Estimation de la pose 3D d'objets dans un environment industriel

Giorgia Pitteri

THÈSE - THESIS

présentée à - defended in

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par - by

Giorgia PITTERI

POUR OBTENIR LE GRADE DE - TO GET THE DEGREE OF

DOCTEUR - DOCTOR

INFORMATIQUE - COMPUTER SCIENCE

3D OBJECT POSE ESTIMATION IN INDUSTRIAL CONTEXT

Date de soutenance - defense date : 26 Novembre 2020

Devant la commission d'examen composée de - With a review board composed of :

Thierry	Chateau	Pr.	Université Clermont Auvergne	Rapporteur
Eric	Marchand	Pr.	Université de Rennes	Rapporteur
Pierrick	Coupé	DR CNRS	Université de Bordeaux	Président du jury
Frédéric	Devernay	Chercheur	Amazon	Examinateur
Slobodan	Ilic	Dr.	Siemens Corporate Technology	Examinateur
Aurélie	Bugeau	MDC HDR	Université de Bordeaux	Co-directrice de thèse
Vincent	Lepetit	Chercheur	École des ponts ParisTech	Co-directeur de thèse

– 2020 –

Titre

Estimation de la pose 3D d'objets dans un environment industriel

Résumé

La détection d'objets 3D et l'estimation de leur pose à partir d'images sont très importantes pour des tâches comme la robotique et la réalité augmentée et font l'objet d'intenses recherches depuis le début de la vision par ordinateur. D'importants progrès ont été réalisés récemment grâce au développement des méthodes basées sur l'apprentissage profond. Ce type d'approche fait néanmoins face à plusieurs obstacles majeurs qui se révèlent en milieu industriel, notamment la gestion des objets contenant des symétries et la généralisation à de nouveaux objets jamais vus par les réseaux lors de l'apprentissage.

Dans cette thèse, nous montrons d'abord le lien entre les symétries d'un objet 3D et son apparence dans les images de manière analytique expliquant pourquoi les objets symétriques représentent un défi. Nous proposons alors une solution efficace et simple qui repose sur la normalisation de la rotation de la pose. Cette approche est générale et peut être utilisée avec n'importe quel algorithme d'estimation de pose 3D.

Ensuite, nous abordons le deuxième défi: la généralisation aux objets jamais vus pendant l'apprentissage. De nombreuses méthodes récentes d'estimation de la pose 3D sont très efficaces mais leur succès peut être attribué à l'utilisation d'approches d'apprentissage automatique supervisé. Pour chaque nouvel objet, ces méthodes doivent être re-entrainées sur de nombreuses images différentes de cet objet, ces images n'étant pas toujours disponibles. Même si les méthodes de transfert de domaine permettent de réaliser l'entrainement sur des images synthétiques plutôt que sur des images réelles, ces sessions d'entrainement prennent du temps, et il est fortement souhaitable de les éviter dans la pratique.

Nous proposons deux méthodes pour traiter ce problème. La première méthode s'appuie uniquement sur la géométrie des objets et se concentre sur les objets avec des coins proéminents, ce qui est le cas pour un grand nombre d'objets industriels. Nous apprenons dans un premier temps à détecter les coins des objets de différentes formes dans les images et à prédire leurs poses 3D, en utilisant des images d'apprentissage d'un petit ensemble d'objets. Pour détecter un nouvel objet dans une image donnée, on identifie ses coins à partir de son modèle CAD, on détecte également les coins visibles sur l'image et on prédit leurs poses 3D. Nous introduisons ensuite un algorithme de type RANSAC qui détecte et estime de manière robuste et efficace la pose 3D de l'objet en faisant correspondre ses coins sur le modèle CAO avec leurs correspondants détectés dans l'image.

La deuxième méthode surmonte les limites de la première et ne nécessite pas que les objets aient des coins spécifiques et la sélection hors ligne des coins sur le modèle CAO. Elle combine l'apprentissage profond et la géométrie 3D, et repose sur une représentation réduite de la géométrie 3D locale pour faire correspondre les modèles CAO aux images d'entrée. Pour les points sur la surface des objets, cette représentation peut être calculée directement à partir du modèle CAO; pour les points de l'image, nous apprenons à la prédire à partir de l'image elle-même. Cela établit des correspondances entre les points 3D sur le modèle CAO et les points 2D des images. Cependant, beaucoup de ces correspondances sont ambiguës car de nombreux points peuvent avoir des géométries locales similaires. Nous utilisons alors Mask-RCNN sans l'information de la classe des objets pour détecter les nouveaux objets sans ré-entraîner le réseau et ainsi limiter drastiquement le nombre de correspondances possibles. La pose 3D est estimée à partir de ces correspondances discriminantes en utilisant un algorithme de type RANSAC.

Mots-clés

Vision artificielle, Détection d'objets 3D, Estimation de la pose d'objets 3D, Apprentissage profond

Title

3D Object Pose Estimation in Industrial Context

Abstract

3D object detection and pose estimation are of primary importance for tasks such as robotic manipulation, augmented reality and they have been the focus of intense research in recent years. Methods relying on depth data acquired by depth cameras are robust. Unfortunately, active depth sensors are power-hungry or sometimes it is not possible to use them. It is therefore often desirable to rely on color images. When training machine learning algorithms that aim to estimate objects' 6D poses from images, many challenges arise, especially in an industrial context that requires handling objects with symmetries and generalizing to unseen objects, that means objects never seen by the networks during training.

In this thesis, we first analyze the link between the symmetries of a 3D object and its appearance in images. Our analysis explains why symmetrical objects can be challenging when training machine learning algorithms to predict their 6D pose from images. We then propose an efficient and simple solution that relies on the normalization of the pose rotation. This approach is general and can be used with any 6D pose estimation algorithm.

Then, we address the second main challenge: the generalization to unseen objects. Many recent methods for 6D pose estimation are robust and accurate but their success can be attributed to supervised Machine Learning approaches. For each new object, these methods have to be retrained on many different images of this object, and those images are not always available.Even if domain transfer methods allow for training such methods with synthetic images instead of real ones—at least to some extent—such training sessions take time, and it is highly desirable to avoid them in practice. We propose two methods to handle this problem. The first method relies only on the objects' geometries and focuses on objects with prominent corners, which covers a large number of industrial objects. We first learn to detect object corners of various shapes in images and also to predict their 3D poses using training images of a small set of objects. To detect a new object in a given image, we first identify its corners from its CAD model; we also detect the corners visible in the image and predict their 3D poses. We then introduce a RANSAC-like algorithm that robustly and efficiently detects and estimates the object's 3D pose by matching its corners on the CAD model with their detected counterparts in the image. The second method overcomes the limitations of the first one as it does not require objects to have specific corners and the offline selection of the corners on the CAD model. It combines Deep Learning and 3D geometry and relies on an

embedding of the local 3D geometry to match the CAD models to the input images. For points at the objects' surface, this embedding can be computed directly from the CAD model; for image locations, we learn to predict it directly from the image itself. This establishes correspondences between 3D points on the CAD model and 2D locations of the input images. However, many of these correspondences are ambiguous as many points may have similar local geometries. We also show that we can use Mask-RCNN in a class-agnostic way to detect the new objects without retraining and thus drastically limit the number of possible correspondences. We can then robustly estimate a 3D pose from these discriminative correspondences using a RANSAC-like algorithm.

Keywords

Computer Vision, 3D Object Detection, 3D Pose Estimation, Deep Learning

Acknowledgements

At the end of my PhD journey, it is time for some acknowledgements since this thesis would not have been completed without the help and the support of many people to whom I am extremely grateful.

First and foremost, my sincere gratitude goes to my supervisor Vincent Lepetit, who gave me the opportunity to pursue my PhD and supported me during these years. He has been a great supervisor, always available for interesting discussions and encouraging words. I am really grateful to him for always believing in my work and in me even when I did not.

I would also like to thank my other supervisor Aurélie Bugeau. Even tough she was my supervisor only for the last year of the PhD, she has always been there for me, both from the scientific and personal side.

I am also grateful to Slobodan Ilic, for our interesting discussions on Skype and for letting me visit his research group at Siemens for a short period, during which I had the possibility to meet and discuss with his very talented PhD students. Undoubtedly, a special thanks of course goes to all the members of my committee: Erich Marchand and Thierry Chateau for the time and effort invested in the review of this thesis and Pierrick Coupé and Frédéric Devernay for accepting to be part of the jury.

I also want to thank all the people I met during these years both at LaBRI in Bordeaux and at Écoles des Ponts in Paris. Many thanks to my collegues and "favorite French guys", Michaël and Pierre. We soon became great friends and I could not have finished this thesis without their support. I will miss our coffee breaks, our adventures around the world and our long deadline nights.

I cannot forget my Italian friends who helped me get through the difficult times, and for all the emotional support, entertainment, and caring they provided even if we were far away and we saw each other only once a year. In particular I would like to thank Elisa, Francesco, Giulia, Alessandro and Enrica. A big thanks also to my cousin Teresa who always told me to "bite the bullet" and never give up.

Last but not least I would like to thank the three most important people in my life. Thanks to my parents, my role models, for their unconditional love, encouragement and support throughout my studies and experiences. This thesis is also for them. A big thanks to Nicola who first helped me find the courage to step outside my comfort zone and start the PhD and then supported me through these years. Living away and seeing each other once a month was not easy but we did it (and Easyjet can thank us for all the flight tickets we bought)! I could not have done this PhD without his support, his love and his patience.

Ai miei genitori
A Nicola

To my parents
To Nicola

Summary

List of Figures

List of Tables

Acronyms list

The following table lists all acronyms that are used hereafter in the document.

AR	Augmented Reality
NN	Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
RANSAC	Random Sample Consensus
PnP	Perspective-n-Point
LSE	Local Surface Embedding
RGB-D	RGB Color + Depth image
IoU	Intersection over Union
DoF	Degrees of Freedom
ICP	Iterative Closest Point

Résumé en Français

La vision par ordinateur est en train de faire des pas de géant pour faciliter notre vie quotidienne et la rendre beaucoup plus sûre. Les chercheurs et ingénieurs développent des robots capables d'effectuer des tâches difficiles, des voitures avec des systèmes de vision intégrés pour aider les gens à conduire prudemment, des algorithmes pour aider les médecins à diagnostiquer les maladies à partir d'images médicales, etc. La gamme d'applications qui bénéficient des progrès de la vision par ordinateur est vaste. Cependant, il existe encore de nombreux défis à relever lors du développement d'applications de vision par ordinateur. Par exemple, comprendre le monde en 3D à partir d'images ou de vidéos, c'est-à-dire identifier les objets présents dans une image, estimer où ils se trouvent en dans l'espace, à quelle distance ils sont de la caméra, etc. semble naturel et facile pour les humains, mais reste extrêmement difficile dans le cas général pour un système informatique.

Un premier grand pas vers une plus grande performance pour des tâches de vision par ordinateur a été fait grâce au développement des réseaux de neurones artificiels, des modèles inspirés des réseaux de neurones biologiques qui constituent le cerveau humain, en particulier avec l'avènement de puissants GPU qui ont rendu leur entraînement plusieurs fois plus rapide, beaucoup moins cher et plus précis. Les réseaux de neurones profonds ou les réseaux de neurones convolutifs sont des réseaux artificiels qui, contrairement aux techniques classiques de vision par ordinateur, peuvent apprendre à extraire automatiquement de meilleures caractéristiques d'image sans nécessiter d'étape de prétraitement. Cependant, ces réseaux nécessitent une grande quantité de données de formation pour apprendre à résoudre des tâches visuelles.

L'apprentissage profond a été appliqué aussi à l'estimation de la pose d'objets 3D à partir des images avec un grand succès. Ce problème a de nombreuses applications mais il reste difficile car il est affecté par les défis majeurs de la vision par ordinateur tels que les conditions d'éclairage, les occlusions et les encombrements, les changements de point de vue et la disponibilité limitée des quantités de données d'entraînement annotées.

Estimation de la pose 3D des objets

L'estimation de la pose d'un objet 3D est une tâche de vision par ordinateur qui déduit la pose 3D d'un objet dans une image ou une vidéo, ou de manière équivalente, le problème de la détermination de la position et de l'orientation d'une caméra par rapport à un objet donné. Cette pose 3D a 6 degrés de liberté : 3 pour la

translation 3D et 3 pour la rotation 3D qui peuvent être représentées par différentes représentations de pose comme une matrice de rotation 3D ou un quaternion.

Il est important de souligner que dans le problème d'estimation de pose 3D, les caméras sont souvent supposées calibrées. Cela signifie que les *paramètres intrinsèques de la caméra* (la distance focale, les points principaux de la caméra et la distorsion de l'objectif) sont connus tandis que les *paramètres extrinsèques de la caméra* (la translation et la rotation) doivent être estimés. Enfin, les méthodes d'estimation de pose 3D considèrent souvent une seule image. Cependant, les algorithmes d'estimation de pose 3D peuvent initialiser ou réinitialiser les méthodes de suivi d'objets 3D.

Ce problème a différentes variantes en fonction du type d'objet dont nous voulons prédire la pose, du type de capteur utilisé pour acquérir les données d'entrée et de l'objectif final de l'algorithme. Dans cette thèse nous traiterons l'estimation de la pose 3D des objets rigides quand plusieurs instances de plusieurs objets sont présent dans l'image. Nous utiliserons comme données d'entrée uniquement une simple image couleur, sans information de profondeur, ainsi qu'un modèle CAO des objets sans information de texture. Disposer de la profondeur rend le problème beaucoup plus facile, mais les capteurs ont des défauts : ils ont gourmands en énergie, et peuvent échouer sur les objets spéculaires.

Une autre façon possible de capturer des informations de profondeur serait d'utiliser plusieurs systèmes de caméras. Cependant, cela devient impossible en raison du coût et des efforts nécessaires pour mettre en place un système calibré et synchrone de plusieurs caméras, en particulier pour les applications de suivi d'objets 3D. Pour ces limitations des capteurs de profondeur, la communauté de recherche se concentre davantage sur le problème de l'estimation de pose 3D à partir d'images RGB uniquement et c'est ce que nous considérons également dans cette thèse.

Même si d'importants progrès ont été réalisés récemment grâce au développement des méthodes basées sur l'apprentissage profond, ce type d'approche fait néanmoins face à plusieurs obstacles majeurs qui se révèlent en milieu industriel, notamment la gestion des objets contenant des symétries et la généralisation à de nouveaux objets jamais vus par les réseaux lors de l'apprentissage.

Gestion des objets symétriques

De nombreux objets de notre vie quotidienne ou dans des contextes industriels présentent des symétries, ou du moins des « quasi-symétries » quand un petit détail empêche l'objet d'avoir une symétrie parfaite. Ces symétries créent des ambiguïtés lorsque l'on cherche à estimer la pose 3D de l'objet à partir d'images, comme nous l'expliquerons. Pendant longtemps, ce problème n'a pas été pris en compte par les chercheurs et le problème de l'estimation de la pose d'objets 3D n'a été résolu que sur des scénarios simples. Pour mieux comprendre le problème posé par les symétries d'un objet, nous pouvons considérer la figure 1. L'objet bleu a une symétrie de rota-

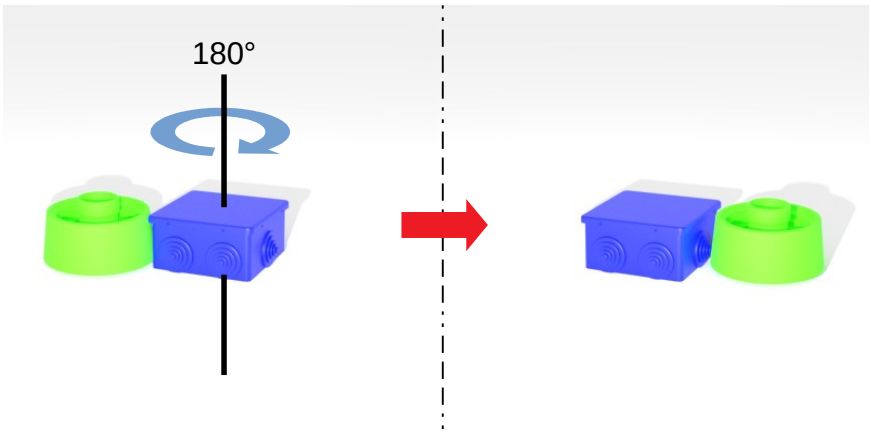


Figure 1: Deux vues de la même scène avant et après rotation de 180° autour de l'axe vertical de l'objet bleu.

tion autour de l'axe vertical. Si nous appliquons une rotation de 180° autour cet axe, cet objet a la même apparence. Plus généralement, lorsqu'un objet O a une certaine symétrie, il existe un ou plusieurs mouvements rigides tels que, si nous appliquons ces mouvements rigides à la pose de l'objet, l'apparence de l'objet est préservée. En autres termes, deux images d'un objet symétrique peuvent être identiques mais pas correspondent à la même pose. Si on considère une image $\boldsymbol{I}$ d'un objet O sous la pose $\boldsymbol{p}$, il n'existe pas de fonction $\mathcal{F}$ telle que

$$\mathcal{F}(\boldsymbol{I}) = \boldsymbol{p}\,, \tag{1}$$

et il est donc impossible de l'apprendre avec un réseau de neurones. Par exemple, si un réseaux des neurones est entrainé pour prédire la pose avec une fonction de coût carré entre la vérité terrain et la prédiction, ce réseau apprendrait à prédire la moyenne des pose possibles. Ce résultat n'aurait pas de sens.

Nous expliquons dans le chapitre 5 le lien entre les symétries d'un objet 3D et ses apparences dans les images en détail et nous montrons pourquoi il ne suffit pas de restreindre les poses dans certaines intervalles.

Nous proposons en suite une solution simple et analytique pour gérer ce problème basée sur la normalisation de la rotation de la pose problème. Cette solution est générale et peut être introduite dans n'importe quel algorithme d'estimation de pose 3D.

Généralisation à de nouveaux objets

Nous avons déjà fait remarqué que l'avènement de l'Apprentissage Profond a amélioré les performances de l'estimation de pose 3D à partir des images. Cependant, ces approches permettent d'obtenir d'excellentes performances dans l'estimation de

la pose d'objet 3D, au moins lorsque suffisamment d'images d'entraînement sont disponibles sous différentes poses. Un modèle entraîné sur certains objets ne fonctionnera pas aussi bien sur d'autres objets, qui ne faisaient pas partie des échantillons d'apprentissage. C'est une grosse limitation surtout dans les contextes industriels où les nouveaux objets sont courants et il n'est pas facile d'en avoir beaucoup d'images de formation.

Même si les méthodes de transfert de domaine permettent d'entrainer les réseaux profonds avec des images synthétiques au lieu de vraies au moins dans une certaine mesure, ces sessions de'entrainement prennent du temps, et il est hautement souhaitable de les éviter dans la pratique, en particulier dans les contextes industriels. Nous proposons dans cette thèse une méthode capable de prédire la pose 3D d'un objet jamais vu par le réseau profond pendant son entrainement, sans avoir besoin des plusieurs images de ce nouveau objet et ni de nouvelle sessions d'entrainement.

Utilisation de coins 3D générique

Dans le chapitre 6, nous proposons une première approche pour estimer la pose 3D de nouveaux objets. Cette méthode ne nécessite pas d'apprentissage supplémentaire ni d'images d'entrainement pour les nouveaux objets et nous considérons un scénario où des modèles CAO pour les nouveaux objets existent, mais pas nécessairement des images d'entrainement. C'est souvent le cas dans les contexte industriels, où un objet est construit à partir de son modèle CAO. Nous nous appuyons sur les coins que nous apprenons pour détecter et estimer les poses 3D lors d'une étape hors ligne. Notre approche se concentre sur les objets industriels qui ont souvent des coins comme partie commune.

Détecter ces coins et déterminer leurs poses 3D est la base de notre approche. Nous utilisons l'apprentissage profond et nous entrainons l'architecture Faster R-CNN sur un petit ensemble d'objets pour détecter les coins et prédire leurs poses 3D. Nous utilisons la représentation des poses 3D introduite par Crivellaro et al. (2018): La pose 3D d'un coin est prédite sous la forme d'un ensemble de reprojections 2D de 3D points virtuels. Cette représentation est pratique pour notre objectif car plusieurs coins peuvent être facilement combiné pour calculer la pose de l'objet. En plus, cette approche est robuste aux occlusions. Grâce à cette représentation de pose, il suffit d'avoir 1 ou 2 coins pour prédire la pose de l'objet.

Cependant, nous devons relever un défi qui se pose avec les coins, et que a été ignoré dans Crivellaro et al. (2018): en raison de ses symétries, la pose 3D d'un coin est souvent ambigu et défini uniquement par un ensemble de rotations rigides. Nous, par conséquent, introduisez un algorithme robuste et efficace qui considère toutes les possibles poses 3D des coins détectés, pour enfin estimer les poses 3D de les nouveaux objets.

Cette méthode a, toutefois, des limitations : elle se limite seulement a des objets avec des coins et elle nécessite qu'un expert sélectionne hors ligne les coins sur les

modèles CAO des nouveaux objets. Pour supprimer cette exigence et rendre notre solution plus générale, nous avons introduit une nouvelle méthode.

Utilisation de descripteurs locaux de la surface des objets

Dans le chapitre 7, nous considérons toujours le même probleme mais proposons une nouvelle méthode basée sur des correspondances denses entre l'image couleur d'entrée et le modèle CAO, au lieu de correspondances de coin.

Étant donné que nous voullons une approche capable de fonctionner dans le contexte industriel, nous voulons gérer les objets symétriques, sans texture, ambigus ou partiellement occultés. De plus, nous ne voulons plus sélectionner manuellement les points 3D virtuels sur les coins. Notre nouvelle approche combine l'apprentissage automatique et la géométrie 3D : comme des travaux précédents (Brachmann et al., 2016; Zakharov et al., 2019; Park et al., 2019), nous établissons des correspondances denses entre les pixels de l'image et les points 3D sur la modèle CAO, car ils ont montré que cela donne des poses précises. Cependant, il existe une différence fondamentale entre ces travaux et le nôtre : ils prédisent les cordonnes 3D des points de l'objet car ils peuvent entraîner un modèle d'apprentissage automatique à l'avance pour prédire ces coordonnées 3D des pixels dans une image donnée. Dans notre cas, nous souhaitons éviter toute les phases d'apprentissage de nouveaux objets et donc pour nost predire els cordonnes 3D des points de l'objet est impossible.

Nous utilisons donc sur une stratégie différente et nous introduisons un descripteur capturant la géométrie locale des points 3D qui se trouvant sur la surface de l'objet. Nous entrainons un réseau profond à prédire ces descripteur par pixel en utilisant un petit nombre d'objets, et nous utilisons le modèle entrainée sur les images de nouveaux objets. En mettant en correspondance ces descripteur prédits sur l'image avec les descripteur calculés pour les points 3D sur la surface de l'objet, nous obtenons des correspondances 2D-3D à partir desquelles nous estimons la pose 3D de l'objet avec une algorithme de RANSAC et d'un solveur PnP.

Cette approche est conceptuellement simple, robuste aux occlusions et fournit une pose 3D précise. Cependant, pour réussir, une attention particulière est nécessaire. Premièrement, les descripteurs doivent être invariants de rotation. Deuxièmement, à cause des symétries et de cette invariance de rotation, de nombreuses correspondances entre pixels et points 3D sont possibles *a priori* et la complexité de trouver un ensemble de correspondances correctes peut devenir exponentielle. Nous contrôlons cette complexité de deux manières. Nous nous concentrons sur les descripteurs d'image qui sont les plus discriminants car ils ont moins de correspondances potentielles. Nous observons également que l'architecture Mask R-CNN (He et al., 2017) peut prédire les masques de nouveaux objets lorsqu'ils sont entraînés sans aucune information de classe, et ainsi segmenter de nouveaux objets sans réentraînement. Nous l'utilisons pour contraindre les ensembles de correspondances dans RANSAC à se trouver sur le même masque, et ainsi réduire considérablement le nombre d'échantillons à considérer dans RANSAC.

Chapter 1

Introduction

Table of contents

In 1966 a famous computer scientist, Marvin Minsky, instructed a graduate student to connect a camera to a computer and have it describe what it sees. More than 50 years later, we are still working on it.

Computer vision is taking giant steps nowadays, facilitating our daily life and making it much safer. Researchers and engineers have developed robots able to perform complicated tasks, cars with integrated vision systems to help people drive carefully, algorithms to help doctors to diagnose diseases from medical images, to highlight only a few examples. The range of applications that benefit from advances in computer vision is huge. However, there are many challenges to face when developing computer vision applications. For example, understanding the real 3D world from images or videos, which objects are in the image, how they are located in 3D, how far they are from the camera seems natural and easy for humans but it is not so easy for computers systems. Early approaches to solving visual tasks consisted of representing the objects with patterns, like edges or corners, and then implemented methods to search for these features in the images. These features need to describe each object. However, object appearance can change a lot depending on the viewpoint, the lighting conditions and occlusions. Therefore, there is the need to represent each object from all possible views and under all imaging conditions. This process is highly time-consuming, and researchers have soon started studying how to automatically extract those features.

Machine learning helped researchers make computers learn these patterns to solve visual tasks by considering a lot of examples, called training data, without the necessity to be preprogrammed. Early machine learning-based approaches require a first step of preprocessing where image features are extracted and a second learning step where these features are fed into a classifier, such as a support vector machine, which makes decisions about how to solve the visual task.

A first big step towards higher accuracy in computer vision tasks was made by Neural Networks (NNs), models inspired by the biological neural networks that constitute the human brain, especially with the advent of powerful GPUs that made training them multiple times faster, much cheaper and more accurate. A subcategory of NNs is the Deep Neural Networks (DNNs) or Convolutional Neural Networks (CNNs). Unlike classical computer vision techniques or NNs, DNNs can learn to extract better image features automatically without requiring any preprocessing step. By learning features from the entire input image, DNNs do not suffer from any loss of information and learn better features than the hardcoded ones. However, these networks require a large amount of training data to solve visual tasks. The concept of DNNs has been around for some time but it was not considered a successful and reliable tool by the research community. This conception changed in 2012, when the Deep Neural Network (DNN) *AlexNet*, developed by Krizhevsky et al. (2012), won the image recognition ImageNet challenge by a large margin. Since that, more and more computer scientists are using CNNs to successfully solve many tasks of

computer vision, making it one of the most remarkable breakthroughs over the last decade. Thanks to computers with more computational power and the availability of much more training data such as videos and pictures, essential for deep machine learning, that was much more possible.

One field of computer vision that benefited a lot of the advent of Deep Learning (DL) is 3D object pose estimation. This problem has plenty of applications and it is interesting as well as difficult because it is affected by the major challenges of computer vision such as lighting conditions, occlusions and clutters, viewpoint changes and limited annotated training data as we will see in Sections 1.2 and 1.3.

1.1 3D Object Pose Estimation

3D object pose estimation is a computer vision task that infers the 3D pose of an object in an image or video, or equivalently, the problem of determining the position and orientation of a camera relative to a given object. This 3D pose has 6 degrees of freedom (DoF): 3 for the 3D translation and 3 for the 3D rotation that can be represented by different pose representations such as a 3D rotation matrix or a quaternion. Because of these 6 DoF the terms 3D pose and 6D pose refer to the same concept and the research community often refers to this problem also as 6D pose estimation.

It is important to underline that 3D pose estimation differs from 2D object pose estimation (or 2D detection) where the objective is to localize the object in 2D space relative to an image or video frame. Indeed, it is much more challenging since it transforms an object in a 2D image into a 3D object by adding a z-dimension to the prediction. It is important to underline that in the 3D pose estimation problem, the cameras are supposed to be calibrated. It means that the *camera intrinsic parameters* (the focal lengths, the camera principal points and the lens distortion) are known while the *camera extrinsic parameters* (the translation and the rotation components) have to be predicted. Finally, the 3D pose estimation problem is intended from a single frame. However, 3D pose estimation algorithms can initialize or re-initialize 3D object tracking methods.

3D pose estimation has different variants based on the type of the object of which we want to predict the pose, the type of sensor used to acquire the input data and the final goal of the algorithm.

Rigid vs non-rigid objects: a non-rigid object is made of moving parts or deformation parameters. Their positions and shapes have many more DoF and specific pose representations have been developed. On the contrary, a rigid object is fixed and, in this case, a rigid body transformation is a correct representation of an object's pose.

RGB cameras vs depth-sensors: 3D pose estimation methods can be classified into two big categories depending on if they exploit both the color and the depth information or only the color.

Relying on depth data make algorithms more robust. Depth information contains many insights about the shape of the object and it is independent of lighting conditions that, as we will see in Section 1.3, is one of the major challenges in computer vision tasks from RGB images. Furthermore, discontinuities in depth are a strong indicator of object boundaries which make algorithms able to detect objects in the image even with a cluttered background and the depth values constraints the z-component of the unknown object translation. When depth is available, the initial 3D pose estimate can be coarse since it can be refined later by aligning a 3D model of the object with the local depth at the estimated position, like for example with the *Iterative Closest Point* (ICP) algorithm (Chen and Medioni (1991), Zhang (2017)). On the contrary, methods relying on RGB inputs are less robust since they are affected by illumination changes and clutter background since object boundaries are less prominent. Refinement algorithms that exploit the depth information cannot be used here and it means the pose estimates should be very accurate. This high accuracy is very complicated to obtain especially the translation value along the z-axis because the distance of the object from the camera changes only slightly in the image compared to the distance in 3D.

