3 Profondeur locale d'une pièce

Les réseaux d'estimation de translation sur Z et d'orientation sont nourris en partie par le patch de profondeur de l'objet. La profondeur locale $P_D \in [0, 1]$ est obtenue par un réseau de neurones encodeur-décodeur inférant la distance à la caméra de chaque pixel de façon probabiliste. Des tests sont conduits sur des images réelles de LINEMOD ainsi que de MULTITUDE après un entraînement réalisé sur des données uniquement synthétiques.

4.3.1 Architecture

À partir du patch RGB de l'objet P_I, le réseau de profondeur infère le patch de profondeur P_D. Ce réseau produit une image de la même taille que l'entrée tout en réalisant un passage de la 2D vers la 3D. Pour réaliser une telle transformation, un réseau encodeur-décodeur est utilisé. Ce réseau (Fig. 4.2) diffère légèrement de celui employé pour la segmentation (Fig. 3.1) puisqu'il comporte des couches entièrement connectées entre la phase d'encodage et de décodage. Ces calculs supplémentaires dans l'espace latent assurent la bonne compréhension de la géométrie de la pièce (Table. 4.1).

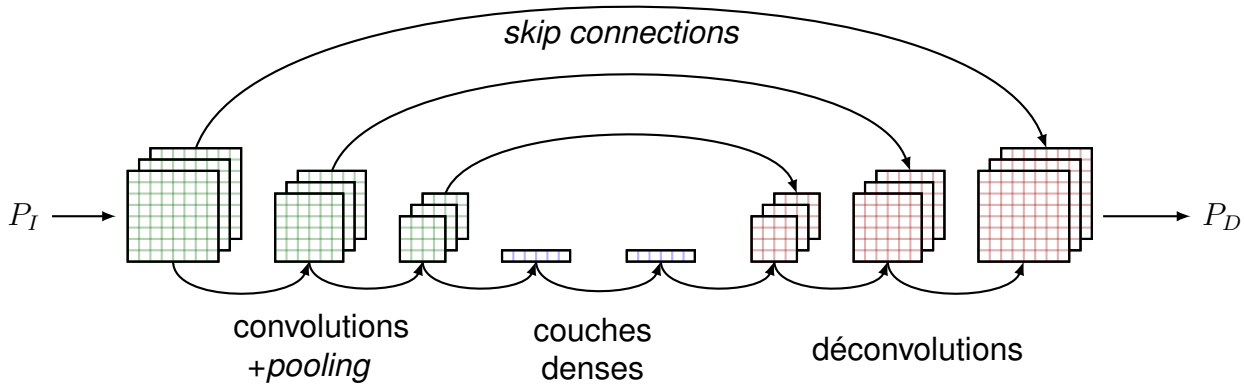


FIGURE 4.2 – Réseau encodeur-décodeur pour l'estimation de profondeur.

TABLE 4.1 – Détails des couches du réseau d'estimation de profondeur.

Encodage		Espace latent		Décodage	
couche	taille	couche	taille	couche	taille
conv$_{11}$	$3, 32$	dense$_1$	4096	**deconv**$_{11}$	$3, 256, 2$
conv$_{12}$	$3, 64$	dense$_2$	4096	skip$_1$	conv$_{53}$+deconv$_{11}$
pooling$_1$	2	redim.	$\rightarrow 4 \times 4 \times 256$	deconv$_{12}$	$3, 128, 1$
conv$_{21}$	$3, 64$			**deconv**$_{21}$	$3, 128, 2$
conv$_{22}$	$3, 96$			skip$_2$	conv$_{47}$+deconv$_{21}$
pooling$_2$	2			deconv$_{22}$	$3, 128, 1$
conv$_{31}$	$3, 96$			**deconv**$_{31}$	$3, 96, 2$
conv$_{32}$	$3, 128$			skip$_3$	conv$_{31}$+deconv$_{31}$
pooling$_3$	2			deconv$_{32}$	$3, 96, 1$
conv$_{41}$	$3, 128$			**deconv**$_{41}$	$3, 64, 2$
conv$_{42}$	$3, 128$			deconv$_{42}$	$3, 64, 1$
$\downarrow$				**deconv**$_{51}$	$3, 32, 2$
conv$_{47}$	$3, 128$			deconv$_{52}$	$3, 1, 1$
conv$_{48}$	$3, 256$			skip$_4$	entrée+deconv$_{52}$
pooling$_4$	2			deconv$_{61}$	$3, 2, 1$
conv$_{51}$	$3, 256$				
conv$_{52}$	$3, 256$				
conv$_{53}$	$3, 256$				
conv$_{54}$	$3, 512$				
pooling$_5$	2				

Détails :

conv	Couche de convolution
deconv	Couche de déconvolution
dense	Couche entièrement connectées
redim	Remodelage du tenseur
skip	Saut de connexion
pooling	Augmentation de la taille de l'image
taille	Taille des filtres, nombre de filtres, pas

Le réseau réalise une régression par pixel de la profondeur normalisée $\bar{P_{W_z}}$. En transformant les données de $[-1, 1]$ vers $[0, 1]$, une non-linéarité en sigmoïde peut être placée en sortie pour être utilisée dans une fonction de coût. Cependant, si les régressions usuelles utilisent une fonction de coût basée sur une distance L_2 ou une fonction de Hubert entre la prédiction et la vérité, l'inférence apparaît bruitée dans ces deux cas et la rétro-projection Π^{-1} n'est pas fidèle. Une autre approche consiste à prédire

l'image de profondeur P_D ainsi que son inverse $1 - P_D$ en employant une non-linéarité *softmax* aux deux canaux de l'image de sortie. La somme d'un pixel (u, v) du premier canal avec celui du deuxième canal valant 1, l'entraînement peut être réalisé à l'aide d'une entropie croisée $\mathcal{L}_D$ avec la vérité terrain $\hat{P_D}$ pour $i = 1, ..., N$ exemples :

$$\mathcal{L}_D = \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{u,v} \left(\hat{P_{Di}} \log \left(P_{Di} \right) \right) \right) \tag{4.1}$$

La fonction $\mathcal{L}_D$ permet naturellement un débruitage de la sortie P_D et donc un lissage du nuage de points par Π^{-1}. Ceci permet d'éliminer les valeurs aberrantes et par la suite, d'optimiser le raffinement de la pose par l'algorithme de recalage.

4.3.2 Performances

Le réseau de profondeur utilise une descente de gradient *Adam* [52] et tous les paramètres sont initialisés selon une distribution uniforme de Glorot [26]. Le réseau est implémenté en Theano Lasagne et entraîné durant 1000 epochs. Les poids donnant la plus faible erreur sur un jeu de validation composé d'images virtuelles aléatoires sont conservés. La MSE est utilisée en tant que métrique sur les pixels des images de profondeurs compris entre 0 et 1.

LINEMOD

Le jeu de données LINEMOD dispose d'une vérité de profondeur obtenue par une caméra de type Kinect. Cependant, les données sont peu précises (discrétisation de la distance sur Z grossière, nombreuses valeurs aberrantes) et souvent incomplètes (surfaces non détectées par le capteur). Le réseau de profondeur est alors entraîné sur des images synthétiques issues d'OpenGL. À chaque epoch, entre 1000 et 2000 images sont générées, selon une sphère de Fibonacci. Lors des tests, 50% du jeu de données est utilisé. Les performances de l'estimation de profondeur sont présentées dans le tableau 4.2.

TABLE 4.2 – Erreur en inférence de profondeur sur LINEMOD.

Métrique	Objet			
	APE	**CAN**	**CAT**	**DUCK**
MSE	0.0011	0.0013	0.0024	0.0012

La profondeur inférée pour LINEMOD ne présente que peu d'exemples aberrants. Des difficultés sont néanmoins présentes sur les textures de objets (yeux du chat et du canard) et la géométrie de l'arrosoir (trous au niveau de la pomme). Ces spécificités sont néanmoins importantes pour le réseau puisqu'elles constituent des caractéristiques fortement discriminantes. Les modèles CAO des objets sont donc modifiés en conséquence : l'absence de géométrie à ces endroits correspond ainsi aux textures de l'objet dans l'image réelle (Fig. 4.3).

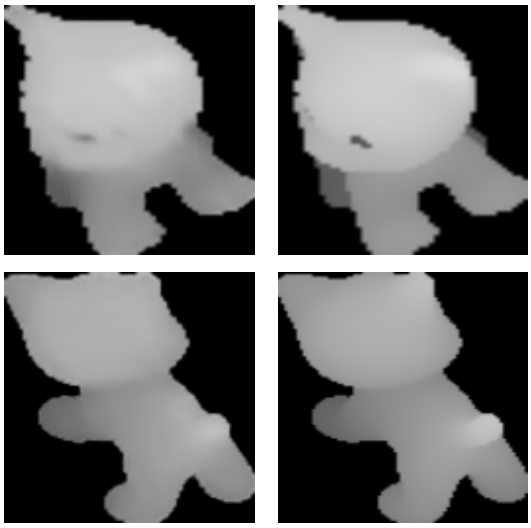


FIGURE 4.3 – Exemples d'inférences de profondeur locale sur le chat de LINEMOD (gauche) et vérité de profondeur (droite).

MULTITUDE

Pour MULTITUDE, les images de profondeur vérités sont obtenues par OpenGL selon des poses issues d'un découpage de la 3-sphère $SO(3)$. À chaque epoch, entre 1000 et 2000 images sont générées avec des paramètres de scène aléatoires. Les performances de l'estimation de profondeur sont présentées dans le tableau 4.3.

TABLE 4.3 – Erreur en inférence de profondeur sur MULTITUDE.

Métrique	Performances
MSE	0.015

La profondeur inférée est visuellement convaincante et la MSE entre prédiction et vérité est faible. Les images d'inférences montrent également que le réseau comprend la géométrie de la pièce et notamment le trou de l'embouchure (Fig. 4.4). Les surfaces planes apparaissent légèrement bruitées mais les angles formés par les saillances sont respectés assurant ainsi un bon fonctionnement du futur algorithme de recalage.

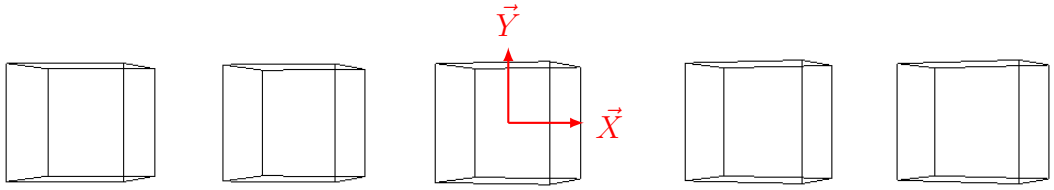
FIGURE 4.4 – Exemples d'inférence de profondeur sur des images réelles MULTITUDE.

4.4 Orientation et translation sur Z

Une fois la profondeur locale de l'objet obtenue, elle est utilisée afin d'inférer par l'intermédiaire de deux réseaux convolutifs, l'orientation de l'objet et sa translation sur Z. Cependant, l'image de profondeur est un patch ne comportant aucune information sur la taille relative de l'objet ni sur sa position dans le plan de l'image source. Les estimations de translation et d'orientation étant dépendantes de ces informations, la modalité d'entrée du réseau est complétée avec les coordonnées normalisées du patch dans l'image originale (notées P_c dans la fig. 4.1). Le réseau d'orientation produisant une orientation sous la forme d'un quaternion unitaire, une fonction de coût issue d'une métrique de $SO(3)$ est employée lors de l'apprentissage. Les performances sont étudiées sur les jeux de données LINEMOD et MULTITUDE.

4.4.1 Équivalences en projection sur $SO(3)$

Lorsqu'une pièce est projetée dans le plan caméra selon une pose $p_1 \in SE(3)$, son contexte local dans l'image (patch) peut être très proche de celui d'une autre pièce vue selon $p_2 \in SE(3)$ (Fig. 4.5).



(θ, t_x) : $(0, -6.5)$ $(-9°, -3.1)$ $(-18°, 0)$ $(-27°, 3.5)$ $(-35°, 6.9)$

FIGURE 4.5 – Équivalences locales des projections d'un cube unitaire pour une rotation de θ sur $\vec{Y}$ et une translation de t_x sur $\vec{X}$

Ainsi, le patch local de la pièce ne permet pas d'estimer correctement son orientation puisque la translation dans la scène n'est pas connue. Pour palier à ce défaut, les coordonnées locales des pixels de la pièce dans l'image originale, P_C (Éq. 4.2), sont fournies au réseau d'orientation à l'aide de la boîte englobante de taille (b_w, b_h) centrée sur (b_x, b_y) et d'un *patching* Gaussien (Éq. 3.6). Puisque le patch de la pièce est local, l'aire de l'objet dans l'image originale est perdue. Ainsi, cette modalité est également

essentielle pour estimer la translation sur $\vec{Z}$, puisque la taille de l'objet est révélatrice de son éloignement à la caméra.

$$
\begin{aligned}
P_{C_x} &= \frac{2}{w}\begin{pmatrix} \vdots & & \vdots \\ b_x - \frac{b_w}{2} & \dots & b_x + \frac{b_w}{2} \\ \vdots & & \vdots \end{pmatrix} - 1 \\
P_{C_y} &= \frac{2}{h}\begin{pmatrix} \dots & b_y - \frac{b_h}{2} & \dots \\ & \vdots & \\ \dots & b_y + \frac{b_h}{2} & \dots \end{pmatrix} - 1 \\
P_C &= F_y^T \left[P_{C_x}, P_{C_y}\right] F_x
\end{aligned}
\tag{4.2}
$$

4.4.2 Architecture des réseaux

Les réseaux d'orientation et d'estimation de translation sur $\vec{Z}$ possèdent des architectures similaires composées de convolutions puis de couches denses (Fig. 4.6). Ces deux réseaux sont nourris avec l'image de profondeur P_D et les coordonnées des pixels P_C afin de former une image de résolution $128 \times 128 \times 3$. Les noyaux des filtres sont de tailles décroissantes mais leur nombre augmente durant le passage de l'image dans le réseau. Ceci permet d'isoler progressivement des caractéristiques de plus en plus détaillées. La couche dense$_3$ (Fig. 4.6) est une couche de caractéristiques permettant de réaliser un compromis entre les performances du réseau et sa taille. À la lumière de [19], cette couche est dotée de 64 neurones donnant les meilleurs résultats en estimation de pose.

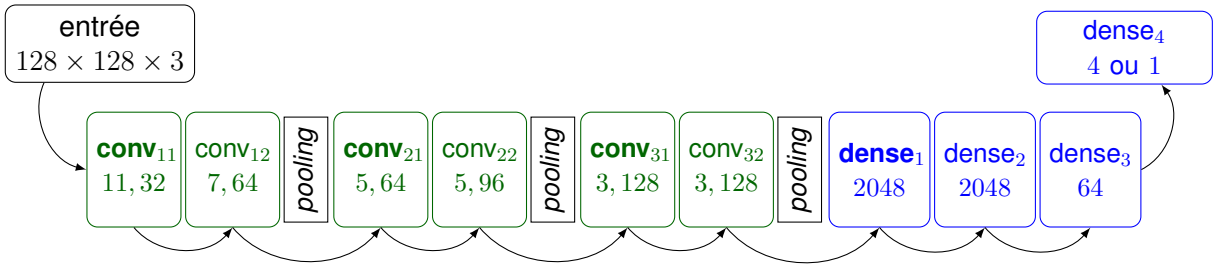


FIGURE 4.6 – Architecture du réseau d'orientation (4 neurones en sortie) et de translation sur $\vec{Z}$ (un neurone en sortie). Les couches en caractères gras possèdent une normalisation par batch.

Tout comme le réseau de profondeur, ces deux réseaux emploient des fonctions d'activations *ReLU*. La dernière couche du réseau de translation sur $\vec{Z}$ n'utilise en revanche pas de non-linéarité et régresse directement une valeur numérique en centimètres (ou millimètre selon les données) en employant une simple distance L_2 comme fonction de coût. Le réseau d'orientation prédit un quaternion q composé de 4 valeurs dans l'intervalle $[-1, 1]$. Ainsi, la non-linéarité de sa dernière couche est une fonction tangente hyperbolique *tanh*. Puisqu'un quaternion doit être unitaire pour représenter une rotation, la prédiction doit être normalisée avant d'être utilisée dans une fonction de coût issue d'une métrique de l'espace $Sp(1)$.

4.4.3 Apprentissage du réseau d'orientation

Deux quaternions unitaires q et $-q$ réalisent la même rotation (Annexe B). Une fonction de coût alors construite sur une distance L_2 peut diverger si la prédiction vaut l'opposé du quaternion vérité $\hat{q}$: $(\hat{q} - (-q))^2 = 4q^2 \neq 0$. Cette incertitude peut être levée en restreignant au préalable $Sp(1)$ aux quaternions disposant d'une première composante positive. Néanmoins, ces travaux de thèse proposent d'utiliser tout l'espace des quaternions à l'aide d'une métrique valide sur $Sp(1)$, afin que le réseau puisse comprendre au mieux la géométrie de la pièce. Pour quantifier l'écart entre deux rotations représentées par $R_1, R_2 \in SO(3)$, une première méthode repose sur l'utilisation de la norme de Frobénius mesurant l'écart à l'identité de la composition de R_1 et de R_2^T définissant ainsi une métrique sur $SO(3)$ [87] :

$$E_F = ||I - R_1 R_2^T||_F \tag{4.3}$$

L'ensemble des matrices de rotation $SO(3)$ est un groupe de Lie compact et isomorphe à la 3-sphère. Tous les éléments de son espace tangent $so(3)$ (matrices antisymétriques) peuvent être projetés sur la 3-sphère à l'aide d'une correspondance exponentielle (Annexe C). Il est alors simple de paramétrer par λ, la géodésique entre R_1 ($\lambda = 0$) et R_2 ($\lambda = 1$) sur la 3-sphère afin d'obtenir toutes les rotations intermédiaires :

$$R_{R_1 \to R_2}(\lambda) = R_1 e^{\lambda \log(R_1^T R_2)} \tag{4.4}$$

La distance géodésique E_G est obtenue en intégrant la métrique Riemannienne :

$$E_G = \int_0^1 \langle \dot{R(t)}, \dot{R(t)} \rangle^{1/2} dt = || \log(R_1^T R_2) ||_F \tag{4.5}$$

Les métriques E_F (Éq. 4.3) et E_G (Éq. 4.5) peuvent également s'interpréter avec des quaternions q_1 et q_2 :

$$\begin{aligned} E_F &= 2\sqrt{2(1 - |q_1.q_2|^2)} \\ E_G &= 2 \arccos(|q_1.q_2|) \end{aligned} \tag{4.6}$$

Ces deux métriques ont un comportement similaire en particulier lorsque les deux rotations sont proches (Fig. 4.7).