At a first glance, it appears that using RGB data with depth information is the right choice to make. Unfortunately, active depth sensors are power-hungry or sometimes it is not possible to use them. We can think of Time-of-Flight (ToF) or RGB-D cameras for example. The former type of camera uses infrared light to capture depth information in this way: the sensor emits a light signal, which hits the object and returns to the sensor. The time it takes to bounce back is then measured and provides depth-mapping values. The latter adds per-pixel depth information to the RGB image by first projecting Infra-Red light dots on the scene, and capturing images in the Infra-Red range. It then computes the depth by measuring the displacement of the dots. Both these cameras lack performance in the outdoor scenes because of the interference of sunlight. Another possible way to capture depth information would be to use multiple camera systems to solve some ambiguities that raise when using RGB inputs. However, this becomes impractical due to the cost and effort in setting up a calibrated and synchronous system of multiple cameras, especially for 3D tracking applications. The depth information can be also provided by LiDAR sensors. However LiDAR sensors can not be easily adapted in industrial context and have mostly been used in outdoor localization, for example for mobile robotics (Delobel et al. (2015)) and autonomous driving (Geiger et al. (2012)).

For these limitations of depth sensors, the research community is focusing more on the problem of 3D pose estimation from RGB images only and it is what we consider in this thesis as well.

Single object vs Multi Objects pose estimation: In the single object pose estimation, the objective is to predict the 3D pose of a unique object instance, where an object instance means a specific object of the real word different from others in terms of shape and material. This is different from the concept of object class or category which includes all objects of a particular class. In this task, the object is often considered visible in the image. In multi objects pose estimation the DNNs are trained with multiple objects simultaneously and, given an image, the system should estimate the poses of all the objects in it. This is more challenging since some objects can be not visible and/or occluded by other objects and an object detector is often required and added to the pipeline to know which objects are visible. A particular case of multi objects pose estimation is multi instances pose estimation. In this case, multiple instances of the same object can appear in the scene.

1.1.1 Problem statement

Early methods for 3D pose estimation focus on simple scenarios where a single object lies on a uniformly planar surface that is not en entirely realistic scenario. To bridge the gap between pure research and industrial applications, more difficult scenarios with high clutter, occlusion and all the limitations that real applications imply need to be considered.

For this reason, in this thesis, we consider the most difficult scenario, and we focus on the problem of multiple object instances pose estimation in a single RGB image, also defined as Vivo (**v**ary number of **i**nstances of a **v**arying number of **o**bjects) task in the BOP challenge [1] (Hodaň et al., 2018). The methods we will discuss in this thesis exploit the power of Deep Learning and so they belong to the learning-based methods category. According to the BOP challenge instructions, the general set-up for these approaches is:

Firstly, at training time, a method is given a training set $T = \{T_o\}_{o=1}^{n}$, where o is an object identifier. Training data T_o may have different forms, such as a 3D mesh model of the object or a set of RGB-D images showing object instances in known 3D poses. Secondly, at test time, the method is provided with the image I and a list $L = [(o_1, n_1), ..., (o_m, n_m)]$, where n_i is the number of instances of object o_i present in the image I. The goal is to estimate the 3D pose of all the instances of object o_i visible enough in image I. Each estimate is given by a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$.

In this set-up, DNNs can successfully work with objects that have previously been seen during the training phase. If this does not happen, DNNs cannot predict the pose of a new object. In an industrial context, this is a substantial limitation since often happens the need to estimate the 3D pose of new objects when only their CAD models are available but not training images. On the contrary, a learning-based

[1] https://bop.felk.cvut.cz/home/

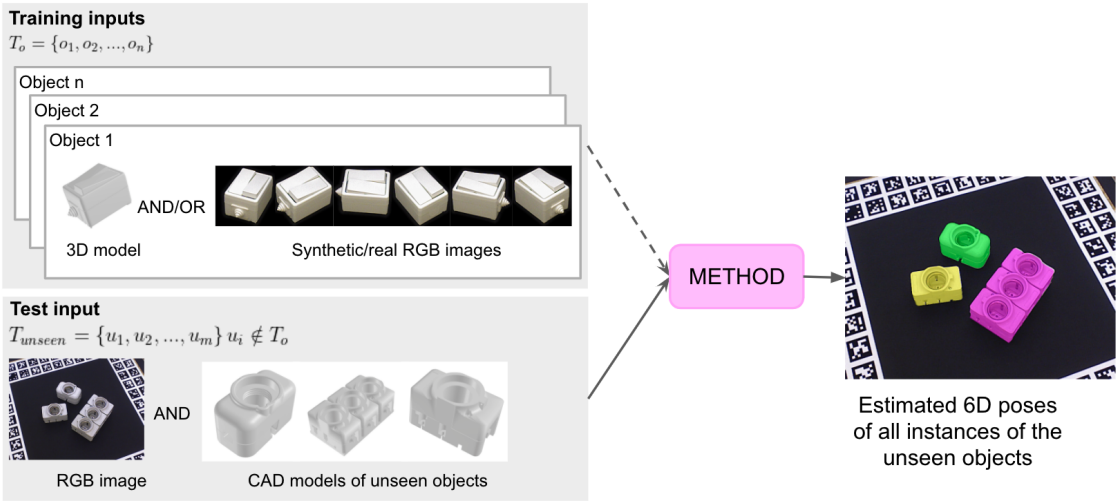


Figure 1.1: Representation of our extended problem statement.

3D pose estimation algorithm able to work with unseen objects would be a powerful tool in the industry since the need for a large number of images for training learning-based methods make them often unsuitable for industrial applications (Section 1.2).

This problem is not solved yet, and only recently, the research community realized its importance and started tackling it. We, therefore, take this direction and in this thesis, in Chapters 6 and 7, we will propose two approaches towards a possible solution for 3D pose estimation of unseen objects problem. The set-up for these methods, shown in Figure 1.1, is as follows:

The training set $T = \{T_o\}_{o=1}^{n}$ is still used at training time but at test time the method is provided with another list $L_{unseen} = [(u_1, n_1), ..., (u_m, n_m)]$, where $u_i \notin T$ is an object present in the image I that has not been seen during training and where n_i is the number of instances of object u_i in the image I.

To summarize, our objective is to solve the 3D pose estimation of objects that are typical in industrial contexts, which means considering all the challenges it implies (as we will discuss in Section 1.3) without doing any simplification. Our methods should deal with:

- Symmetrical, ambiguous, texture-less objects that are often similar to others.
- Unavailability depth sensors. The input of the algorithms should be an RGB image.
- Images have a cluttered background and objects in the image can be occluded by others.
- Multiple instances of the same object can be in the image.

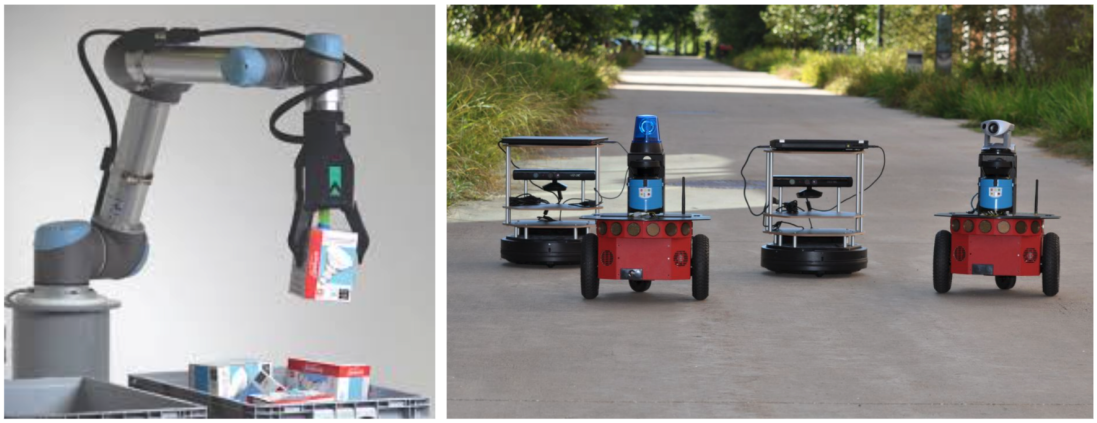


Figure 1.2: Robotics applications. **Left:** A robotic arm grasping an object [a]. **Right:** Robotic platforms navigating in an outdoor environment [b].

[a] https://www.therobotreport.com/grasp-sight-picking-evolve-robots/
[b] https://robohub.org/robot-teams-create-supply-chain-to-deliver-energy-to-explorer-robots/

1.2 Applications

The recovery of the 3D pose of an object is an important problem in the computer vision field since it has great impact to many rapidly evolving technology areas. Here we will detail these applications.

Robotics: Two applications in robotics can benefit from the advances of 3D object pose estimation: object grasping, for example in assembly lines or automated warehouses and navigation (Figure 1.2). If a robot wants to grasp an object, it must detect it and estimate its 3D pose. Similarly, a mobile robot needs a representation of the environment and a representation of its belief regarding its pose in this environment. For this reason, the robots must detect the objects around themselves and estimate their 3D poses. Once the objects are localized in the scene, the robot could navigate to the objects' positions in the 3D space avoiding obstacles.

Augmented Reality: Augmented reality (AR) allows us to seamlessly insert virtual objects in an image sequence. To accomplish this goal, synthetic elements must be rendered and aligned in the scene in an accurate and visually acceptable way. The solution to this problem can be related to a pose estimation or, equivalently, a camera localization process. Two examples of AR are shown in Figure 1.3.

Autonomous driving: A self-driving car benefits from advances in 3D object pose estimation since it requires accurate object detection and pose estimation for

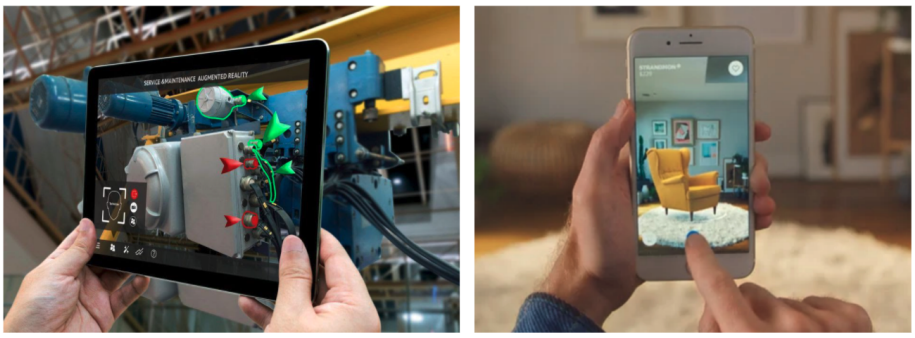


Figure 1.3: Augmented reality applications. **Left:** AR can provide operators with visual instructions and enhance remote support and communications[a]. **Right:** A mobile app using AR technology to scan a room and design the space by placing objects in the digital image of the room to create a new environment with the new products [b].

[a] https://www.i-scoop.eu/industry-40-virtual-reality-vr-augmented-reality-ar-trends/
[b] https://www.ikea.com/au/en/customer-service/mobile-apps

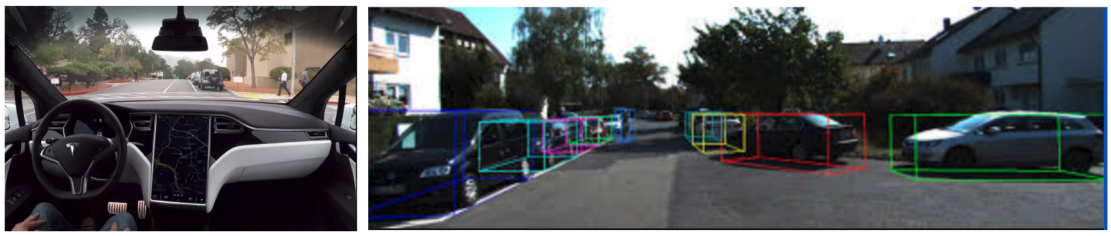


Figure 1.4: Example of an autonomous navigation system [a] that exploits a 3D pose estimation algorithm to detect the cars in the streets and their 3D pose (Geiger et al., 2012).

[a] https://www.tesla.com/videos/autopilot-self-driving-hardware-neighborhood-short

a vehicle to navigate in the 3D space without human assistance avoiding collisions with pedestrians, cyclists and cars. Figure 1.4 shows an example of a self-driving car and the output of a 3D object pose estimation algorithm applied to cars. However, autonomous driving is not only limited to cars, and it can be used for different types of vehicles such as trains and drones as well.

1.3 Challenges

Many challenges need to be tackled in practice when estimating the 3D poses of objects from RGB images and that is the reason why the 3D pose estimation problem is not yet solved, except for simple scenarios. We detail the most critical challenges below:

Illumination changes, Figure 1.5(a): The object appearance can largely change under different illuminating conditions. Illumination parameters can change the overall magnitude of light intensity reflected from an object, as well as the shadows visible in an image. Also, the color of the light can produce changes in the image. Methods for 3D object pose estimation that rely on the appearance of the objects are highly affected by this problem. Indeed to be robust to light changes these algorithms need to be trained with a large set of images of the objects under varying illumination and these images are not always available or they are not easy to create synthetically.

Viewpoint changes: Appearance-based methods also suffer from viewpoint changes since object appearance varies when the object is viewed from different angles. To be robust to these changes these methods need to see objects under a lot of viewpoints during the training phase.

Clutter background and Occlusions, Figure 1.5(b): 3D pose estimation algorithms tend to fail when there are cluttered backgrounds and objects partially occlude each other. This happens mostly because of two factors: the first is that these methods often rely on components representing the entire objects globally and second because random objects in the background might be similar or have similar parts to the target objects and act as distractors. To deal with the problem of occlusions some methods look only locally at some parts of the objects while others try to train DNNs on images with a lot of occluded objects.

Texture-less objects, Figure 1.5(c)(d)(e): Early approaches to pose estimation relied on the presence of texture or pattern on the objects' surfaces because stable features can be detected and matched relatively easily and efficiently on patterns. These approaches fail on texture-less objects since they lack discriminative parts. Dealing with these objects is essential since they are very common in the industry.

Ambiguity, Figure 1.5(e): Repetitive patterns, multiple instances of the same object, symmetrical or quasi-symmetrical objects are very common in practice especially in industrial contexts. These symmetries create ambiguities when aiming to estimate the 3D pose of the object from images with machine learning algorithms that confuse the networks during the training phase preventing them from learning. We will address this problem in Chapter 5.

Training data: To train supervised methods for 3D object pose estimation a lot of ground truth 3D object pose data are required. Annotating 3D poses of the objects in the images requires a great effort and the publicly available datasets for this task are limited. On the contrary, to solve the challenges described above these

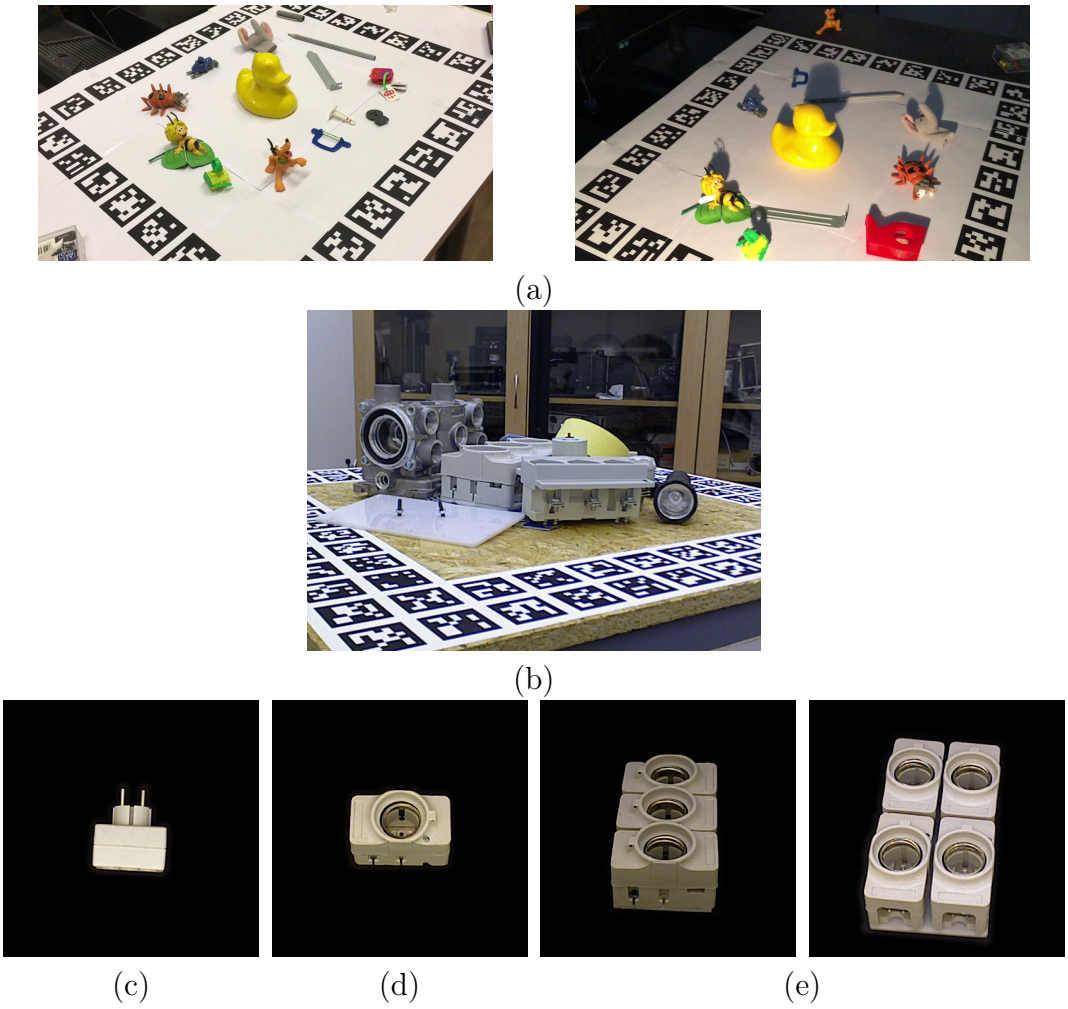


Figure 1.5: Some challenges that occur when estimating the 3D poses of the objects from RGB images. (a) Different lighting conditions can induce strong variations in appearance of objects. The images have been taken from the ALCN dataset (Rad et al., 2017). (b) Partial occlusions. (c) (d) (e) Ambiguities that arise because of symmetries (c) or quasi-symmetries (d) of the objects, or because some objects (e) are composed of smaller objects (d) repeated multiple times. These images have been taken from the T-LESS dataset (Hodaň et al., 2017).

algorithms need to see a lot of variations in light, viewpoints and occlusions during the training. One common approach to gain robustness against this problem is to augment the training set and this can be done online during training by changing the brightness of the image, the color, simulating occlusions or adding noise. Unfortunately, this is often not enough. To simulate different light sources, high occlusions or heavy cluttered random background like real scenes research scientists often need to generate synthetic training images, and that leads to another

challenge: the domain gap between synthetic training images and real test images makes methods trained on synthetic images do not generalize well. Researchers are doing a significant effort even in this direction to overcome this limitation.

Scalability: AR applications and robotics should work with a large set of objects. The latest 3D pose estimation datasets have up to 33 objects. A question then arises: how many objects can the deep learning-based algorithms handle? Adding more and more objects, are these algorithms still going to perform well? That is the problem of scalability.

Generalization to unseen objects: Since learning-based methods are trained on images of specific objects, they are not able to generalize to new objects at test time. This means that every time a new object appears, these methods need to be retrained on many different images of the new objects. Unfortunately, these images are not always available, especially in the industry. A possible solution would be training these methods on synthetic images but sometimes it is highly desirable to avoid training sessions since they take time. Time is precious in industry and it is rather preferred to have a method able to generalize to unseen objects without any retraining. This aspect is also implicitly linked to the problem of scalability.
Generalization to unseen objects is the main challenge that the work described in this thesis deal with, and we will discuss in Chapters 6 and 7 two approaches to solve this core problem.

1.4 Contributions

This thesis covers the following peer-reviewed accepted publications:

- *On Object Symmetries and 3D Pose Estimation from Images*
  Giorgia Pitteri, Michaël Ramamonjisoa, Slobodan Ilic and Vincent Lepetit
  International Conference on 3D Vision, 2019

- *CorNet: Generic 3D Corners for 3D Pose Estimation of New Objects without Retraining*
  Giorgia Pitteri, Slobodan Ilic and Vincent Lepetit
  International Conference on Computer Vision Workshops, 2019

- *3D Object Detection and Pose Estimation of Unseen Objects in Color Images with Local Surface Embeddings*
  Giorgia Pitteri, Aurélie Bugeau, Slobodan Ilic and Vincent Lepetit
  Asian Conference on Computer Vision, 2020

1.5 Outline

The remainder of this thesis is structured as follows:

- In Chapter 2, we introduce some notions and background on computer vision and deep learning necessary to understand the thesis.

- In Chapter 3 we present an overview of the state-of-the-art methods in the field of 3D object detection and pose estimation. In particular, we focus on recent works that handle symmetrical objects and unseen objects.

- In Chapter 4, we detail the open-source datasets most used to train and evaluate the 3D pose estimation methods. We will explain which dataset we chose to evaluate our methods and all the reasons behind our choice. We also present our synthetic photo-realistic dataset.

- Chapter 5 presents a general approach based on the normalization of the pose rotation to handle symmetrical objects that can be integrated into any 3D pose estimation pipelines.

- Chapter 6 presents our first approach to estimate the 3D pose of unseen objects by learning to detect corners in the image and estimate the 3D poses of the corners during an offline stage.

- Chapter 7 presents our second approach to estimate the 3D pose of unseen objects by using an embedding of local 3D geometry to match the CAD model and 2D locations of the input images.

- Finally, in Chapter 8, we conclude this thesis discussing some limitations and possible future directions to solve them.

Chapter 2

Background on Computer Vision

Table of contents

When discussing 3D object pose estimation, it is essential to review the relationship of the 3D world and its depiction in 2D images, that means how a camera projects 3D structures onto the image plane. In this chapter, we define the necessary geometry background.

2.1 Perspective Camera Model

A camera provides a mapping between the 3D world and a 2D image. Different approaches to model the transformation of a 3D point in world coordinates to a 2D point in image coordinates exist. In this thesis, we use the perspective projection model by assuming a pinhole model of the camera. In this model, the image plane is located between the camera and the scene plane, and the camera axis, called the focal axis, is orthogonal to the image plane. The camera can be described in matrix form as:

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \tag{2.1}$$

where $\mathbf{K}$ is a matrix of the intrinsic parameters, $\mathbf{R} \in SO(3)$ is the camera rotation matrix, $\mathbf{t} \in \mathbb{R}_3$ is the camera translation. The intrinsic matrix $\mathbf{K}$ is described as:

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where $\mathbf{f} = (f_u, f_v)^T$ is the focal length of the camera in pixels and $\mathbf{c} = (c_u, c_v)^T$ is the principal point offset.

Given a 3D point in the world $P = [X, Y, Z, 1]^T$, in a homogeneous coordinates system, the coordinates of the pixel in the image are obtained as:

$$\mathbf{x} = \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

By dividing the first two coordinates of $\mathbf{x} = [\lambda u, \lambda v, \lambda]^T$ by the third coordinates, it is possible to obtain the pixel coordinates (u, v).

2.2 Perspective-n-Point

The definition of Perspective-n-Point (PnP) is as follows:

Given a set of n 3D points in the world coordinate system and their corresponding 2D projections in the image, assuming the camera intrinsic parameters and the camera distortion coefficients are known, find the 3D rotation and 3D translation of

the object with respect to the camera.

Because a rigid body transformation has 6 DoF, it is necessary to get information of at least three pairs of corresponding points to solve a PnP problem. Principally to avoid the impact of noise, it is better if more points are observed. It is also important that these points are further away from being coplanar. Most of the available solutions are applicable for the typical case in which $n > 3$ but solutions applicable for $n = 3$ exist. Most of these methods assume the correspondences are noisy-free. We will discuss in the next section a method to get rid of outlier points.

P3P: If we observed 3 points, we end up with a system of polynomial equations that generally has 8 solutions. 4 solutions are behind the camera. Only 4 solutions are physically possible, and they can be obtained via computation algebra methods classified into iterative, non-iterative, linear, and non-linear ones (R. Haralick and Nolle, 1991).

EPnP: In most cases, it is better to have four or more correspondences problem in PnP. Redundancy in a larger set of points decreases the interference of noise. However, having a large set of points increase the complexity of the procedure to get the rigid body transformation, and it is necessary to increase the efficiency of the PnP algorithms in order to run in real-time. EPnP, proposed by Lepetit et al. (2009), achieves a complexity of $\mathcal{O}(n)$ with a reasonable accuracy in the transformation estimation. The central idea of this method is to express the n 3D points as a weighted sum of four non-coplanar virtual points.

Iterative PnP: The Iterative PnP method is based on a Levenberg-Marquardt optimization. In this case, the function finds such a pose that minimizes the reprojection error, that is the sum of squared distances between the observed projections in the image and the observed points in the object coordinates system.

Choosing the method depends on the type of applications. For example, EPnP and P3P are faster than Iterative PnP at finding an optimal solution. However, EPnP and P3P are not incredibly robust in front of planar surfaces.

2.3 RANSAC

The methods proposed above to solve the PnP problem are susceptible to outliers in the correspondences set. For example, if we learn the 2D locations in the image of 3D points with a regressor, they are likely to be noisy. For this reason, PnP is often used in conjunction with a more robust estimator. The Random Sample Consensus (RANSAC), proposed by Fischler and Bolles (1981), is an iterative method to find a mathematical model from a set of observed noisy data that contains outliers.

This robust estimator starts with a minimal random selected subset from the entire observed data, and it computes a fitting model using only the elements of this subset. Then it checks if the other data fit the estimated model. Data that fit the model, also called inliers, are used to form the consensus set while others, the outliers, are discarded. The RANSAC algorithm iteratively repeats these steps until the consensus set has enough inliers. In the case of 3D object pose estimation, the set of 2D-3D correspondences can have some outliers, especially if these matches are predicted with a regressor. These outliers are the cause of wrong pose estimation. That is why RANSAC and PnP are often used together to remove outliers from the set of correspondences and to have a more accurate pose estimation.

2.4 Deep Learning

In Chapter 1 and 3 we show how Deep Learning boosts the performances of many computer vision tasks including 3D pose estimation and say that CNNs are becoming the first choice for many computer vision scientist. In this thesis we will present methods that exploit the CNNs. We, therefore, review the fundamental blocks of a CNN and how we can train this network.

Convolutional Layer: A CNN is composed of convolutional layers, hence the name, and thanks to this *convolutional* operator it extracts features from the input image automatically. A convolution unit has a receptive field with given weights (also called filter) that is shifted step by step across a 2-dimensional array of input values, such as the pixels of an image (usually there are several such filters). By sliding these learned filters over the input image, we obtain activation maps that give responses to that filter at every spatial position. These activation maps represent the extracted features from the image and they can be used in the higher levels of the network.

Activation Function: If we use only convolutions we end up with a linear regression model that it is not enough to represent complex functions that map an image input to an output. To bring non-linearity to the network, we add some activation functions. Their aim is to bound the values of the neurons and decide if they should activate or not. The most common activation functions used are:

- Tanh: $f(x) = tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$
- Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$
- ReLu: $f(x) = max(0, x)$

Pooling layer: A pooling layer is needed to reduce the dimensionality of each feature maps by downsampling them. There are different pooling types such as Max Pooling, Average Pooling and Sum Pooling. A spatial neighborhood of size $n \times m$ is selected in the feature map and the max, the average or the sum of the elements in the neighborhood are taken.

Fully Connected Layer: The fully connected layer takes the feature maps obtained with the set of convolutional and pooling layers to perform the final classification or regression task. The term "fully connected" means that every neuron in the previous layer is connected to every neuron on the next layer, as in regular NNs.

Cost function: A CNN learns to map a set of inputs to a set of outputs from training data. This map is equivalent to a set of weights the model can use in order to make good predictions. The best set candidate of weights is chosen with an optimization algorithm, in which a cost function is minimized. This cost function, often referred as loss, is a function error between the predicted outputs of the CNN and the correct ground truth. There are different types of loss functions and which one to use depends on the task we want to solve.

Optimizer: Receptive field weights in convolutional layers start from an initialization value, and then they are learnt during a training phase. In this phase, an optimization algorithm updates the values of these weights to minimize a cost function or objective function. Since Deep Learning requires a large amount of training data, it is important to have a fast and scalable optimization algorithm. Some researches are working on this optimization process, providing fast and accurate solutions. In our methods we use the Momentum or the Adam optimization algorithms.

2.5 Conclusion

In this chapter, we reviewed some fundamental concepts in computer vision and Deep Learning that are essential to understand the work of this thesis. In the next chapter, we present an overview of the state-of-the-art methods in the field of 3D object detection and pose estimation.

Chapter 3

Related work

Table of contents

In this chapter, we present an overview of 3D pose estimation methods of rigid objects. 3D pose estimation has a long history in computer vision with a large variety of proposed methods and collecting all methods in a structured review is not easy. There are interesting and well-structured reviews on 3D pose estimation in the literature: Marchand et al. (2016) with a survey on pose estimation for AR applications, Sahin et al. (2020) with a mathematical-model-based categorization of the methods and Lepetit (2020) with a survey on recent advances in 3D object pose estimation through several milestone methods.

3D pose estimation methods can be divided into two categories based on the information they exploit, that means if they use or not the depth information. We already discussed the limitation of depth sensors showing that the research community currently focuses more on methods based on RGB-only techniques and, for this reason, we will focus on them. Inspired by Lepetit (2020), we separate deep learning-based methods from early approaches since the advent of DL marked a turning point in the history of 3D pose estimation from RGB images. We, therefore, briefly review "traditional" methods that are not using DL to then focus on learning-based approaches explained why Machine Learning helped to improve the results.

We will finally detail methods that tackle some specific challenges in 3D pose estimation discussed in Section 1.3: handling unseen objects, symmetrical objects and the lack of training data.

3.1 Early Approaches to 3D object pose estimation

Early approaches focused more on 3D tracking problems and they have been summarized in different survey such as Marchand et al. (2016) and Lepetit and Fua (2005). To estimate the 3D pose of an object they were based on matching local simple features extracted from an image like corners or edges to features in a 3D model of the object and most of them relied on some prior knowledge about the 3D pose.

A first family of these approaches is the edge-based methods. Some of them try to match the projections of the target object 3D edges with some extracted image contours while others look at strong gradients in the image. In this last case, a first estimate of the object's pose is required. Harris and Stennett (1990), with their video rate object tracker RAPID, Lowe (1991) and Armstrong and Zisserman (1995) propose to describe the object of interest as a set of 3D geometric primitives such as lines and conics. By taking edges or control points on edges, they find the 3D pose by matching the contours in the image with the reprojected 3D geometric primitives with pose hypotheses. Despite being very fast at run-time, these methods lack robustness. If the edges are not very well detected, the object's pose is not accurate. These errors are frequent since they arise from occlusions, different lighting conditions and background clutter. For these reasons, some methods have

been proposed to make RAPID more robust such as Marchand et al. (1999) and Drummond and Cipolla (2002).

These methods rely on prior knowledge on the object's pose. Usually, they try to estimate the pose at the time instant $t+1$ knowing the pose at time t. Methods able to estimate the pose without relying on prior knowledge on it exploit features points often called keypoints. These methods propose to match keypoints between the input image with a reference image of the target object previously captured during an offline stage. The 3D pose of the object associated with this reference image is known, and by matching images, it is possible to get the final 3D pose. Different methods were successful for detecting and matching keypoints via descriptors such as SIFT (Lowe, 2004) and the faster methods, including SURF (Bay et al., 2008) and ORB (Rublee et al., 2011). Unfortunately, these approaches work well on textured objects but tend to fail with texture-less objects since only a few local features can be reliably extracted.

To deal with this problem, some methods relied on template matching, such as Hinterstoisser et al. (2012). In these methods, the 3D space is discretized and the object's appearance is represented for each discretized pose with a template robust to lighting conditions. Then the methods search the right template that matches up with the input image to estimate the pose. The main drawback of these approaches is that they need to create many templates and they are not strongly robust to occlusions.

3.2 Recent Approaches to 3D object pose estimation

To understand why Machine Learning became successful also in 3D object pose estimation from images, we can reformulate this problem as finding a mapping from the input image to the pose representation. The input space of this mapping is thus the images space that can be very large while the output space is smaller since it counts only 6 parameters. This difference in the space dimensions makes a purely algorithmic mapping hard to find. On the contrary, Machine learning techniques allow machines to find this mapping automatically by only looking at data during a training phase. These techniques learn a mapping from a features space to the poses space. This features space has a lower dimension than images space and features need to be extracted from input images with some image processing techniques. This phase is very delicate: if the features are not of good quality, there is no hope the pose estimation would be sufficiently accurate and a risk of losing information is always present in the feature extraction phase.

Deep Learning allows us to avoid this pre-processing phase to extract features and, in the last years, has been successfully applied to the problem of finding a mapping directly from the image to the 3D objects' poses. The literature on deep learning-based methods is growing faster and faster and here we present some mile-

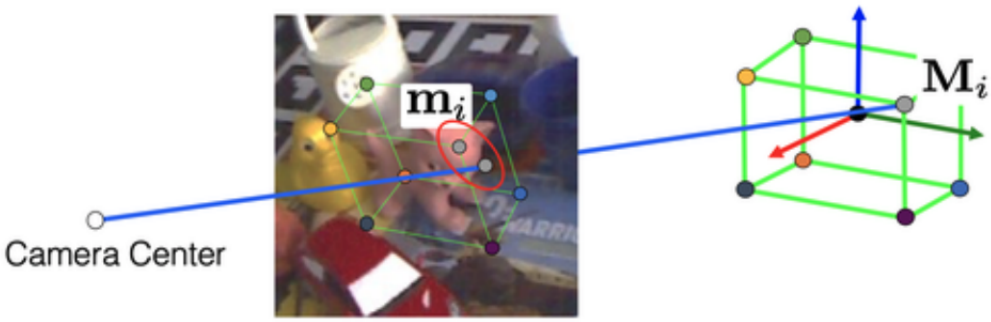


Figure 3.1: The pose representation developed in BB8 and Crivellaro et al. (2018) is now a common approach to predict the pose, by first predicting the 2D reprojections of some 3D points and then using a PnP algorithm on these 2D-3D correspondences. The image has been taken from Lepetit (2020).

stone methods that, at the time they have been published, beat the state-of-the-art results or introduced innovative solutions.

Two of the first methods that exploit the power of Deep Learning are BB8 (Rad and Lepetit, 2017) and SSD-6D (Kehl et al., 2017). Instead of directly regressing the full 3D pose of the object, BB8 regresses local 2D-3D correspondences and then computes the 3D pose from them. More specifically, it first learns to segment the objects in the image with a two-level coarse-to-fine object segmentation. Then it applies a Deep Network on the image window centered on the detected object to predict the 2D reprojections of the corners of the 3D object's bounding box. Figure 3.1 shows the corners of the 3D object's bounding box (M_i) and how they reproject into the image (m_i).Thanks to these 2D-3D correspondences, it predicts the 3D pose with a PnP algorithm (see Section 2.2). The last step in BB8 is a refinement of the first pose estimate. To improve the predictions of the 2D reprojections and the pose estimate, both the input image and the rendering of the object with the estimated pose are fed to another network to improve the pose estimate.

The idea of representing the 3D pose with the 2D reprojections of 3D points, introduced in BB8 and Crivellaro et al. (2018) is now often used since it makes the network optimization easier. Regressing the pose directly in the form of a 3D rotation and 3D translation is more difficult because pose ambiguities can introduce convergences issues. To train a deep network, an optimization algorithm on a cost function is then required. This loss function is basically an error between the ground truth pose and the estimated one. If we represent the pose with a 3D translation and a 3D rotation, we need to evaluate both translation and rotation errors that have different order of magnitude. To get rid of this unbalance in the loss, a meta parameter is needed and it is not always easy to find the correct value. On the

contrary, by representing the pose with the 2D reprojections, only a simple Euclidean distance between 2D coordinates has to be considered.

In SSD-6D, the 3D pose estimation problem is addressed as a classification problem instead of a regression one. SSD-6D extends the Single Shot Multibox Detector (SSD) architecture (Liu et al., 2016), developed originally for 2D object detection, also to predict the 3D poses of the objects with a single architecture. It proposes to discretize the pose based on the 3D pose decomposition into the direction of view over a half-sphere and in-plane rotation. They use the same paradigm of SSD, and they compute 6 feature maps at multiple scales. Each feature map is convolved with prediction kernels to provide, from each feature map location, the scores for object class identity, the discrete viewpoint and in-plane rotation. Given the viewpoint and the in-plane rotation it is easy to get the 3D rotation and from the 2D detected bounding box it is possible to infer the 3D translation. Two main advantages are that a single network is needed to perform 2D object detection and 3D pose prediction at the same time and that the training of the network can be done on only synthetic data without requiring a labeled data, differently from BB8. However, to perform this multi-task training, a single loss function made of a weighted sum of different terms is needed and tuning these weights can be difficult.

Both SSD-6D and BB8 need a refinement step to achieve accurate poses, and this takes time. YOLO-6D (Tekin et al., 2018) proposes a method, end-to-end trainable, that does not require additional post-processing and so it is much faster than the previous ones. YOLO-6D relies on YOLO (Redmon et al., 2016) for 2D object detection and it predicts the object poses in the form of the 2D projections of the corners of the 3D bounding boxes as BB8 plus the reprojection of the 3D objects' centroid. The model in YOLO-6D takes as input a single-color image, process it with the CNN and divides the image into a 2D regular grid containing SxS cells. For each cell of the grid, the model outputs the 2D locations of the 9 control points, the class probabilities and a confidence value.

PoseCNN (Xiang et al., 2018) claims that the pose representation as 2D locations of 3D control points is not able to handle heavy occlusions and symmetrical objects. It, therefore, regresses the objects' poses directly in the form of a 3D translation and 3D rotation. The key idea behind PoseCNN is to decouple the pose estimation task into different components. Specifically, PoseCNN performs 3 sub-tasks: semantic segmentation, 3D translation estimation and 3D rotation regression. The architecture has a shared backbone network and 3 different branches: (1) the first branch predicts for each pixel of the image the object label, (2) the second branch predicts a unit vector towards the 2D object center and the 3D distance between the object and the camera center, (3) the third branch regresses the 3D rotation for each bounding box in the form of a quaternion by taking as input only the features extracted inside the bounding boxes. From the predicted unit vectors and the semantic labels, a Hough voting layer computes the voting score to select the object center and obtain

the 3D translation for each object. Thanks to these multiple united vectors and this voting strategy, PoseCNN can work even when the objects are occluded. To train the architecture it uses the ADD metric and the ADD-S metric to deal with symmetrical objects. As we will see in Section 6.3, the ADD metric computes the average distance between two transformed model points using the ground truth pose and the estimated pose and the ADD-S metric is its variation for symmetrical objects. Deep-6DPose (Do et al., 2018) relies on Mask R-CNN to jointly detect, segment and recover 3D poses of object instances from a single RGB image. They propose to add a new branch to Mask R-CNN to directly regress the 3D pose without any pose refinement. Their main contribution is the decoupling of the pose parameters into a translation and a rotation terms so the latter can be regressed via a Lie algebra representation. The pose branch is parallel with the detection and segmentation branches and the architecture is entirely end-to-end trainable.

Handling occlusions: Some methods focus on the problem of objects under large partial occlusions, such as Oberweger et al. (2018), Hu et al. (2019) and Peng et al. (2019b). They use similar pose representations as in BB8 but combine multiple predictions or obtain predictions by looking only at some local image information. In this way, local image information not disturbed by occlusions provides good pose predictions while poses predicted with local image information disturbed by occlusions can be filtered out with a robust estimation. In particular, Oberweger et al. (2018) use the same pose representation of BB8 but they predict the 2D reprojections of the 3D points in the form of heatmaps. To deal with occlusions, instead of looking at the entire input image, they predict these heatmaps from multiple small patches independently and accumulate the results to obtain accurate and robust predictions.

Hu et al. (2019) claim that looking at the object as a full entity is not robust to occlusions, and they propose a method where each visible part of the object contributes a local pose prediction. To do that, they propose a two-streams architecture made of a shared encoder and two decoders. They superimpose an $S \times S$ grid on the input image and for each cell of the grid, they predict the label of the object that belongs to the image patch related to the cell and the 2D locations of 3D control points selected on the 3D object model. In practice, instead of predicting the 2D location in image coordinates, they predict the 2D displacement from the center of the corresponding grid cell. This technique allows a better and faster convergence. In this way, each visible part of the objects contributes a local pose prediction and with a RANSAC algorithm, they retrieve the best pose among the many possible hypotheses. This method is faster than Oberweger et al. (2018) since it does not require an expensive sliding windows strategy.

Peng et al. (2019b) select 3D control points on the objects' surfaces, and they introduce a Pixel-wise Voting Network (PVNet) to regress pixel-wise unit vectors pointing to the 2D reprojections of the 3D control points and use these vectors to

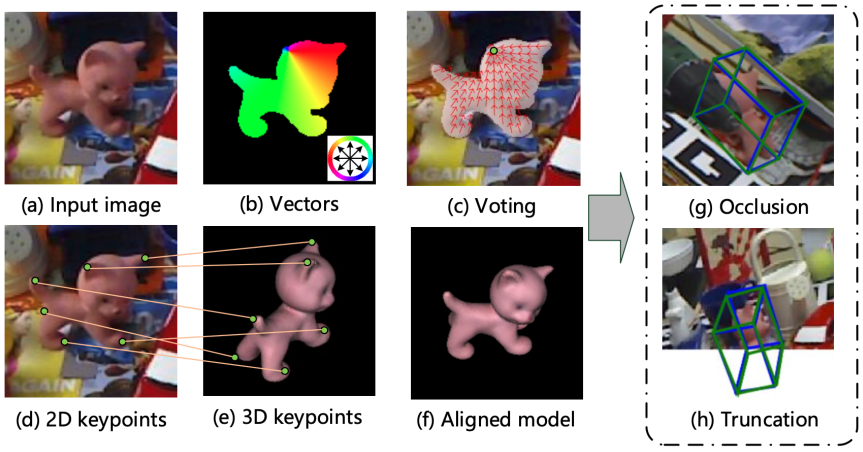


Figure 3.2: PVNet: an example of looking "locally" to be robust to partial occlusions and truncation. The image has been taken from Peng et al. (2019b).

vote for the final reprojections using RANSAC, as shown in Figure 3.2. In this way, the location of an invisible part can be inferred from the visible parts. This vector-field representation is useful to represent even keypoints outside of the image borders. Furthermore, the RANSAC voting scheme allows to get rid of outliers and to associate a confidence score for each keypoints. In this way, PnP can identify consistent correspondences to obtain the final 3D pose.

A more recent work, HybridPose (Song et al., 2020) was inspired by PVNet and proposes to use an hybrid pose representation made of keypoints, edge vectors and symmetry correspondences. In this way, they can handle occlusions: When some features are a bad prediction, they rely on others to obtain the right 3D poses.

Dense correspondences: Instead of predicting sparse 2D-3D correspondences, some recent methods propose to predict dense correspondences thanks to the encoder-decoder architectures. DPOD (Zakharov et al., 2019) and Pix2Pose (Park et al., 2019) propose to predict for each pixel of a target image its 3D coordinates in the object's coordinate system. From these correspondences, it is possible to predict the pose using RANSAC and PnP. DPOD represents the 3D object coordinates via a correspondence UV map that is a textured 2-channels image with values ranging from 0 to 255. In this way, instead of regressing the 3D object coordinates, they address the problem as a discrete color class classification problem. They claim this allows faster convergence and superior quality of 2D-3D matches. DPOD has been demonstrated to be capable to run in real-time. Pix2Pose together with the 3D object coordinates regresses a confidence value (based on the error of the prediction),

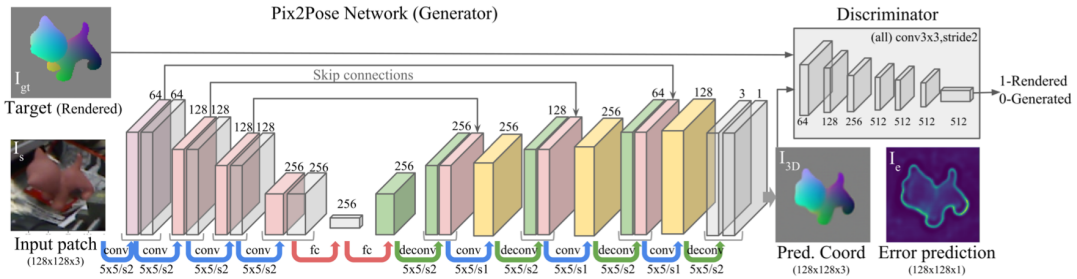


Figure 3.3: Pix2Pose architecture to regress the 3D object coordinates from a color image. The image has been taken from Park et al. (2019).

as shown in Figure 3.3, that allows to filter out a lot of 2D-3D matches at inference time to speed up the pose estimation algorithm. It also introduces a novel loss to deal with symmetrical objects.

Li et al. (2019) propose *Coordinates-Based Disentangled Pose Network* (CDPN) at the same time of DPOD and Pix2Pose. They noticed that regressing the 3D object's pose from dense 2D-3D correspondences is effective for the 3D rotation but not for 3D translation. They argue that rotation and translation should be treated differently for their significant difference and so they propose CDPN, which disentangles the pose to predict rotation and translation separately. Their proposed pipeline starts with a zoom on the image window around the object and then two different networks are trained to predict the translation and the rotation respectively. The rotation is obtained with PnP on the estimated dense 3D coordinates while the translation is regressed directly from the input image.

All the methods presented so far aim to explicitly learn a mapping from the input image to the 3D poses. Almost at the same time as these methods, a completely different approach has been proposed by Sundermeyer et al. (2020b), shown in Figure 3.4. They do not try to learn the mapping from 3D pose annotations during training explicitly, but they implicitly learn an embedding from an image of the object with an augmented auto-encoder (AAE). At run time, to predict the pose, they compare this embedding with a codebook that can be previously created offline by sampling views around the target objects and associating the embeddings of these views to the corresponding rotations. The orientation associated with the embedding of the codebook with the highest cosine similarity with the predicted embedding is the predicted orientation. The translation can be recovered from the object bounding box 2D location and scale. The architecture is made of an encoder, that aims to learn the embedding from the image, and a decoder that reconstructs the input image from the embedding. A Domain Randomization Technique is pro-

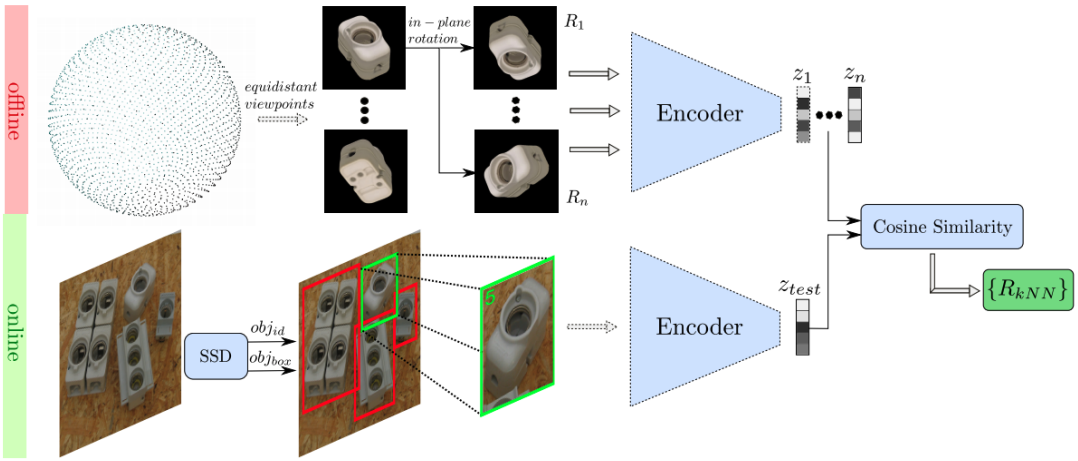


Figure 3.4: AAE is trained to learn an image embedding. A codebook is created offline from the encodings of discrete synthetic object views. Online, at test time, the AAE estimates the embeddings and the 3D pose relative to the most similar embeddings in the codebook. The image has been taken from Sundermeyer et al. (2020b).

posed together with the AAE framework to make the encodings invariant to noise, environment and sensors variation. Since the AAE does not learn any 1-to-1 mapping from the image of an object and the 3D pose, it is capable to work well with symmetrical objects, which are common in industrial contexts. AAE has been recently developed in its new version in Sundermeyer et al. (2020a) which we will discuss later in Section 3.2.3.

Recently, Hodan et al. (2020b) propose EPOS, a new method where the objects are represented by compact surface fragments. A surface fragment j of the object i is a portion of the object surface defined as follows: $S_{ij} = \{\boldsymbol{x} | \boldsymbol{x} \in S \wedge d(\boldsymbol{x}, \boldsymbol{g}_{ij}) < d((\boldsymbol{x}, \boldsymbol{g}_{ik})\}$, $\forall k \in J$, $k \neq j$, where $d(.)$ is the Euclidean distance of two 3D points and $\{\boldsymbol{g}_{ij}\}_{j=1}^{n}$ are preselected fragment centers. This representation is also able to handle symmetrical objects. An encoder-decoder network is trained to predict at each pixel: (1) the object label this pixel belongs to, (2) the fragment label, (3) the 3D location of the pixel on the predicted fragment, as shown in Figure 3.5. To handle all these many to many 2D-3D correspondences they propose a new robust estimator that performs better than the standard PnP-RANSAC variants.

Labbé et al. (2020) proposes an approach to estimate the pose of multiple objects from multiple views that also works with symmetrical objects. It exploits a set of RGB images to obtain a single consistent scene interpretation. It also shows an approach for 3D pose estimation from a single view who won the last BOP challenge. They train a deep network similar to Li et al. (2018), explained in Section 3.2.3, and they use a rotation parametrization introduced in Zhou et al. (2019) instead

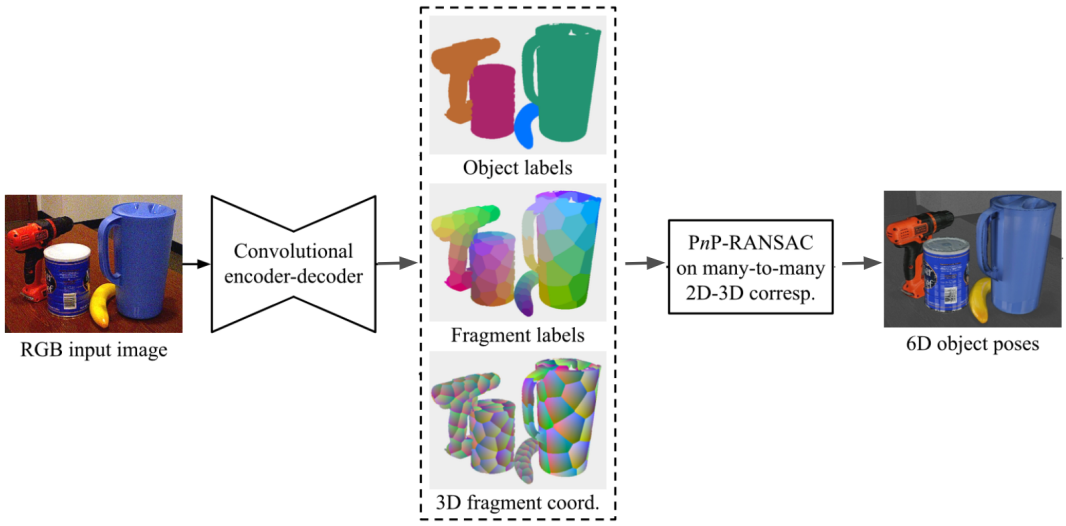


Figure 3.5: Pipeline of EPOS (Hodan et al., 2020b). Given an image, the network predicts at each pixel the object label, the fragment label and the 3D location on the fragment. The image has been taken from Hodan et al. (2020b).

of using quaternions for a more stable training. Finally they disentangle depth and translation prediction in the loss following Simonelli et al. (2019).

Hu et al. (2020) proposes a method to boost the performances of correspondences-based methods allowing them to be end-to-end trainable. We already discussed these methods that are composed of two stages: a first stage to establish 2D-3D correspondences between 3D points and 2D predicted image locations and a second stage to estimate the pose with RANSAC that cannot be trainable. The method presented in Hu et al. (2020) can be added to any correspondences-based method, right after the network that predicts the 2D locations of the 3D points. The proposed deep architecture takes as input clusters of 2D locations and it regresses the 3D pose.

3.2.1 3D pose estimation of symmetrical objects

Although all recent learning-based works have been very successful at predicting the 3D pose of objects, most of them do not consider objects with symmetry. Only recently proposed methods aim to tackle objects with pose ambiguities, and we detail them in this section. BB8 is probably the first work that mentioned the difficulty of predicting the 3D pose of objects with symmetries using Deep Networks. BB8, when naively applied to symmetrical objects, fails because symmetrical objects have a similar appearance but different 2D locations of the 3D points. This happens also to all methods that learn 2D-3D correspondences from images. The authors propose to use images of the object under rotation in a restricted range, such that the

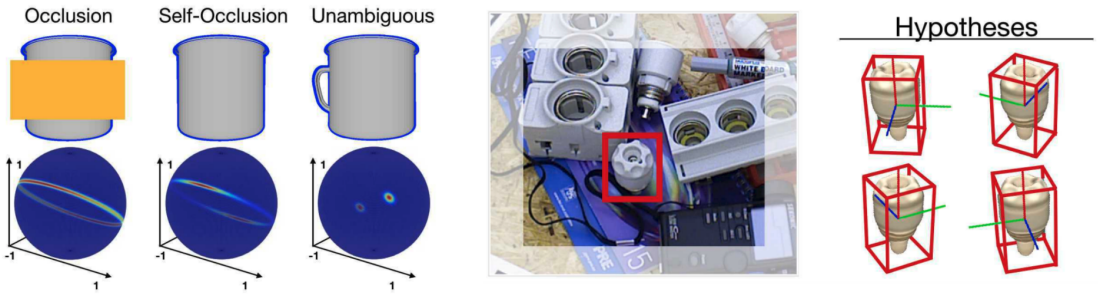


Figure 3.6: **Left:** The method of Manhardt et al. (2019) can detect and predict the ambiguities that arise because of intrinsic symmetries or occlusions automatically. **Right:** They predict M hypotheses for the 3D pose **p**. Each hypothesis is visually identical from the current viewpoint. The images have been taken from Manhardt et al. (2019).

training set does not contain ambiguous images. To recover the object pose under a larger range of rotation, BB8 train a classifier to predict the actual range where the rotation is. They tested their approaches on some objects from the T-LESS dataset (Section 4.5). Although the solution is not general, it inspired us for our method to tackle object symmetries explained in Chapter 5.

Sundermeyer et al. (2018), which we have already mentioned in Section 3.2, show that their learned embedding is ambiguity agnostic, in the sense that visually ambiguous poses will map to the same code in the latent space. They perform pose estimation by matching the code obtained from an image of the object with a precomputed code table covering the 6D pose space. While this approach is very fascinating, it does not provide an analytical study of the ambiguities. Corona et al. (2018) learns to compare an input image with a set of renderings of the object under many views, to predict the most similar view and the rotational symmetries of the object. This also requires to discretize the possible rotations, while we predict a continuous 3D pose. Manhardt et al. (2019) also consider a learning-based approach, tackle ambiguities raised by partial occlusions in addition to rotational symmetries, that is when an occluder hides a part of an object, so that it is not possible to estimate the pose exactly anymore (Figure 3.6). This is done by training a network to predict multiple poses so that only one has to correspond to the actual pose. At test time, the network predicts multiple poses, which are expected to represent the distribution over the possible poses. By contrast with this learning-based approach, we explicitly consider the ambiguities that can raise under symmetries in the method proposed in Chapter 5.

Brégier et al. (2018) does not consider pose prediction using regression or machine learning but they introduce the concept of proper symmetries group in a survey that aims to cover ambiguities and a pose representation specific to a metric on 3D poses, as a finite set of vectors $\mathcal{R}(\mathcal{P})$ of at most 12 dimensions, depending on the proper

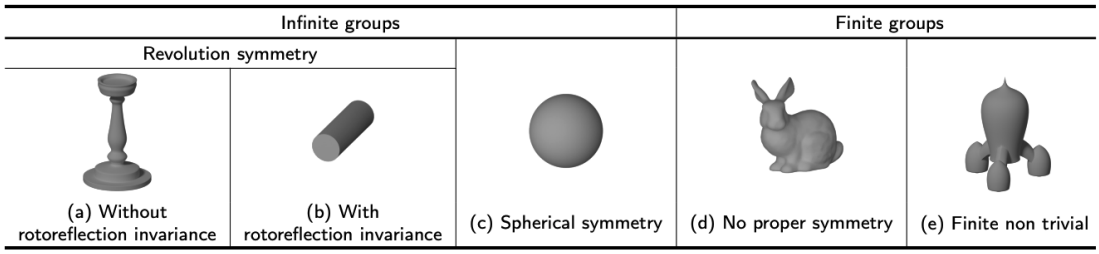


Figure 3.7: Classification of the potential groups of proper symmetries. The image has been taken from Brégier et al. (2018).

symmetry class of the object. As shown in Figure 3.7, they propose three pose representations: A pose of an object without proper symmetries is represented by a 12D point (the 3D position of the object's centroid, $\mathbf{t}$ and the rotation matrix $\mathbf{R}$ plus a matrix $\mathbf{\Lambda}$ to scale the rotation to account for the object's geometry), a pose of an object with finit non-trivial proper symmetry group is represented by several of those 12D points and a pose of revolution objects is represented by 6D vectors consisting in the centroid's position and the direction of the revolution axis, scaled to account for the object's geometry. Our method to solve the ambiguities created by symmetrical objects is based on this concept of proper symmetry group.

Henderson and Ferrari (2018) notices that symmetries produce multiple modes in the distribution $Q(\theta|\boldsymbol{I})$ over 3D poses θ. They, therefore, enforce a uniform prior $P(\theta)$ over symmetrical poses to successfully approximate Q. However, they do not explicitly report results on (quasi)-symmetrical objects such as those of T-LESS (Hodaň et al., 2017).

3.2.2 3D object pose estimation by training on synthetic images

Relying on synthetic images for training 3D pose estimation algorithms is an important research direction to overcome the challenge of the limited number of real training images. These images can be generated by rendering the CAD models of the objects when they are available, as we assume in this thesis. 3D models can be created very quickly and, for industrial applications, a detailed 3D model often exists before the real object is even created. There is however a domain gap between real and synthetic images, which has to be considered to make sure the method generalizes well to real images.
A straightforward approach is to train a convolutional network for some problems such as 2D detection on real images and use the first part of the network for extracting image features Hinterstoisser et al.; Kehl et al. (2017). Then, a network taking these features as input can be trained on synthetic images. This is easy to do, but it is not clear how many layers should be used exactly.