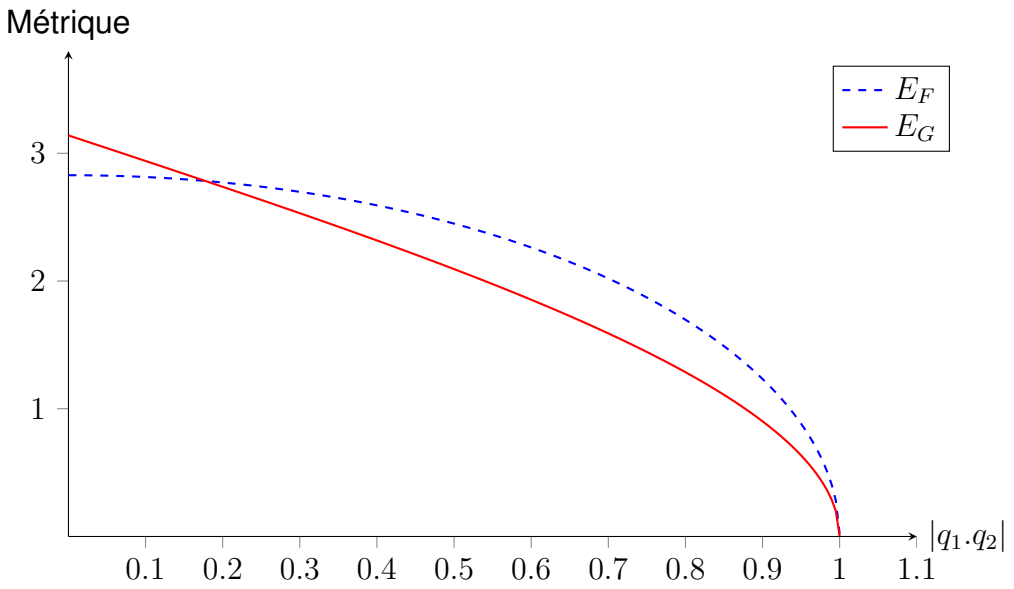


FIGURE 4.7 – Comportement des métriques E_F (Éq. 4.3) et E_G (Éq. 4.5) selon $|q_1.q_2|$.

L'emploi de la métrique E_G dans une fonction de coût neuronale n'est cependant pas idéal puisque la rétro-propagation du gradient de $\cos^{-1}$ est coûteuse. La métrique E_F possédant un simple produit scalaire est en revanche bien plus adaptée. L'apprentissage est donc réalisé avec une fonction de coût $\mathcal{L}_\mathcal{F}$ basée sur E_F, tandis que l'évaluation est réalisée par E_G entre le quaternion normalisé prédit q et la vérité $\hat{q}$.

$$\mathcal{L}_F = \sqrt{1 - (q.\hat{q})^2} \tag{4.7}$$

4.4.4 Performances

De façon similaire au réseau de profondeur, les réseaux d'orientation et de translation sur Z sont entraînés durant 1000 epochs. Les poids donnant les meilleurs résultats sur un jeu de données de validation composé de poses aléatoires générées en début d'apprentissage, sont gardés pour les tests. Le réseau de translation utilise une descente de gradient *Adam* tandis que le réseau d'orientation utilise des moments de Nesterov [73]. Tous les paramètres sont initialisés selon une distribution uniforme de Glorot [26]. **Chaque réseau est évalué en utilisant les inférences du réseau de profondeur ainsi, chaque erreur peut être vue comme une erreur cumulée dans le pipeline.** Les deux métriques utilisées sont la distance géodésique E_G (Éq. 4.6), la moyenne des erreurs sur les angles d'Euler en convention $E-Z-Y-Z$ notée E_E et la valeur absolue de la différence des translations AD.

LINEMOD

Pour évaluer les réseaux sur LINEMOD, 50% du jeu de données est utilisé. À chaque epoch, entre 1000 et 2000 images sont générées, selon une sphère de Fibonacci. Les performances en estimation d'orientation et de translation sont présentées dans le tableau 4.4.

TABLE 4.4 – Erreurs cumulées des réseaux depuis les patchs segmentés P_I pour LINEMOD.

Tâche	Métrique	Objet			
		APE	**CAN**	**CAT**	**DUCK**
Orientation	E_E	$7.14°$	$9.54°$	$9.91°$	$11.54°$
	E_G	$8.33°$	$11.18°$	$10.26°$	$13.22°$
Translation	AD	1.96cm	2.90cm	1.67cm	2.65cm

L'inférence d'orientation offre en moyenne une erreur géodésique proche de $11°$. Le canard possède une géométrie moins complexe que les autres objets, il dispose donc naturellement de moins de caractéristiques géométriques ce qui diminue les performances du réseau. Les erreurs de translation sont proches de 2 cm pour tous les objets ce qui permet de correctement initialiser l'algorithme de recalage.

MULTITUDE

Pour MULTITUDE, les images vérités sont obtenues par OpenGL selon des poses obtenues par découpage de la 3-sphère $SO(3)$. À chaque epoch, entre 1000 et 2000 images sont générées. Les performances en estimation d'orientation et de translation sont présentées dans le tableau 4.5.

TABLE 4.5 – Erreurs cumulées des réseaux depuis les patchs segmentés P_I pour MULTITUDE.

Tâche	Métrique	Données	
		Virtuelles	**Réelles**
Orientation	E_E	$7.87°$	$15.22°$
	E_G	$8.45°$	$16.3°$
Translation	AD	3.1cm	4.21cm

Les résultats en estimation de pose pour MULTITUDE sont corrects sur des données virtuelles aléatoires. Pour le jeu de données réelles, l'erreur en translation est proche de 4cm et l'erreur géodésique proche de $16°$ offrant des résultats visuellement convaincants (Fig. 4.8) mais non suffisamment précis pour un dévracage automatisé. Pour raffiner la pose fournie par les réseaux, un algorithme de recalage est par la suite employé.

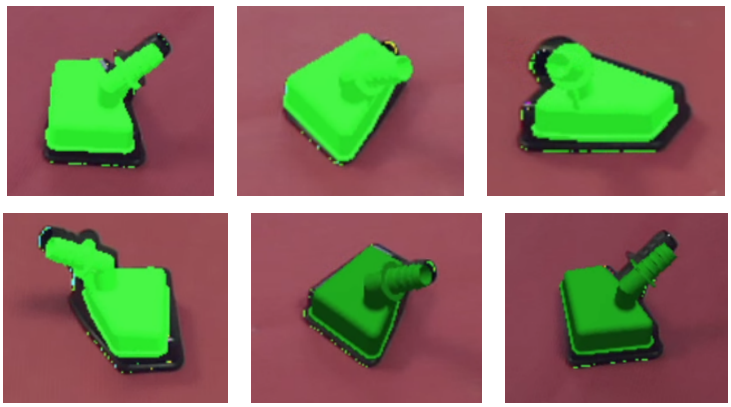


FIGURE 4.8 – Exemples d'inférences pour MULTITUDE (en vert la prédiction).

4.5 Rétro-projection et recalage

L'estimation de pose par réseaux de neurones reste grossière et ne convient pas à un dévracage automatisé. Puisque le canal de profondeur est disponible, une algorithme de recalage par nuage de points de type ICP est implémenté. En rétro-projetant la carte de profondeur à l'aide de la translation estimée sur Z, le modèle CAO de l'objet est initialement placé selon le quaternion prédit puis recalé dans le nuage afin d'optimiser la pose.

4.5.1 Initialisation

En utilisant la translation t_z et les coordonnées P_C, le nuage de point de la scène représentant la surface visible discrétisée de l'objet, v_s, est obtenu par rétro-projection de l'image de profondeur P_D via Π^{-1} (Éq. 2.9) tel que $v_s = \Pi^{-1}(P_{C_x}, P_{C_y}, P_D, t_z, r_{\text{obj}})$ (Fig. 4.9).

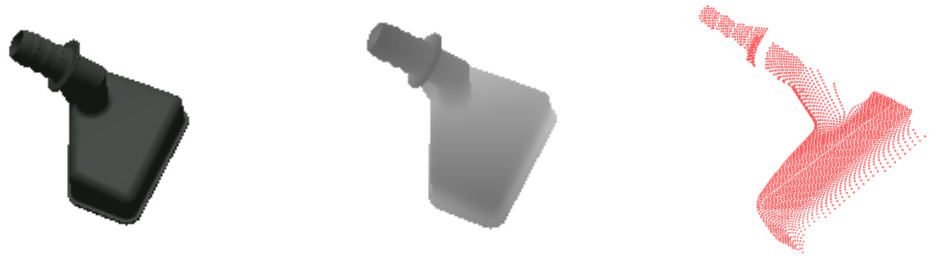


FIGURE 4.9 – Pièce en RGB (gauche) avec sa profondeur P_D (centre) et sa rétro-projection par Π^{-1} (droite)

En plaçant le modèle CAO de la pièce v_m selon le centroïde de v_s et en l'orientant selon q, un algorithme de recalage est employé pour chercher à aligner le nuage de point obtenu v_m avec la surface v_s (Fig. 4.10).

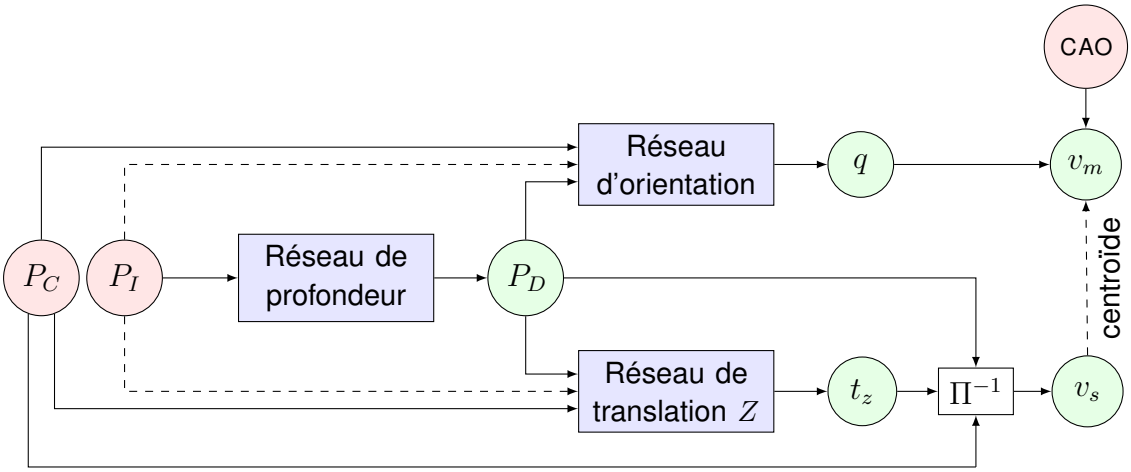


FIGURE 4.10 – Obtention des nuages de points v_s et v_m à l'aide de l'image P_I, des coordonnées P_C et des trois réseaux de neurones fournissant la profondeur P_D, l'orientation q et la translation t_z.

Puisque l'ICP établit des correspondances entre des jeux de points, le modèle CAO est au préalable échantillonné uniformément par un découpage récursif de la boîte englobante du modèle selon un *octree*, puis par projection du centre des boîtes sur ses surfaces triangulaires. La distance entre deux points du nuage surfacique v_s étant en moyenne $d = \sqrt{(t_z/f_X)^2 + (t_z/f_Y)^2}$, l'échantillonnage du modèle est réglé sur $d/2$ afin de réduire les temps de calcul liés à l'attribution des points entre v_m et v_s (garder une même densité entre le modèle et la scène n'est en général pas une bonne stratégie pour un algorithme ICP [23]). Le nuage v_s ne contient que la surface du modèle CAO visible à la caméra, ainsi, v_m est coupé à mi-hauteur sur l'axe Z. Ceci permet notamment de réaliser une simulation simpliste d'un Z-test OpenGL à condition que l'objet ne présente pas de détails complexes volumiques au sein de sa géométrie.

Le recalage repose ensuite sur un algorithme de la librairie OpenCV fonctionnant par sélection de correspondances entre v_m et v_s par *Picky* ICP [108]. L'évaluation de la distance entre les nuages est issue d'une métrique linéaire point-à-plan [68]. Cette métrique étant linéaire, il est possible de minimiser la distance entre les points des nuages mis en correspondances par une simple décomposition en valeurs singulières SVD.

4.5.2 Estimation des normales

Le calcul de la métrique point-à-plan nécessite cependant les normales en chaque point des nuages v_m et v_s. Si les normales du modèle CAO sont connues à priori, ce n'est pas le cas de la surface v_s. La normale en un point de v_s peut être obtenue en considérant la moyenne des normales aux triangles formés avec ses plus proches voisins. Plutôt que d'employer un arbre k-d pour rechercher les plus proches voisins de façon optimale, l'image de profondeur P_D est utilisée avec une 4-connectivité. Chaque normale est obtenue en moyennant les normales des triangles formés par le voisinage d'un pixel (Fig. 4.11).

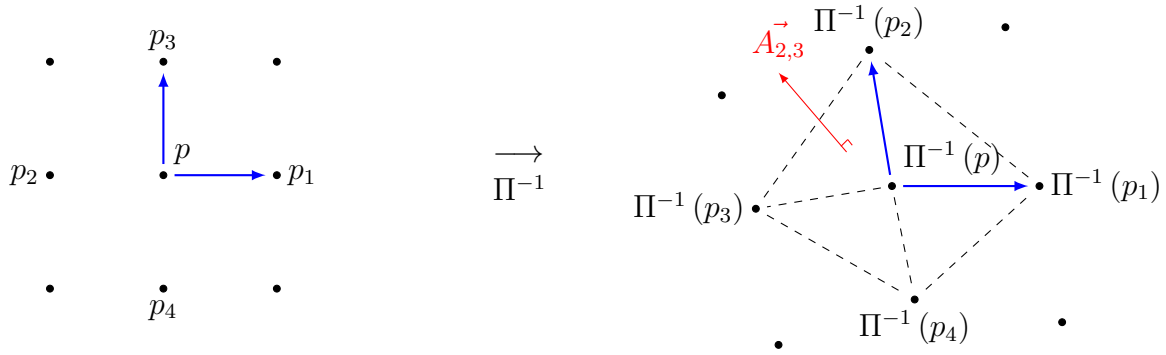


FIGURE 4.11 – Calcul des normales aux points de la surface rétro-projetée. Le voisinage d'un pixel p de la carte de profondeur (gauche) est rétro-projeté via Π^{-1} pour construire les normales aux triangles par produit vectoriel (droite).

Soient un pixel $p = (i, j)$, ses 4 voisins p_k, ainsi que leurs coordonnées dans la scène après rétro-projection notées par simplification $\Pi^{-1}(p)$, la normale au pixel $\vec{n}$, est obtenue par :

$$\begin{aligned} \vec{A_{k,l}} &= \left(\Pi^{-1}(p_k) - \Pi^{-1}(p)\right) \wedge \left(\Pi^{-1}(p_l) - \Pi^{-1}(p)\right) \\ \vec{n} &= \frac{1}{4}\left(\frac{\vec{A_{1,2}}}{||\vec{A_{1,2}}||} + \frac{\vec{A_{2,3}}}{||\vec{A_{2,3}}||} + \frac{\vec{A_{3,4}}}{||\vec{A_{3,4}}||} + \frac{\vec{A_{4,1}}}{||\vec{A_{4,1}}||}\right) \end{aligned} \tag{4.8}$$

Tout comme la rétro-projection Π^{-1} (Éq. 2.9), il est possible d'optimiser ce calcul sous forme tensorielle, en définissant le voisinage d'un pixel par des translations unitaires de l'image P_D ($\pm 1\vec{u}$, $\pm 1\vec{v}$). Le nuage et les normales sont alors obtenues en quelques millisecondes (Fig. 4.12).

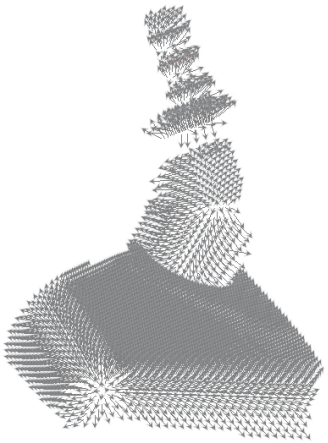


FIGURE 4.12 – Normales aux points du nuage obtenues par calcul tensoriel en utilisant la 4-connectivité de P_D.

4.5.3 Fonctionnement

L'algorithme ICP cherche à aligner deux représentations différentes de l'inférence des réseaux : l'une basée sur l'inférence de $(R, \vec{t}) \in SE(3)$ et l'autre sur la précision du nuage de point P_D. Ces deux représentations dépendent de la translation inférée, t_z, seule composante que l'algorithme de recalage ne peut donc optimiser. Cependant, si un premier recalage du nuage v_m est suffisamment proche de la vérité, le masque binaire issu de la projection OpenGL correspondante possède un contour similaire à celui de la segmentation d'instances P_M. La comparaison d'IOU donne donc un critère d'éloignement de la pièce à la caméra. La translation t_z offrant la meilleure IOU est gardée pour regénérer le nuage v_s à partir de P_D, et par la suite, replacer v_m. Une nouvelle passe d'ICP est ensuite effectuée pour obtenir la transformation $(R, \vec{t})$ finale (Fig. 4.13).