Generative Adversarial Networks (GANs, Goodfellow et al. (2014)) and Domain Transfer have been used to make synthetic images more realistic (Bousmalis et al. (2016); Müller et al. (2018); Bousmalis et al. (2017); Zhu et al. (2017); Ganin et al. (2016); Long et al. (2015); Tzeng et al. (2015); Lee et al. (2018); Zakharov et al. (2018)). Zakharov et al. (2018) chose to make real depth images closer to clean synthetic depth images. It requires however careful augmentation to create realistic synthetic depth maps. Because synthetic depth maps are easier to render than color images, Rad et al. (2018a) propose to learn a mapping between features for depth maps and features for color images using an RGB-D camera. Another interesting approach is domain randomization (Tobin et al., 2017), which generates synthetic training images with a random appearance by applying drastic variations to the object textures and the rendering parameters to improve generalization. Sundermeyer et al. (2018) present another domain randomization approach based on autoencoders to train a pose estimation network from CAD models and deal with pose ambiguities.

Recently, Park et al. (2020) claim it is difficult to obtain textured 3D models and annotate the poses of the objects in real scenarios. They, therefore, propose NOL, a Neural Object Learning, that takes as input few observations from cluttered images and texture-less 3D models of the objects in the images to learn to associate the right texture to the model's vertices and then generate synthetic images for training a 3D pose estimation method.

These works can exploit synthetic images to work with unseen objects when the CAD models or few images are available (Park et al., 2020). However, they also require a training phase for new objects that is what it is preferable to avoid in industrial applications. We therefore review the few and recent works who handle unseen objects at test time without retraining.

3.2.3 3D object pose estimation of unseen objects

Methods presented in Section 3.2 fail at predicting the 3D poses of new target objects, or, in other words, objects that the networks do not see during the training phase. AR and robotics applications, especially in industrial contexts, have to deal with new targets objects whose CAD model is available but not training images. Only recently the problem of predicting the 3D pose of unseen objects with Machine Learning has been tackled. The first possible solution is to generate synthetic training images of new objects and train the networks on them. We discussed these methods in Section 3.2.2. However, training sessions are still required and they take time. When time is precious, there is the need to completely avoid new training sessions and directly having methods capable to generalize to unseen objects. One early approach targeting texture-less objects is to rely on templates Hinterstoisser et al. (2012). They learn to match the templates of an object under different viewpoints with the input image with a similarity score by looking at some color gradient

features and surface normal features. To deal with texture-less objects, they keep only the main color gradient features located on the contour of the object silhouette.

Deep Learning has also been applied to such problems, by learning to compute a descriptor from pairs or triplets of object images (Wohlhart and Lepetit (2015); Balntas et al. (2017); Zakharov et al. (2017); Bui et al. (2018)) that efficiently capture both the object identity and the 3D pose. These approaches do not require re-training, as they only require to compute the descriptors for images of the new objects. However, it requires many images from points of view sampled around the object. It may be possible to use synthetic images, but then, some domain transfer has to be performed. Nevertheless, the main drawback of these approaches is the lack of robustness to partial occlusions, as the descriptor is computed for whole images of objects. It is also not clear how it would handle ambiguities, as it is based on metric learning on images.

Wang et al. (2019) tackle the problem of unseen instances of seen categories. In their scenario, CAD models of unseen object instances are not available at test time. They proposed a method similar to the concept of predicting "object coordinates" based on finding correspondences between object pixels to normalized coordinates in a shared object description space. This approach can generalize well to unseen object instances of seen categories but not to unseen categories and they do not rely only on RGB images but they exploit depth map to obtain the 3D poses. Li et al. (2018), motivated by the fact that the difference in performances between RGB and RGB-D methods is the lack of an effective RGB based refinement step, propose DeepIM, a pose refiner able to refine a given initial pose even of unseen objects. DeepIM is a network that, taken an input image and the rendered one generated with the CAD model of the object and the pose estimate, predicts a relative SE(3) transformation that matches the rendered view with the observed image. They predict the rotation and the translation separately. They also remark that predicting the rotation in the camera coordinate system also translates the object. They thus set the center of rotation to the object center. For the axes of the coordinate system, the authors remark that using those of the object's 3D model is not a good option since the network should learn them for each object and it would not generalize to unseen objects. They, therefore, propose to use the axes parallel to the axes of the camera coordinate system, which makes the network generalize much better. The translation update is predicted as a 2D translation on the image plane, plus a delta along the z-axis of the camera on a log-scale. Although this idea is very promising DeepIM has also been tested on unseen objects, but only on very simple synthetic images with constant lighting. Therefore, it is not clear how this method can work on real images of unseen objects with different lighting and noise conditions. Even more recently, Sundermeyer et al. (2020a) propose an extension of Sundermeyer et al. (2018) able to generalize to new objects. Thanks to the single-encoder-multi-decoder architecture, they can learn an interleaved encoding

where general features can be shared across multiple instances of novel categories. However, to achieve competitive results, they need to use depth information and refine the pose with an ICP algorithm.

3.3 Conclusion

In this chapter, we reviewed traditional and current 3D pose estimation approaches with a focus on RGB-only methods because of the limitations of depth sensors discussed in Chapter 1.1. In particular, we gave a review of methods from 2017 where the exploitation of the power of Deep Learning for 3D object pose estimation started. This literature review shows that these recent methods based on CNNs are very promising to solve the task of 3D pose estimation from RGB images. As shown in the report of the BOP challenge 2020 (Hodan et al., 2020a), some RGB-only methods based on CNNs finally outperform the methods that exploit the depth information.

However, these approaches require large amounts of data or long training sessions for each object and they are, therefore, not able to generalize to new objects. We also showed that there are a few works that try to handle complex objects such as texture-less, ambiguous and symmetrical objects. In the next chapter, before presenting our approaches inherent to the 3D pose estimation problem, we will present the different datasets available and the one we choose to train and evaluate our methods.

Chapter 4

6D Pose Estimation Datasets

Table of contents

Data are one of the most critical aspects of all applications using DL, for training, evaluating and comparing the methods. Researchers have put a lot of effort in annotating images for 3D pose estimation in recent years. Different datasets are publicly available nowadays such as LineMOD (Hinterstoisser et al., 2012), Occluded LineMOD (Brachmann et al., 2014), YCB-Video (Xiang et al., 2018) and TLESS (Hodaň et al., 2017). All these datasets have different features and different levels of difficulty. In this chapter we will detail these datasets, the ones most used in literature.

Contribution: We developed a Blender based pipeline to generate the Synthetic T-LESS dataset, a dataset made of photorealistic synthetic images, to train the deep networks.
The code is available at https://github.com/MichaelRamamonjisoa/SyntheT-Less. Note that, at the time we created Synthetic T-LESS, the *BlenderProc* (Denninger et al., 2019) tool was not released yet.

4.1 LineMOD Dataset

This dataset was created to evaluate the method proposed in Hinterstoisser et al. (2012). They captured 15 video sequences of 15 3D objects mainly of different geometry (ape, bench vise, bowl, can, cat, cup, driller, duck, glue, hole-puncher, iron, lamp, phone, cam and eggbox) as shown in Figure 4.1. Each object was placed on the center of a planar board with markers attached to it and for each object, more than 1100 real RGB-D images are recorded with the corresponding 3D pose annotations. The sequence provides views from 0-360 ° around the object, 0-90 ° tilt rotation, ±45 ° in-plane rotation and $650 - 150$ mm object distance. The goal of the method proposed in Hinterstoisser et al. (2012) is to estimate the pose of a single object in the image, which is known to be present. That is why only the object centered in the image has the corresponding pose ground truth but other 3D objects are placed on the planar board and the background of the scenes is cluttered. The advent of DL was not already started when this dataset was created and therefore there is not a division between training and test images.

4.2 Occluded-LineMOD Dataset

The Occluded-LineMOD dataset (Brachmann et al., 2014) is an extended version of the LineMOD dataset, created for the goal of 3D pose estimation of multiple objects especially in the case of occlusions. In this dataset, it can be observed up to 70-80 % occluded objects. The images are the same as LineMOD but the ground truths are extended. In Occluded-LineMOD all 3D objects in the scene have the annotated ground truth pose and not only the one at the center of the image as in LineMOD.

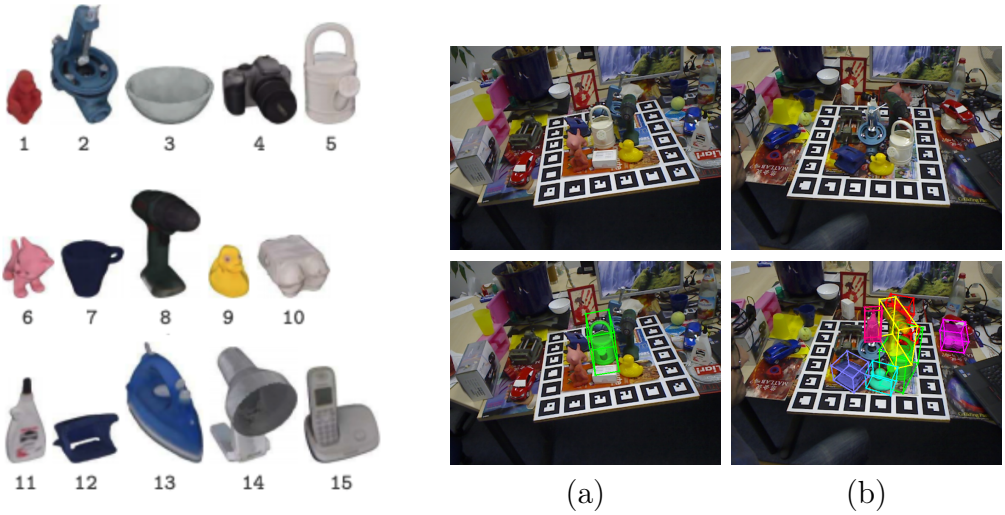


Figure 4.1: **Left:** LineMOD and Occluded-LineMOD objects. **Right:** (a) LineMOD RGB image with the corresponding ground truth only for the object placed at the center. (b) Occluded LineMOD RGB image with the ground truths for all the objects.

This makes the dataset count 10k images of 20 objects (both textured and textureless) captured each under three different lighting conditions: bright artificial light, darker natural light, and directional spotlight.

4.3 YCB-Video Dataset

The YCB-Video dataset (Xiang et al., 2018) is a large-scale RGB-D dataset which contains 3D poses of 21 objects from the YCB object set (Calli et al., 2017), as shown in Figure 4.2, in 92 video sequences with a total of more than 100K frames. In particular 12 sequences are used for testing and the remaining 80 sequences for training. Besides, the dataset contains 80k synthetically rendered images, which can be used for training as well. The dataset has been designed for benchmarking robotic manipulation research in real daily life applications. The objects are textured and they vary from cereal boxes, plates, scissors, drills, etc. The test images are challenging due to the presence of significant image noise, different illumination levels, and large occlusions.

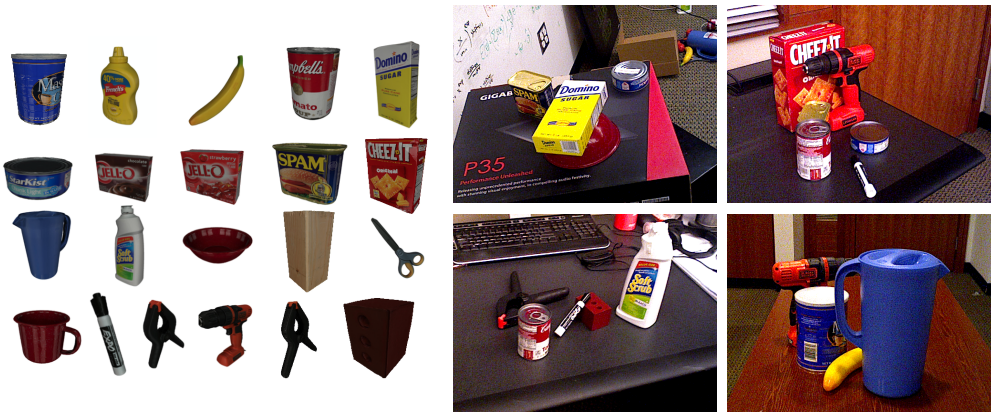


Figure 4.2: **Left:** YCB-Video dataset objects. **Right:** Examples of test images of the YCB-Video dataset.

4.4 HomebrewedDB Dataset

The HomebrewedDB dataset (Kaskman et al., 2019) features 33 objects, more specific 17 toys, 8 household and 8 industry-relevant objects, as shown in Figure 4.3. This dataset is the one with the highest number of objects. It is mainly intended to deal with the problem of scalability of the 3D pose estimation methods. The dataset features 13 scenes of more than 1300 images that span a range of complexity from simple, such as 3 objects on a plain background, to complex like highly occluded scenes with 8 objects and extensive clutter.

4.5 T-LESS Dataset

The T-LESS (Hodaň et al., 2017) dataset features thirty industrial objects (such as commodity electrical parts) which have no significant texture, discriminative color or distinctive reflectance properties. Figure 4.4 shows the reconstructed 3D models of these objects. They often have similarities in both shape and size and some objects are composed of other smaller objects. It is not easy to select discriminative shape features on these objects because of the presence of similar-looking object classes, along with the presence of similar-looking distractors (such as planar surfaces) and out-of-training objects (such as books, scissors). The dataset has 20 different test scenes, each of which consists of 504 test images. This dataset has significant variability in complexity and it is still very challenging since it was built to test 3D pose estimation algorithms in a very complicated situation: texture-less objects in different viewpoints, clutter background, occlusion, multi instances of the same object class in the image and similar-looking distractors. This is the reason behind the fact this dataset was not very famous at the time it was released. Only

Figure 4.3: **Top:** HomebrewedDB objects. **Bottom:** HomebrewedDB test images from a simple scene with a uniform background to a difficult one with clutter background and high level of occlusions.

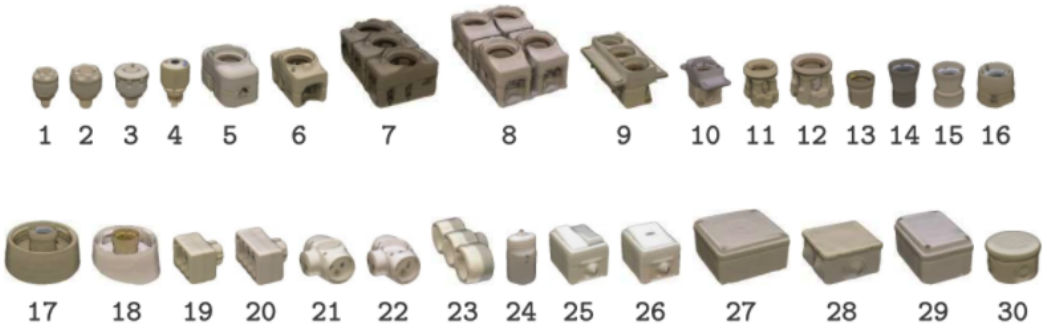


Figure 4.4: Reconstructed 3D models of the 30 objects from T-LESS.

recently researchers, motivated by the will to develop algorithms robust to all the challenges of the real words, started to test their method on this dataset.

4.6 Our choice

These mentioned datasets are the most frequently used to test performances of 3D pose estimation, but other datasets are available: ITODD (Drost et al., 2017), RU-APC (Rennie et al.), IC-BIN (Doumanoglou et al., 2016), IC-MI (Tejani et al., 2014) and TUD-L/TYO-L (Hodaň et al., 2018).

These datasets have different levels of difficulty depending on the type of objects and test images. For example, evaluating the algorithms on LineMOD is relatively easy since only one target object in the images has the annotated 3D pose and it is visible most of the time, while in Occluded LineMOD the difficulty increases because of the occlusions between the objects. Also, the type of 3D objects is different and it can make the dataset more or less challenging. LineMOD, Occluded-LineMOD and YCB-Video have both texture and texture-less objects and only a few of them are symmetrical. On the contrary, objects from T-LESS are texture-less, symmetrical and have similarities between them.

Ideally, a 3D object pose estimation algorithm should be robust to the type of data and sensors used to capture them to be suitable for real industrial applications. Research scientists are making a significant effort to reduce the gap between research and industrial applications and recently the BOP challenge has been proposed to test the methods on different benchmarks. The results of this challenge show that methods that beat state-of-the-art methods on a particular dataset are not always able to generalize and work well on other datasets because learning-based algorithms are very susceptible to the data they are trained on. For example, a network trained on textured objects would hardly work with texture-less ones and vice versa, or a network trained on images where no occluded objects appear is not going to work with an occluded object. For this reason, 3D pose estimation from RGB images is a not yet general solved problem.

The industrial applications of 3D pose estimation are the focus of this thesis with all the challenges that this scenario implies. For this reason, we need to handle symmetrical, texture-less and ambiguous objects with no discriminative parts on cluttered images with occlusions and distractors. Therefore, we decided to evaluate all the proposed methods on the T-LESS dataset, since it is one of the most challenging dataset nowadays and the one that represents better an industrial application.

4.6.1 Synthetic T-LESS

One difficulty in the T-LESS dataset is the big gap between training and testing images. As shown in Figure 4.5, training images are captured under controlled conditions, they depict individual objects against a black background and they have a limited range of poses and illuminations condition. On the other hand, test images have massive viewpoint changes and objects in multiple instances affected by clutter

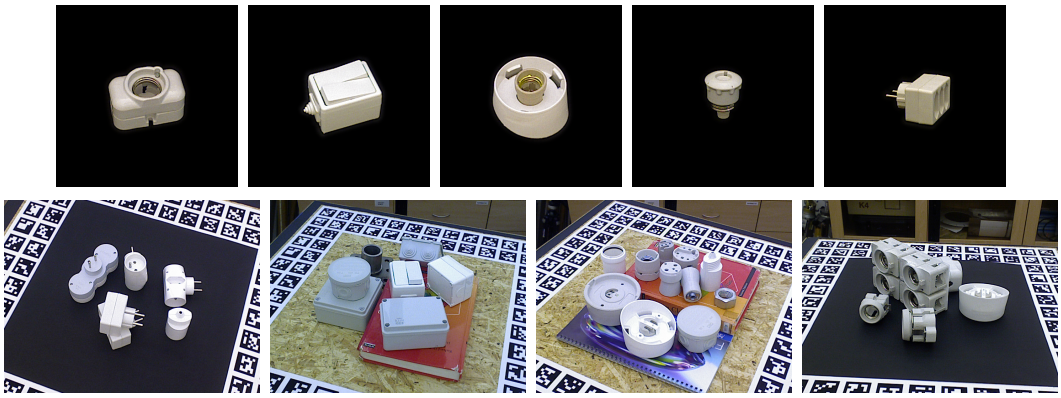


Figure 4.5: Training and test images from T-LESS showing the gap between them. **Top:** A single training image for each object. The object is always located in the center over a black background. **Bottom:** some test images from different test scenes. Some images have a black background but others have more cluttered ones. Objects often occlude each others and some "distractors" appear.

and occlusions. Furthermore, as mentioned above, other unlabeled objects are in the test scenes acting as distractors for the deep networks.

A network trained on the training images of T-LESS is not going to perform well on the test images. A general approach in Deep Learning applications to overcome the problem is *data augmentation*. It can be done online by changing the brightness of an image, adding different types of noise, adding patches to the image to simulate occlusions, adding a random background to increase the complexity of a uniform background. Also, 2D transformations can be applied, such as 2D translation and 2D rotation. Unfortunately, we soon realized that even applying data augmentation techniques our learning-based approaches could not generalize to these test images because of a lack of pose viewpoints and occlusions. We decided to tackle this problem by generating synthetic images. A first approach would be to render the object's CAD models on a random image with a random pose but images generated in this way lack realism because of the presence of flying objects. We, therefore, put some effort into generating photorealistic synthetic images with T-LESS objects and we create the Synthetic T-LESS dataset. For all the approaches we will present in this thesis, we use this Synthetic T-LESS to train and validate the networks. To solve the problem of the domain gap between training and test images, the synthetic generated images have partial occlusions and illumination variations. Synthetic T-LESS is made of 30K samples, generated using the CAD models provided in the original T-LESS dataset with Cycles, a photorealistic rendering engine of the open-source software Blender. Each sample of our dataset is generated using a random set $\mathcal{S}$ of objects taken from the T-LESS dataset, using random grayscale color (from dark-gray to white) for each of them. Each object of $\mathcal{S}$ is initially set up with

Light	Strength	$\sim \mathcal{U}(30, 90)$
	Position	$\theta \sim \mathcal{U}(0, 360)$ $\phi \sim \mathcal{U}(1, 80)$
	Color	$\sim \mathcal{U}(0, 0.65)$
Pose	Rotation	$\theta \sim \mathcal{U}(0, 360)$ $\phi \sim \mathcal{U}(1, 80)$
	Translation	$T_x \sim \mathcal{U}(-0.075, 0.075)$ $T_y \sim \mathcal{U}(-0.075, 0.075)$ $T_z = max(-4, (-4 \cdot \frac{object_{idx}}{n_{objects}}))$
Objects		$\sim \mathcal{U}(1, 9)$

Table 4.1: Distribution of random values used for generating the synthetic images.

Dataset	T-LESS (primesense)		Synthetic T-LESS
	Train	Test	
Number of samples	38K	10K	30K
Illumination variation	None	Small	Strong
Occlusion	No	Yes	Yes
Multi-objects images	No	Yes	Yes
Object color variation	None	Small	Small
Background variation	None	Small	Strong

Table 4.2: Comparison between the T-LESS dataset and our Synthetic T-LESS dataset.

a random pose, and we let the objects fall on a randomly textured plane, using Blender's physics simulator. Because the objects can collide together, their final pose on the table is also random. Illumination randomization is performed by varying the level of ambient light and randomizing a point light source in terms of position, strength, and color. This often results in strong cast shadows, as can be seen in Figure 4.6 and Figure 4.7. Table 4.1 shows how we select the values randomly.

A comparison with the original T-LESS (primesense) dataset is given in Table 4.2. Although we are working only with RGB images, for all samples of our dataset, we also generate normals, depth, contours and object instances maps as shown in Figure 4.6. This can help train Deep Networks for the different tasks such as semantic segmentation and 3D scene understanding.

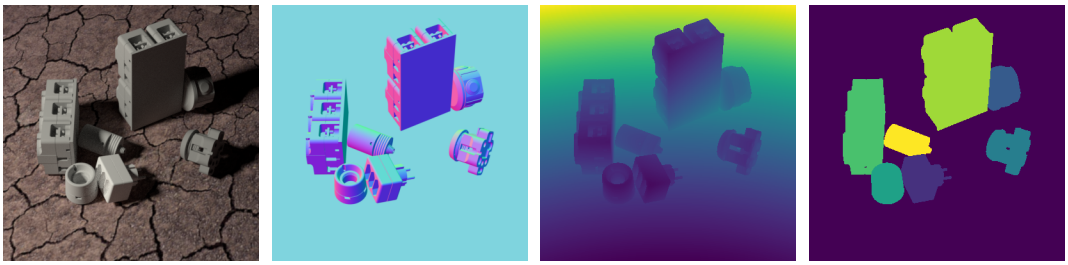


Figure 4.6: A sample of our Synthetic T-LESS Dataset. From left to right: RGB, normals, depth and object instance masks.

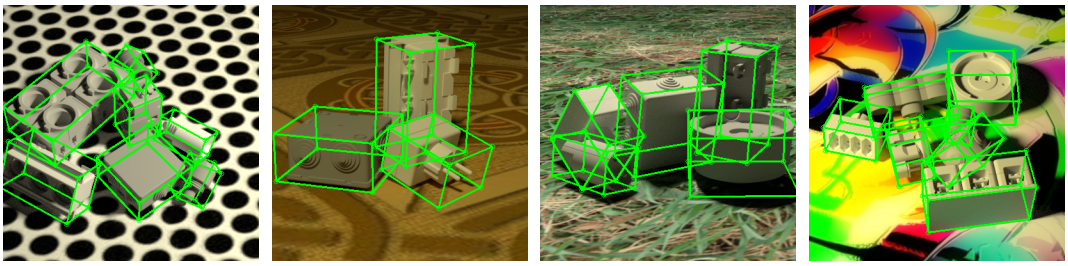


Figure 4.7: Sample images from our Synthetic T-LESS dataset. All objects in each image are annotated with their classes and 3D poses.

4.7 Conclusion

In this chapter, we presented the available 3D pose estimation datasets most used in the literature for training and evaluation purposes. We discussed the differences between them in term of difficulty of the test images and the type of the objects. We finally explained why we chose the T-LESS dataset and described our Synthetic T-LESS dataset that extends it. In the next chapter, we will present an approach to handle symmetrical objects with the aim of estimating the 3D poses of objects with Machine Learning techniques.

Chapter 5

6D Pose Estimation of Symmetrical Objects

Table of contents

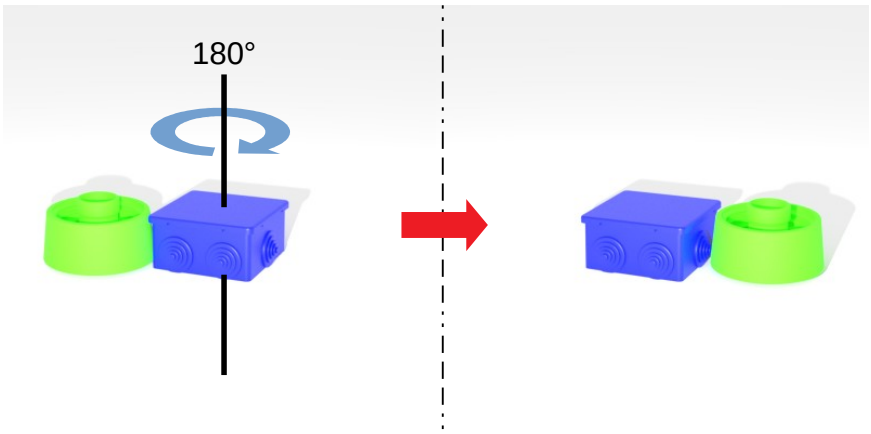


Figure 5.1: Two views of the same scene before and after a rotation of 180° around the vertical axis of the blue object. Since this object is symmetrical, it has the same appearance but its pose is different. The same happens for the green object, which has a continuous symmetry around its vertical axis.

In this chapter, we first address one of the main challenges in 3D object pose estimation discussed in Section 1.3: the ambiguities that raise with symmetrical objects. We explain the link between the symmetries of a 3D object and its appearances in images and we provide a simple and analytical solution to handle the problem based on the normalization of the pose rotations. The method we propose can handle texture-less, symmetrical and ambiguous objects in cluttered images. Furthermore, it is general and can be integrated into any 6D pose estimation framework that learns a mapping from a RGB image to a pose representation.

5.1 Overview

Many objects of our daily life or from industrial contexts exhibit symmetries, or at least "quasi-symmetries" when only a small detail prevents the object to have a perfect symmetry. These symmetries create ambiguities when aiming to estimate the 3D pose of the object from images. For a long time, this problem has not been considered by researchers and the problem of 3D object pose estimation has been solved only on simple case scenarios.

In this chapter, we explain why exactly symmetries can be a problem for 3D pose estimation algorithms and we then provide a simple solution that is general and can be introduced in any 3D pose estimation algorithm.

To better understand the problem raised by the symmetries of an object, let's first consider Figure 5.1. The blue object has rotational symmetry around the vertical axis. If we apply a rotation of 180° around this axis, this object has the very same appearance. More generally, when an object O has some symmetry, there exist one

or more rigid motions such that, if we apply these rigid motions to the object pose, the appearance of the object is preserved. Formally, we consider the set

$$\mathcal{M}(O) = \{\boldsymbol{m} \in \mathrm{SE}(3) \text{ such that } \forall \boldsymbol{p} \in \mathrm{SE}(3),\ \mathcal{R}(O, \boldsymbol{p}) = \mathcal{R}(O, \boldsymbol{m}.\boldsymbol{p})\}\ , \tag{5.1}$$

where $\mathcal{R}(O, \boldsymbol{p})$ is the image of Object O under pose $\boldsymbol{p}$ (ignoring lighting effects), $\boldsymbol{m}$ is a rigid motion related to the symmetry, and $\boldsymbol{m}.\boldsymbol{p}$ is the pose after applying motion $\boldsymbol{m}$. $\mathcal{M}(O)$ is thus the set of rigid motions $\boldsymbol{m}$ that preserve the visual aspect of a given object. It is easy to see that it forms a subgroup of $SE(3)$. Brégier et al. (2018) call the elements of $\mathcal{M}(O)$ *proper symmetries*.

In other words, two images of a symmetrical object can be identical but not correspond to the same pose. If we consider an image $\boldsymbol{I}_1 = \mathcal{R}(O, \boldsymbol{p})$ of an object O under pose $\boldsymbol{p}$ and a motion $\boldsymbol{m} \in \mathcal{M}(O)$, then, the image $\boldsymbol{I}_2$ of object O under pose $\boldsymbol{m}.\boldsymbol{p}$ is equal to image $\boldsymbol{I}_1$, that means $\boldsymbol{I}_2 = \mathcal{R}(O, \boldsymbol{m}.\boldsymbol{p}) = \mathcal{R}(O, \boldsymbol{p}) = \boldsymbol{I}_1$. There is therefore no function

$$\mathcal{F}(\boldsymbol{I}) = \boldsymbol{p} \tag{5.2}$$

that can provide the pose $\boldsymbol{p}$ of object O given an image $\boldsymbol{I}$. Any attempt to learn such a function, for example with a Deep Network, would fail. For example, if a network is trained to predict the pose using the squared loss between the ground truth poses and the predicted poses, it would converge to a model predicting the average of the possible poses for an input image, which is of course meaningless.

Only a few works, presented in Chapter 3, consider the problem of symmetrical objects. Sundermeyer et al. (2018) solves this problem by learning a mapping to a latent representation of the pose; Brégier et al. (2018) introduced a representation of the pose that differs from rigid motions and suitable for their similarity metric between two poses; Manhardt et al. (2018) learns to predict several poses so that at least one pose corresponds to the ground truth; Rad and Lepetit (2017) rely on image mirroring to deal with some symmetries. Recent published works can handle symmetrical objects by representing the object surface via fragments such as Hodan et al. (2020b) or by optimizing the networks over losses that handle ambiguities (Wang et al. (2019), Labbé et al. (2020)). While these works propose interesting solutions, in this chapter, we present a general analytic approach to the problem. It will give insights on the learning-based methods and yields a simple way to solve the ambiguities due to symmetries.

Contributions. The contributions of this chapter are:

- We explain the link between the symmetries of a 3D object and its appearance in images and we show why restricting the poses within some ranges is not enough.

- We propose a simple and analytical solution to handle the problem based on the normalization of the pose rotation.

The work presented in this chapter was presented at the *International Conference on 3D Vision*, 2019 (Pitteri et al.).

5.2 Method

In this section, we study the effect of symmetries on algorithms aiming to learn the mapping between an image of an object and its 6D pose, and we show how we can derive a simple method for handling these symmetries. The proposed method is general and can be integrated into any 3D pose estimation algorithm. We will describe in Section 5.3 how this method can be integrated within a Faster R-CNN framework. In the remainder of this chapter we will refer to our method also as normalization procedure.

5.2.1 Mapping Ambiguous Rotations

Let's consider the set $\mathcal{M}(O)$ already introduced in Eq. (5.1). In practice, the motions in $\mathcal{M}$ are usually in the form $\boldsymbol{m} = [R, \mathbf{0}]$ with $R \in \mathrm{SO}(3)$, that means objects have mostly rotational symmetries. A translation component different from $\mathbf{0}$ would correspond to an object with translation symmetries, for example, a long building with windows of similar appearances.

We thus first define the notion of ambiguous rotations: We say that two rotations R_1 and R_2 are ambiguous if they result in the same object appearance, that is if $\mathcal{R}(O, [R_1, T_1]) = \mathcal{R}(O, [R_2, T_2])$. This defines an equivalence relationship $R_1 \sim R_2$. If $R_1 \sim R_2$, then it is not possible from an image to distinguish between rotation R_1 and R_2 when predicting the pose. Predicting R_1, or R_2, or any rotation $R \sim R_1$ is equally good. This is the idea behind the ADI metric (Hinterstoisser et al., 2012).

As illustrated in Figure 5.2, a natural idea to aim at preventing trouble during learning is, therefore, to first map equivalent rotations to a unique rotation, which we call a canonical rotation. This means that during training, training images with the same object appearance will be assigned the same rotation after mapping. The transformation $\mathcal{F} : \boldsymbol{I} \mapsto \boldsymbol{p}$ of Eq. 5.2 will thus become a function and we will be able to learn it with a Deep Network for example. This implies that at inference, the network will predict the canonical rotation for a given input image, which is the best that can be done in the presence of symmetries.

Given the set $\mathcal{M}(O)$ of the object's proper symmetries, we are therefore looking for an operator $\mathrm{Map}(\cdot)$ on $\mathrm{SO}(3)$ that can map ambiguous 3D rotations to a single rotation such that $\mathrm{Map}(R_1) = \mathrm{Map}(R_2) \iff R_1 \sim R_2$ $(\star)$ holds.

Proposition 1. *Given a proper symmetry group $\mathcal{M}(O)$, let us define* Map *operator as:*

$$\mathrm{Map}(R) = \hat{S}^{-1} R, \quad \forall R \in \mathrm{SO}(3), \tag{5.3}$$

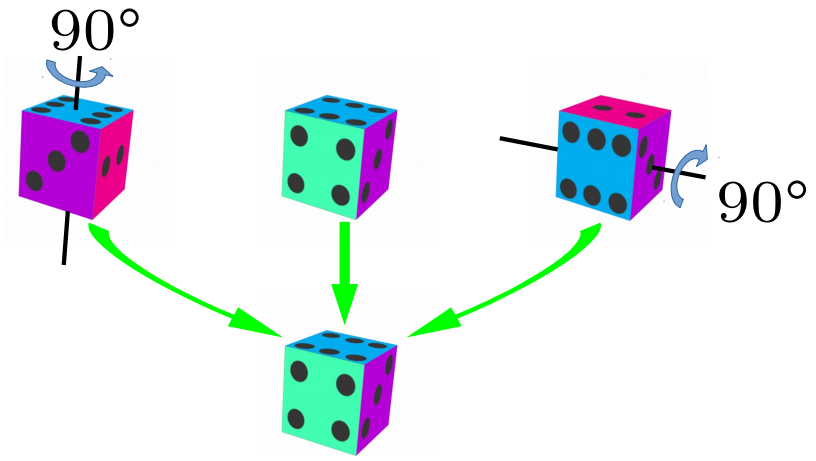


Figure 5.2: Mapping of 3 ambiguous poses to the same pose. We consider here a uniform object and the colors and dots on the faces are only to visualize the different poses. The left and right poses are remapped to the reference pose in the middle.

with

$$\hat{S} = \underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - I_3|_F, \tag{5.4}$$

where $|\cdot|_F$ *is the Froebenius norm. Then* Map *verifies the mapping property* ($\star$).

Proof. To simplify the notations, let us consider that $\mathcal{M}(O)$ is made only of the rotation components. By definition of $R_1 \sim R_2$ and $\mathcal{M}(O)$:

$$R_1 \sim R_2 \quad \Leftrightarrow \quad \exists! \quad S_{12} \in \mathcal{M}(O) \quad \text{such that} \quad R_1 = S_{12}R_2\,. \tag{5.5}$$

Let us consider the solution of the optimization problem in Eq. (5.4) for R_1:

$$\hat{S}_1 = \underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R_1 - I_3|_F\,. \tag{5.6}$$

then, by replacing R_1 with the expression in Eq. (5.5)

$$\hat{S}_1 = \underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}S_{12}R_2 - I_3|_F\,. \tag{5.7}$$

We introduce the variable T such that $S = S_{12}T$. Since S and S_{12} belong to $\mathcal{M}(O)$ and $\mathcal{M}(O)$ is a group, T also belongs to $\mathcal{M}(O)$. We can therefore perform the following change of variable:

$$\hat{S}_1 = S_{12} \underset{T \in \mathcal{M}(O)}{\arg\min} |T^{-1}R_2 - I_3|_F\,, \tag{5.8}$$

which is equal to:

$$\hat{S}_1 = S_{12}\hat{S}_2, \tag{5.9}$$

with

$$\hat{S}_2 = \underset{S\in\mathcal{M}(O)}{\arg\min} |S^{-1}R_2 - I_3|_F. \tag{5.10}$$

Therefore

$$R_1 \sim R_2 \quad \Leftrightarrow \quad \text{Map}(R_1) = \hat{S}_1^{-1}R_1 = \hat{S}_2^{-1}S_{12}^{-1}S_{12}R_2 = \hat{S}_2^{-1}R_2 = \text{Map}(R_2)\,. \tag{5.11}$$

□

5.2.2 Implementation of the Map operator

If $\mathcal{M}$ is discrete, implementing the operator Map is trivial, as it is only a matter of iterating over the elements of $\mathcal{M}$ to find the minimum. However, $\mathcal{M}$ can be continuous for some objects. This is the case for generalized cylinders and spheres (Brégier et al., 2018). For spheres, Map is also trivial as it can always return the identity transformation, for example.

For generalized cylinders, implementing the operator Map is more complex. In this case, $\mathcal{M}$ can be written as:

$$\mathcal{M}(O) = \{R_\alpha^{\boldsymbol{u}} : \alpha \in [0, 2\pi)\}\,, \tag{5.12}$$

where $R_\alpha^{\boldsymbol{u}}$ is the rotation around axis $\boldsymbol{u}$ of amount α.

The Froebenius norm in Eq. (5.4) can be rewritten as

$$|S^{-1}R - I_3|_F = |D|_F = \text{Trace}(D^T D)\,, \tag{5.13}$$

with $D = S^{-1}R - I_3$. After some derivations:

$$|S^{-1}R - I_3|_F = 6 - 2\cdot\text{Trace}(S^T R)\,. \tag{5.14}$$

Proof. Let:

$$\hat{S} = \underset{S\in\mathcal{M}(O)}{\arg\min} |S^{-1}R - I_3|_F^2 = \arg\min \text{Trace}(D^T D), \tag{5.15}$$

with $D = S^{-1}R - I_3$.

We have:

$$D^T D = R^T S S^T R - R^T S - S^T R + I_3 \quad = 2I_3 - (R^T S + S^T R)\,, \tag{5.16}$$

and thus

$$\text{Trace}(D^T D) = 6 - \text{Trace}(R^T S) - \text{Trace}(S^T R) \quad = 6 - 2\cdot\text{Trace}(S^T R)\,, \tag{5.17}$$

where we used the fact that $\text{Trace}(I_3) = 3$ and $\text{Trace}(A^T) = \text{Trace}(A)$.

We note:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} . \tag{5.18}$$

In the object's coordinate system, S writes as:

$$S = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{5.19}$$

Then:

$$\begin{aligned} \text{Trace}(D^T D) &= 6 - 2 \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \\ &= 6 - (R_{11} + R_{22}) \cos(\alpha) + (R_{12} - R_{21}) \sin(\alpha) . \end{aligned} \tag{5.20}$$

Since $\hat{S} = \underset{S \in \mathcal{M}(O)}{\arg\min} \text{Trace}(D^T D)$ and S is parameterized by α, Equation (5.15) can be rewritten as a minimization over α such that:

$$\begin{aligned} \hat{\alpha} &= \underset{\alpha \in [0, 2\pi)}{\arg\min} \text{Trace}(D^T D) \\ &= \underset{\alpha \in [0, 2\pi)}{\arg\max} (R_{11} + R_{22}) \cos(\alpha) - (R_{12} - R_{21}) \sin(\alpha) . \end{aligned} \tag{5.21}$$

This is solved analytically by solving $\frac{\partial \text{Trace}(D^T D)}{\partial \alpha} = 0$ for α. The solution of Equation (5.15) is then:

$$\hat{S} = \begin{bmatrix} \cos(\hat{\alpha}) & -\sin(\hat{\alpha}) & 0 \\ \sin(\hat{\alpha}) & \cos(\hat{\alpha}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{with } \hat{\alpha} = \arctan\left(\frac{R_{21} - R_{12}}{R_{11} + R_{22}} \right) . \tag{5.22}$$

5.2.3 Discontinuities of $\mathcal{F}$ After Mapping

After applying the Map operator, there are no pose ambiguities anymore, that is two similar images are assigned the same rotation. However, a new difficulty arises: The transformation $\mathcal{F}(\boldsymbol{I}) \rightarrow \boldsymbol{p}$ is now discontinuous around some rotations and this is problematic when using Deep Networks to learn $\mathcal{F}$, as Deep Networks can only approximate continuous functions (Cybenko, 1989; Hornik, 1991; Hanin, 2017).

To understand why these discontinuities happen, let us consider an example, more exactly the rectangular object seen from the top as in Figure 5.3. $\mathcal{M}(O)$ is

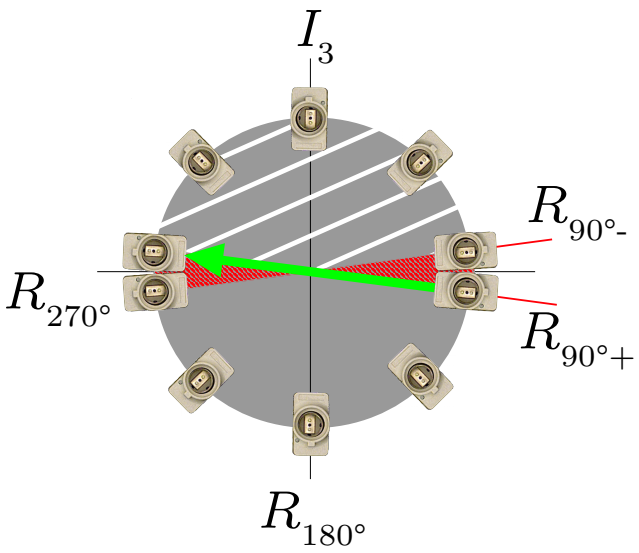


Figure 5.3: Discontinuities of $\mathcal{F}$ after applying the Map operator, for an object with one axis of symmetry and a π-symmetry. All poses are mapped to a pose in the hashed region by operator Map introduced in Section 5.2.1. Since $\text{Map}(R^z_{\pi/2+\epsilon}) = R^z_{\epsilon-\pi/2}$ (visualized by the green arrow) and $\text{Map}(R^z_{\pi/2-\epsilon}) = R^z_{\pi/2-\epsilon}$, there exists a hazardous region (in red) where $\mathcal{F}$ is discontinuous.

made of two rotations around the $\boldsymbol{z}$-axis: The identity matrix, and the rotation of angle π, and $\mathcal{M}(O) = \{I_3, R^{\boldsymbol{u}}_{\pi}\}$. If a training image is annotated with rotation $R^{\boldsymbol{z}}_{\pi/2+\epsilon}$, this rotation will be mapped by operator Map to rotation $R^{\boldsymbol{z}}_{\epsilon-\pi/2}$; If a training image is annotated with pose $R^{\boldsymbol{z}}_{\pi/2-\epsilon}$, this rotation will be mapped to itself that is $R^{\boldsymbol{z}}_{\pi/2-\epsilon}$. By making ϵ converge to 0, it can be seen that there is a discontinuity of $\mathcal{F}$ around images annotated with rotations π before mapping.

Another way of looking at the problem is to notice that images of the object annotated with rotations $R^{\boldsymbol{z}}_{\epsilon-\pi/2}$ and $R^{\boldsymbol{z}}_{\pi/2-\epsilon}$ look very similar, but with very different rotations. A Deep Network would have to learn to predict very different poses for very similar images.

5.2.4 Solving the Discontinuities

The discontinuities only occur when $\mathcal{M}$ is discrete: It can be seen from Eq. (5.22) that in the case of a generalized cylinder, the Map operator is continuous. Otherwise, we avoid these discontinuities by introducing a partition of SO(3) made of two subsets Ω_1 and Ω_2. For each subset, we train a different regressor to predict the pose. We will therefore have two regressors $\mathcal{F}_1$ and $\mathcal{F}_2$ instead of only one. In this way, both $\mathcal{F}_1$ and $\mathcal{F}_2$ will be continuous over their respective domains.

We describe below our method on an example and then extend it to the general

case.

- **One Symmetry Axis,** $M = 2$

  Let us consider again the rectangular object pictured in Figure 5.3, and already discussed in Section 5.2.3. For this object, we have $\mathcal{M}(O) = \{I_3, R^{\boldsymbol{u}}_{\pi}\}$. We can notice that $\mathcal{M}$ and Map generate a partition of SO(3) made of two subsets:

  $$\Omega_1 = \{R : \hat{S}(R) = I_3\} \text{ and } \Omega_2 = \{R : \hat{S}(R) = R^{\boldsymbol{u}}_{\pi}\}, \tag{5.23}$$

  where $\hat{S}(R)$ is the rotation of Eq. (5.4) when applying Map to R.

  However, this partition will not solve our problem: We already know that $\mathcal{F}$ is not continuous on Ω_1. We must therefore introduce a new partition of SO(3). For this partition, we consider the new set:

  $$\begin{aligned}\sqrt{\mathcal{M}}(O) &= \{(R^{\boldsymbol{u}}_{k\pi/2}) : k \in \mathbb{Z}\} \\ &= \{I_3, R^{\boldsymbol{u}}_{\pi/2}, R^{\boldsymbol{u}}_{\pi}, R^{\boldsymbol{u}}_{3\pi/2}\},\end{aligned} \tag{5.24}$$

  and the partition it generates with Map:

  $$\Omega^{(k)} = \{R : \hat{S}(R) = R^{\boldsymbol{u}}_{k\pi/2}\}. \tag{5.25}$$

  As shown in Figure 5.4, no part $\Omega^{(k)}$ includes any discontinuity. Moreover, for a rotation in $\Omega^{(2)}$, there is another rotation in $\Omega^{(0)}$ that generates the same object appearance. The same yields for $\Omega^{(3)}$ and $\Omega^{(1)}$.

  We, therefore, take $\Omega_1 = \Omega^{(0)}$ for the domain of regressor $\mathcal{F}_1$, and $\Omega_2 = \Omega^{(1)}$ for the domain of regressor $\mathcal{F}_2$. $\mathcal{F}_1$ and $\mathcal{F}_2$ thus do not suffer from discontinuities nor ambiguity. They are sufficient enough to estimate the object pose under any rotation since we can map this rotation to a rotation either in Ω_1 or Ω_2 corresponding to the same appearance. To do so, we introduce a new mapping Map$'$ derived from Map such that:

  $$\begin{aligned}&\forall R \in SO(3),\ \text{Map}'(R) = (\hat{S}^{-1}R, \delta) \quad \text{such that} \\ &(\hat{S}, \delta) = \begin{cases} (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - I_3|_F, 1) & \text{if Map}(R) \in \Omega_1\ , \\ (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - R^{\boldsymbol{u}}_{\pi/2}|_F, 2) & \text{otherwise}\ , \end{cases}\end{aligned} \tag{5.26}$$

  During training, given a training image $\boldsymbol{I}$ annotated with rotation R, we compute $(\hat{S}^{-1}R, \delta) \leftarrow \text{Map}'(R)$ and train regressor $\mathcal{F}_\delta$ to predict rotation $\hat{S}^{-1}R$ from $\boldsymbol{I}$.

  During inference, given a test image $\boldsymbol{I}$ of an object, we need to know which regressor we should invoke to predict the pose. To do so, during training, we

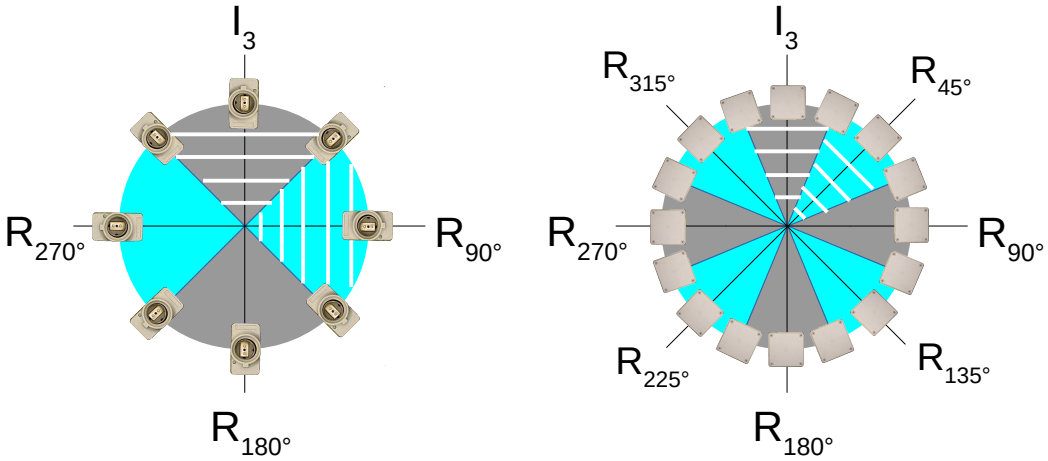


Figure 5.4: Partitions for an object with one axis of symmetry with $M = 2$ (**left**) and $M = 4$ (**right**) as defined in Section 5.2.4. Rotations in areas filled with one color should be mapped to a rotation in the hashed region of the same color to avoid discontinuities. Two different regressors $\mathcal{F}_1$ and $\mathcal{F}_2$, one for each color, are used to predict poses for each hashed region.

train a classifier $\mathcal{C}$ to predict which regressor we should invoke to compute the pose, that is we train $\mathcal{C}$ to predict δ from $\boldsymbol{I}$. For rotations close to the boundary between Ω_1 and Ω_2, the prediction for $\mathcal{C}$ can become ambiguous. However, in this case, the ambiguity is not a problem in practice: Even if the classifier predicts the wrong regressor to use close to the boundary between Ω_1 and Ω_2, both regressors can correctly predict poses close to this boundary.

- **One Symmetry Axis, Arbitrary** M

  Let us now generalize to an object O with an arbitrary amount of symmetries around a single axis u. These symmetries are necessarily periodic around $\boldsymbol{u}$ with angular period $f_\alpha = 2\pi/M$: Rotating O around u by any angle multiple of f_α does not change its appearance. The proper symmetry group $\mathcal{M}(O)$ for such an object is:

$$\mathcal{M}(O) = \left\{\left(R^{\boldsymbol{u}}_{2\pi/M}\right)^m\right\}_{m\in\mathbb{N}} = \{R^{\boldsymbol{u}}_{2m\pi/M}\}_{m\in\mathbb{N}}. \tag{5.27}$$

  $\sqrt{\mathcal{M}}(O)$ of Eq. (5.24)) becomes:

$$\sqrt{\mathcal{M}}(O) = \left\{\left(R^{\boldsymbol{u}}_{\pi/M}\right)^m\right\}_{k\in\mathbb{N}} = \{R^{\boldsymbol{u}}_{m\pi/M}\}_{m\in\mathbb{N}}, \tag{5.28}$$

and mapping Map$'$ of Eq. (5.26) becomes:

$$\forall R \in SO(3), \quad \text{Map}'(R) = (\hat{S}^{-1}R, \delta) \quad \text{such that}$$
$$(\hat{S}, \delta) = \begin{cases} (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - I_3|_F, 1) & \text{if Map}(R) \in \Omega_1 \ , \\ (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - R^{\boldsymbol{u}}_{\pi/M}|_F, 2) & \text{otherwise} \ , \end{cases} \tag{5.29}$$

where $\Omega_1 = \{R : \hat{S}(R) = I_3\}$.

We can use Map$'$ the same way as in the previous subsection to train and use to regressors $\mathcal{F}_1$ and $\mathcal{F}_2$.

- **General Case**

  In the general case, each rotation R in $\mathcal{M}$ can be written in the form:

$$R = R^{\boldsymbol{u}}_{2\pi/M}.R^{\boldsymbol{v}}_{2\pi/N}.... \quad \text{with } M, N, .. \in \mathbb{N}, \tag{5.30}$$

  where $\boldsymbol{u}$, $\boldsymbol{v}$, etc. are rotation axes. Most common objects have at most 2 axes of symmetries, but it is possible to imagine objects with more, for example, a golf ball. To keep the notations as simple as possible, we will stick to only two axes, as it is easy to extend to more axes from there.

  $\sqrt{\mathcal{M}}(O)$ becomes:

$$\sqrt{\mathcal{M}}(O) = \{R^{\boldsymbol{u}}_{m\pi/M}.R^{\boldsymbol{u}}_{n\pi/N}\}_{(m,n)\in\mathbb{N}^2}, \tag{5.31}$$

  and mapping Map$'$ becomes:

$$\forall R \in SO(3), \text{Map}'(R) = (\hat{S}^{-1}R, \delta_1, \delta_2) \text{s.t. } (\hat{S}, \delta_1, \delta_2) =$$
$$\begin{cases} (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - I_3|_F, 1, 1) \text{ if Map}(R) \in \Omega_{1,1} \ , \\ (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - R^{\boldsymbol{u}}_{\pi/M}|_F, 2, 1) \text{ if Map}(R) \in \Omega_{2,1} \ , \\ (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - R^{\boldsymbol{v}}_{\pi/N}|_F, 1, 2) \text{ if Map}(R) \in \Omega_{1,2} \ , \\ (\underset{S \in \mathcal{M}(O)}{\arg\min} |S^{-1}R - R^{\boldsymbol{u}}_{\pi/M}R^{\boldsymbol{v}}_{\pi/N}|_F, 2, 2) \text{ otherwise} \end{cases} \tag{5.32}$$

  where $\Omega_{1,1} = \{R : \hat{S}(R) = I_3\}$, $\Omega_{2,1} = \{R : \hat{S}(R) = R^{\boldsymbol{u}}_{\pi/M}\}$, and $\Omega_{1,2} = \{R : \hat{S}(R) = R^{\boldsymbol{v}}_{\pi/N}\}$. It means that in this case, we have to train 4 different regressors $\mathcal{F}_{1,1}$, $\mathcal{F}_{2,1}$, $\mathcal{F}_{1,2}$, and $\mathcal{F}_{2,2}$ according to δ_1 and δ_2, and the classifier $\mathcal{C}$ to predict a class index in $[0; 3]$.

5.2.5 Method Summary

We can summarize our method developed above as follow. We distinguish between generalized cylinders and objects with discrete symmetries.

- **Generalized cylinders:** Given a training image $\boldsymbol{I}$ annotated with rotation R, we train a single regressor $\mathcal{F}$ to predict $\text{Map}(R)$ using Eq. 5.3 from $\boldsymbol{I}$. At inference time, given a test image $\boldsymbol{I}$, we simply have to invoke $\mathcal{F}$ to predict the object pose from $\boldsymbol{I}$.

- **Objects with discrete symmetries:** Given a training image $\boldsymbol{I}$ annotated with rotation R, we apply Map' to R using Eq. (5.29) or Eq. (5.32) depending on the number of symmetry axes. Map' provides the rotation to be associated with $\boldsymbol{I}$ for training, as well as the index of the regressor $\mathcal{F}_i$ to train. In addition to training the regressors, we need to train classifier $\mathcal{C}$ to predict the index of the regressor to use. At inference time, we first invoke classifier $\mathcal{C}$ to predict which regressor we should use from $\boldsymbol{I}$, and then, invoke this regressor to predict the object pose from $\boldsymbol{I}$.

5.3 Framework: a Faster-RCNN based architecture

Our proposed method is general and it can be integrated into any 6D pose estimation algorithm to improve the performances on symmetrical objects. As we discussed in Chapter 3, there are plenty of deep learning-based 6D pose estimation pipelines proposed in recent years and a lot of different modalities. We decide to integrate our method into a correspondence-based approach. We, therefore, rely on a Deep Network to predict the 2D reprojections of some 3D points on the CAD models of the objects. In particular, we use the same setting as in Rad and Lepetit (2017); Tekin et al. (2018); Tremblay et al. (2018); Peng et al. (2019a) for simplicity and we predict the objects' 6D poses in the form of the 2D reprojections of the 8 corners of the 3D bounding boxes. From these 2D reprojections, it is possible to estimate a 6D pose using a PnP. However, our approach to handle symmetries is general, and using any other representation of the pose, with quaternions for example, is also possible. As a Deep Network to predict the 2D reprojections we rely on Faster R-CNN Ren et al. (2015), a powerful object detector. To explain our framework, it is essential to give a brief overview of Faster R-CNN since its architecture is very complex, as shown in Figure 5.5.

5.3.1 Faster R-CNN

The Faster R-CNN architecture was developed to solve the task of object detection, that can be explained in the following way:

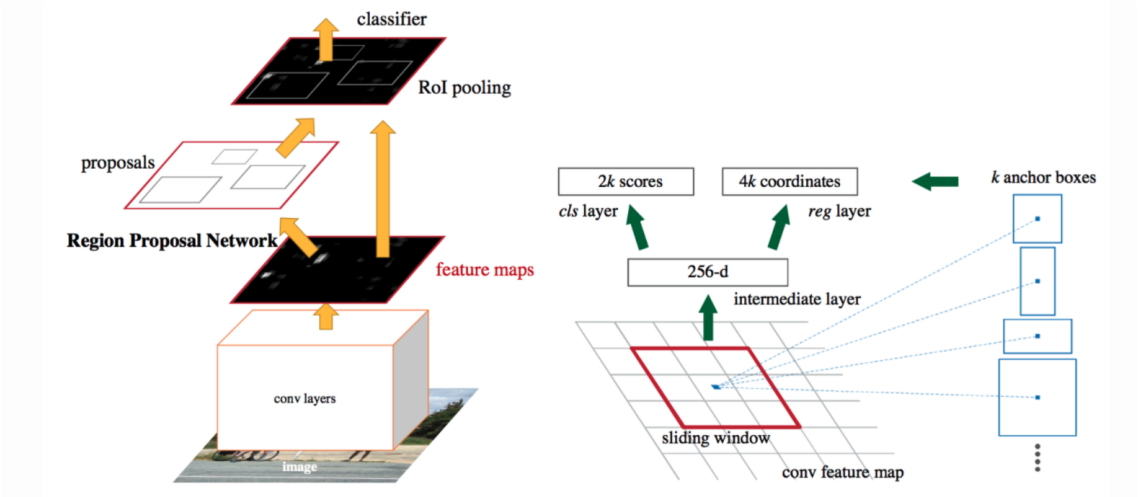


Figure 5.5: Overview of Faster R-CNN architecture. The image has been taken from Ren et al. (2015).

Given a color image, the network outputs a list of object classes and where these objects are located in the image in the form of the coordinates of the objects bounding boxes.

The input image is passed through a pre-trained CNN up until an intermediate layer, ending up with a convolutional feature map. This CNN acts as a feature extractor. This is a common practice used in the context of *Transfer Learning*, especially when the network is trained on a small dataset using the weights of a network pre-trained on a larger one. This feature map is fed into a Region Proposal Networks (RPN) which finds a predefined number of region proposals that may contain objects using the concept of anchors. Anchors are offline configured bounding boxes of a fixed size which are placed uniformly throughout the original image, as shown in Figure 5.5. The RPN outputs the probability of each anchor to contain an object and, in that case, it refines the bounding box location around the object. Using the features extracted by the CNN and the bounding boxes with relevant objects, a layer called Region of Interest (ROI) Pooling extracts those features inside the bounding boxes. These features are then fed into the last step of Faster R-CNN, the Region-based Convolutional Neural Network (R-CNN), where a classifier predicts the label of the object (or classify it as background if there is not an object), and a regressor adjusts the bounding box of the proposal and localize the object more accurately according to the predicted class.