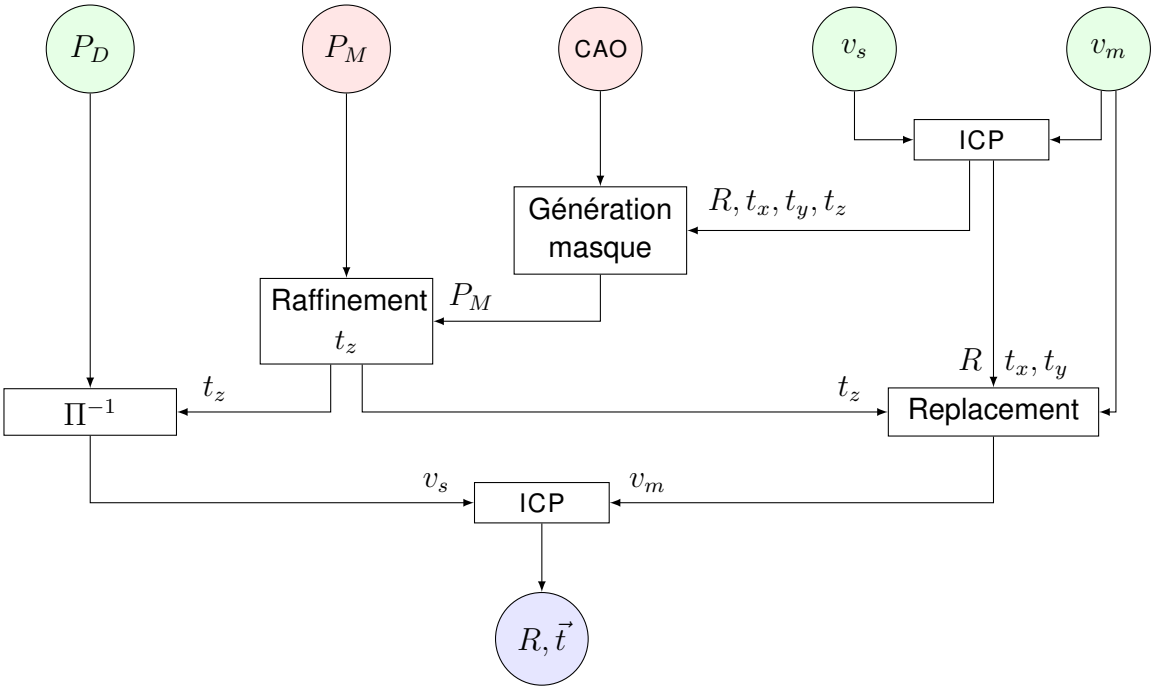


FIGURE 4.13 – Estimation de la translation (t_x, t_y) et raffinement de (R, t_z) par ICP et comparaison entre le masque regénéré et l'original P_M.

4.5.4 Performances

Trois métriques sont employées pour évaluer l'estimation de pose finale sur $i = \{1, ..., N\}$ exemples : ADD [40], *2D-proj* et 5cm5°.

ADD

La métrique ADD quantifie la proportion d'exemples, dont l'écart moyen entre les sommets du modèle CAO vérité $\hat{V}$ et prédit V, est inférieur à 10% du diamètre de l'objet.

$$\text{ADD} = \frac{1}{N}\sum_{i=1}^{N}\left[||\hat{V}_i - V_i||_2 < 0.1 \times \text{Diam}_{\text{objet}}\right] \tag{4.9}$$

2D-proj

La métrique *2D-proj* quantifie la proportion d'exemples, dont l'écart moyen entre les sommets du modèle CAO vérité $\hat{V}$ et prédit V, rétro-projetés selon la matrice d'homographie H, est inférieur à 5 pixels.

$$\text{2D-proj} = \frac{1}{N}\sum_{i=1}^{N}\left[||H.\hat{V}_i - H.V_i||_2 < 5\text{px}\right] \tag{4.10}$$

5cm5°

La métrique 5cm5° quantifie la proportion d'exemples dont l'écart entre la translation vérité $\hat{T}$ et prédite T est inférieur à 5 (centimètres ici), et dont l'écart angulaire moyen sur les angles d'Euler vérités $(\hat{\phi}, \hat{\theta}, \hat{\psi})$ et prédits (ϕ, θ, ψ) est inférieur à $5°$ (convention $E - Z - Y - Z$ ici).

$$5\text{cm}5° = \frac{1}{N}\sum_{i=1}^{N}\left[||(\hat{\phi}_i, \hat{\theta}_i, \hat{\psi}_i) - (\phi_i, \theta_i, \psi_i)||_2, ||\hat{T}_i - T_i||_2) < (5, 5)\right] \tag{4.11}$$

LINEMOD

Les résultats d'estimation de pose sur LINEMOD sont comparées avec diverses méthodes de l'état de l'art dans le tableau 4.6.

TABLE 4.6 – Comparaison des métriques avec les méthodes de l'état de l'art.

Objet	Métrique	Méthode				
		LINEMOD [41]	BB8 [78]	SSD-6D [50]	DeepIM [63]	**Proposée avec ICP**
APE	5cm5°	34.40	80.20	-	90.38	81.51
	2D-proj	85.20	96.60	-	98.38	93.81
	ADD	33.20	40.40	76.30	76.95	72.13
CAN	5cm5°	48.40	76.80	-	92.81	88.50
	2D-proj	70.80	91.20	-	99.70	68.74
	ADD	62.90	64.10	93.10	96.46	85.71
CAT	5cm5°	34.60	79.90	-	87.62	86.66
	2D-proj	84.20	98.80	-	98.70	93.00
	ADD	42.70	62.60	89.30	82.14	78.34
DUCK	5cm5°	22.00	53.20	-	85.16	86.56
	2D-proj	73.10	92.20	-	98.50	87.13
	ADD	30.20	44.30	80.00	77.65	63.43

Ces métriques montrent que malgré l'apprentissage des réseaux sur des images uniquement synthétiques, le recalage par IOU permet de compenser l'erreur introduite par le biais virtuel-réel et d'atteindre des performances au dessus de certaines techniques utilisant le jeu d'entraînement réel de LINEMOD. L'inférence de la profondeur est un atout pour cet algorithme montrant ainsi que les architectures *end-to-end* ne sont pas toujours la meilleure stratégie. Dans le cas du canard, la métrique 5cm5° dépasse l'état de l'art *DeepIM*. Pour l'arrosoir, la métrique *2D-proj* est faible ce qui peut s'expliquer par la anse de l'objet très importante dans une métrique de rétro-projection vers le plan 2D.

MULTITUDE

Les résultats d'estimation de pose sur MULTITUDE après la phase de recalage par ICP sont présentés dans les tableaux 4.7 et 4.8.

TABLE 4.7 – Erreurs cumulées des réseaux depuis les patchs segmentés P_I pour MULTITUDE avant et après le recalage par ICP.

Tâche	Métrique	Avant ICP	Après ICP
Orientation	E_G	$16.3°$	$5.26°$
Translation	AD	4.21cm	1.09cm

TABLE 4.8 – Métriques d'estimation de pose sur BREATHER.

Métrique	Performances
5cm5°	75.13
2D-proj	85.11
ADD	72.89

La précision après l'algorithme de recalage est d'environ 1cm pour la translation et de $5°$ géodésique pour l'orientation. Les métriques 5cm$5°$, *2d-proj* et ADD possèdent des valeurs similaires à LINEMOD (Table. 4.8). L'emploi d'un ICP est une nouvelle fois pertinent puisqu'il permet de compenser l'erreur géodésique initiale forte (Fig. 4.14).

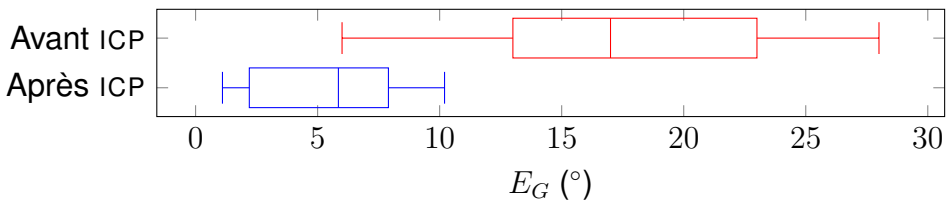


FIGURE 4.14 – Dispersion des erreurs géodésiques avant et après la phase de recalage par ICP.

Cependant, l'algorithme de recalage souffre en performances lorsque l'erreur géodésique est telle que le modèle CAO et le nuage rétro-projeté ont peu de points alignés dès l'initialisation. L'exemple de gauche de la figure 4.15 montre que l'embouchure de la pièce avant la phase d'ICP est mal positionnée. S'agissant d'une caractéristique importante pour le recalage, la pose finale n'est pas correcte. L'exemple de droite montre la pertinence du recalage de la translation t_z lorsque la pose est correctement estimée après une phase d'ICP.

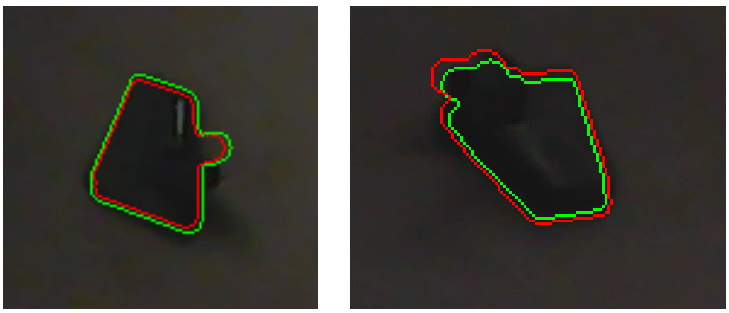


FIGURE 4.15 – Exemples d'inférence sur fond noir avant ICP (rouge) et après (vert) sur MULTITUDE.

Synthèse et conclusion du chapitre

Ce chapitre propose un pipeline d'estimation de pose à partir d'une image 2D d'une pièce segmentée dans une scène. Dans un premier temps, un réseau encodeur-décodeur possédant des couches denses dans son espace latent, permet d'inférer la carte de profondeur de la scène. Ce réseau est particulièrement robuste au biais virtuel-réel et permet d'obtenir des résultats fiables à partir d'un entraînement uniquement réalisé sur des images de synthèse. Cette carte est ensuite utilisée avec les coordonnées de l'objet dans la scène dans deux réseaux convolutifs, afin d'obtenir une première estimation de l'orientation de l'objet sous la forme d'un quaternion ainsi que sa translation sur l'axe Z. Le réseau d'orientation est construit avec une fonction de coût issue d'une métrique sur l'espace des quaternions unitaires $Sp(1)$ et permet d'inférer les quaternions q ou $-q$ représentant la même orientation. L'estimation de pose étant encore grossière, un module de recalage est ensuite employé. En utilisant la rétro-projection de la carte de profondeur avec l'estimation de translation sur Z, le modèle CAO de l'objet initialisé selon le quaternion prédit, est aligné dans le nuage de points afin d'affiner la pose. La translation sur Z est également raffinée par une mise en correspondance du masque binaire de la pièce dans l'image et celui obtenu par OpenGL à l'aide du modèle CAO recalé. Cette méthode offre des résultats proches de l'état de l'art en utilisant des métriques standards sur LINEMOD. Pour MULTITUDE, les résultats suffisent à une préhension absorbant les défauts de poses.

CHAPITRE 5

DÉPLOIEMENT DU MODULE DE VISION

Introduction

Ce chapitre propose différentes études et implémentations du module de vision pour l'estimation de pose en situation industrielle.

Dans un premier temps, une étude de la précision angulaire du jeu d'entraînement est proposée afin d'évaluer l'impact du nombre de poses choisies sur la 3-sphère, sur l'estimation de pose (Sec. 5.1). Puis, différents éléments permettant de choisir une caméra adaptée à la problématiques sont présentés. L'objectif est de sélectionner une caméra offrant peu d'effets de perspectives et permettant de limiter les effets de flous liés à la profondeur de champ. Afin de produire des images d'entraînement réalistes, un algorithme d'estimation de paramètres de scène est détaillé (Sec. 5.2). Son fonctionnement repose sur un raffinement d'intervalles de paramètres selon la minimisation d'une MSE entre des images générées et des images réelles d'un objet vu sous une même pose. Afin d'utiliser au mieux le module neuronal, la caméra doit au préalable être calibrée. Cette calibration concerne à la fois la focale de la lentille mais également ses coefficients de distorsions issus de modèles mathématiques. Le module neuronal devant être implémenté en usine à divers endroits, une architecture de déploiement reposant sur des communications réseaux entre un contrôleur et un serveur est présentée (Sec. 5.3). Elle permet notamment la gestion de plusieurs réseaux de neurones de façon centralisée. Enfin, le choix de la zone préhension de la pièce après son estimation de pose est introduite. Le calcul de la dépose est réalisé par un automate et permet de déposer la pièce sur un repère intermédiaire avant sa saisie précise pour son placement sur un module d'assemblage.

5.1 Précision de la pose

5.1.1 Incrément angulaire

Le réseau de neurones d'orientation est entraîné sur des images générées d'après des poses de $SO(3)$. Ces N orientations sont issues d'un tirage uniforme sur la 3-sphère (Éq. 2.13) à chaque epoch. Comme mentionné dans le chapitre sur la génération de données, le nombre de poses N agit comme une graine lors du tirage (Éq. 2.10, Éq. 2.13). La génération de N poses et de $N + \delta$ (avec au moins $\delta > 50$) poses offrent des orientations qui ne se recoupent qu'en certaines zones de l'espace des rotations (bien que quantitativement complexe à estimer) (Fig. 5.1). Ainsi, en choisissant N aléatoirement dans un intervalle, les images vues à chaque epoch sont pour la majorité différentes. Ceci permet d'obtenir un réseau de neurones réalisant une régression de pose plutôt qu'une classification parmi N orientations fixées.

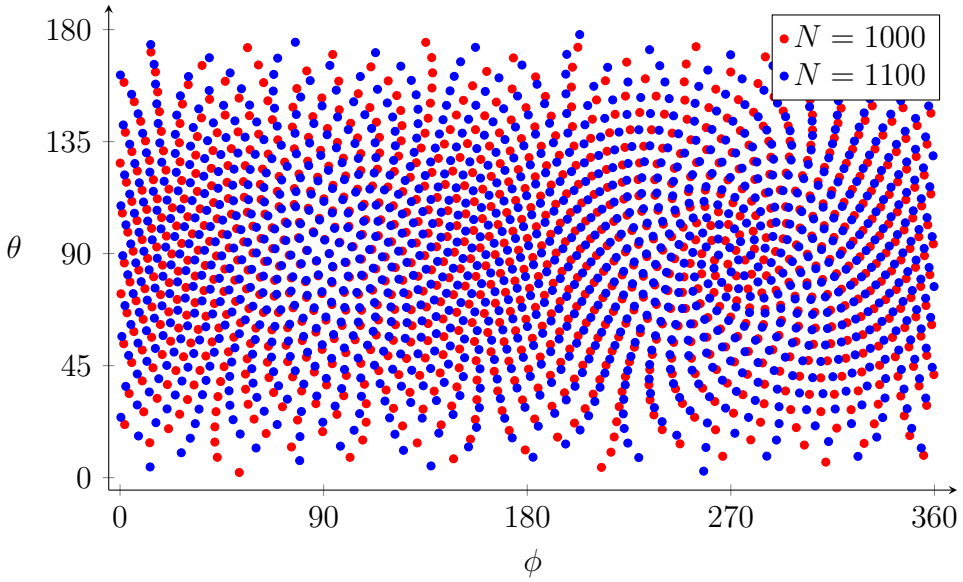


FIGURE 5.1 – Différences entre les angles ϕ et θ pour un tirage de Fibonacci de 1000 et 1100 points.

Durant l'entraînement, le coefficient d'apprentissage est diminué tous les 5 epochs (ou toutes les $5 \times N$ images). En fonction du nombre de tirages N, la dynamique d'apprentissage varie donc et la précision finale exprimée en terme d'erreur géodésique E_G n'est pas constante (Table. 5.1).

TABLE 5.1 – Précision de l'estimation de pose sur MULTITUDE en fonction du nombre de tirages sur la sphère N.

Génération	N	100-200	500-600	1K-1.5K	5K-6K	10K-11K
Orientation	E_G	$28.25°$	$23.71°$	$17°$	$16.51°$	$16.45°$

L'erreur géodésique E_G selon le nombre total d'images vues par le réseau durant l'apprentissage N_a montre qu'un nombre de tirages compris entre 1000 et 1500 par epoch offre des performances similaires à des tirages plus conséquents. Ainsi, la diminution du coefficient d'apprentissage toutes les 5000 images offre un comportement optimal pour le réseau (Fig. 5.2).

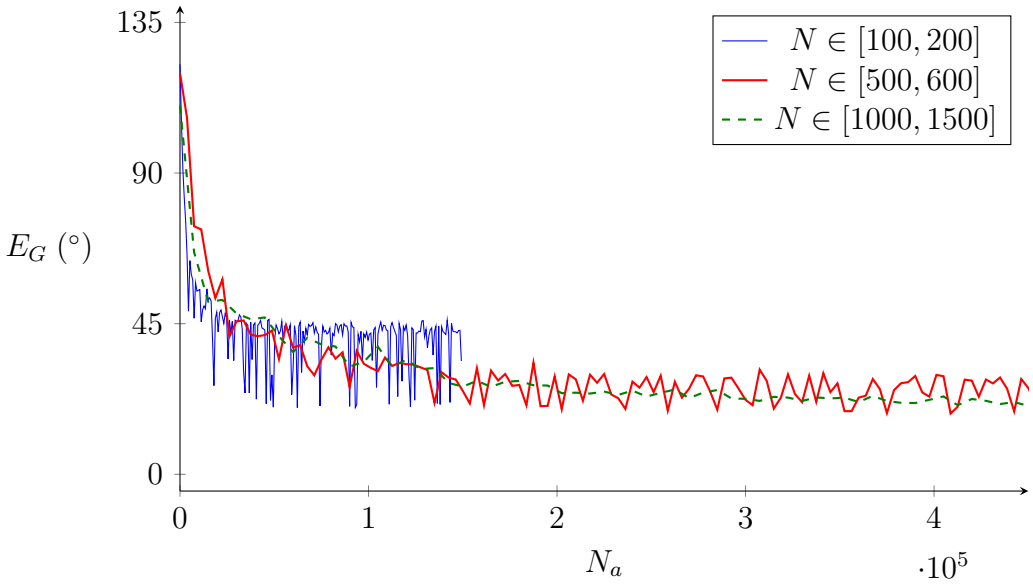


FIGURE 5.2 – Précision E_G sur des données de test réelles en fonction du nombre total d'images virtuelles vues lors de l'apprentissage N_a, avec N images par epoch.

5.1.2 Matériel

Effets de perspective

Les premières publications de thèse [58] présentaient une estimation de pose à partir d'images prises par une caméra télécentrique (objectif n'offrant aucun effet de perspective) (Fig. 5.3).

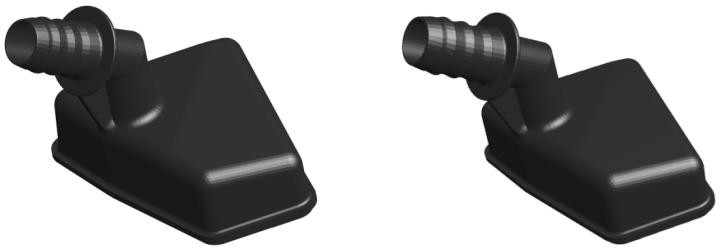


FIGURE 5.3 – Effet d'une caméra télécentrique (gauche) et entocentrique (droite).

Les résultats montrent que le réseau de neurones tend à apprendre avec plus de facilités, l'orientation de pièces n'offrant aucun effet de perspective tout en inférant correctement sur des images entocentriques (avec perspectives) virtuelles (Table. 5.2).