The complete model is trained in a multi-task way. The loss is a weighted sum of 4 different loss terms: a classification and a regression loss for both the RPN and R-CNN steps. The first term is the RPN classification loss, a binary cross-entropy, to say if an anchor contains an object or not. The second term is the RPN regression

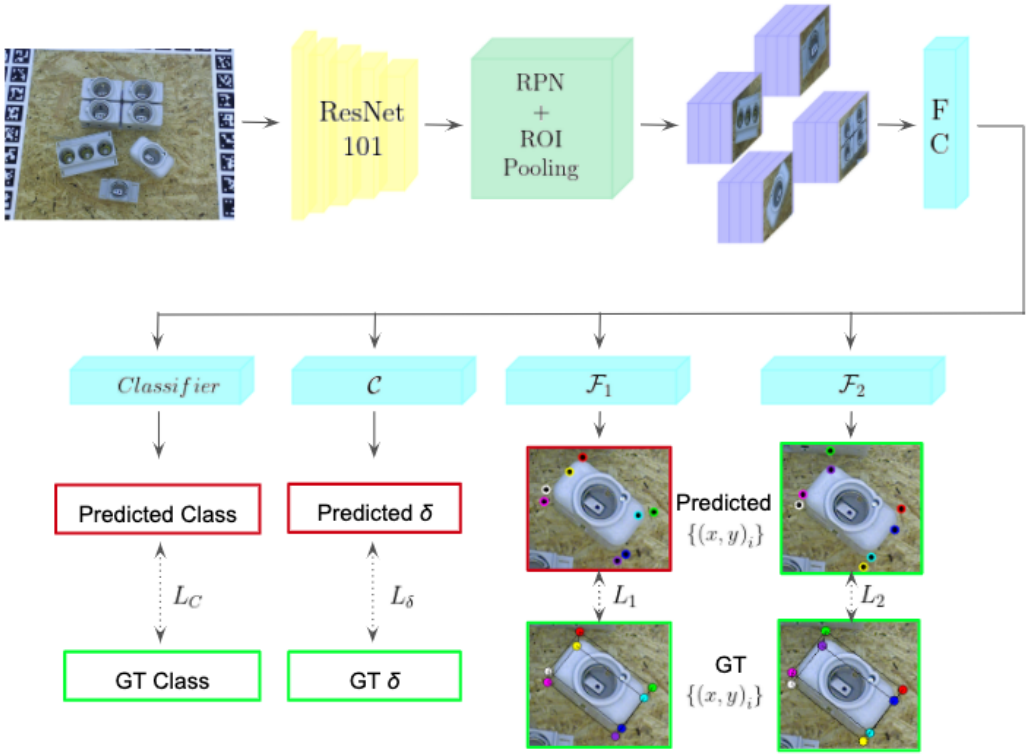


Figure 5.6: Our architecture for implementing our approach. It is built on top of the Faster R-CNN architecture, to which we add specific branches: One for each regressor $\mathcal{F}_i$, and one for classifier $\mathcal{C}$ to learn to choose between the regressors. In this example, the object has only one angle of symmetry around one axis. For this reason, only two regressors $\mathcal{F}_1$ and $\mathcal{F}_2$ are needed and the aim of the classifier $\mathcal{C}$ is to choose which regressor to use.

loss to adjust the anchor bounding box. This proposed loss is a smoothed L1-norm, which is a L1 loss when the error is under a certain threshold σ. To compute this loss only positive anchors, that means anchors that contain an object, are used. An anchor contains an object if its *Intersection over Union (IoU)* with its relative ground truth is higher than a threshold. The smooth L1 loss is also used as R-CNN regression loss, while cross-entropy is used for the R-CNN classification.

5.3.2 Faster R-CNN based framework

We integrate our normalization method into the Faster R-CNN architecture. We add some branches in the R-CNN part after the ROI Pooling, as shown in Figure 5.6 and detailed below.

Pose regressor $\mathcal{F}$ branch. We add a specific branch to the Faster R-CNN architecture to predict the 2D coordinates of each 3D corner for each regressor. This branch is implemented as a fully connected multi-layer perceptron and takes as input the output shared single-channel feature-map extracted with the ROI Pooling. Without taking into account any symmetry, the output of this branch would have size $n_{points} \times 2 \times n_{classes}$, where n_{points} is the number of 3D points whose 2D reprojections we want to predict and $n_{classes}$ corresponds to the number of objects in the dataset we use for training and evaluating the method. In our case $n_{points} = 8$ (the 3D corners of the 3D object's bounding box) and $n_{classes} = 30$ (the number of the objects in T-LESS). In our method, to handle the symmetries, the branch has size $n_{points} \times 2 \times n_{classes} \times n_{regressors}$ where $n_{regressors}$ corresponds to the number of regressors $\mathcal{F}_i$ is used.

Classifier $\mathcal{C}$ branch. We also add a specific multi-layer perceptron branch to Faster R-CNN to implement the classifier $\mathcal{C}$. The output size of this branch is $n_{classes} \times n_{regressors}$. Training data for this branch is obtained using Eq. (5.26).

We train our method in a multi-task learning way by adding the new task-attached loss, with a respective weight, to the global loss term of Faster R-CNN. More precisely, we include in the global loss term the loss functions $\mathcal{L}_1$ used for training the two regressors $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively, and the classifier loss $\mathcal{L}_\delta$ for the regressor classifier $\mathcal{C}$. For $\mathcal{L}_1$ we used smoothed L1-norm, while for the $\mathcal{C}$ we took the logistic regression function:

$$\mathcal{L}_\delta = -(p^* \cdot log(p) + (1 - p^*) \cdot log(1 - p)) \tag{5.33}$$

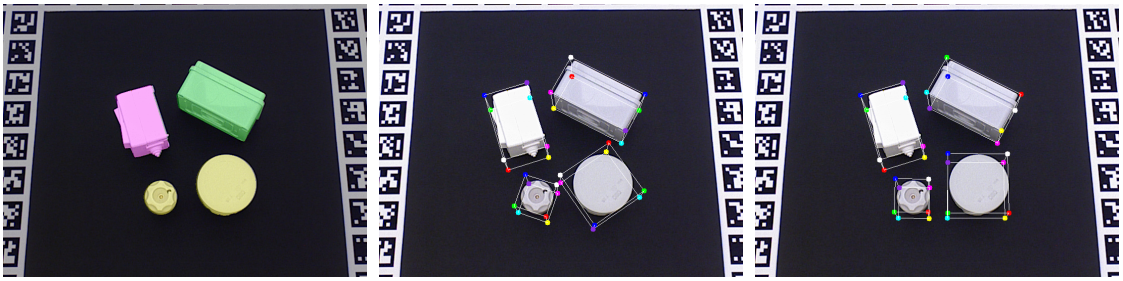
where p is the output of the last sigmoid layer of classifier $\mathcal{C}$ and p^* its associated ground truth. Regressors 1 and 2 are taken as classes 0 and 1 respectively, hence $p* = 1$ corresponds to using regressor 2 for pose estimation. At training time, for each iteration, we used the ground truth value of the classifier $\mathcal{C}$, thus knowing which regressor should be trained. We implemented this choice with the help of a binary mask applied to the outputs of the two regressors.

5.3.3 Ground truth generation

To generate ground truth data, we use prior knowledge on the symmetries of the objects, reported in Table 5.1. We compute the partition of $SO(3)$ space based on the object's symmetry type and we get the corresponding number of regressors $\mathcal{F}_i$. We apply the mapping Map$'$ to the object's rotation. We then project the 3D corners of the object bounding boxes with the normalized object's pose to get the 2D locations in the image as shown in Figure 5.7.

No symmetries		21, 22, 24, 25, 26
Continuous symmetry		1, 2, 3, 4, 13, 14, 15, 16, 17, 18, 24, 30
Discrete symmetries	180°, $\boldsymbol{z}$-axis	5, 6, 7, 8, 9, 10, 11, 12, 29
	180°, $\boldsymbol{y}$-axis	19, 20, 23
	90°, $\boldsymbol{z}$-axis	27, 28

Table 5.1: Prior knowledge of symmetries of the T-LESS objects.



(a) (b) (c)

Figure 5.7: Ground truths generation for training the network for a different type of symmetrical objects. (a) Input RGB image. The pink object has not symmetries, the green one is symmetrical along the $\boldsymbol{z}$-axis of 90° and the yellow ones have a continuous symmetry of 2π along the $\boldsymbol{z}$-axis. (b) Ground truths of the 2D locations of the 3D bounding box corners without any normalization. (c) Ground truths after applying the normalization procedure.

5.4 Evaluation

In this section, we detail how we evaluated our approach, and show its effectiveness on objects with various types of symmetries. As mentioned in Section 4.6, the evaluation is done on the T-LESS dataset while the training of the network has been done using images of Synthetic T-LESS.

Firstly, we show the impact of our normalization procedure and our simultaneous learning of ambiguous poses and analyze how they helped stabilize the learning process. As shown in Figure 5.8, the loss of our Faster R-CNN-based implementation converges only when the rotations are normalized using our normalization procedure, indicating that something is incorrect in the loss function in absence of normalization. In this case, the loss continues to oscillate around the average value of all possible poses.
In Figure 5.9, we show what happens in practice for three possible types of objects: Two generalized cylinders (objects 30 and 3), an object with an axis of symmetry (object 29), and an object without any symmetry (object 26). When dealing with non-symmetrical objects, the network can learn the 6D pose with and without the

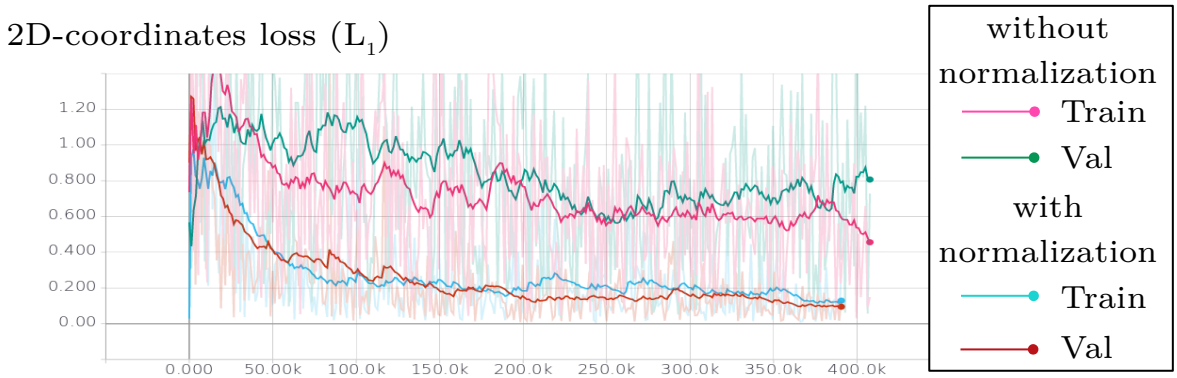


Figure 5.8: Learning curves on the training and validation sets of our Faster R-CNN based implementation. Without our normalization described in Section 5.2, the network fails to converge to a satisfying solution. More exactly, it converges to a local minimum where all keypoints collapse at the center of the object—see Figure 5.9.

normalization procedure. On the opposite, when the objects are symmetrical, without our normalization, the network learns the average between all the possible poses ending up predicting a pose collapsed to the center of the object.

We now evaluate our method using the *Visible Surface Discrepancy* (VSD) error function introduced by Hodaň et al. (2018).

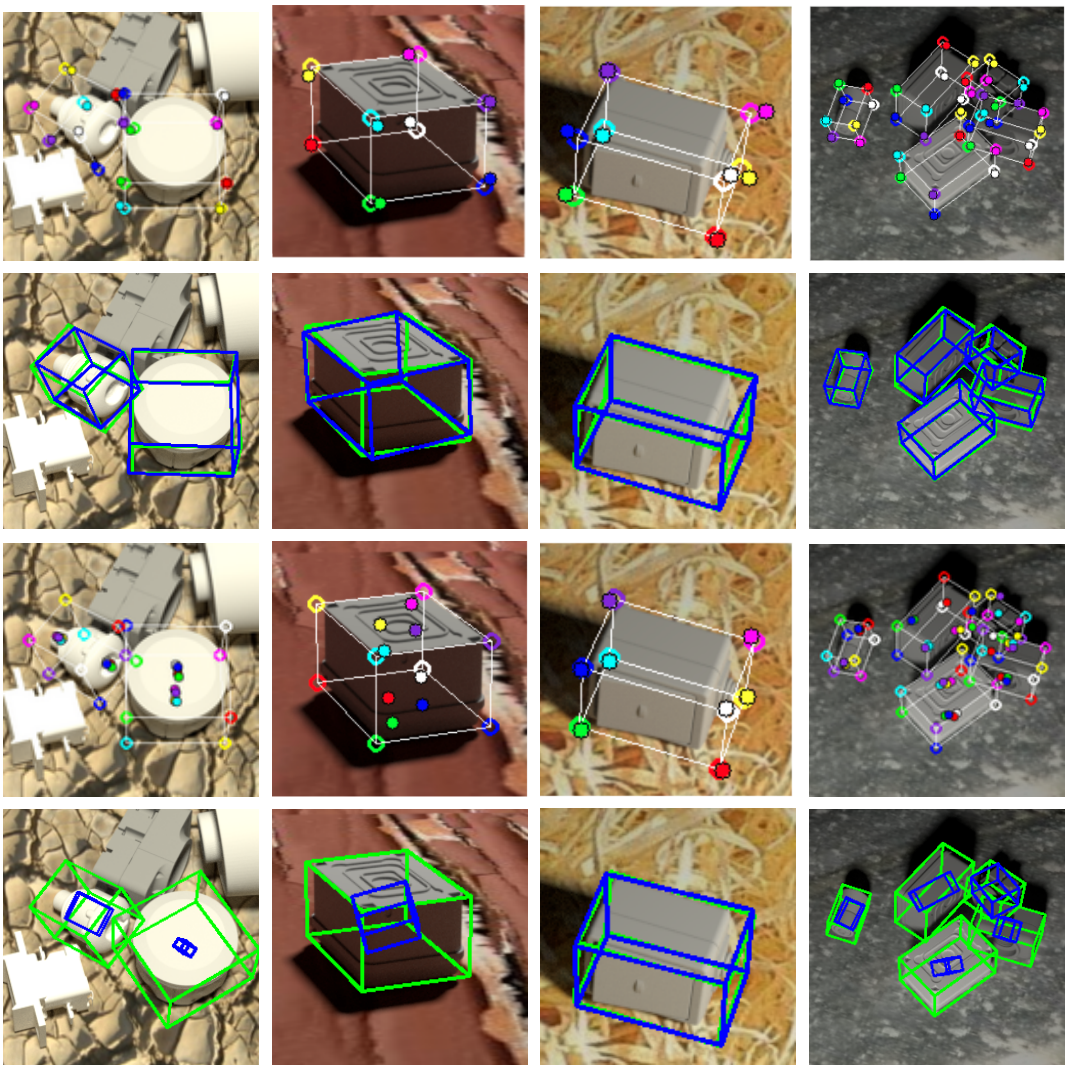
The VSD metric evaluates the pose error in a way that is invariant to the pose ambiguities due to object symmetries. It is computed from the distance between the estimated and ground truth visible object surfaces in the following way:

$$err_{\text{VSD}}(\hat{S}, \bar{S}, S_I, \hat{V}, \bar{V}, \tau) = \underset{p \in \hat{V} \bigcup \bar{V}}{\text{Mean}} \begin{cases} 0, & \text{if p} \in \hat{V} \bigcap \bar{V} \wedge |\hat{S}(p) - \bar{S}(p)| < \tau \\ 1, & \text{otherwise} \end{cases} \tag{5.34}$$

where $\hat{S}$ and $\bar{S}$ are distance maps obtained by rendering the object model in the estimated and ground truth poses respectively. The distance maps are compared with the distance map S_I of the test image I to obtain the visibility masks $\hat{V}$ and $\bar{V}$, that are the sets of pixels where the object model is visible in image I, as shown in Figure 5.10. In Table 5.2, we report the mean VSD recall of 6D object poses at $err_{\text{VSD}} < 0.3$ with tolerance $\tau = 20mm$ and $> 10\%$ object visibility and we compare our method to the method of Sundermeyer et al. (2018).

The object 3D orientation and translation along the $\boldsymbol{x}$-and $\boldsymbol{y}$-axes are typically well estimated. Although most of the translation error is along the $\boldsymbol{z}$-axis, it is unsurprising since we do not use or regress the depth information. To have a meaningful evaluation of our results in terms of VSD, we keep the ground truth of the translation along the $\boldsymbol{z}$-axis in our pose predictions.

Finally, in Figure 5.11 we show some qualitative results on T-LESS test scenes.



(a) (b) (c) (d)

Figure 5.9: Pose estimation results with (**first and second rows**) and without (**third and fourth row**) our normalization approach for (a) generalized cylinders, (b) an object with an axis of symmetry, (c) an object without any symmetry, and (d) a typical scene from our Synthetic T-LESS dataset. The first and third rows show the 2D reprojections of the 3D control points prediction while the second and fourth rows the pose estimation results. The green and blue bounding boxes correspond to the ground truth and estimated poses respectively. Without our normalization, the network learns to predict the average between all the possible poses for symmetrical objects, which is of course meaningless.

5.5 Conclusion

In this chapter, we studied the subtle problems that arise when training a machine learning method to predict the 6D pose of an object with symmetries from color

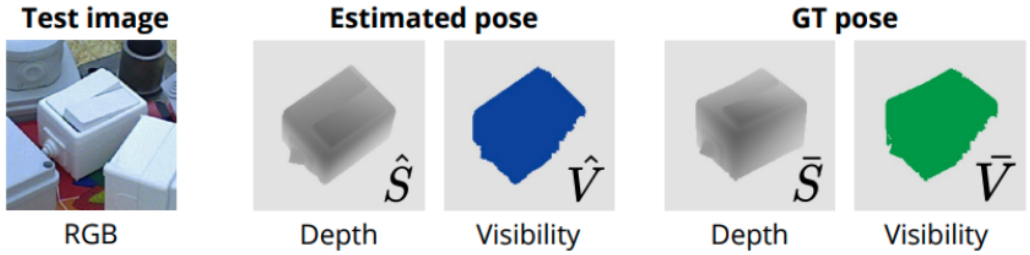


Figure 5.10: Quantities used in the calculation of VSD when no depth information is used. **Left:** Input RGB image. **Center:** Distance map $\hat{S}$ and visibility mask $\hat{V}$ obtained by rendering the object with the estimated pose. **Right:** $\bar{S}$ and visibility mask $\bar{V}$ obtained by rendering the object with the ground truth pose. The images have been taken from Hodaň et al. (2018).

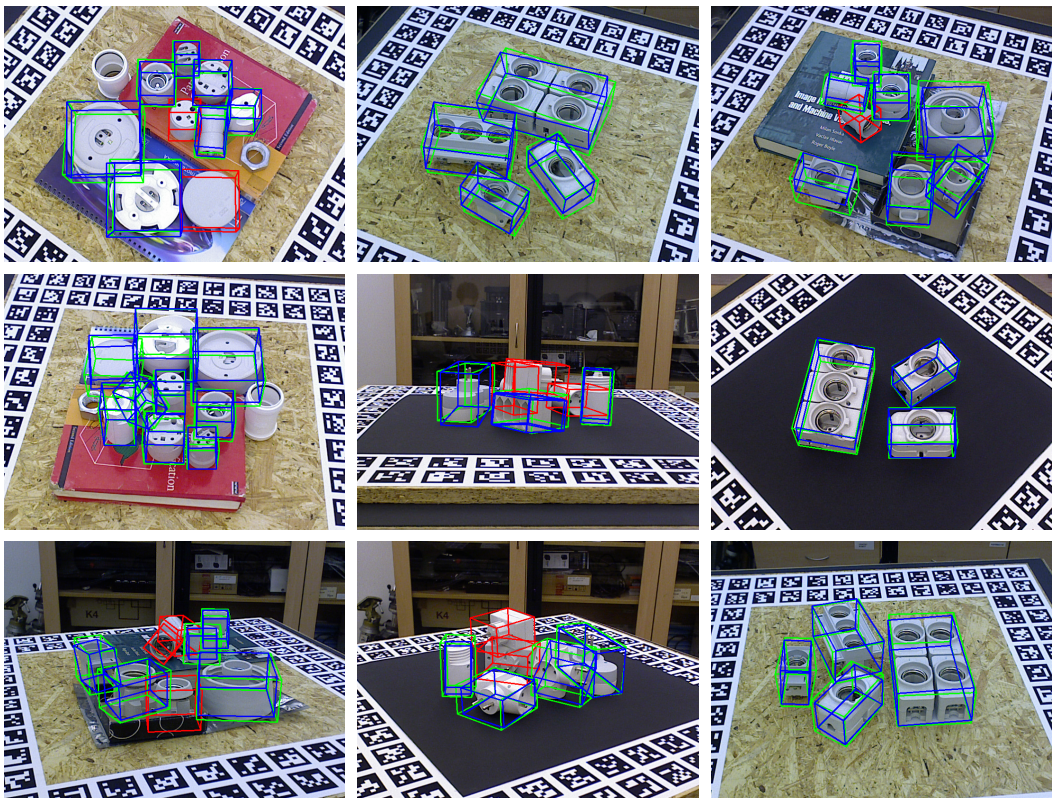


Figure 5.11: Some qualitative results on test scenes of the T-LESS dataset. Green and blue bounding boxes are the ground truth and estimated poses respectively while the red bounding boxes correspond to missed detections.

images. We presented a simple and analytic method based on the normalization

	Sundermeyer *et al.* Sundermeyer et al. (2018)			Ours
Object	SSD	Retina	GT BBox	Faster R-CNN
1	5.65	8.87	12.33	26.35
2	5.46	13.22	11.23	56.14
3	7.05	12.47	13.11	83.33
4	4.61	6.56	12.71	32.98
5	36.45	34.80	66.70	44.54
6	23.15	20.24	52.30	98.33
7	15.97	16.21	36.58	87.74
8	10.86	19.74	22.05	17.09
9	19.59	36.21	46.49	52.54
10	10.47	11.55	14.31	5.43
11	4.35	6.31	15.01	27.97
12	7.80	8.15	31.34	43.08
13	3.30	4.91	13.60	48.54
14	2.85	4.61	45.32	42.19
15	7.90	26.71	50.00	47.10
16	13.06	21.73	36.09	42.18
17	41.70	64.84	81.11	56.83
18	47.17	14.30	52.62	19.31
19	15.95	22.46	50.75	27.53
20	2.17	5.27	37.75	32.16
21	19.77	17.93	50.89	41.19
22	11.01	18.63	47.60	49.10
23	7.98	18.63	35.18	26.08
24	4.74	4.23	11.24	41.34
25	21.91	18.76	37.12	44.37
26	10.04	12.62	28.33	23.80
27	7.42	21.13	21.86	33.78
28	21.78	23.07	42.58	35.10
29	15.33	26.65	57.01	15.92
30	34.63	29.58	70.42	36.17
Mean	14.67	18.35	36.79	41.27

Table 5.2: T-LESS: Object recall for $err_{vsd} < 0.3$ on all Primesense test scenes (the higher the better).

of the pose rotation that is agnostic to the exact pose representation and the pose prediction model. Our method can therefore be included in current and future developments for properly handling objects with symmetries.

One main limitation of the system is that the prior knowledge of the object symmetries is needed. A direct extension of our work could be to automatically detect the object symmetries. We will discuss it in Section 8.2. In the next chapter, we will present an approach able to face another challenge in 3D pose estimation: the generalization to unseen objects.

Chapter 6

CorNet: 6D Pose Estimation of Unseen Objects using Generic 3D Corners

Table of contents

In this chapter, we address another main challenge in 3D object pose estimation discussed in Section 1.3: the generalization of deep learning-based methods to *unseen* objects, that means objects never seen during the training, that is also linked to the problem of scalability of these methods. In this chapter, we develop a first approach to estimate the 3D pose of new target unseen objects that does not require additional learning nor training images for new objects but only the CAD models for the target objects. We first give an overview of the problem tackled and then we explain our method. We detail the 3D object pose representation we use and how we can exploit single parts of the object to obtain the full 3D pose, by tacking care of all the ambiguities that raise because of the symmetries of the object's parts. We finally evaluate our approach on some scenes of T-LESS.

6.1 Overview

DL approaches achieve great performance in 3D object pose estimation, at least when enough training images are available under different poses. A model trained on some objects is not going to perform as well on other objects, which were not part of the training samples. This is a significant limitation especially in industrial contexts where new target objects are often present and it is not easy to have a lot of training images of them. Even if domain transfer methods allow for training such methods with synthetic images (Hinterstoisser et al.; Kehl et al., 2017; Sundermeyer et al., 2018) instead of real ones (Bousmalis et al., 2017; Zhu et al., 2017; Ganin et al., 2016; Long et al., 2015; Tzeng et al., 2015; Lee et al., 2018; Rad et al., 2018a; Zakharov et al., 2018) at least to some extent, such training sessions take time, and it is highly desirable to avoid them in practice, especially in industrial contexts.

In this chapter, we develop a first approach to estimate the 3D pose of new target unseen objects. The proposed method does not require additional learning nor training images for new objects and we consider a scenario where CAD models for the target objects exist, but not necessarily training images. This is often the case in industrial settings, where an object is built from its CAD model. We rely on corners that we learn to detect and estimate the 3D poses during an offline stage. Our approach focuses on industrial objects. Industrial objects are often made of similar parts, and corners are a dominant common part, as shown in Figure 6.1.

Detecting these corners and determining their 3D poses is the basis for our approach. We follow a deep learning approach and train Faster R-CNN on a small set of objects to detect corners and predict their 3D poses. We use the representation of 3D poses introduced by Crivellaro et al. (2018): The 3D pose of a corner is predicted in the form of a set of 2D reprojections of 3D virtual points, and this is convenient for our purpose since multiple corners can be easily combined to compute the object pose when using this representation. Furthermore, this approach is robust to occlusions and in fact, thanks to this pose representation we only need 1 or 2 corners to predict the pose of the object.

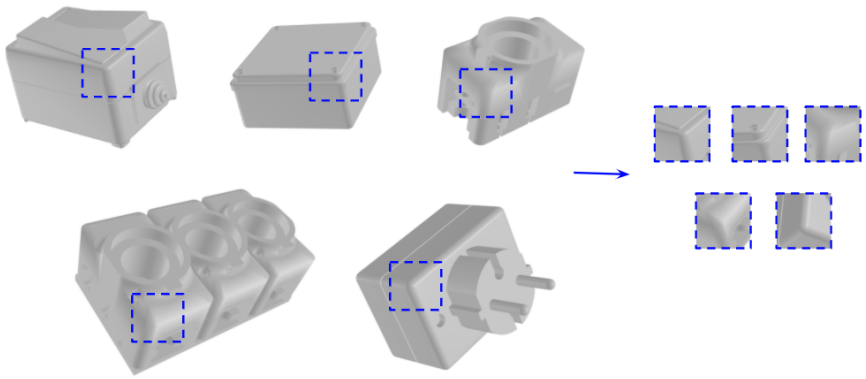


Figure 6.1: Example of industrial CAD models with similar corners.

However, we need to take care of a challenge that arises with corners, and that was ignored in Crivellaro et al. (2018): Because of its symmetries, the 3D pose of a corner is often ambiguous and defined only up a set of rigid rotations. We, therefore, introduce a robust and efficient algorithm that considers the multiple possible 3D poses of the detected corners, to estimate the 3D poses of the new objects finally.

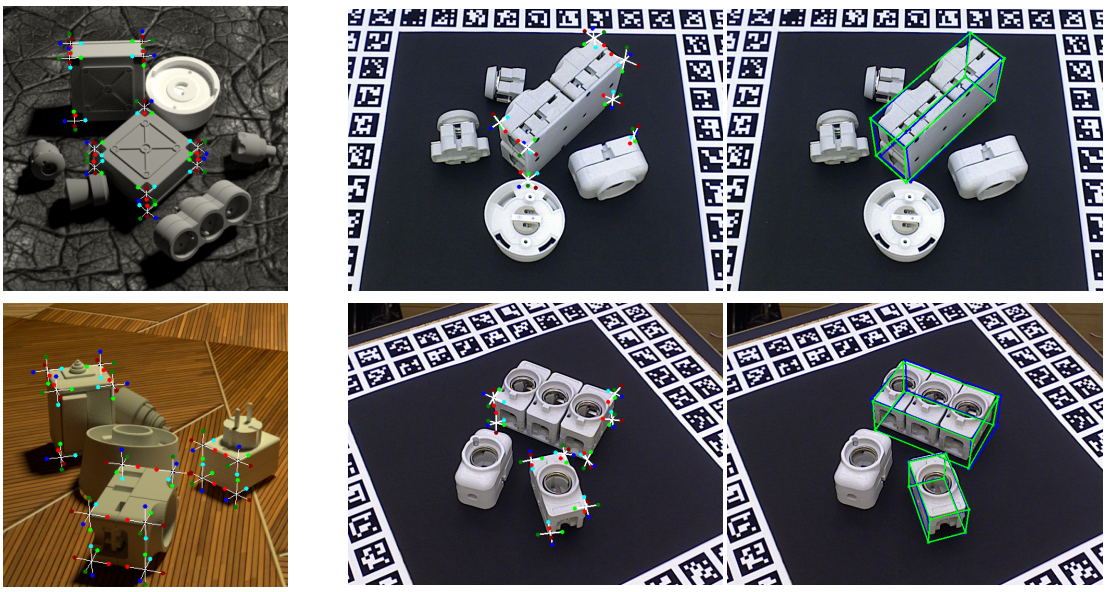
Contributions. The contributions of this chapter are:

- We exploit the 3D pose representation of Crivellaro et al. (2018) and extend it in the case of multiple corners by handling the ambiguities caused by their intrinsic symmetries.
- We propose a new approach to estimate the 3D poses of unseen objects with prominent corners from RGB images.