TABLE 5.2 – Erreur géodésique en fonction du type de caméra utilisé sur un jeu de test d'images virtuelles (apprentissage sur deux angles uniquement (ϕ, θ)).

		Apprentissage	
		Télécentrique	Entocentrique
Inférence	Télécentrique	$3.30°$	$7.22°$
	Entocentrique	$9.74°$	$9.80°$

En pratique, un objectif télécentrique couvrant tout un vrac à une distance d'au moins $1\text{m}50$ n'existe pas puisque la taille de l'objectif doit être du même ordre de grandeur que la scène à visualiser (réservé en général aux petites pièces pour du contrôle de forme ou de qualité). L'inférence doit donc être réalisée sur un objectif entocentrique avec une focale adaptée afin de limiter au maximum les effets de perspectives et de se rapprocher d'un comportement télécentrique.

Résolution de l'image

La pièce étudiée possède une boîte englobante d'environ $70 \times 70 \times 70$ mm^3. Pour un bac de taille $P_{W_y} \times P_{W_x} = 400 \times 600$ mm^2, ceci correspond à une présence d'environ 5 pièces en largeur et 8 pièces en longueur. Les réseaux de neurones utilisant des patchs de taille 128×128 px, la taille d'image idéale (cas dans lequel les patchs des pièces sont peu redimensionnés) du bac est alors de $I_h \times I_w = 640 \times 1024$ px. En plaçant la caméra à une distance d'environ $P_{W_z} = 1500$ mm du bac, la distance focale exprimée en pixel correspondante est selon l'équation 2.3 de :

$$\begin{aligned} f_X &= I_w \frac{P_{W_z}}{P_{W_x}} = 2560 \text{ px} \\ f_Y &= I_h \frac{P_{W_z}}{P_{W_y}} = 2400 \text{ px} \end{aligned} \tag{5.1}$$

Pour un capteur photosensible industriel standard de $2/3$" correspondant à une taille de $s_w \times s_h = 8.8 \times 6.6$ mm^2, la distance focale exprimée en millimètres est selon l'équation 2.2 :

$$f \approx \frac{s_w f_X}{I_w} = 22 \text{ mm} \tag{5.2}$$

Les calculs précédents sont conduits pour une pièce située à $P_{W_z} = 1500$ mm de la caméra. Il est important de tenir compte de la hauteur du bac de $h = 300$ mm afin d'étudier la différence de taille en pixels de pièces placées en haut et en bas du bac.

Soit z la profondeur du bac avec $z = 0$ pour le haut du bac et $z = h$ pour le fond. La taille d'une pièce de $P_{W_y} \times P_{W_x} = 70 \times 70$ mm^2, en pixels, est selon la profondeur z :

$$p_x(z) = \frac{f I_w P_{W_x}}{s_w (P_{W_z} + z)} \tag{5.3}$$

La différence de taille selon la profondeur entière du bac est simplement :

$$\Delta p_x = \frac{f I_w P_{W_x} h}{s_w P_{W_z} (P_{W_z} + h)} \tag{5.4}$$

Pour la caméra sélectionnée, on trouve $\Delta p_x \approx 20$ px ce qui représente une diminution de 15% de la taille d'une pièce en haut de bac.

Profondeur de champ

L'optique doit offrir une profondeur de champ suffisante pour obtenir une image nette sur la totalité de la hauteur du vrac. Pour un capteur $2/3$", la gamme de taille du cercle de confusion CdC est obtenue en ramenant l'acuité visuelle humaine ($\frac{1}{60^\circ}$) à la taille du capteur. Pour une distance de 250 mm, le plus petit détail distinguable est d'environ 0.2 mm. Une visualisation à 250 mm donne une taille de scène visuelle de $2 \times 250 \times \tan \frac{60^\circ}{2} = 290$ mm.

En ramenant au capteur de largeur 8.8 mm, le diamètre du cercle de confusion vaut :

$$CdC = 0.2 \times \frac{8.8}{290} = 0.00783 \text{ mm} \tag{5.5}$$

En général, il est préférable de définir une gamme de diamètres suite aux diverses approximations. Un calcul simple consiste à approximer le diamètre en divisant la diagonale du capteur (11 mm ici) par une constante : 1440, 1730 ou 3000. On considère donc un CdC dans l'intervalle $[0.0036, 0.00763]$ mm.

La profondeur de champ est également déterminée par l'ouverture de l'objectif N en général une valeur parmi $\{1, 1.4, 2, 2.8, 4, 5.6, 8, 11, 16, 22\}$. On peut montrer que le point le plus proche d_{proche} net ainsi que le plus loin d_{loin} pour une mise au point d, est obtenue ainsi :

$$d_{\text{proche}} = \frac{d}{1 + \frac{CdC \times N \times (d-f)}{f^2}} \quad \text{et} \quad d_{\text{loin}} = \frac{d}{1 - \frac{CdC \times N \times (d-f)}{f^2}} \tag{5.6}$$

On obtient donc la profondeur de champ PdC :

$$PdC = \frac{2 \times N \times CdC \times f^2 \times D \times (D-f)}{f^4 - N^2 \times CdC^2 \times (D-f)^2} \tag{5.7}$$

Pour que tout le bac soit net, la mise au point est placée à $d = 1650$ mm. Pour obtenir $PdC > 300$ mm, l'ouverture doit être de minimum 8 afin de respecter le critère sur toute la gamme de CdC. Ce paramètre nous donne un premier point net à 1.3 m et un dernier point net à 2 m couvrant ainsi tout le vrac.

5.2 Biais réel-virtuel

Les réseaux de neurones étant entraînés sur des images générées, il convient d'assurer un réalisme maximal afin de limiter le biais réel-virtuel introduit. Dans un premier temps, les paramètres de scène (ombrage de Phong) sont calculés par un simple algorithme en comparant des paires d'images réelles et virtuelles. Ceci permet en ayant connaissance du plastique employé et de pigments de couleurs, d'établir un abaque des paramètres pour entraîner un réseau de neurones sur de futures pièces. Dans un second temps, la calibration de la caméra est détaillée. L'image du bac couvrant la totalité du plan image de la caméra, le modèle doit tenir comptes des distorsions de la lentille. En modifiant le *vertex shader* OpenGL, des distorsions radiales et tangentielles peuvent être introduites afin d'aider l'algorithme à calculer des coefficients plausibles.

5.2.1 Paramètres de scène

Afin de limiter le biais réel-virtuel introduit lors de la génération de données, les paramètres de scène doivent être correctement réglés afin de générer des images réalistes. Une méthode consiste à utiliser les images réelles obtenues par homographie (Sec. 2.3) pour obtenir des intervalles de confiance de chaque paramètre de l'ombrage de Phong. Les paramètres de scène optimaux sont trouvés par affinage des intervalles de confiance initiaux (en général l'intervalle $[0, 1]$). L'optimisation est effectuée par une minimisation de la MSE (sans arrière-plan) entre l'image réelle issue de l'homographie et l'image générée par OpenGL (avec des paramètres parmi les intervalles de confiance) selon la même pose. Une fois les paramètres obtenus pour chaque image du jeu d'images réelles, l'intervalle de confiance final est simplement délimité par les valeurs minimales et maximales de paramètres (en éliminant éventuellement les valeurs aberrantes). Un exemple est donné pour le paramètre de brillance m_b (Sec. 2.1.2), dans l'algorithme 1.

Algorithme 1 : Calcul de l'intervalle de confiance M_b

Données : Image réelles $\left\{\hat{I}_i\right\}_{i=1,\dots,N}$, poses associées $\left\{(R,\vec{t})_i\right\}_{i=1,\dots,N}$
intervalle initial $\left[\underline{m_b},\overline{m_b}\right]$, incrément initial δ_b, précision ϵ
Résultat : Intervalle de confiance M_b
Fonction `trouverIntervalleOptimal(`$\hat{I}$,R,$\vec{t}$,$\underline{m_b}$,$\overline{m_b}$,δ_b,ϵ`)`
  tant que $\delta_b > \epsilon$ **faire**
    $\underline{e}, \hat{m_b}, m_b \leftarrow 10^7, 0, \underline{m_b}$
    tant que $m_b \leq \overline{m_b}$ **faire**
      $I \leftarrow$OPENGL$(R, \vec{t}, m_b)$
      $e \leftarrow$MSE$(I, \hat{I})$
      si $e < \underline{e}$ **alors**
        $\underline{e}, \hat{m_b} \leftarrow e, m_b$
      $m_b \leftarrow m_b + \delta_b$
    $\underline{m_b}, \overline{m_b} \leftarrow \hat{m_b} - \delta_b, \hat{m_b} + \delta_b$
    $\delta_b \leftarrow \delta_b/2$
  retourner $\underline{m_b}, \overline{m_b}$
début
  $\underline{M_b}, \overline{M_b} \leftarrow [\ \ ], [\ \ ]$
  pour $i \leftarrow 1$ **à** N **faire**
    Ajouter `trouverIntervalleOptimal(`$\hat{I}_i$,R_i,$\vec{t_i}$,$\underline{m_b}$,$\overline{m_b}$,δ_b,ϵ`)` à $\underline{M_b}, \overline{M_b}$
  $M_b \leftarrow$min$(\underline{M_b})$,max$(\overline{M_b})$

La méthode proposée cherche en priorité à optimiser la composante ambiante l_a, puis diffuse l_d, spéculaire l_s, la brillance du matériau m_b et enfin la position de la lumière dans la scène $\vec{l}$. À chaque nouvelle optimisation de paramètres, les précédents sont sauvegardés dans le *fragement shader*. L'algorithme est relancé plusieurs fois afin d'éviter les valeurs aberrantes. Pour générer une image réaliste, les paramètres de scènes sont aléatoirement tirés dans les intervalles de confiance obtenus (Fig. 5.4, Fig. 5.5). Cette technique permet également d'élaborer un abaque des paramètres à employer pour simuler un matériau. En ayant connaissance du plastique employé et des pigments de couleur, des images synthétiques d'une future pièce similaire peuvent être générées sans fabrication au préalable de l'objet.

FIGURE 5.4 – Images de LINEMOD (haut) et rendus après optimisation des paramètres de scène (bas).

FIGURE 5.5 – Images de MULTITUDE (haut) et rendus après optimisation des paramètres de scène (bas).

5.2.2 Calibration

La lentille n'étant pas parfaite, elle génère des distorsions biaisant les coordonnées de l'information projetée. Considérons la projection P_C et son équivalent après distorsion P_{Cd}. La modélisation mathématique du phénomène revient à construire une fonction δ, si possible inversible, telle que :

$$\begin{gathered}(P_{C_x}, P_{C_y}) = \delta(P_{Cd_x}, P_{Cd_y}) \\ \text{et} \\ (P_{Cd_x}, P_{Cd_y}) = \delta^{-1}(P_{C_x}, P_{C_y})\end{gathered} \tag{5.8}$$

La distorsion radiale δ_R construite par déplacement des coordonnées relatives à un centre de distorsion (c_{dx}, c_{dy}) (non nécessairement égal au centre de l'image (c_x, c_y)) selon une loi polynomiale de coefficients $\{K_R\}$, est un modèle connu. Soit le rayon $r = ||P_{Cd} - c_d||_2$:

$$\delta_R(P_{Cd}) = P_{Cd} + (P_{Cd} - c_d) \times \sum_{i=1}^{K_R} K_{R_i} r^{2i} \tag{5.9}$$

Lorsque tous les coefficients K_{r_i} sont positifs, la distorsion est dite en pique-aiguille (Fig 5.6). Lorsqu'ils sont tous négatifs, la distorsion est en tonneau. Dans les autres cas, la distorsion est complexe (ou en moustache).

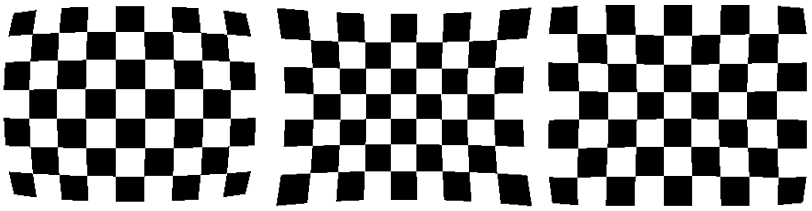


FIGURE 5.6 – De gauche à droite : distorsions par modèle radial en tonneau, pique-aiguille et moustache.

Ce modèle peut être complété par une distorsion tangentielle δ_T basée sur une forme non linéaire composée de coefficients $\{K_T\}$ et différente selon les axes $\vec{x}$ et $\vec{y}$:

$$\begin{aligned}\delta_T(P_{Cd})_x &= (K_{T_1}(r^2 + 2(P_{Cd_x} - c_{d_x})^2) + 2K_{T_2}(P_{Cd_x} - c_{d_x})(P_{Cd_y} - c_{d_y})) \times \sum_{i=0}^{K_T} K_{T_{i+2}} r^{2i} \\ \delta_T(P_{Cd})_y &= (2K_{T_1}(P_{Cd_x} - c_{d_x})(P_{Cd_y} - c_{d_y})) + K_{T_2}(r^2 + 2(P_{Cd_y} - c_{d_y})^2)) \times \sum_{i=0}^{K_T} K_{T_{i+2}} r^{2i}\end{aligned} \tag{5.10}$$

Le modèle complet de distorsion est la somme des deux précédents :

$$P_C = \delta(P_{Cd}) = \delta_R(P_{Cd}) + (\delta_T(P_{Cd})_x, \delta_T(P_{Cd})_y) \tag{5.11}$$

Notons que si la distorsion radiale inverse δ_R^{-1} peut être obtenue de façon exacte, ce n'est pas le cas de la distorsion tangentielle, qui nécessite une méthode itérative [20].

Les algorithmes d'estimation de distorsions éprouvent des difficultés lorsque la lentille possède de fortes distorsions. La focale de la caméra peut être diminuée au profit de plus fortes distorsions et inversement (il s'agit d'un compris). Cependant, en implémentant la distorsion au sein du *vertex shader* d'OpenGL, il est possible d'estimer l'erreur de rétroprojection des cibles d'un échiquier connu. Dans un premier temps, la focale de la caméra est estimée sur une zone peu distordue de l'image (majoritairement au voisinage du centre de l'image). Les coefficients de distorsions sont ensuite estimés par minimisation de l'erreur de rétroprojection des N cibles P_{C_i} dans le plan image.

$$\Theta^\star = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} ||\delta(P_{C_{di}}; \Theta) - P_{C_i}||_2^2 \tag{5.12}$$

Cette minimisation peut être effectuée avec un algorithme de réduction d'intervalle de confiance à l'image de celui employé pour estimer les paramètres de scène (Alg. 1), ou bien par une descente de gradient sur les paramètres des modèles de distorsion.

5.3 Implémentation

L'implémentation du module de vision pour l'estimation de pose doit être effectuée au sein d'une production industrielle. Dans un premier temps, une architecture réseau composée d'un contrôleur et d'un serveur est proposée. Cette dernière permet une centralisation des réseaux de neurones exploitables par des contrôleurs situés dans l'usine. Une fois la pose obtenue, une zone de préhension de la pièce est sélectionnée puis, le bras de robot contrôlé par un automate assure la saisie de la pièce et sa dépose sur un repère intermédiaire.

5.3.1 Déportation des calculs

Les normes de sécurité, l'emplacement disponible ainsi que les problématiques de refroidissement, empêchent l'utilisation d'une machine de calcul dédiée à l'entraînement et l'inférence des réseaux de neurones dans un environnement industriel. De plus, la machine doit servir à l'hébergement de plusieurs réseaux et doit rester accessible sur différents postes de dévracage. Il convient alors de déporter les calculs du module de vision.

La solution retenue est l'utilisation d'un contrôleur vidéo dialoguant via une interface Ethernet (en protocole HTTP) avec un serveur de calcul. Le serveur HTTP est bâti sur un *framework* Nodejs (serveur web en JavaScript) dialoguant avec le contrôleur par SocketIO (module de Nodejs permettant des communications en temps réel). Le serveur dialogue également avec les réseaux de neurones qu'il héberge en SocketIO (avec son implémentation en Python). Ceci permet un dialogue rapide et bidirectionnel : le serveur ou les réseaux de neurones peuvent envoyer des messages aux contrôleurs vidéo pour informer sur leur disponibilité ou leur charge et les contrôleurs peuvent demander des inférences aux réseaux (Fig. 5.7).

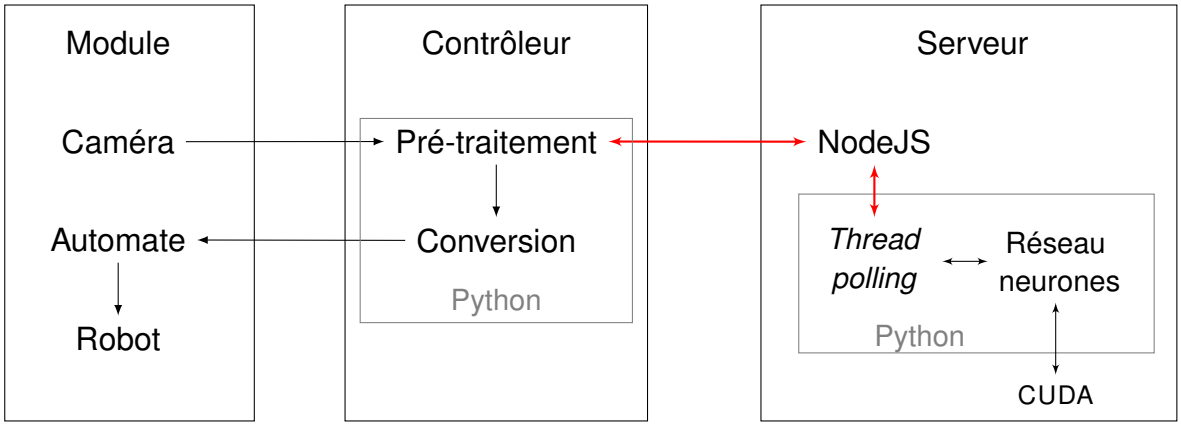


FIGURE 5.7 – Implémentation d'un réseau de neurones dans l'usine composée d'un contrôleur et d'un serveur Nodejs. Les liens en rouge représentent une communication en SocketIO.

La caméra montée dans le module de dévracage, envoie un flux vidéo par USB au contrôleur dans un script Python. Ce script est paramétré par un opérateur pour sélectionner un réseau de neurones propre à une application allant recevoir ce flux vidéo. Pour un même contrôleur, plusieurs applications peuvent être sélectionnées. Dans le futur ceci permettra par exemple d'estimer la pose d'une pièce mais également de réaliser des vérifications de qualité (présence ou non de défauts...).

Le flux vidéo devant être transmis aux réseaux de neurones peut être lourd. Si plusieurs applications de dévracage fonctionnent en parallèle, les commutateurs du réseau interne de l'usine seraient rapidement saturés. Ceci nécessite donc un pré-traitement de l'information afin de ne garder que quelques images pertinentes. Le script envoie les trames en SocketIO au serveur Nodejs avec les identifiants du ou des réseaux concernés et le type de requête (demande d'inférence, statut...). Le serveur dispose d'un module Nodejs permettant de lire et envoyer des requêtes en SocketIO venant du réseau Ethernet mais également, de communiquer en SocketIO à travers un *thread* de *polling* avec les applications Python hébergeant les différents réseaux de neurones.

Si l'identifiant du réseau n'existe pas, ce dernier est créé en Python et Nodejs transmet un message d'attente au contrôleur. Lorsque la communication est entièrement établie, le *thread* de *polling* lit régulièrement dans un tampon la liste des requêtes reçues de Nodejs. Ces requêtes sont interprétées puis transmises au réseau de neurones pouvant travailler avec le GPU et notamment avec la librairie CUDA (librairie de parallélisation des calculs dans les cartes graphiques). Les réponses des réseaux (statut, charge, résultat de l'inférence) sont remontées à Nodejs puis passent dans le réseau Ethernet en SocketIO avant d'être traitées par le contrôleur. Un module de conversion interprète le message du serveur et peut par exemple post-traiter l'inférence d'un réseau de neurones. Dans le cas d'un dévracage, ce module assure la conversion d'une pose en un signal robot interprété par un automate (Fig. 5.8).

```
INFO:: RECEIVED REQUEST getNetworksList FROM CLIENT dTV6OT1wYJ1u2DvvAAAA
INFO:: NEW NETWORK DETECTED
INFO:: RECEIVED REQUEST getNetworkInfoWithId FROM CLIENT dTV6OT1wYJ1u2DvvAAAA
INFO:: RECEIVED REQUEST subscribeToNetwork FROM CLIENT dTV6OT1wYJ1u2DvvAAAA
SUCCESS:: CLIENT dTV6OT1wYJ1u2DvvAAAA SUBSCRIBED TO Typ96T6kqqjSI23oAAAB
INFO:: RECEIVED REQUEST getNetworkArchitecture FROM CLIENT dTV6OT1wYJ1u2DvvAAAA TO NETWORK Typ96T6kqqjSI23oAAAB
INFO:: RECEIVED REQUEST getDatasetParams FROM CLIENT dTV6OT1wYJ1u2DvvAAAA TO NETWORK Typ96T6kqqjSI23oAAAB
INFO:: RECEIVED MESSAGE FROM NETWORK Typ96T6kqqjSI23oAAAB TO CLIENT dTV6OT1wYJ1u2DvvAAAA
INFO:: NETWORK 1 DISCONNECT FROM SOCKET Typ96T6kqqjSI23oAAAB
{
  network_id: 1,
  network_name: 'reseau_1',
  network_socket_id: 'Typ96T6kqqjSI23oAAAB',
  network_status: 'off'
}
```

FIGURE 5.8 – Exemple de transmissions par SocketIO. Le contrôleur (client) s'inscrit aux événements d'un réseau et peut demander diverses informations notamment l'architecture, puis se déconnecte.

Cette architecture d'implémentation permet de gérer plusieurs réseaux dans divers endroits de l'usine. Le serveur possède une interface web permettant de consulter l'état du serveur (charge, mémoire...) mais également celui des réseaux (visualisation des couches, architecture...) (Fig. 5.9). L'interface ne se limite pas à la simple consultation puisqu'il est possible d'initialiser ou stopper des réseaux selon l'application souhaitée. Pour le dévracage, un réseau peut être instancié et entraîné en spécifiant la nouvelle pièce à produire (notamment son plastique et sa coloration). En cas d'erreur, il est facile d'isoler le réseau mis en cause et d'appliquer rapidement un correctif en pleine production et ceci à distance. De plus, les protocoles employés étant standards, l'implémentation ne nécessite pas de matériel propriétaire pour fonctionner.

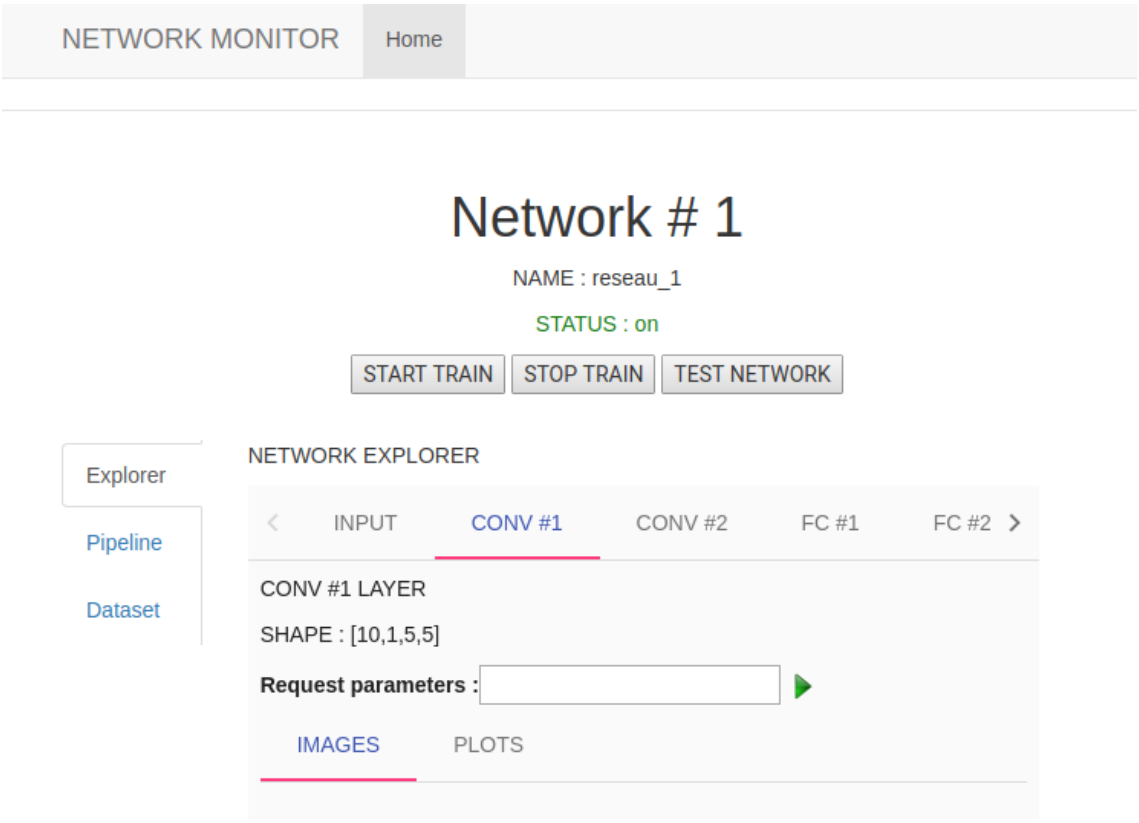


FIGURE 5.9 – Image de l'interface web du serveur dans la fonctionnalité d'exploration des couches d'un réseau.

Cette solution de déploiement est efficace puisqu'elle ne nécessite au final que d'implémenter un simple contrôleur vidéo (ne demandant que peu de ressources matérielles) sur le poste de dévracage. L'opérateur est ensuite guidé dans l'interface utilisateur pour utiliser la machine de calcul centralisée et exploiter les réseaux de neurones.

5.3.2 Préhension

Lorsque une instance est correctement identifiée avec sa pose associée, un bras de robot muni d'un préhenseur est chargé de la saisir sur une des zones de préhension autorisées. Chaque pièce dispose d'une base de données des préhensions possibles comportant chacune le type de préhension (pince, ventouse...), la position à la surface ainsi que la normale associée (Fig. 5.10).

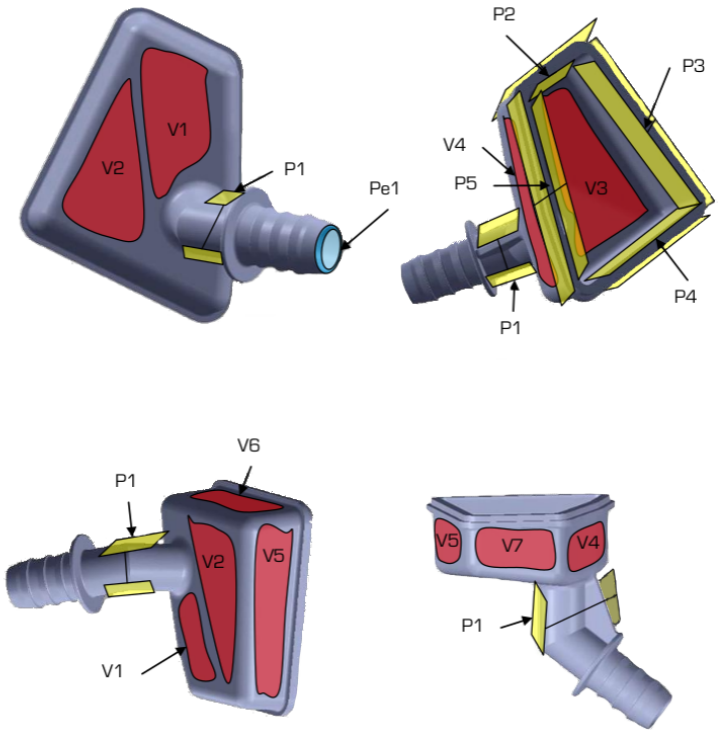


FIGURE 5.10 – Exemple de base de données des préhensions d'une pièce produite par ventouse (V_i), pince (P_i) et pince à mors ouvrants (P_{e_i}).

La préhension est retenue lorsque la normale à la surface de préhension est contenue dans un cône axé sur Z et ouvert de $30°$. La faisabilité de la prise est également établie selon la présence de points du nuage rétro-projeté au sein du cône. L'approche du préhenseur est calculée en 4 points selon la normale à la surface de préhension. L'automate dispose d'une zone de sécurité pour laquelle il calcule les éventuelles collisions entre le bac et le préhenseur. Les points d'approche du robot sont envoyés à l'automate qui calcule en langage pour PLC la trajectoire pour les atteindre. La dépose de la

pièce sur un support intermédiaire est également assurée en PLC dans l'automate. Ce calcul est néanmoins complexe puisque l'automate n'autorise par l'utilisation de matrices ni de quaternion. Il convient alors de transformer la pose sous forme d'un triplet Eulérien selon la convention utilisée par le robot. Puisque le robot ne travaille qu'en rotation intermédiaire, le calcul de la transformation doit être effectué depuis le repère de l'outil (TCP). Pour cela, la conversion d'une matrice de $SO(3)$ vers des angles d'Euler est réalisée dans l'automate [92] (Annexe B). Cet algorithme est dans un premier temps testé dans un environnement virtuel avant d'être implémenté dans l'automate (Fig. 5.11).

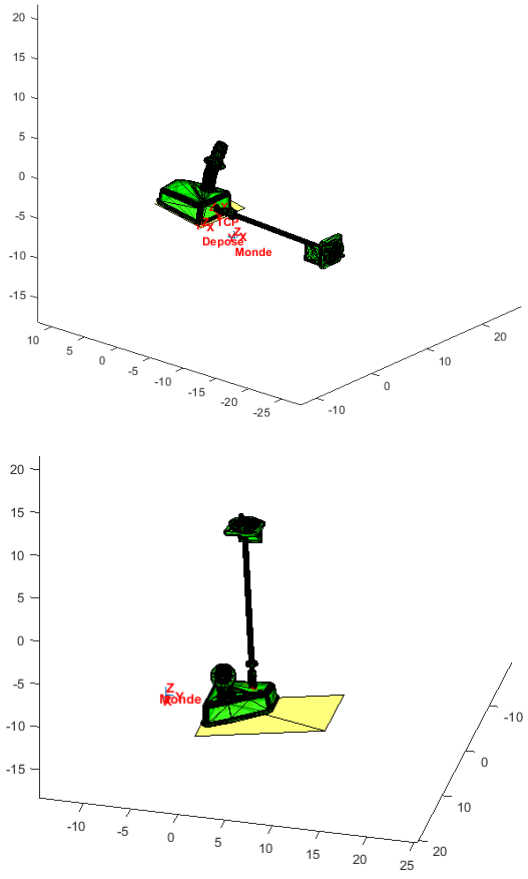


FIGURE 5.11 – Exemple de déposes dans un environnement virtuel (saisies en V_2 et V_6).

Dans le cas d'une préhension par l'une des zones du dessous de pièce (saisie en V_3), le bras de robot ne peut pas déposer la pièce sans collisions. Pour contrer ce problème, une dépose par fourchette est envisagée. Dans cette situation, l'automate fournit 4 points d'approche en spline au bras de robot afin de déposer au mieux la pièce contre la fourchette (Fig. 5.12).

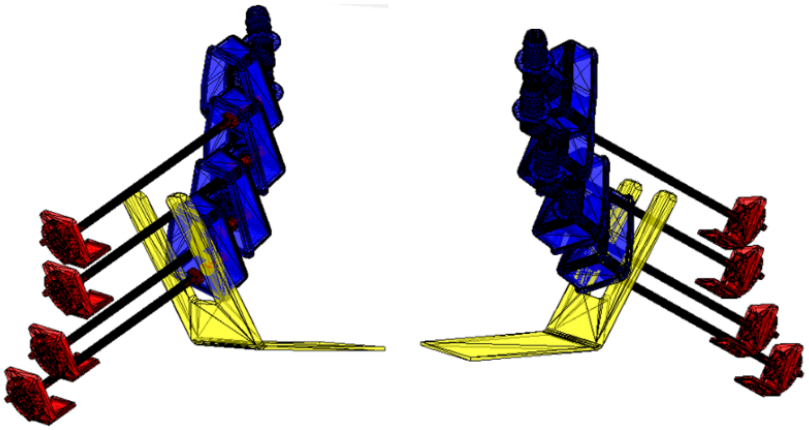


FIGURE 5.12 – Dépose par fourchette lorsque la pièce est saisie par le dessous (saisie en V_3).

Ce module de dépose s'intègre avec d'autres solutions puisqu'il fonctionne avec le module de dévracage industriel testé en début de thèse (Table. 3) mais également avec le module neuronal proposé.

Synthèse et conclusion du chapitre

Ce chapitre présente les détails de l'implémentation du module neuronal au sein d'une industrie. Les différents choix sur les données d'entraînement des réseaux de neurones sont explicités. Le nombre de poses N à générer par itération influence peu les performances du réseau puisque N agit comme un tirage aléatoire sur l'espace des rotations. Chaque itération permet donc de découvrir des nouvelles poses et de contrer des éventuels effets de classification. Les paramètres de scène sont calculés par un simple algorithme de minimisation de la MSE entre images réelles et virtuelles. Cette technique permet d'obtenir des rendus réalistes et d'établir un abaque des paramètres pour tous les types de plastiques utilisés dans l'entreprise. La caméra employée dans le module neuronal dispose d'une focale adaptée à la scène (couverture du bac et effets de perspectives réduits). Le calcul de l'ouverture au préalable, assure une zone de netteté de l'image sur toute la hauteur du bac. Afin d'utiliser au mieux la caméra, la distance focale doit être calibrée et les éventuelles distorsions de le lentille compensées par des modèles mathématiques. Puis, pour utiliser les réseaux de neurones dans l'usine, une architecture de déploiement composée d'un serveur de calcul communiquant avec des contrôleurs est proposée. Elle permet notamment de centraliser les calculs et de vérifier l'état des machines durant la production. Enfin, le choix de la zone de préhension est détaillé. Ce choix est basé majoritairement sur la normale à la surface de préhension devant être contenue dans un cône d'accessibilité sur l'axe Z. Les mouvements du robot sont programmés en langage automate pour la préhension mais aussi pour la dépose dont le module est validé virtuellement avant d'être implémenté. Ce module de dépose a été implémenté avec succès dans l'usine avec la solution de dévracage industriel testée en début de thèse et s'intègre avec le module neuronal proposé.

CONCLUSIONS ET PERSPECTIVES

Réalisme des données

Estimer la pose d'un objet dans une scène par des réseaux de neurones nécessite au préalable une grande quantité d'images de l'objet vu sous différentes poses. En pratique, il est impossible d'obtenir de telles données puisque l'objet est contraint de tenir sur une de ses positions d'équilibre. De plus, les photographies résultantes sont dépendantes de la luminosité de la scène durant la session de prises de vues. Ces travaux de thèse proposent alors d'employer des images synthétiques pour l'entraînement des réseaux de neurones. Ceci permet de générer des données durant l'entraînement avec une grande versatilité (modifications de lumières, choix de la pose...). De plus, la pièce industrielle n'a pas besoin d'être produite *a priori* pour débuter l'entraînement des réseaux.

Ces images sont générées d'après le modèle CAO de l'objet à travers un pipeline OpenGL disposant d'un modèle mathématique de coloration de pixels. Ce modèle (ombrage de Phong) dispose de 3 composantes principales de luminosité et demeure assez simple. Afin d'établir des paramètres de scène offrant le meilleur réalisme, un algorithme basé sur la correspondance entre une paire d'images réelle et virtuelle et proposé. Bien que le module neuronal doit être opérationnel sans avoir produit la pièce au préalable, ce processus nécessite néanmoins de disposer d'une base de données d'images réelles de l'objet avec la pose associée. Cependant, puisque les plastiques employés dans l'entreprise sont prédéfinis, un abaque peut être construit avec des pièces déjà produites employant un matériau similaire. L'algorithme de choix des paramètres de scène n'est donc pas indispensable à tout nouvel objet à dévraquer.

Il est possible de construire un modèle de coloration plus complet basé sur des textures plastiques extraites de pièces déjà produites. Il faut noter que des *shaders* plus lourds augmentent fortement le temps de génération des données notamment dans le cas de *shaders* de luminosité par pixel (*per-pixel lighting*). De plus, un grand nombre des implémentations de ces modèles sont propriétaires et la liberté de leur utilisation en industrie n'est pas assurée. Ces travaux montrent qu'un modèle de Phong offre un réalisme suffisant pour inférer la pose d'objet depuis des images réelles après un entraînement sur des images synthétiques. Dans des futurs travaux, des implémentations de *shaders* plus complexes peuvent éventuellement être proposées.