The work presented in this chapter was presented at the *6th International Workshop on Recovering 6D Object Pose* of the *International Conference on Computer Vision*, 2019 (Pitteri et al., 2019).

6.2 Method

In this section, we will describe our approach. We first describe how we learn to detect corners and predict their 3D poses. We then present our algorithm to estimate the 3D poses of new objects in an input image, from the corners detected in this image. Our pipeline, made of a corner detection block and a pose block, is represented in Figure 6.3.



(a) (b)

Figure 6.2: Given a small set of objects from the T-LESS dataset Hodaň et al. (2017), we learn to detect corners of various appearances and shapes and to estimate their 3D poses using synthetic renderings (a). Then, given only the CAD model of new objects with corners, we can detect these objects and estimate their 3D poses, *without any new training phase* (b). The green bounding boxes correspond to the ground truth poses and the blue bounding boxes to the poses estimated with our method.

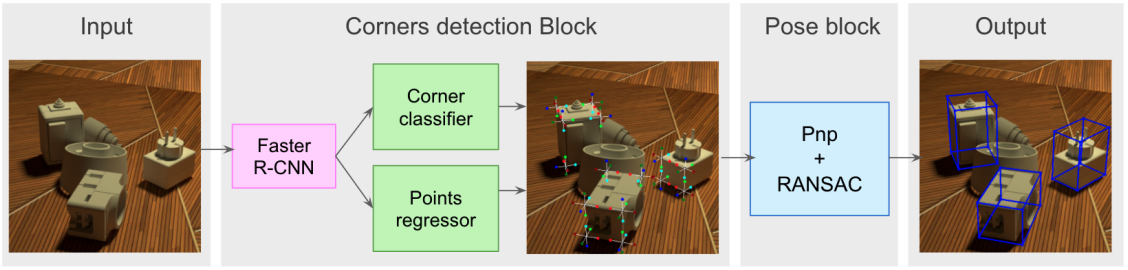
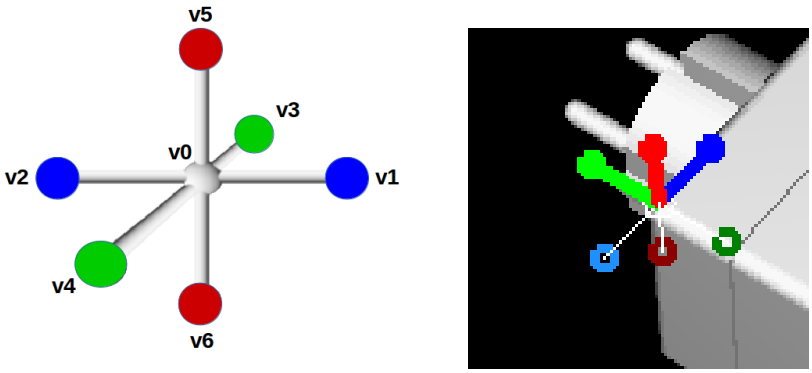


Figure 6.3: Overview of our approach. We modified Faster R-CNN to detect generic corners in images and predict their 3D poses. Our pose estimation algorithm, which is an extension of RANSAC, estimates the 3D poses of full objects from these detections.

6.2.1 Corner Detection and 3D Pose Estimation

We use the representation of the 3D pose of a part introduced in Crivellaro et al. (2018) to represent the 3D pose of our corners. This representation is made of the 2D reprojections of a set of 3D control points. Its main advantage is that it is easy to combine the 3D poses of multiple parts to compute the 3D pose of the object by



(a) (b)

Figure 6.4: 3D pose representation of an object part from Crivellaro et al. (2018). (a) Seven 3D control points arranged to span 3 orthogonal directions are assigned to each part. (b) Given an image patch of the part, Crivellaro et al. (2018) predicts the 2D reprojections of these control points and computes the 3D pose of the objects from these 3D-2D correspondences.

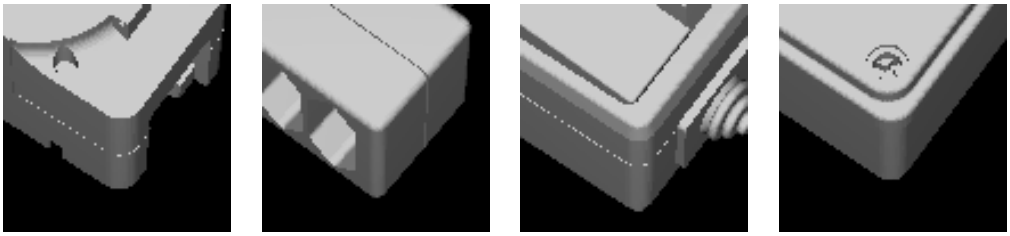


Figure 6.5: The difference with Crivellaro et al. (2018) is that our corners are *generic* in the sense that they can correspond to corners of various shapes and appearances, as corners from different objects can be different, while Crivellaro et al. (2018) considers parts from object instances. This allows us to consider new objects without retraining. The other difference is that we handle pose ambiguities, which occur in the case of corners because of their symmetries.

solving a PnP problem. These control points are only "virtual", in the sense that they do not have to correspond to specific image features. As shown in Figure 6.4, we consider seven 3D control points for each part, arranged to span 3 orthogonal directions and the center of the part, as in Crivellaro et al. (2018).

While Crivellaro et al. (2018) performed detection and pose prediction with two separate networks, we rely on the Faster R-CNN framework as it is common practice now for various problems: We kept the original architecture to obtain region proposals that correspond to parts and added a specific branch to predict the 2D coordinates of each control point. This branch is implemented as a fully connected

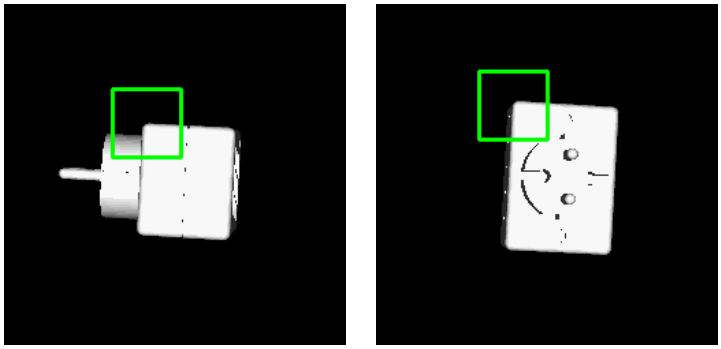


Figure 6.6: The same corner can look the same under different 3D poses. This implies that it is possible to predict the 3D pose of a corner only up to some rigid motions.

two-layer perceptron. The size of its output is $2 \times N_v$, where N_v is the number of control points for a detected corner, and with $N_v = 7$ in practice. For training, we used the default hyperparameters used in Ren et al. (2015) and the same loss function to predict the object class (corner vs background). We also added to the global loss term of Faster R-CNN a squared loss for learning to predict the reprojections of the control points. Because of the symmetries of corners, this part requires some care and we discuss it in the next subsection.

To train Faster R-CNN, we used a small number of objects exhibiting different types of corners, shown in Figure 6.5, and created synthetic images of these objects for training. Two examples of these images are shown in the first row of Figure 6.2. These images are created by randomly placing the training objects in a simple scene made of a plane randomly textured, and randomly lighted, similar to what we did for our Synthetic T-LESS in section 4.6.1. The only difference with our Synthetic T-LESS is that we select only the objects with prominent corners from T-LESS. In practice, we noticed that we did not need to apply transfer learning to take care of the domain gap between our synthetic images and the real test images of T-LESS. This is probably because we consider only local parts of the images, and because the test images of T-LESS are relatively noise-free. Given the CAD models of these objects, we can select the control points in 3D and project them in the synthetic images using their ground truth poses. In this way, we obtain the 2D ground truth reprojections of the control points needed to train the network.

6.2.2 Ambiguities between Corner Poses and How to Handle Them

As shown in Figure 6.6, many ambiguities happen when trying to predict the 3D pose of a corner from its appearance. Such ambiguities do not happen in the problems considered by Crivellaro et al. (2018) and are due to the symmetries of corners. Figure 6.7 shows that, given the image of a corner, there are in general 3 possible

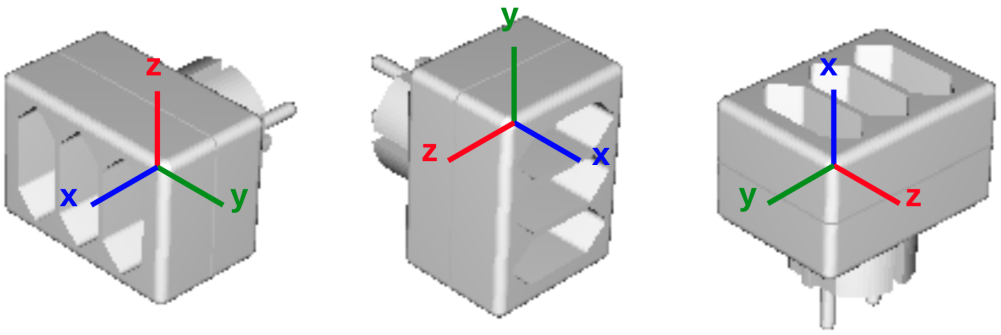


Figure 6.7: Given the image of a corner, there are in general 3 possible 3D poses that correspond to this image. Therefore, three arrangements of 3D virtual points are possible.

3D poses that correspond to this image. A standard squared loss would focus on the annotated pose, and penalize the two other possible poses.

More exactly, from one possible 3D pose $\boldsymbol{p}$, it is possible to generate the two other poses by applying rotations around the corner. In our case, since we represent the pose with the 2D reprojections of the virtual points, this can also be done by permuting the 2D reprojections properly. We, therefore, introduce two permutations Σ_1 and Σ_2 which operate on the 2D reprojections and train Faster R-CNN to predict the virtual point reprojections by introducing for each training image a term motivated by Xiang et al. (2018):

$$\mathcal{L}_p = \min_{\boldsymbol{p} \in \{\boldsymbol{p}_{\text{ann}}, \Sigma_1(\boldsymbol{p}_{\text{ann}}), \Sigma_2(\boldsymbol{p}_{\text{ann}})\}} |\boldsymbol{p}_{\text{pred}} - \boldsymbol{p}|^2 \,, \tag{6.1}$$

where $\boldsymbol{p}_{\text{ann}}$ and $\boldsymbol{p}_{\text{pred}}$ are the annotated and predicted poses respectively for the training image.

Given a pose predicted by Faster R-CNN, we can generate the 2 other possible poses by applying Σ_1 and Σ_2. This is used in our pose estimation algorithm described in the next subsection.

6.2.3 Pose Estimation Algorithm

We represent a new object to detect as a set $\mathcal{C} = \{\mathcal{C}_1, .., \mathcal{C}_{N_C}\}$ of N_C 3D corners. This can be done using only the CAD model of the object. Each corner is made of N_v 3D virtual points: $\mathcal{C}_i = \{M_{i,1}, .., M_{i,N_v}\}$ expressed in the object coordinate system.

From our Faster R-CNN framework, given an input image, we obtain a set $\mathcal{D} = \{d_1, \ldots, d_{N_d}\}$ of N_D detected corners d_j. Each detected corner d_j is made of N_v predicted 2D reprojections: $d_j = [m_{j,1}, \ldots, m_{j,N_v}]$.

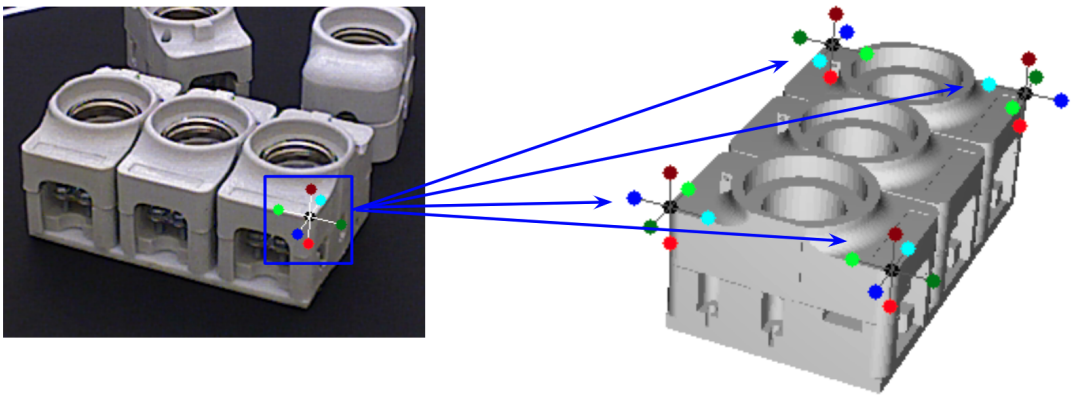


Figure 6.8: A detected corner in 2D can be matched to multiple 3D corners on the CAD model. The RANSAC-based pose estimation algorithm will retrieve the good pose hypothesis among all.

The pseudocode for our detection and pose estimation algorithm is given as Alg. 1. To deal with the erroneously detected parts, we use the same strategy as RANSAC. By matching the detected corners d_j with their 3D counterparts $\mathcal{C}_i$, it is possible to compute the 3D pose of the object using a PnP algorithm, followed by a Levenberg-Marquardt optimization to refine the pose. Since each corner is represented by $N_v = 7$ points, it is possible to compute the pose from a single match. As explained in Section 6.2.2, each detected corner can correspond to 3 possible arrangements of virtual points, and we apply Σ_1 and Σ_2 to the $m_{j,k}$ reprojections to generate the 3D possible poses for the detected corners. Furthermore one object can have similar corners and each detected corner can be matched to different 3D corners, as shown in Figure 6.8.
To find the best pose among all these 3D possible poses, we compute a similarity score as the cross-correlation between the gradients of the image and the image gradients of the CAD model rendered under the 3D pose. We finally keep the pose with the largest similarity score as the estimated pose.

6.3 Evaluation

In this section, we present and discuss the results of our pose estimation algorithm. We first describe the metrics we use, and then we show a quantitative analysis of object detection and pose estimation as well as qualitative results. All the results are computed on the challenging T-LESS dataset.

Algorithm 1 Pose estimation algorithm

$\mathcal{C} \leftarrow \{C_i\}_i$, the set of 3D corners on the new object. Each 3D corner C_i is made of 7 3D control points, expressed in the coordinate system of the new object.
$\mathcal{D} \leftarrow \{d_j\}_j$, the set of 2D detected corners in the input image. Each 2D corner d_j is made of 7 2D image locations.

procedure POSE_ESTIMATION($\mathcal{C}$, $\mathcal{D}$)
    $poses \leftarrow []$ ▷ Set of possible poses and their scores
    for $C \in \mathcal{C}$ **do**
        for $d \in \mathcal{D}$ **do**
            for $\Sigma \in \{I, \Sigma_1, \Sigma_2\}$ **do**
                corr $\leftarrow (C, \Sigma(d))$ ▷ 2D-3D correspondence
                $pose \leftarrow$ PNP(corr) ▷ 3D pose estimate
                $nb_{\text{inliers}} \leftarrow$ COMPUTE_INLIERS($pose, \mathcal{C}, \mathcal{D}$)
                if $nb_{\text{inliers}} > \tau_{\text{inliers}}$ **then**
                    REFINE($pose, \mathcal{C}, \mathcal{D}$) ▷ Compute pose using all the inliers
                    $s_{pose} \leftarrow$ SCORE($pose, \mathcal{C}, \mathcal{D}$)
                    Add $(pose, s_{pose})$ to $poses$
    return $pose$ with best s_{pose} in $poses$

procedure SCORE($pose$, $\mathcal{C}$, $\mathcal{D}$)
    $s \leftarrow 0$
    $template \leftarrow ImageGradients(rendering(model, pose)$
    $edges_{input} \leftarrow ImageGradients(input_{image})$
    $s \leftarrow Cross\_Correlation(edges_{input}, template)$
    return s

6.3.1 Metrics

To evaluate our method, we use the percentage of correctly predicted poses for each sequence and each object of interest, where a pose is considered correct based on the ADD metric:

$$ADD = \frac{1}{\mathcal{V}} \sum_{\mathbf{M} \in \mathcal{V}} ||\text{Tr}_{(\hat{\mathbf{R}},\hat{\mathbf{t}})}(\mathbf{M}) - \text{Tr}_{(\bar{\mathbf{R}},\bar{\mathbf{t}})}(\mathbf{M})||_2 \,. \tag{6.2}$$

This metric is based on the average distance in 3D between the model points after applying the ground truth pose and the estimated one. A pose is considered correct if the distance is less than 10% of the object's diameter. $\mathcal{V}$ is the set of object's vertices, $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ the estimated pose and $(\bar{\mathbf{R}}, \bar{\mathbf{t}})$ the ground truth pose, and $\text{Tr}_{\mathbf{R},\mathbf{t}}(\cdot)$ a rigid transformation by rotation $\mathbf{R}$ and translation $\mathbf{t}$.

For objects with ambiguous poses due to symmetries, we consider the ADI metric:

$$ADI = \frac{1}{\mathcal{V}} \sum_{\mathbf{M_1} \in \mathcal{V}} \underset{\mathbf{M_2} \in \mathcal{V}}{min} ||\text{Tr}_{(\hat{\mathbf{R}},\hat{\mathbf{t}})}(\mathbf{M_1}) - \text{Tr}_{(\bar{\mathbf{R}},\bar{\mathbf{t}})}(\mathbf{M_2})||_2 \,, \tag{6.3}$$

where the error is calculated as the average distance to the closest model point instead of the corresponding one. If an object is symmetric, both its pose $(\bar{\mathbf{R}}, \bar{\mathbf{t}})$ and

its symmetrical one are correct because it is not possible to distinguish them from the input image.

6.3.2 Results

The complexity of the test scenes varies from several isolated objects on a clean background to very challenging ones with multiple instances of several objects with a high amount of occlusions and clutters. Only a few previous works present results on the challenging T-LESS dataset. To the best of our knowledge, when this method was proposed, the problem of pose estimation of new objects not seen at training time had not been addressed yet and no comparison was possible.

To evaluate our method, we split the objects from T-LESS into two sets: One set of objects seen by the network during the training and one set of objects never seen and used for evaluation at testing time. More specifically, we train our network on corners extracted from Objects #6, #19, #25, #27 and #28 and test it on Objects #7, #8, #20, #26 and #29 on T-LESS test scenes #02, #03, #04, #06, #08, #10, #11, #12, #13, #14 and #15. Figure 6.9 shows the T-LESS objects of this setup.

6.3.3 Dataset

We use synthetic images for training the network. These images are generated as explained in Section 4.6.1 with a slight difference: instead of rendering the CAD models of all the objects from T-LESS, we render only the training objects we use in this set-up. For each synthetic image, we generate the ground truths for the 2D locations of the 3D control points.

We previously select, for each corner of the object, the 3D control points. Then, given the ground truth pose of the object, we project each set of 3D control points in 2D and we associate a bounding box to each corner to train the points regressor and the corner classifier respectively. We associate these ground truths only to corners that are visible enough in the image. In the previous chapter, we looked at the object globally, and we consider also objects partially occluded. In this method, we are looking locally at the corners of the objects. For this reason, we need to pay attention to the visibility of the corners not to train a network with ambiguous information. To train the network with the loss in Eq. 6.1, we apply the two permutations we introduced, Σ_1 and Σ_2, to the ground truths of the 2D locations.

2D Detection: We first evaluate our method in terms of 2D detection. Even this task is challenging on the T-LESS dataset given our setting, as the objects are very similar to each other.

Most of previous works separate the detection task from the pose estimation. For example, in Rad and Lepetit (2017), the authors present a method that first detects objects through a segmentation approach and then use the corresponding crop of the image to estimate the pose. Some works only focus on pose estimation:

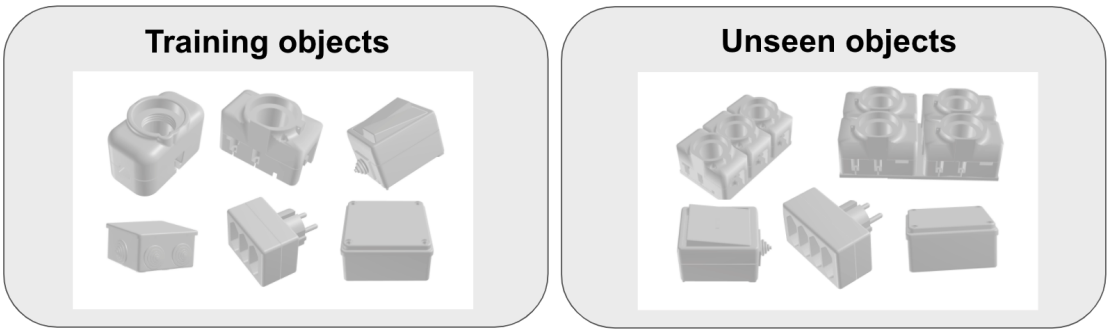


Figure 6.9: T-LESS Objects used in our evaluation set-up. **Left:** Objects seen by the network during the training. **Right:** Unseen objects used to evaluate our method.

Scene: Obj	$AD\{D\|I\}_{10\%}$	$AD\{D\|I\}_{20\%}$	$AD\{D\|I\}_{30\%}$	detection [%]
02: 7	68.3	80.1	83.7	67.3
03: 8	57.9	72.5	78.7	76.3
04: 26	28.1	47.2	56.2	48.3
04: 8	21.2	53.0	68.2	35.7
06: 7	36.8	61.7	78.7	73.7
08: 20	10.0	40.4	56.1	34.1
10: 20	27.8	47.2	58.3	30.0
11: 8	58.8	74.9	85.3	74.3
12: 7	23.1	44.6	47.7	54.6
13: 20	26.6	57.3	69.0	52.9
15: 29	48.0	59.1	76.7	38.3
14: 20	10.0	24.6	31.6	44.0
Average	34.7(±18.5)	55.2(±15.2)	65.9(±15.6)	52.5(±16.2)

Table 6.1: Our quantitative results on T-Less test Scenes #02, #03, #04, #06, #08, #10, #11, #12, #13, #14, #15. The last column reports the detection accuracy. $AD\{D|I\}_{*\%}$ refers to the ADD/ADI metric value and $*$ stands for the percentage of the diameter used to compute this value. We consider the object to be detected if the IoU between the rendering of the object with the pose estimate and with the ground truth is higher then 0.4.

Sundermeyer et al. (2018) use the ground-truth crops of each object of the scene to avoid the detection step.

In this work, we cannot access images of objects on which the pose estimation is done. Thus, it is not possible to train a separate object detection network or segmentation network to solve this problem. Our method returns the 3D poses of the objects directly. To evaluate the detection accuracy, we therefore use the 2D bounding boxes computed from the reprojections of the CAD models under the

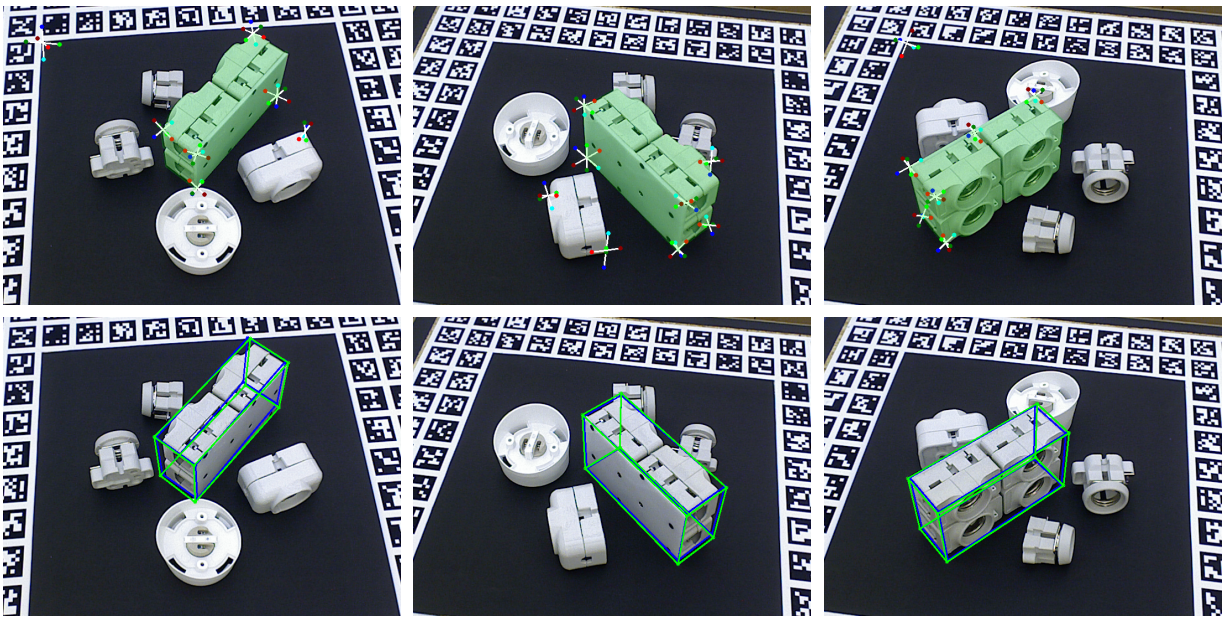


Figure 6.10: Some qualitative results on Object #8 in Scene #03 of the T-LESS dataset. **First row:** 2D detection results. **Second row:** 3D Pose Estimation results. Green and blue bounding boxes are the ground truth and estimated poses respectively.

estimated 3D pose.

We report our detection accuracy in the last column of Table 6.1. The accuracy is measured in terms of *Intersection over Union* (IoU) between the rendering of the object with the estimated pose and the rendering of the object with the ground truth pose. An object is considered correctly detected in the frame if $IoU > 0.4$. Our method succeeds an average of 52.5% of good detection without any detection or segmentation priors.

3D Pose Estimation We evaluate the pose estimation on images where the object of interest has been detected. For each object of our experiments, we compute the ADD metric in Eq. (6.2). Table 6.1 reports the scores for three percentages of object diameters. For symmetrical objects, we report the ADI in Eq. (6.3) metric instead of ADD. The object 3D orientation and translation along the x-and y-axes are typically well estimated. Most of the translation error is along the z-axis, as it is usually the case of other algorithms for 3D pose estimation from color images.

To conclude the evaluation of our method, we present several qualitative results obtained on the tested scenes of the T-LESS dataset in Figures 6.10-6.15. Each top row shows the results of the corners detection part with the unseen objects colored in green while each bottom row shows the estimated 3D poses. Green boxes are ground truth 3D bounding boxes while blue boxes are bounding boxes we predicted using our pose estimation pipeline. Some scenes are very challenging. Here, the background is highly textured compared to the objects, and the scenes are

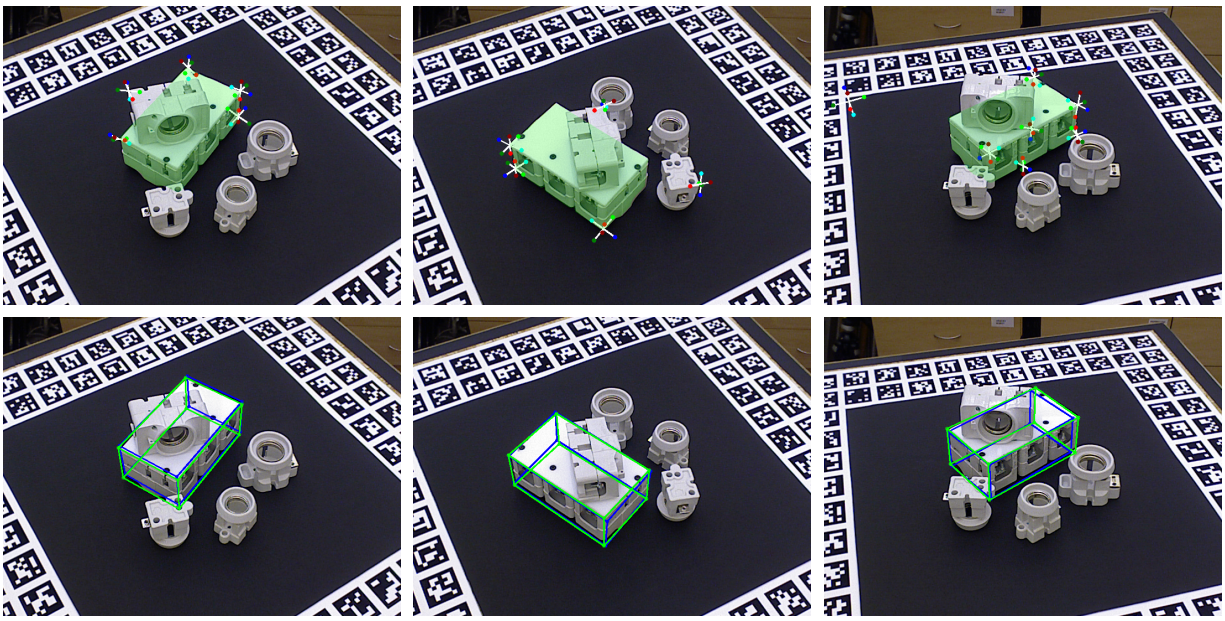


Figure 6.11: Some qualitative results on Object #7 in Scene #06 of the T-LESS dataset.