Depuis l'émergence des réseaux adverses génératifs (GAN) [29], la translation d'image est facilitée (par exemple une scène vue de jour transformée en une scène vue de nuit...). Puisque les réseaux sont entraînés sur des images virtuelles, un biais réel-virtuel est introduit. Un GAN peut être employé afin de transformer une image synthétique en une image disposant d'un meilleur réalisme. Cependant, l'introduction de réalisme sur le rendu d'un objet est complexe car il nécessite un ajout d'entropie dans l'image à des endroits précis (saillances lumineuses liées à la géométries, tâches, marquages...). Pour contrer ce défaut, il est envisagé de transformer une image réelle en une image à l'apparence synthétique. L'image servant à l'inférence de la pose serait au préalable transformée par un GAN afin d'absorber les défauts liés à la réalité. Les réseaux d'estimation de pose étant entraînés sur des images virtuelles, offriront alors de meilleures performances. Cependant, l'entraînement des GANs aujourd'hui requiert un grand nombre d'images et demeure toujours instable. Ceci suppose alors de disposer d'une grande base de données d'images réelles de pièces avec la pose associée ce qui n'est pas envisagé pour le moment.

Segmentation d'instances

Les différentes architectures implémentées pour répondre à la problématique de segmentation d'instances ainsi que les expériences associées, nécessitent d'être approfondies. En particulier, d'autres architectures [37] offrent de très bonnes performances pour des problématiques de segmentation variées et n'ont pas été testées durant la thèse. Également, la prédiction de cartes angulaires par encodeur-décodeur peut permettre de résoudre la problématique. Les pièces étant angulairement classifiées depuis leur centroïde dans l'image, il est possible de détecter le motif central et de parcourir les écarts angulaires pour former le masque binaire de chaque instance. Cette détection de motif est similaire à un module d'attention visuel auparavant implémenté sous la forme d'un LSTM. Ces travaux de thèse s'étant principalement concentrés sur une solution neuronale, un algorithme *ad-hoc* peut parfois offrir de meilleures performances en dépit d'un temps de calcul plus élevé.

Il est important de noter que l'étape de segmentation d'instances n'est pas critique dans le cas de l'estimation de pose. Plusieurs alternatives aux bacs sont envisagées (convoyeur, tambour vibrant), afin de séparer les instances entre elles. Cependant, ces technologies sont plus onéreuses et moins modulaires qu'un simple bac ce qui suppose de modifier le cahier des charges initial.

Précision de l'estimation de pose

L'estimation de pose est réalisée au moyen de trois réseaux de neurones. Un premier encodeur-décodeur est chargé d'inférer la carte de profondeur d'une pièce dans la scène. En employant une fonction de coût probabiliste, la carte produite est peu bruitée et permet par la suite d'exploiter la profondeur en tant que nuage de points. Cette modalité est employée avec les coordonnées locales des pixels de la pièce dans l'image source dans deux réseaux. Le premier infère l'orientation de la pièce sous la forme d'un quaternion unitaire. L'emploi d'une fonction de coût issue d'une métrique de $Sp(1)$ permet d'obtenir une erreur géodésique proche de $20°$ sur des images réelles. Le second réseaux infère la translation sur l'axe Z de l'objet dans la scène. Cette translation permet de rétro-projeter la profondeur locale afin de construire un nuage de points de l'objet dans la scène. Le modèle CAO de l'objet est ensuite orienté selon le quaternion et la translation prédits, puis recalé dans le nuage de points par un algorithme ICP afin d'obtenir la pose finale.

Ce recalage est efficace pour diminuer l'erreur d'estimation de pose liée au biais réel-virtuel introduit lors de la génération des images d'entraînement. Son initialisation sur un quaternion estimé au préalable par un réseau de neurones permet de converger efficacement vers une solution qui n'est pas un minimum local. Ce pipeline d'estimation de pose montre des résultats proches de l'état de l'art sur LINEMOD alors que la majorité des travaux utilise des images réelles pour l'entraînement des réseaux de neurones. Les résultats sur MULTITUDE sont cependant moins prometteurs puisque le jeu de données est plus complexe notamment du fait des variations de luminosité et de la pose sur $SE(3)$. Néanmoins, ces performances sont suffisantes pour employer un préhenseur flexible permettant d'absorber les erreurs d'estimation de pose.

Cette stratégie reste dépendante de la bonne inférence de profondeur puisque les réseaux d'orientation et de translation n'utilisent pas l'image RGB. L'architecture encodeur-décodeur est très robuste au biais réel-virtuel contrairement aux réseaux convolutifs classiques. L'emploi de la modalité RGB avec la profondeur dans ces réseaux ajoute un nouveau biais faisant chuter les performances de l'estimation d'orientation. Cependant, employer uniquement la profondeur est limité dans le cas où la pose n'est différentiable que par une information de texture ou de couleur dans la scène.

Annexes

ANNEXE A

ENSEMBLE DES QUATERNIONS $\mathbb{H}$

A.1 Produit de Hamilton sur $\mathbb{H}$

Soient deux quaternions $q_{1,2} = a_{1,2} + \vec{v_{1,2}}$ avec $\vec{v_{1,2}} = b_{1,2}i + c_{1,2}j + d_{1,2}k$. Le produit des deux quaternions s'écrit :

$$q_1 q_2 = a_1 a_2 + a_2 \vec{v_1} + a_1 \vec{v_2} + \vec{v_1}\vec{v_2}$$

en particulier :

$$\vec{v_1}\vec{v_2} = (b_1 i + c_1 j + d_1 k)(b_2 i + c_2 j + d2k)$$
$$= b_1 b_2 i^2 + b_1 c_2 ij + b_1 d_2 ik + c_1 b_2 ji + c_1 c_2 j^2 + c_1 d_2 jk + d_1 b_2 ki + d_1 c_2 kj + d_1 d_2 k^2$$

avec les relations des quaternions (Éq. 1.4) :

$$\begin{aligned}\vec{v_1}\vec{v_2} &= -(b_1 b_2 + c_1 c_2 + d_1 d_2) + (b_1 c_2 k - b_1 d_2 j - c_1 b_2 k + c_1 d_2 i + d_1 b_2 j - d_1 c_2 i) \\ &= -\vec{v_1}.\vec{v_2} + \begin{pmatrix} i & j & k \end{pmatrix} \begin{pmatrix} c_1 d_2 - d_1 c_2 \\ d_1 b_2 - b_1 d_2 \\ b_1 c_2 - c_1 b_2 \end{pmatrix} \\ &= -\vec{v_1}.\vec{v_2} + \vec{v_1} \wedge \vec{v_2}\end{aligned} \tag{A.1}$$

au final :

$$q_1 q_2 = a_1 a_2 - \vec{v_1}.\vec{v_2} + a_2 \vec{v_1} + a_1 \vec{v_2} + \vec{v_1} \wedge \vec{v_2}$$

ANNEXE B

ÉQUIVALENCES ENTRE $\theta + \vec{u}$, $SO(3)$ ET $Sp(1)$

B.1 De $\theta + \vec{u}$ vers $SO(3)$

Toute matrice de rotation R peut être obtenue à partir d'une représentation axe-angle selon les paramètres θ (angle de rotation) et $\vec{u}$ (vecteur unitaire de l'axe de rotation). En partant de la formule de rotation de Rodrigues (Fig. B.1) tournant $\vec{v}$ en $\vec{v'}$:

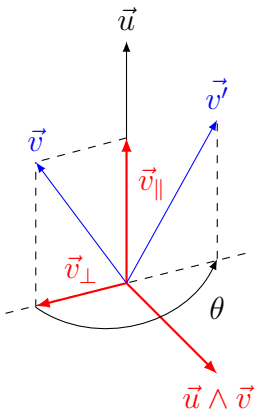


FIGURE B.1 – Vecteurs $(\vec{v}_{\parallel}, \vec{v}_{\perp})$ de la formule de rotation de Rodrigues.

$$\vec{v'} = \vec{v}_{\parallel} + \cos(\theta)\vec{v}_{\perp} + \sin(\theta)(\vec{u} \wedge \vec{v})$$

avec :

$$\begin{aligned} \vec{v}_{\perp} &= \vec{v} - \vec{v}_{\parallel} \\ \vec{v}_{\parallel} &= \vec{u}(\vec{u}.\vec{v}) \end{aligned} \tag{B.1}$$

au final :

$$\vec{v'} = \cos(\theta)\vec{v} + \sin(\theta)(\vec{u} \wedge \vec{v}) + \vec{u}(\vec{u}.\vec{v})(1 - \cos(\theta))$$

En prenant une matrice antisymétrique $[u]_{\wedge} = U$ pour représenter le produit vectoriel, on montre que :

$$\begin{aligned} U &= \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix} \\ \vec{v'} &= (I + \sin(\theta)U + (1 - \cos(\theta))U^2)\vec{v} \end{aligned} \tag{B.2}$$

En développant l'expression $(I + \sin(\theta)U + (1 - \cos(\theta))U^2)$ on obtient la forme de R explicitée dans l'équation 1.3.

Également, la correspondance exponentielle entre $so(3)$ et $SO(3)$ (Annexe C) permet de construire une matrice de rotation à l'aide d'un élément de l'espace tangent $so(3)$ (matrice antisymétrique). Il est possible de montrer que cet élément correspond exactement à la représentation angle-axe en utilisant un vecteur $\vec{\omega} = \theta\vec{u}$ tel que :

$$R = e^{[\vec{\omega}]_{\wedge}} = \exp \begin{pmatrix} 0 & -\theta u_3 & \theta u_2 \\ \theta u_3 & 0 & -\theta u_1 \\ -\theta u_2 & \theta u_1 & 0 \end{pmatrix} \tag{B.3}$$

B.2 De $\theta + \vec{u}$ vers $Sp(1)$

Soient un quaternion unitaire $q = e^{\theta u} = \cos(\theta) + \sin(\theta)\vec{u}$ et un vecteur $\vec{v} \in \mathbb{R}^3$. Montrons que la conjugaison de $\vec{v}$ par q équivaut à une rotation de 2θ autours de $\vec{u}$ du

vecteur $\vec{v}$.

$$\begin{aligned} q\vec{v}q^{-1} &= (\cos(\theta) + \sin(\theta)\vec{u})\vec{v}(\cos(\theta) - \sin(\theta)\vec{u}) \\ &= (\cos(\theta)\vec{v} + \sin(\theta)\vec{u}\vec{v})(\cos(\theta) - \sin(\theta)\vec{u}) \\ &= \cos^2(\theta)\vec{v} - \cos(\theta)\sin(\theta)\vec{v}\vec{u} + \cos(\theta)\sin(\theta)\vec{u}\vec{v} - \sin^2(\theta)\vec{u}\vec{v}\vec{u} \\ &= \cos^2(\theta)\vec{v} + \cos(\theta)\sin(\theta)(\vec{u}\vec{v} - \vec{v}\vec{u}) - \sin^2(\theta)\vec{u}\vec{v}\vec{u} \end{aligned}$$

En particulier en utilisant le produit de Hamilton (Éq. A.1) :

$$\vec{u}\vec{v} - \vec{v}\vec{u} = 2\vec{u} \wedge \vec{v} \underbrace{-\vec{u}.\vec{v} + \vec{v}.\vec{u}}_{=0}$$

et :

$$\begin{aligned} \vec{u}\vec{v}\vec{u} &= (\vec{u} \wedge \vec{v} - \vec{u}.\vec{v})\vec{u} \\ &= (\vec{u} \wedge \vec{v})\vec{u} - \vec{u}(\vec{u}.\vec{v}) \\ &= (\vec{u} \wedge \vec{v}) \wedge \vec{u} - \underbrace{(\vec{u} \wedge \vec{v}).\vec{u}}_{=0} - \vec{u}(\vec{u}.\vec{v}) \\ &= \underbrace{(\vec{u}.\vec{u})}_{=1}\vec{v} - 2\vec{u}(\vec{u}.\vec{v}) \end{aligned}$$

On obtient donc

$$\begin{aligned} q\vec{v}q^{-1} &= \cos^2(\theta)\vec{v} + 2\cos(\theta)\sin(\theta)(\vec{u} \wedge \vec{v}) - \sin^2(\theta)(\vec{v} - 2\vec{u}(\vec{u}.\vec{v})) \\ &= (\cos^2(\theta) - \sin^2(\theta))\vec{v} + 2\cos(\theta)\sin(\theta)(\vec{u} \wedge \vec{v}) + \vec{u}(\vec{u}.\vec{v})(2\sin^2(\theta)) \end{aligned}$$

Au final en utilisant les relations trigonométriques usuelles :

$$q\vec{v}q^{-1} = \cos(2\theta)\vec{v} + \sin(2\theta)(\vec{u} \wedge \vec{v}) + \vec{u}(\vec{u}.\vec{v})(1 - \cos(2\theta)) \tag{B.4}$$

On retrouve la formule de Rodrigues effectuant une rotation de 2θ autours de $\vec{u}$. Ainsi, tout quaternion unitaire $q = \pm(\cos(\theta/2) + \sin(\theta/2)\vec{u})$ réalise une rotation de θ autours de $\vec{u}$.

B.3 De $Sp(1)$ vers $SO(3)$

Soit un quaternion unitaire $q = a + \vec{u}$ avec $\vec{u} = (u_1, u_2, u_3)$ et $\vec{v} \in \mathbb{R}^3 = (v_1, v_2, v_3)$. Le conjugué de $\vec{v}$ par q peut être mis sous forme matricielle permettant ainsi de passer de

$\mathbb{H}$ vers $SO(3)$.

$$\begin{aligned}\vec{v'} = q\vec{v}q^{-1} &= (a+\vec{u})\vec{v}(a-\vec{u}) \\ &= a^2\vec{v} + a(\vec{u}\vec{v} - \vec{v}\vec{u}) - \vec{u}\vec{v}\vec{u} \\ &= a^2\vec{v} + 2a(\vec{u}\wedge\vec{v}) - (||\vec{u}||^2\vec{v} - 2\vec{u}(\vec{u}.\vec{v}))\end{aligned}$$

En projetant sur chaque dimension :

$$\begin{aligned}v'_1 &= (a^2 - ||\vec{u}||^2 + 2u_1^2)v_1 + (2u_1u_2 - 2au_3)v_2 + (2u_1u_3 + 2au_2)v_3 \\ v'_2 &= (2u_2u_1 + 2au_3)v_1 + (a^2 - ||\vec{u}||^2 + 2u_2^2)v_2 + (2u_2u_3 - 2au_1)v_3 \\ v'_3 &= (2u_3u_1 - 2au_2)v_1 + (2u_3u_2 + 2au_1)v_2 + (a^2 - ||\vec{u}||^2 + 2u_3^3)v_3\end{aligned}$$

En sachant que $a^2 + ||\vec{u}||^2 = 1$ on a finalement :

$$\vec{v'} = \begin{pmatrix} 1-2u_2^2-2u_3^2 & 2u_1u_2 - 2au_3 & 2u_1u_3 + 2au_2 \\ 2u_2u_1 + 2au_3 & 1-2u_1^2-2u_3^2 & 2u_2u_3 - 2au_1 \\ 2u_3u_1 - 2au_2 & 2u_3u_2 + 2au_1 & 1-2u_1^2-2u_2^2 \end{pmatrix}\vec{v} \tag{B.5}$$

ANNEXE C

MÉTRIQUES SUR $SO(3)$ ET $Sp(1)$

C.1 Géodésique Riemannienne sur $SO(3)$

Le groupe spécial orthogonal 3 noté $SO(3)$ comporte l'ensemble des matrices orthogonales de $O(3)$ dont le déterminant est de 1. Il inclut donc naturellement l'ensemble des matrices de rotation dans l'espace Euclidien $\mathbb{R}^3$. Le groupe $SO(3)$ (tout comme $O(3)$) est un groupe de Lie disposant de son algèbre de Lie notée $so(3)$. Les éléments de $so(3)$ sont les éléments de l'espace tangent à la variété $SO(3)$ à l'identité I (non démontré ici). La correspondance exponentielle $so(3) \to SO(3)$ est alors classiquement établie par une série :

$$\begin{aligned} so(3) &\to SO(3) \\ R &\mapsto e^R = \sum_{k=0}^{+\infty} \frac{1}{k!} R^k \end{aligned} \tag{C.1}$$

Depuis l'identité I, toute matrice de rotation peut être trouvée en parcourant $so(3)$ puis en projetant sur $SO(3)$ par la correspondance exponentielle $e^A \in SO(3)$ avec $A \in so(3)$. En suivant la direction A sur $so(3)$, la courbure sur la variété peut être obtenue en paramétrant par $t \in [0, 1]$:

$$R_{I \to e^A}(t) = e^{tA} \tag{C.2}$$

Si l'on cherche la courbure liant une rotation R_1 à R_2, on multiplie à gauche par R_1 pour débuter à l'orientation R_1 ($R(0) = R_1$) :

$$R_{R_1 \to e^A}(t) = R_1 e^{tA} \tag{C.3}$$

Puis, avec la condition $R(1) = R_2$ on résout le système $R_1 e^A = R_2$ pour trouver $A = \log(R_1^T R_2)$.

$$\begin{aligned} R_1 e^{tA} &= R_2 \\ e^{tA} &= R_1^T R_2 \\ A &= \log(R_1^T R_2) \end{aligned} \tag{C.4}$$

La forme final de la courbure s'écrit donc :

$$R_{R_1 \to R_2}(t) = R_1 e^{t \log(R_1^T R_2)} \tag{C.5}$$

La longueur L parcourue sur la courbure $R(t)$ est obtenue sur une variété Riemannienne par la somme des normes des vecteurs tangents pour toute valeur de t. La norme est évidemment définie par le produit scalaire sur $so(3)$ noté $\langle , \rangle$ tel que (Tr dénote l'opérateur de trace matricielle) :

$$\langle \dot{R(t)}, \dot{R(t)} \rangle^{1/2} = ||R(t)^{-1} \dot{R(t)}|| = \sqrt{\frac{1}{2} Tr((R(t)^{-1} \dot{R(t)})(R(t)^{-1} \dot{R(t)})^T)} \tag{C.6}$$

$$L = \int_0^1 \langle \dot{R(t)}, \dot{R(t)} \rangle^{1/2} dt = \int_0^1 ||R(t)^{-1} \dot{R(t)}|| dt \tag{C.7}$$

De façon triviale on obtient :

$$\begin{aligned} \dot{R(t)} &= R_1 e^{t \log(R_1^T R_2)} \log(R_1^T R_2) = R_1 \log(R_1^T R_2) e^{t \log(R_1^T R_2)} \\ R(t)^{-1} &= (R_1 e^{t \log(R_1^T R_2)})^T = (e^{t \log(R_1^T R_2)})^T R_1^T \\ R(t)^{-1} \dot{R(t)} &= (R_1 e^{t \log(R_1^T R_2)})^T = (e^{t \log(R_1^T R_2)})^T R_1^T R_1 e^{t \log(R_1^T R_2)} \log(R_1^T R_2) \end{aligned} \tag{C.8}$$

Comme $R_1 \in SO(3)$ et $(e^{t \log(R_1^T R_2)})^T \in SO(3)$ le produit est simplifié $R(t)^{-1} \dot{R(t)} = \log(R_1^T R_2)$. Il vient alors que :

$$\begin{aligned} L &= \int_0^1 \langle \dot{R(t)}, \dot{R(t)} \rangle^{1/2} dt = || \log(R_1^T R_2)|| \\ L &= \sqrt{\frac{1}{2} Tr(\log(R_1^T R_2) \log(R_1^T R_2)^T)} \end{aligned} \tag{C.9}$$

C.2 Géodésique Riemannienne sur $Sp(1)$

Le logarithme d'un quaternion $q = a + \vec{u}$ est obtenu par :

$$\ln(q) = \ln(||q||) + \frac{\vec{u}}{||\vec{u}||} \arccos\left(\frac{a}{||q||}\right) \tag{C.10}$$

En particulier, pour un quaternion unitaire $\ln(q) = \frac{\vec{u}}{||\vec{u}||} \arccos(a)$. La distance géodésique L définie dans l'équation 4.5 peut alors s'écrire avec des quaternions q_1 et q_2 :

$$L = ||\log(R_1^T R_2)|| = 2 \arccos(|q_1.q_2|) \tag{C.11}$$

ANNEXE D

TIRAGE HOMOGÈNE DE POINTS SUR UNE SPHÈRE

D.1 Icosaèdres

Un icosaèdre est un polyèdre régulier à 20 faces et 12 sommets. Une subdivision d'icosaèdre consiste à fabriquer à partir de chaque surface triangulaire, 4 nouvelles surfaces à partir du milieu des arêtes placés sur la sphère englobante (Fig.D.1).

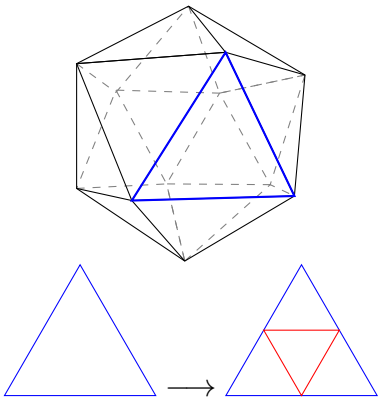


FIGURE D.1 – Découpage triangulaire d'une face d'icosaèdre.

Dans un icosaèdre, chaque face comporte 3 arêtes dont chacune appartient à une autre face. On montre donc que le nombre d'arêtes A est lié au nombre de faces F tel que $A = \frac{3F}{2}$. Chaque subdivision de l'icosaèdre multipliant par 4 le nombre de faces originales (20), le nombre de face après i subdivisions vaut simplement $F_i = 20 \times 4^i$. En utilisant la caractéristique d'Euler d'un polyèdre $\chi = S - A + F = S - \frac{F}{2} = 2$ on obtient trivialement $S_i = 2 + 10 \times 4^i$ (Table. D.1).

TABLE D.1 – Nombre de sommets du polyèdre en fonction du nombre de subdivision de l'icosaèdre initial.

Nombre de subdivisions	0	1	2	3	4	5
Nombre de sommets	12	42	162	642	2562	10242

D.2 Sphère de Fibonacci

D.2.1 Tirages

L'échantillonnage de la surface de la sphère par découpage régulier des intervalles $\phi \in [-\pi, \pi]$ et $\theta \in [0, \pi]$ n'offre pas une discrétisation régulière notamment au niveau des pôles. L'idée est donc d'échantillonner les angles en fonction de la distance à l'équateur.

$$y_i = \frac{2i + 1 - N}{N} \quad x_i = \sqrt{1 - y_i^2} \times \cos\left(2\pi i \left(2 - \Phi\right)\right) \quad z_i = \sqrt{1 - y_i^2} \times \sin\left(2\pi i \left(2 - \Phi\right)\right)$$

D.2.2 Géodésiques

Pour différents nombre de points N, les coordonnées des points sur la sphère $P = \{x, y, z\}_{i=1,\ldots,N}$ sont placées sous forme d'un arbre k-d. Pour chaque point P_i, son plus proche voisin P_j est calculé en distance L_2 puis, la distance géodésique G_i, est obtenue par :

$$G_i = \cos^{-1}\left(\langle P_i, P_j \rangle\right) \tag{D.1}$$

La moyenne des distances géodésique en fonction du nombre de points sur la sphère $G(N)$ (Table. D.3), peut être approximée par une simple régression en puissance (Fig. D.3) :

$$G(N) \approx \frac{3.36}{\sqrt{N}} \tag{D.2}$$

Cette valeur géodésique est bien inférieure à l'encadrement issu du problème de Tammes [10] :

$$G(N) \leq \sqrt{\frac{8\pi}{\sqrt{3}N}} \approx \frac{3.81}{\sqrt{N}} \tag{D.3}$$

TABLE D.2 – Valeurs moyennes des géodésiques pour N points sur la 2-sphère.

N	5	10	50	100	500	1000	2000	5000
$G(N)$	$87.86°$	$59.55°$	$26.99°$	$19.28°$	$8.73°$	$6.11°$	$4.34°$	$2.76°$

Pour s'en convaincre, il est possible de considérer un nombre de points N suffisamment grand pour approximer la surface locale du voisinage d'un point de la sphère à un plan. Le découpage idéal peut être alors un pavage hexagonal de côté a dont la distance entre deux centre d'hexagones voisins, d, est le double de la hauteur du triangle équilatéral de côté a : $d = 2\frac{\sqrt{3}a}{2} = \sqrt{3}a$ (Fig. D.2).

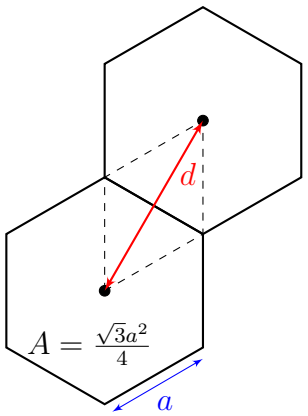


FIGURE D.2 – Surface locale du pavage hexagonal de la sphère approximée par un plan.

La somme des aires des hexagones $6N\frac{\sqrt{3}a^2}{4}$ devant correspondre à l'aire de la sphère unitaire 4π, on en déduit aisément l'équation $6N\frac{\sqrt{3}d^2}{3\times4} = 4\pi$ puis, $d = \sqrt{\frac{8\pi}{\sqrt{3}N}}$. Pour tout découpage, la distance moyenne entre deux proches voisins est forcément inférieure à cette limite (Éq. D.3).

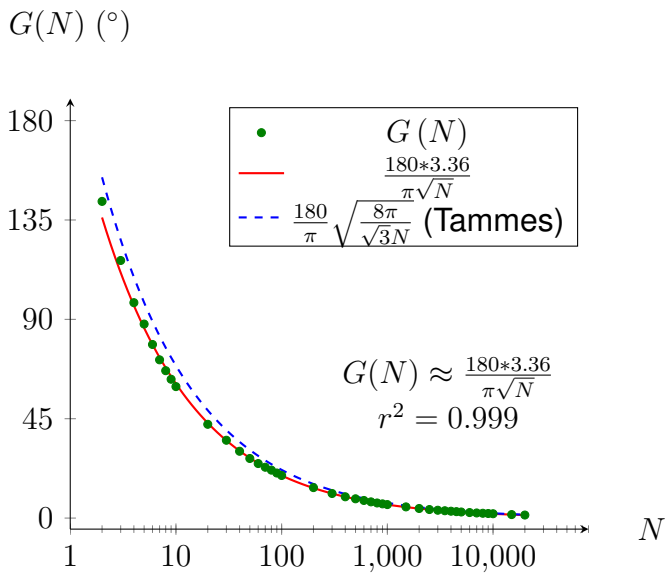


FIGURE D.3 – Régression de G par une loi en puissance selon N.

D.3 Tirage uniforme sur la 3-sphère

Pour sélectionner des orientations homogènes sur l'espace $SO(3)$, il est nécessaire d'échantillonner la 3-sphère étant donnés N points. Soit N le nombre de rotations souhaitées, les point $(x_i, y_i, z_i)_{\{i=1,\dots,N\}}$ sont donnés par la séquence de van der Corput (VDC) en base 2 et 3 tels que :

$$(x_i, yi, zi) = (\text{VDC}(3, i), \text{VDC}(2, i), i) \tag{D.4}$$

Les matrices de rotation résultantes R_i sont alors définies par :

$$V_i = \frac{1}{\sqrt{N}} \begin{pmatrix} \cos(2\pi y_i)\sqrt{z_i} \\ \sin(2\pi y_i)\sqrt{z_i} \\ \sqrt{N - z_i} \end{pmatrix} \quad R_i = -\left(I - 2V_iV_i^T\right) \begin{pmatrix} \cos(2\pi x_i) & \sin(2\pi x_i) & 0 \\ -\sin(2\pi x_i) & \cos(2\pi x_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{D.5}$$

La distance géodésique moyenne entre deux points proches évolue naturellement plus lentement que l'espace $SO(2)$ lorsque N augmente (Table.D.3).

TABLE D.3 – Valeurs moyennes des géodésiques pour N points sur la 3-sphère.

N	5	10	50	100	500	1000	2000	5000
$G(N)$	108°	81.0°	52.6°	38.6°	22.6°	17.5°	13.6°	9.43°

On peut montrer par une régression que la valeur de la distance géodésique moyenne entre deux points voisins sur la 3-sphère, $G(N)$, peut être approximée par :

$$G\left(N\right) \approx \frac{3.47}{\sqrt[3]{N}} \tag{D.6}$$

Pour un nombre $N > 1500$ de points, la distribution des distances géodésiques moyennes entre deux points voisins se concentre sur un pic. On peut alors considérer que le tirage devient quasi-homogène (Fig. D.4).

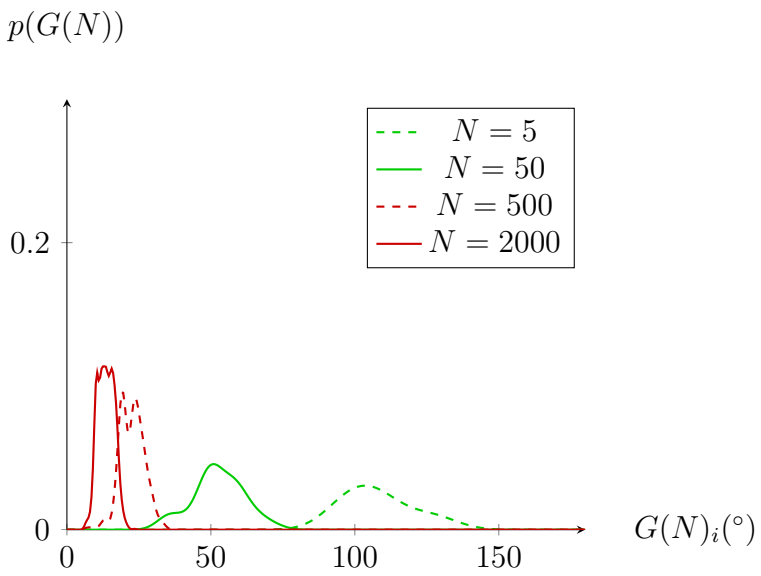


FIGURE D.4 – Distribution de la moyenne des distance géodésiques sur la 3-sphère entre plus proches voisins pour $N = 5, 50, 500, 1000$ et 2000 points.

PUBLICATIONS DE L'AUTEUR

Workshop

- Workshop SFWICT 2017 (papier + poster)
  J. Langlois, H. Mouchère, N. Normand et C. Viard-Gaudin
  3D Pose Estimation of Industrial Part from RGB Image Using Deep Learning
  Sino-French Workshop on Information and Communication Technology, 2017

Conférences internationales

- Conférence ICPRAM 2018 (papier + poster)
  J. Langlois, H. Mouchère, N. Normand et C. Viard-Gaudin
  3D Orientation Estimation of Industrial Parts from 2D Images using Neural Networks
  International Conference on Pattern Recognition Applications and Methods, 2018, pages 409-416
- Conférence QCAV 2019 (papier + présentation)
  J. Langlois, H. Mouchère, N. Normand et C. Viard-Gaudin
  On The Fly Generated Data for Industrial Part Orientation Estimation with Deep Neural Networks
  Quality Control by Artificial Vision, 2019

Communications

- JDOC 2018 (papier + meilleure présentation orale)
  J. Langlois
  Vision industrielle et réseaux de neurones profonds
  Journée des Doctorants, 2018
- GDR Robotique + ISIS - Apprentissage et robotique (présentation)
  J. Langlois, H. Mouchère, N. Normand et C. Viard-Gaudin
  Industrial Part Pose Estimation from Virtual Images with Deep Neural Networks

- Séminaire au vert IRISA et LS2N 2019 (présentation)
  J. Langlois
  Industrial Part Pose Estimation using Deep Neural Networks : from virtual to reality

BIBLIOGRAPHIE

[1] Alexandre ALAHI, Raphael ORTIZ et Pierre VANDERGHEYNST, « Freak : Fast retina keypoint », in : *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, p. 510–517, DOI : 10.1109/CVPR.2012.6247715.

[2] Sharon ALPERT et al., « Image segmentation by probabilistic bottom-up aggregation and cue integration », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.*2* (2011), p. 315–327, DOI : 10.1109/TPAMI.2011.130.

[3] Pablo ARBELÁEZ et al., « Multiscale combinatorial grouping », in : *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, p. 328–335, DOI : 10.1109/CVPR.2014.49.

[4] James ARVO, « Fast random rotation matrices », in : *Graphics Gems III (IBM Version)*, Elsevier, 1992, p. 117–120, DOI : 10.1016/B978-0-08-050755-2.50034-8.

[5] Vijay BADRINARAYANAN, Alex KENDALL et Roberto CIPOLLA, « Segnet : A deep convolutional encoder-decoder architecture for image segmentation », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.*12* (2017), p. 2481–2495, DOI : 10.1109/TPAMI.2016.2644615.

[6] Harry G BARROW et al., *Parametric correspondence and chamfer matching : Two new techniques for image matching*, rapp. tech., SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1977, URL : https://www.sri.com/work/publications/parametric-correspondence-and-chamfer-matching-two-new-techniques-image-matching.

[7] Johannes BAUER, Niko SÜNDERHAUF et Peter PROTZEL, « Comparing several implementations of two recently published feature detectors », in : *IFAC Proceedings Volumes* 40.*15* (2007), p. 143–148, DOI : 10.3182/20070903-3-FR-2921.00027.

[8] Herbert BAY, Tinne TUYTELAARS et Luc VAN GOOL, « Surf : Speeded up robust features », in : *2006 European Conference on Computer Vision*, Springer, 2006, p. 404–417, DOI : 10.1007/11744023_32.

[9] Carlos BELTRÁN et Damir FERIZOVIĆ, *Approximation to uniform distribution in SO(3)*, jan. 2019, arXiv : 1901.10840.

[10] Marcel BERGER, *Géométrie vivante, ou l'échelle de Jacob*, Cassini, 2009, ISBN : 978-2842250355.

[11] Robert C BOLLES et Ronald A CAIN, « Recognising and locating partially visible objects : the local-feature-focus method », in : *Robot Vision*, Springer, 1983, p. 43–82, DOI : 10.1007/978-3-662-09771-7_4.

[12] Yuri Y BOYKOV et M-P JOLLY, « Interactive graph cuts for optimal boundary & region segmentation of objects in ND images », in : *2001 IEEE International Conference on Computer Vision*, t. 1, IEEE, 2001, p. 105–112, DOI : 10.1109/ICCV.2001.937505.

[13] Yuri BOYKOV et Gareth FUNKA-LEA, « Graph cuts and efficient ND image segmentation », in : *International journal of computer vision* 70.*2* (2006), p. 109–131, DOI : 10.1007/s11263-006-7934-5.

[14] Michael CALONDER et al., « Brief : Binary robust independent elementary features », in : *2010 European Conference on Computer Vision*, Springer, 2010, p. 778–792, DOI : 10.1007/978-3-642-15561-1_56.

[15] Liang-Chieh CHEN et al., « Semantic image segmentation with deep convolutional nets and fully connected crfs », in : *arXiv preprint arXiv :1412.7062* (2014).

[16] Yi-Ting CHEN, Xiaokai LIU et Ming-Hsuan YANG, « Multi-instance object segmentation with occlusion handling », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 3470–3478, DOI : 10.1109/CVPR.2015.7298969.

[17] Philip DAVID et al., « SoftPOSIT : Simultaneous pose and correspondence determination », in : *International Journal of Computer Vision* 59.*3* (2004), p. 259–284, DOI : 10.1007/3-540-47977-5_46.

[18] Daniel F DEMENTHON et Larry S DAVIS, « Model-based object pose in 25 lines of code », in : *International journal of computer vision* 15.*1-2* (1995), p. 123–141, DOI : 10.1007/BF01450852.

[19] Andreas DOUMANOGLOU et al., « Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation », in : (2016), arXiv : 1607.02257.

[20] Pierre DRAP et Julien LEFÈVRE, « An exact formula for calculating inverse radial lens distortions », in : *Sensors* 16.*6* (2016), p. 807, DOI : 10.3390/s16060807.

[21] Clement FARABET et al., « Learning hierarchical features for scene labeling », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.*8* (2012), p. 1915–1929, DOI : 10.1109/TPAMI.2012.231.

[22] Dariu M GAVRILA et al., « Multi-feature hierarchical template matching using distance transforms », in : *1998 IAPR International Conference on Pattern Recognition*, t. 98, 1998, p. 439, DOI : 10.1109/ICPR.1998.711175.

[23] Natasha GELFAND et al., « Geometrically stable sampling for the ICP algorithm », in : *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.* IEEE, 2003, p. 260–267, DOI : 10.1109/IM.2003.1240258.

[24] Ross GIRSHICK, « Fast r-cnn », in : *2015 IEEE International Conference on Computer Vision*, 2015, p. 1440–1448, DOI : 10.1109/ICCV.2015.169.

[25] Ross GIRSHICK et al., « Rich feature hierarchies for accurate object detection and semantic segmentation », in : *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, p. 580–587, DOI : 10.1109/CVPR.2014.81.

[26] Xavier GLOROT et Yoshua BENGIO, « Understanding the difficulty of training deep feedforward neural networks », in : *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, p. 249–256.

[27] Steven GOLD et al., « New algorithms for 2D and 3D point matching : Pose estimation and correspondence », in : *Pattern recognition* 31.*8* (1998), p. 1019–1031, DOI : 10.1016/S0031-3203(98)80010-1.

[28] Álvaro GONZÁLEZ, « Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices », in : *Mathematical Geosciences* 42.*1* (2010), p. 49, DOI : 10.1007/s11004-009-9257-x.

[29] Ian GOODFELLOW et al., « Generative adversarial nets », in : *Advances in neural information processing systems*, 2014, p. 2672–2680, URL : http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[30] Iryna GORDON et David G LOWE, « What and where : 3D object recognition with accurate pose », in : *Toward category-level object recognition*, Springer, 2006, p. 67–82, DOI : 10.1007/11957959_4.

[31] David GRANGIER, Léon BOTTOU et Ronan COLLOBERT, « Deep convolutional networks for scene parsing », in : *ICML 2009 Deep Learning Workshop*, t. 3, 6, 2009, p. 109.

[32] Karol GREGOR et al., « Draw : A recurrent neural network for image generation », in : (2015), arXiv : 1502.04623.

[33] Saurabh GUPTA et al., « Learning rich features from RGB-D images for object detection and segmentation », in : *2014 European Conference on Computer Vision*, Springer, 2014, p. 345–360, DOI : 10.1007/978-3-319-10584-0_23.

[34] Bharath HARIHARAN et al., « Hypercolumns for object segmentation and fine-grained localization », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 447–456, DOI : 10.1109/CVPR.2015.7298642.

[35] Bharath HARIHARAN et al., « Simultaneous detection and segmentation », in : *2014 European Conference on Computer Vision*, Springer, 2014, p. 297–312.

[36] Kaiming HE et al., « Deep residual learning for image recognition », in : *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, p. 770–778, DOI : 10.1109/CVPR.2016.90.

[37] Kaiming HE et al., « Mask r-cnn », in : *2017 IEEE International Conference on Computer Vision*, 2017, p. 2961–2969, DOI : 10.1109/ICCV.2017.322.

[38] Kaiming HE et al., « Spatial pyramid pooling in deep convolutional networks for visual recognition », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.*9* (2015), p. 1904–1916, DOI : 10.1109/TPAMI.2015.2389824.

[39] Stefan HINTERSTOISSER et al., « Dominant orientation templates for real-time detection of texture-less objects », in : *2010 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, p. 2257–2264, DOI : 10.1109/CVPR.2010.5539908.

[40] Stefan HINTERSTOISSER et al., « Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes », in : *Asian conference on computer vision*, Springer, 2012, p. 548–562, DOI : 10.1007/978-3-642-37331-2_42.

[41] Stefan HINTERSTOISSER et al., « Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes », in : *2011 IEEE International Conference on Computer Vision*, IEEE, 2011, p. 858–865, DOI : 10.1109/ICCV.2011.6126326.

[42] Stefan HINTERSTOISSER et al., « On pre-trained image features and synthetic images for deep learning », in : *2018 European Conference on Computer Vision*, 2018, p. –, DOI : 10.1007/978-3-030-11009-3_42.

[43] Tomáš HODAŇ et al., « Detection and fine 3D pose estimation of texture-less objects in RGB-D images », in : *2015 IEEE/RSJ International Workshop on Intelligent Robots and Systems*, IEEE, 2015, p. 4421–4428, DOI : 10.1109/IROS.2015.7354005.

[44] Stefan HOLZER et al., « Distance transform templates for object detection and pose estimation », in : *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, p. 1177–1184, DOI : 10.1109/CVPR.2009.5206777.

[45] Sergey IOFFE et Christian SZEGEDY, « Batch normalization : Accelerating deep network training by reducing internal covariate shift », in : (2015), arXiv : 1502.03167.

[46] Omid Hosseini JAFARI et al., « iPose : instance-aware 6D pose estimation of partly occluded objects », in : *Asian Conference on Computer Vision*, Springer, 2018, p. 477–492, DOI : 10.1007/978-3-030-20893-6_30.

[47] Omid Hosseini JAFARI et al., « The best of both worlds : learning geometry-based 6D object pose estimation », in : (2017), arXiv : 1712.01924.

[48] Shuiwang JI et al., « 3D convolutional neural networks for human action recognition », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.*1* (2012), p. 221–231, DOI : 10.1109/TPAMI.2012.59.

[49] Ebrahim KARAMI, Siva PRASAD et Mohamed SHEHATA, « Image matching using SIFT, SURF, BRIEF and ORB : performance comparison for distorted images », in : (2017), arXiv : 1710.02726.

[50] Wadim KEHL et al., « SSD-6D : Making RGB-based 3D detection and 6D pose estimation great again », in : *2017 IEEE International Conference on Computer Vision*, 2017, p. 1521–1529, DOI : 10.1109/ICCV.2017.169.

[51] Alex KENDALL, Matthew GRIMES et Roberto CIPOLLA, « Posenet : A convolutional network for real-time 6-dof camera relocalization », in : *2015 IEEE International Conference on Computer Vision*, 2015, p. 2938–2946, DOI : 10.1109/ICCV.2015.336.

[52] Diederik P KINGMA et Jimmy BA, « Adam : A method for stochastic optimization », in : (2014), arXiv : 1412.6980.

[53] Sven KOSUB, « A note on the triangle inequality for the Jaccard distance », in : *Pattern Recognition Letters* 120 (2019), p. 36–38, DOI : 10.1016/j.patrec.2018.12.007.

[54] Philipp KRÄHENBÜHL et Vladlen KOLTUN, « Efficient inference in fully connected crfs with gaussian edge potentials », in : *Advances in neural information processing systems*, 2011, p. 109–117.

[55] Philipp KRÄHENBÜHL et Vladlen KOLTUN, « Parameter learning and convergent inference for dense random fields », in : *International Conference on Machine Learning*, 2013, p. 513–521.

[56] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON, « Imagenet classification with deep convolutional neural networks », in : *Advances in neural information processing systems*, 2012, p. 1097–1105.

[57] Sanjiv KUMAR et Martial HEBERT, « A hierarchical field framework for unified context-based classification », in : *2005 IEEE International Conference on Computer Vision*, 2005, DOI : 10.1109/ICCV.2005.9.

[58] Julien LANGLOIS et al., « 3D Orientation Estimation of Industrial Parts from 2D Images using Neural Networks. », in : *ICPRAM*, 2018, p. 409–416.

[59] Yann LECUN, Yoshua BENGIO et Geoffrey HINTON, « Deep learning », in : *nature* 521.*7553* (2015), p. 436, DOI : 10.1038/nature14539.

[60] Yann LECUN et al., « Gradient-based learning applied to document recognition », in : *Proceedings of the IEEE* 86.*11* (1998), p. 2278–2324, DOI : 10.1109/5.726791.

[61] Vincent LEPETIT, Francesc MORENO-NOGUER et Pascal FUA, « Epnp : An accurate o (n) solution to the pnp problem », in : *International journal of computer vision* 81.*2* (2009), p. 155, DOI : 10.1007/s11263-008-0152-6.

[62] Stefan LEUTENEGGER, Margarita CHLI et Roland SIEGWART, « BRISK : Binary robust invariant scalable keypoints », in : *2011 IEEE International Conference on Computer Vision*, Ieee, 2011, p. 2548–2555, DOI : 10.1109/ICCV.2011.6126542.

[63] Yi LI et al., « Deepim : Deep iterative matching for 6d pose estimation », in : *2018 European Conference on Computer Vision*, 2018, p. 683–698.

[64] Xiaodan LIANG et al., « Proposal-free network for instance-level object segmentation », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.*12* (2018), p. 2978–2991, DOI : 10.1109/TPAMI.2017.2775623.

[65] Wei LIU et al., « Ssd : Single shot multibox detector », in : *2016 European Conference on Computer Vision*, Springer, 2016, p. 21–37.

[66] Jonathan LONG, Evan SHELHAMER et Trevor DARRELL, « Fully convolutional networks for semantic segmentation », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 3431–3440, DOI : 10.1109/CVPR.2015.7298965.

[67] William E LORENSEN et Harvey E CLINE, « Marching cubes : A high resolution 3D surface construction algorithm », in : *ACM siggraph computer graphics*, t. 21, 4, ACM, 1987, p. 163–169, DOI : 10.1145/37401.37422.

[68] Kok-Lim LOW, « Linear least-squares optimization for point-to-plane icp surface registration », in : *Chapel Hill, University of North Carolina* 4.*10* (2004).

[69] David G LOWE, « Distinctive image features from scale-invariant keypoints », in : *International Journal of Computer Vision* 60.*2* (2004), p. 91–110, DOI : 10.1023/B:VISI.0000029664.99615.94.

[70] David G. LOWE, « Fitting parameterized three-dimensional models to images », in : *IEEE Transactions on Pattern Analysis & Machine Intelligence 5* (1991), p. 441–450, DOI : 10.1109/34.134043.

[71] Francesc MORENO-NOGUER, Vincent LEPETIT et Pascal FUA, « Pose priors for simultaneously solving alignment and correspondence », in : *2008 European Conference on Computer Vision*, Springer, 2008, p. 405–418.

[72] Marius MUJA et al., « Rein-a fast, robust, scalable recognition infrastructure », in : *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, p. 2939–2946, DOI : 10.1109/ICRA.2011.5980153.

[73] Yurii E NESTEROV, « A method for solving the convex programming problem with convergence rate O (1/kˆ 2) », in : *Dokl. akad. nauk Sssr*, t. 269, 1983, p. 543–547.

[74] T ONDA, H IGURA et M NIWAKAWA, « A handling system for randomly placed casting parts using plane fitting technique », in : *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, t. 3, IEEE, 1995, p. 435–440, DOI : 10.1109/IROS.1995.525921.

[75] Mustafa OZUYSAL, Pascal FUA et Vincent LEPETIT, « Fast keypoint recognition in ten lines of code », in : *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Ieee, 2007, p. 1–8, DOI : 10.1109/CVPR.2007.383123.

[76] Manuel G PENEDO et al., « Computer-aided diagnosis : a neural-network-based approach to lung nodule detection », in : *IEEE Transactions on Medical Imaging* 17.*6* (1998), p. 872–880, DOI : 10.1109/42.746620.

[77] Pedro HO PINHEIRO et Ronan COLLOBERT, *Recurrent convolutional neural networks for scene labeling*, rapp. tech., 2014.

[78] Mahdi RAD et Vincent LEPETIT, « BB8 : a scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth », in : *2017 IEEE International Conference on Computer Vision*, 2017, p. 3828–3836, DOI : 10.1109/ICCV.2017.413.

[79] Krisnawan RAHARDJA et Akio KOSAKA, « Vision-based bin-picking : Recognition and localization of multiple complex objects using simple visual cues », in : *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, t. 3, IEEE, 1996, p. 1448–1457, DOI : 10.1109/IROS.1996.569005.

[80] Mengye REN et Richard S ZEMEL, « End-to-end instance segmentation with recurrent attention », in : *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 6656–6664, DOI : 10.1109/CVPR.2017.39.

[81] Shaoqing REN et al., « Faster r-cnn : Towards real-time object detection with region proposal networks », in : *Advances in neural information processing systems*, 2015, p. 91–99.

[82] Lawrence G ROBERTS, « Machine perception of three-dimensional solids », thèse de doct., Massachusetts Institute of Technology, 1963.

[83] Bernardino ROMERA-PAREDES et Philip Hilaire Sean TORR, « Recurrent instance segmentation », in : *2016 European Conference on Computer Vision*, Springer, 2016, p. 312–329.

[84] Edward ROSTEN et Tom DRUMMOND, « Machine learning for high-speed corner detection », in : *2006 European Conference on Computer Vision*, Springer, 2006, p. 430–443.

[85] Ethan RUBLEE et al., « ORB : An efficient alternative to SIFT or SURF », in : *2011 IEEE International Conference on Computer Vision*, t. 11, 1, 2011, p. 2, DOI : 10.1109/ICCV.2011.6126544.

[86] Chris RUSSELL, Pushmeet KOHLI, Philip HS TORR et al., « Associative hierarchical crfs for object class image segmentation », in : *2009 IEEE International Conference on Computer Vision*, IEEE, 2009, p. 739–746, DOI : 10.1109/ICCV.2009.5459248.

[87] R.J. SCHILLING et H. LEE, *Engineering Analysis, a Vector Space Approach*, Wiley et Sons, 1988.

[88] Max SCHWARZ, Hannes SCHULZ et Sven BEHNKE, « RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features », in : *2015 IEEE International Conference on Robotics and Automation*, IEEE, 2015, p. 1329–1335, DOI : 10.1109/ICRA.2015.7139363.

[89] Eitan SHARON, Achi BRANDT et Ronen BASRI, « Fast multiscale image segmentation », in : *2000 IEEE Conference on Computer Vision and Pattern Recognition*, t. 1, IEEE, 2000, p. 70–77, DOI : 10.1109/CVPR.2000.855801.

[90] Eitan SHARON, Achi BRANDT et Ronen BASRI, « Segmentation and boundary detection using multiscale intensity measurements », in : *2001 IEEE Conference on Computer Vision and Pattern Recognition*, 2001, p. 469–476, DOI : 10.1109/CVPR.2001.990512.

[91] Jeffrey S SIMONOFF, *Smoothing methods in statistics*, Springer Science & Business Media, 2012, ISBN : 978-0387947167.

[92] Gregory G SLABAUGH, *Computing Euler angles from a rotation matrix*, 1999, URL : https://www.gregslabaugh.net/publications/euler.pdf (visité le 01/08/2018).

[93] Russell STEWART, Mykhaylo ANDRILUKA et Andrew Y NG, « End-to-end people detection in crowded scenes », in : *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, p. 2325–2333, DOI : 10.1109/CVPR.2016.255.

[94] Sreenivas R SUKUMAR et al., « Towards understanding what makes 3D objects appear simple or complex », in : *2008 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2008, p. 1–8, DOI : 10.1109/CVPRW.2008.4562975.

[95] Christian SZEGEDY et al., « Going deeper with convolutions », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 1–9, DOI : 10.1109/CVPR.2015.7298594.

[96] Alykhan TEJANI et al., « Latent-class hough forests for 3D object detection and pose estimation », in : *2014 European Conference on Computer Vision*, Springer, 2014, p. 462–477, DOI : 10.1007/978-3-319-10599-4_30.

[97] Dissaphong THACHASONGTHAM et al., « 3d object pose estimation using viewpoint generative learning », in : *Scandinavian Conference on Image Analysis*, Springer, 2013, p. 512–521, DOI : 10.1007/978-3-642-38886-6_48.

[98] Antonio TORRALBA, « Contextual influences on saliency », in : *Neurobiology of attention*, Elsevier, 2005, p. 586–592, DOI : 10.1016/B978-012375731-9/50100-2.

[99] Olga VEKSLER, « Image segmentation by nested cuts », in : *2000 IEEE Conference on Computer Vision and Pattern Recognition*, t. 1, IEEE, 2000, p. 339–344, DOI : 10.1109/CVPR.2000.855838.

[100] Tao WANG et al., « End-to-end text recognition with convolutional neural networks », in : *2012 IAPR International Conference on Pattern Recognition*, IEEE, 2012, p. 3304–3308.

[101] Paul WOHLHART et Vincent LEPETIT, « Learning descriptors for object recognition and 3d pose estimation », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p. 3109–3118, DOI : 10.1109/CVPR.2015.7298930.

[102] Yu XIANG et al., « Posecnn : A convolutional neural network for 6d object pose estimation in cluttered scenes », in : (2017), arXiv : 1711.00199.

[103] SHI XINGJIAN et al., « Convolutional LSTM network : A machine learning approach for precipitation nowcasting », in : *Advances in neural information processing systems*, 2015, p. 802–810.

[104] Fisher YU et Vladlen KOLTUN, « Multi-scale context aggregation by dilated convolutions », in : (2015), arXiv : 1511.07122.

[105] JS-C YUAN, « A general photogrammetric method for determining object position and orientation », in : *IEEE Transactions on Robotics and Automation* 5.*2* (1989), p. 129–142, DOI : 10.1109/70.88034.

[106] Shuai ZHENG et al., « Conditional random fields as recurrent neural networks », in : *2015 IEEE International Conference on Computer Vision*, 2015, p. 1529–1537, DOI : 10.1109/ICCV.2015.179.

[107] Jun-Yan ZHU et al., « Toward multimodal image-to-image translation », in : *Advances in neural information processing systems*, 2017, p. 465–476.

[108] Timo ZINSSER, Jochen SCHMIDT et Heinrich NIEMANN, « A refined ICP algorithm for robust 3-D correspondence estimation », in : *2003 IEEE International Conference on Image Processing*, t. 2, IEEE, 2003, p. II–695, DOI : 10.1109/ICIP.2003.1246775.

TABLE DES FIGURES

LISTE DES TABLEAUX

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

Titre : Vision industrielle et réseaux de neurones profonds

Mot clés : Apprentissage automatique, estimation de pose, données synthétiques

Resumé : Ces travaux de thèse présentent une méthode d'estimation de pose de pièces industrielles en vue de leur dévracage à partir d'un système mono-caméra 2D en utilisant une approche par apprentissage avec des réseaux profonds. Dans un premier temps, des réseaux de neurones assurent la segmentation d'un nombre prédéterminé de pièces dans la scène. En appliquant le masque binaire d'une pièce à l'image originale, un second réseau infère la profondeur locale de cet objet. En parallèle des coordonnées de la pièce dans l'image, cette profondeur est employée dans deux réseaux estimant à la fois l'orientation de l'objet sous la forme d'un quaternion et sa translation sur l'axe Z. Enfin, un module de recalage travaillant sur la rétro-projection de la profondeur et le modèle 3D de l'objet, permet d'affiner la pose prédite par les réseaux. Afin de pallier le manque de données réelles annotées dans un contexte industriel, un processus de création de données synthétiques est proposé. En effectuant des rendus aux multiples luminosités, la versatilité du jeu de données permet d'anticiper les différentes conditions hostiles d'exploitation du réseau dans un environnement de production.

Title : Industrial image processing and deep neural networks

Keywords : Machine learning, pose estimation, rendered data

Abstract : This work presents a pose estimation method from a RGB image of industrial parts placed in a bin. In a first time, neural networks are used to segment a certain number of parts in the scene. After applying an object mask to the original image, a second network is inferring the local depth of the part. Both the local pixel coordinates of the part and the local depth are used in two networks estimating the orientation of the object as a quaternion and its translation on the Z axis. Finally, a registration module working on the back-projected local depth and the 3D model of the part is refining the pose inferred from the previous networks. To deal with the lack of annotated real images in an industrial context, an data generation process is proposed. By using various light parameters, the dataset versatility allows to anticipate multiple challenging exploitation scenarios within an industrial environment.