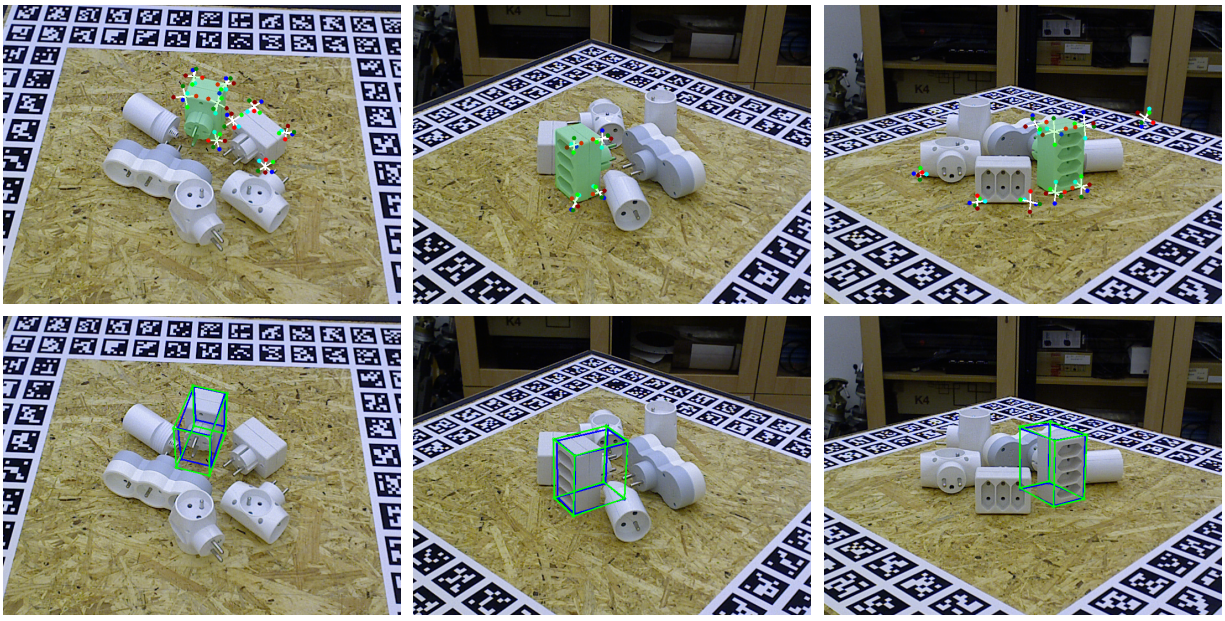


Figure 6.12: Some qualitative results on Object #20 in Scene #10 of the T-LESS dataset.

crowded with unwanted and close objects. Moreover, objects seen by our network during training appear near the objects on which we wanted to test our algorithm. Despite that, we can see that our method succeeds in estimating the pose correctly. Moreover, Figures 6.11, 6.14 and 6.15 show that detecting corners of the objects is a good direction when dealing with "crowded" scenes where partial occlusions often occur.

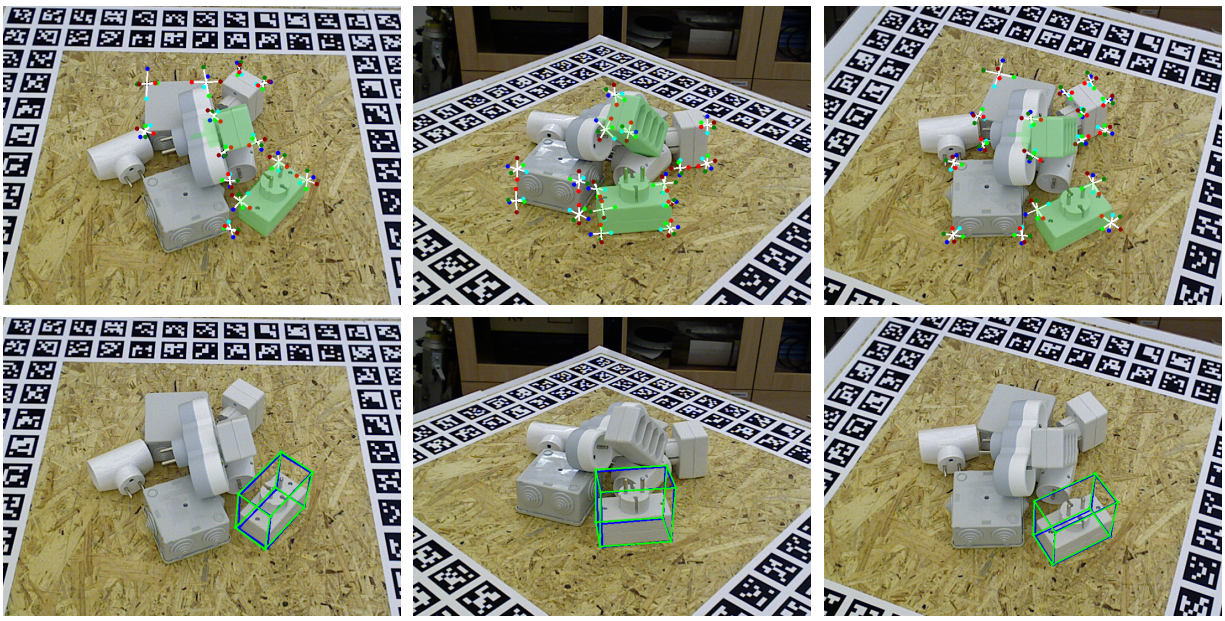


Figure 6.13: Some qualitative results on Object #20 in Scene #13 of the T-LESS dataset.

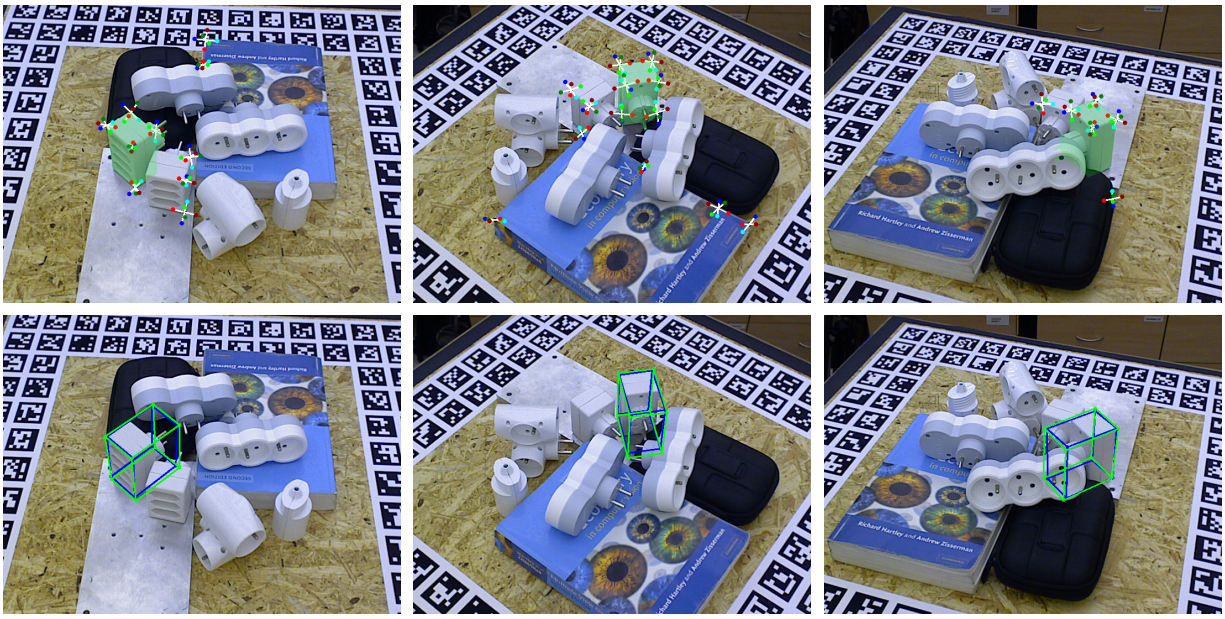


Figure 6.14: Some qualitative results on Object #20 in Scene #14 of the T-LESS dataset.

Computation Times: We implemented our method on an Intel Xeon CPU E5-2609 v4 1.70GHz desktop with a GPU Quadro P5000. Our current implementation takes 300ms for the 3D part detection and 2s for the pose estimation, where most of the time is spent in rendering and cross-correlation. We believe this part could be significantly optimized.

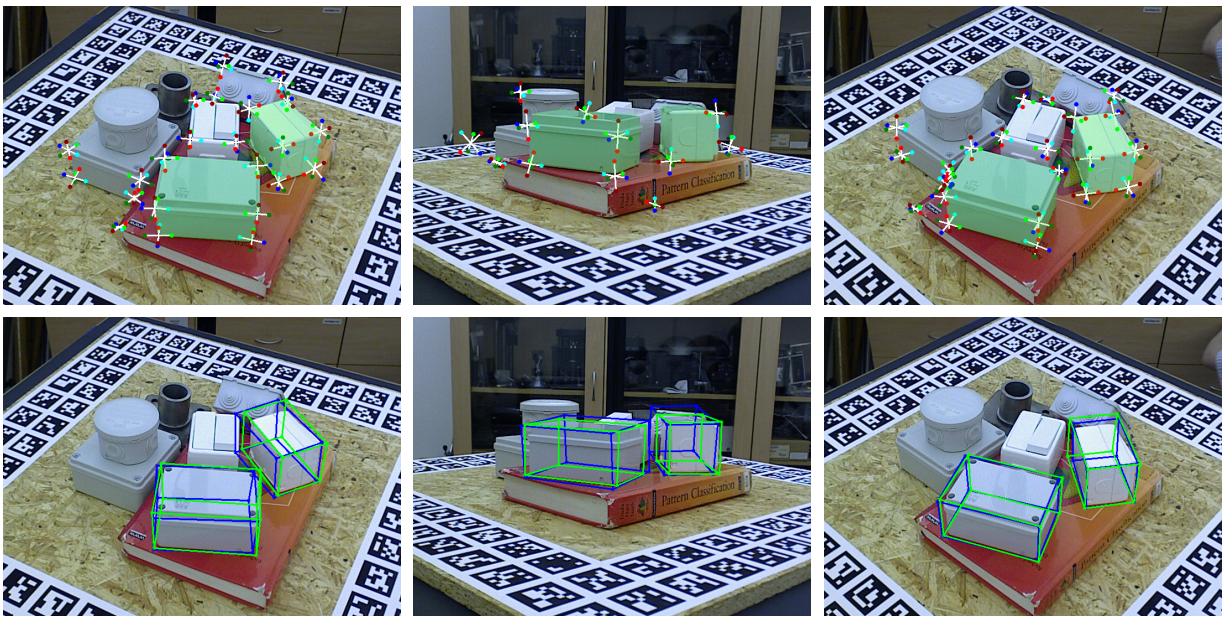


Figure 6.15: Some qualitative results on Object #26 and Object #29 in Scene #15 of the T-LESS dataset.

6.4 Conclusion

In this chapter, we introduced our first approach to the detection and 3D pose estimation of industrial objects in color images that only requires the CAD models of the objects, and no retraining is needed for new objects. We showed that estimating the 3D poses of the corners makes our method able to solve typical ambiguities that raise with industrial objects.

The main disadvantage of this method is that it requires a skilled user to select the 3D control points on the CAD model making the method not fully automatic and that it considers only corners and not other parts, such as edges or quadric surfaces. In the next chapter, we will discuss these limitations and we will propose another approach to predict the 3D poses of unseen objects that is not only limited to objects with prominent corners.

Chapter 7

6D Pose Estimation of Unseen Objects using Local Surface Embeddings

Table of contents

In this capter, we propose another approach for detecting and estimating the 3D poses of objects in images that requires only an untextured CAD model and no training phase for new objects and that overcomes the limitation of the method explained in Chapter 6. After a brief overview of the problem tackled, we explain our method that combines Deep Learning and 3D geometry and how we obtain the final 3D poses of the objects. We evaluate our method on T-LESS and we compare it with our previous method in Chapter 6 and with Sundermeyer et al. (2020a) to show its effectiveness.

7.1 Overview

In Chapter 6 we introduced a novel 3D pose estimation method based on objects' parts to estimate the pose of unseen objects, which is suited to some classes of objects that can be defined by specific sets of parts. In particular, we demonstrated it only on objects with prominent corners, which is quite restrictive. Furthermore, this approach requires an expert to offline select parts on the CAD models of the new objects. To relax this requirement and make our solution more general we introduce a novel method for 3D object detection and pose estimation from color images only based on dense correspondences between the input color image and the CAD model, instead of sparse corner correspondences.

We, again, investigate 3D object pose estimation in an industrial scenario with the challenges this implies: we want to handle symmetrical, textureless, ambiguous, and unseen objects, given only their CAD models. No offline selection of 3D points nor the presence of corners is required this time. By contrast with some previous works, we also do not assume that the ground truth 2D bounding boxes for the objects are available. As shown in Figure 7.1, our approach combines machine learning and 3D geometry: Like previous works (Brachmann et al., 2016; Zakharov et al., 2019; Park et al., 2019), we establish dense correspondences between the image locations and 3D points on the CAD model, as they showed that this yields to accurate poses. However, there is a fundamental difference between these works and ours: they do not generalize to unseen objects and they assume they can train a machine learning model in advance for each object. For these reasons, they can predict directly the 3D coordinates of the pixels of the seen objects in new images.

In our case, we want to avoid any training phase for new objects. Thus, we cannot predict the 3D coordinates of the pixels since this would be an information strictly related to the object. We therefore rely on a different strategy: we introduce an embedding capturing the local geometry of the 3D points lying on the object surface. Given a training set for a small number of objects, we learn to predict these embeddings per pixel for images of new objects. By matching these embeddings with the embeddings computed for 3D points on the object surface, we get 2D-3D correspondences from which we estimate the object's 3D pose using RANSAC and a PnP solver.

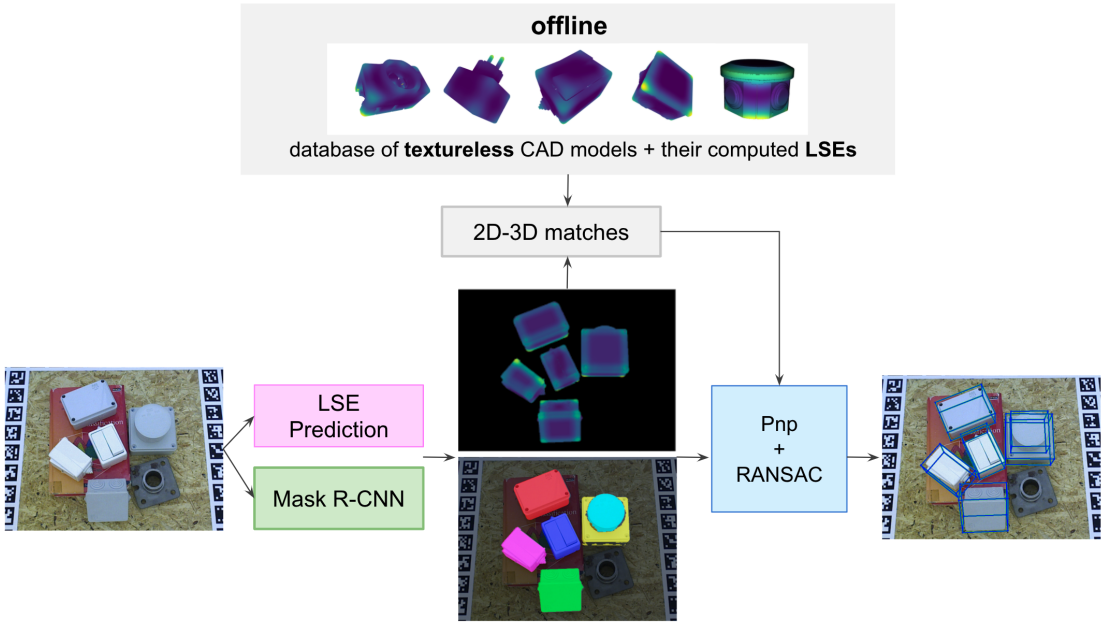


Figure 7.1: Overview of our method. We detect and estimate the 3D poses of objects, given only an untextured CAD model, without having to retrain a deep model for these objects. Given an input RGB image, we predict local surface embeddings (LSEs) for each pixel that we match with the LSEs of 3D points on the CAD models. We then use a PnP algorithm and RANSAC to estimate the 3D poses from these correspondences. We use the predicted masks to constrain the correspondences in a RANSAC sample to lie on the same object, to control the complexity. The LSE prediction network is trained on known objects but generalizes well to new objects. Similarly, we train Mask-RCNN on known objects and use mask R-CNN to segment the objects in the image. Because we train Mask-RCNN in a *class-agnostic* way, it also generalizes to new objects without retraining. Note that we use masks of different colors for visualization only.

This approach is conceptually simple, robust to occlusions, and provides an accurate 3D pose. However, to be successful, some special care is needed. First, the embeddings need to be rotation invariant. Second, because of the symmetries and this rotation invariance, many correspondences between pixels and 3D points are possible *a priori* and the complexity of finding a set of correct correspondences can become exponential. We control this complexity in two ways. We focus on image locations with the most discriminative embeddings as they have less potential correspondences. We also observe that Mask R-CNN (He et al., 2017) can predict the masks of new objects when trained without any class information, and thus can segment new objects without re-training. We use this to constrain the sets of correspondences in RANSAC to lie on the same mask, and thus drastically decrease the number of samples to consider in RANSAC.

In the remainder of the chapter, we describe our method in detail and evaluate

it on the T-LESS dataset, comparing it with our approach proposed in Chapter 6 and a recently published work that address the difficult task of dealing with unseen objects.

Contributions. The contributions of this chapter are:

- We introduce the concept of local embedding of the objects' 3D geometry;
- We propose a new approach to estimate the 3D poses of unseen objects from RGB images;
- We compare with our previous method proposed in Chapter 6 and a recent published work (Sundermeyer et al., 2020a) showing better performances.

The work presented in this chapter will be present at the *Asian Conference on Computer Vision*, 2020 (Pitteri et al., 2020).

7.2 Method

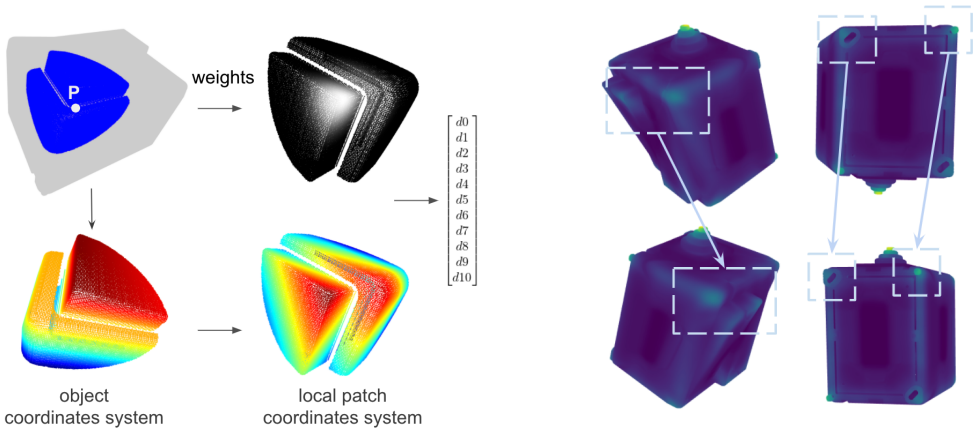
In this section, we will describe our approach. We first explain how we compute the local surface embeddings and how we obtain correspondences between the CAD models and the images. We then describe our pose estimation algorithm.

7.2.1 Local Surface Embeddings

To match new images with CAD models, we rely on embeddings of the local surfaces of the objects. To be able to match these embeddings under unknown poses, they need to be translation invariant and rotation invariant. Achieving translation invariance is straightforward since we consider the local geometry centered on 3D points. Achieving rotation invariance is more subtle, especially because of ambiguities arising in practice with symmetrical objects. This is illustrated in Figure 7.2(b): We need to compute the same embeddings for local geometries that are similar up to a 3D rotation. In this way we can handle ambiguous and symmetrical objects and our method can be more robust to occlusions than the one proposed in Chapter 5.

More exactly, given a 3D point $\mathbf{P}$ on the surface of an object, we define the local geometry as the set of 3D points $\boldsymbol{M}_n$ in a spherical neighborhood centered on $\mathbf{P}$ and of radius r. In practice, on T-LESS, we use $r = 3cm$. To compute a rotation-invariant embedding, we transform these points from the object coordinate system to a local patch coordinate system using a rotation matrix computed from the decomposition of the covariance matrix of the 3D points $\boldsymbol{M}_n$ after centering on $\mathbf{P}$ (Eggert et al., 1997):

$$C = \sum_n \boldsymbol{v}_n \cdot \boldsymbol{v}_n^\top , \tag{7.1}$$



(a) (b)

Figure 7.2: (a): Computation of the LSEs for a given point **P** on a CAD model. We transform the 3D points in the neighborhood of **P** into a rotation-invariant local system and weight them before computing their moments. (b): Visualization of the rotation-invariance property on different parts of the same object. Similar local geometries yield similar LSEs. Through this paper, we represent the LSEs using only their average value mapped to the red, green, blue channels with a color map except for Figures 7.3 and 7.4 that shows all the values.

where $\boldsymbol{v}_n = (\boldsymbol{M}_n - \mathbf{P})$ using a Singular Value Decomposition (SVD):

$$C = L^\top \Sigma R\,. \tag{7.2}$$

R is an orthogonal matrix, but not necessarily a rotation matrix, and small differences in the local geometry can result in very different values for R. We, therefore, apply a transformation to R to obtain a new matrix $\bar{R}$ so that $\bar{R}$ is a suitable rotation matrix. It can be checked that applying $\bar{R}$ to the $\boldsymbol{v}_i$ vectors will achieve rotation invariance for the local surface embeddings.

Let's denote by r_1, r_2, and r_3 the rows of R, and by $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$ the rows of $\bar{R}$. Applying R to the normal n of the object's surface at **P** yields a 3-vector $R \cdot n$ close to either $[0,0,1]^\top$ or $[0,0,-1]^\top$, depending on the orientation of R selected for the SVD. For normalization, we choose that $\bar{R} \cdot n$ should always be closer to $[0,0,1]^\top$. We therefore compute $o = r_3^\top \cdot n$. If o is positive, we take $\bar{r}_3 = r_3$, otherwise, we take $\bar{r}_3 = -r_3$. As a result, $\bar{R} \cdot n$ is always closer to $[0,0,1]^\top$ that to $[0,0,-1]^\top$. Finally, we take $\bar{r}_1 = r_1$ and $\bar{r}_2 = -\bar{r}_1 \wedge \bar{r}_3$, where $\wedge$ denotes the cross-product, which ensures that $\bar{R}$ is a rotation matrix.

We explain now how we define the local surface embeddings. For our experiments, we use the local moments of the local 3D points for simplicity but any other embeddings such as Deng et al. (2018) could also work. Let us denote by $[x_n, y_n, z_n]$

the vectors $\bar{R}\boldsymbol{v}_n$, then local surface embeddings can be computed as:

$$\text{LSE}_{i,j,k}(\mathbf{P}) = \sum_n w_n x_n^i y_n^j z_n^k \,, \tag{7.3}$$

where $w_n = \exp(-|\boldsymbol{v}_n|^2/\sigma^2)$ is a weight associated to each point based on its distance from $\mathbf{P}$ (we use $\sigma = 5$ in practice) and i, j, k are exponents in the range $[0, 1, 2]$. Theoretically it is possible to take all the combinations of exponents but we empirically found that the most discriminative values are computed using: $i \in \{0, 2\}$, $j \in \{0, 2\}$, $k \in \{0, 1, 2\}$, which gives 11 values for the full vector LSE($\mathbf{P}$) as taking $i = j = k = 0$ gives a constant value and is not useful. Finally, we normalize the values of LSE($\mathbf{P}$) to zero mean and unit variance so they have similar ranges. Figure 7.3 displays the embeddings for an example image. In this example case, most of the objects have corners, very discriminative parts. In Figure 7.4 we show an example of LSEs for cylindrical and rounded shape objects. These objects have continuous symmetry and normally we handle these ambiguities during the training of a CNN. Thanks to our rotation-invariant LSEs, any ambiguity raises during training and that makes our method robust to any type of symmetries.

7.2.2 Predicting the local surface embeddings for new images

Given a new CAD model, it is trivial to compute the local surface embeddings on points on its surface. Given a new input image, we would like to also compute the embeddings for the object points visible in this image. We use a Deep Network to perform this task. To do so, we create a training set by generating many synthetic images of known objects under various poses. We also compute the LSEs for all the pixels corresponding to a 3D point of one of the objects. We then train a U-Net-like architecture (Ronneberger et al., 2015) to predict the LSEs given a color image. The U-Net architecture has been proved to preserve the structural integrity of the image reducing the distortion. More details on the architecture and its training are provided in the experimental section.

This training is done once, on known objects, but because the embeddings depend only on the local geometry, the network generalizes well to new objects, as shown in Figure 7.5. Furthermore, the network can predict the LSEs for rounded shape objects and not only for objects with prominent corners as shown in Figure 7.6.

7.2.3 Pose Estimation Algorithm

The pseudocode for our detection and pose estimation algorithm is given in Algorithm 2. Given a new image, we compute the LSEs for each of its pixels using the network described in Section 7.2.2 and establish correspondences between image pixels and object 3D points. However, the number of possible correspondences can quickly become very large, which would yield a combinatorial explosion in the number of the sets of correspondences needed in RANSAC. We control the complexity in two different ways.

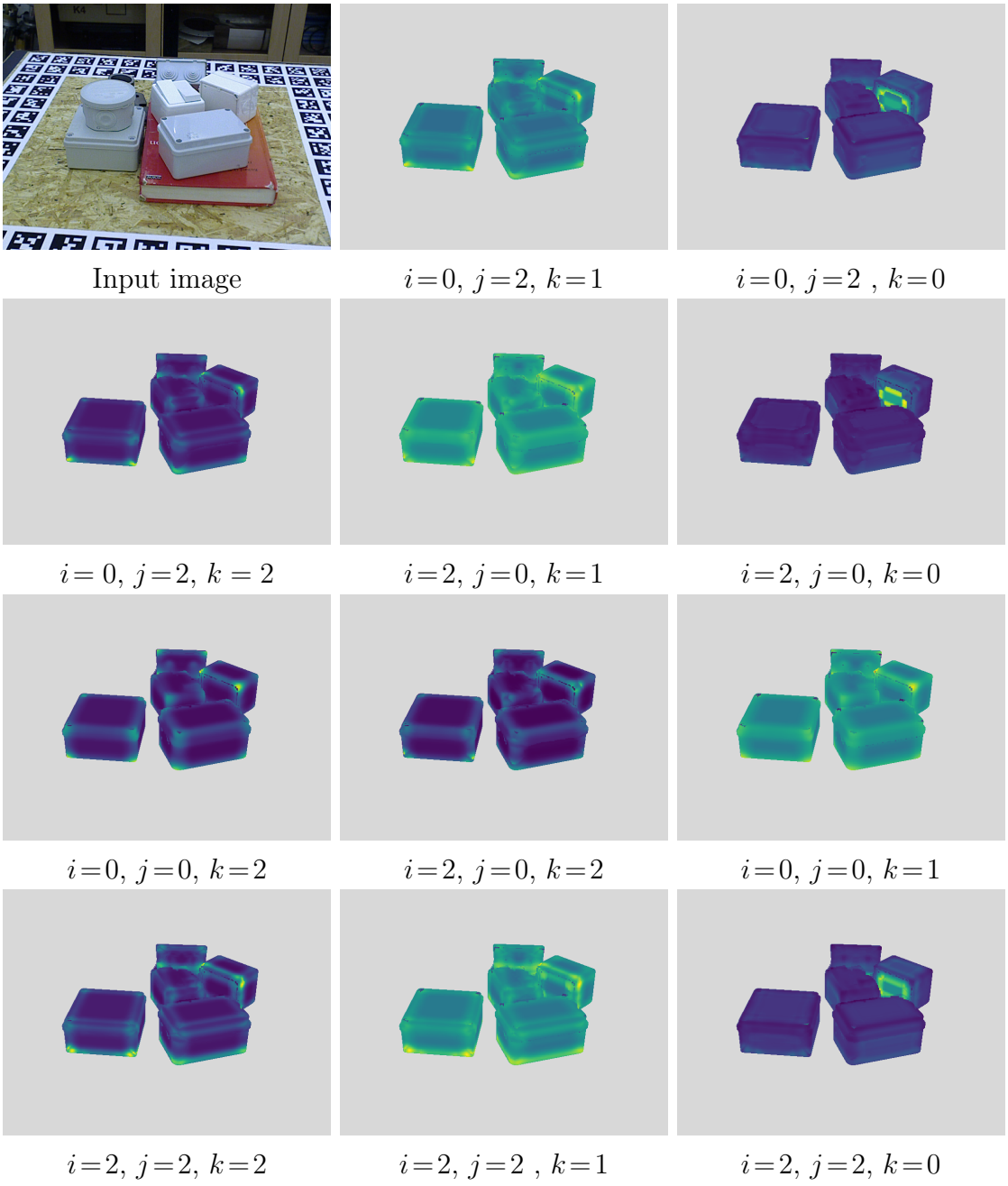


Figure 7.3: Visualization of the 11 coordinates of the LSEs for an example image.

First, we focus on the most discriminative embeddings. Points on planar regions are very common and would generate many correspondences. We discard them by thresholding the embedding values: Points with very low absolute embedding values for the LSEs are removed. Figure 7.7 shows how pixels are selected.

Second, we force the correspondences in each sample considered by RANSAC

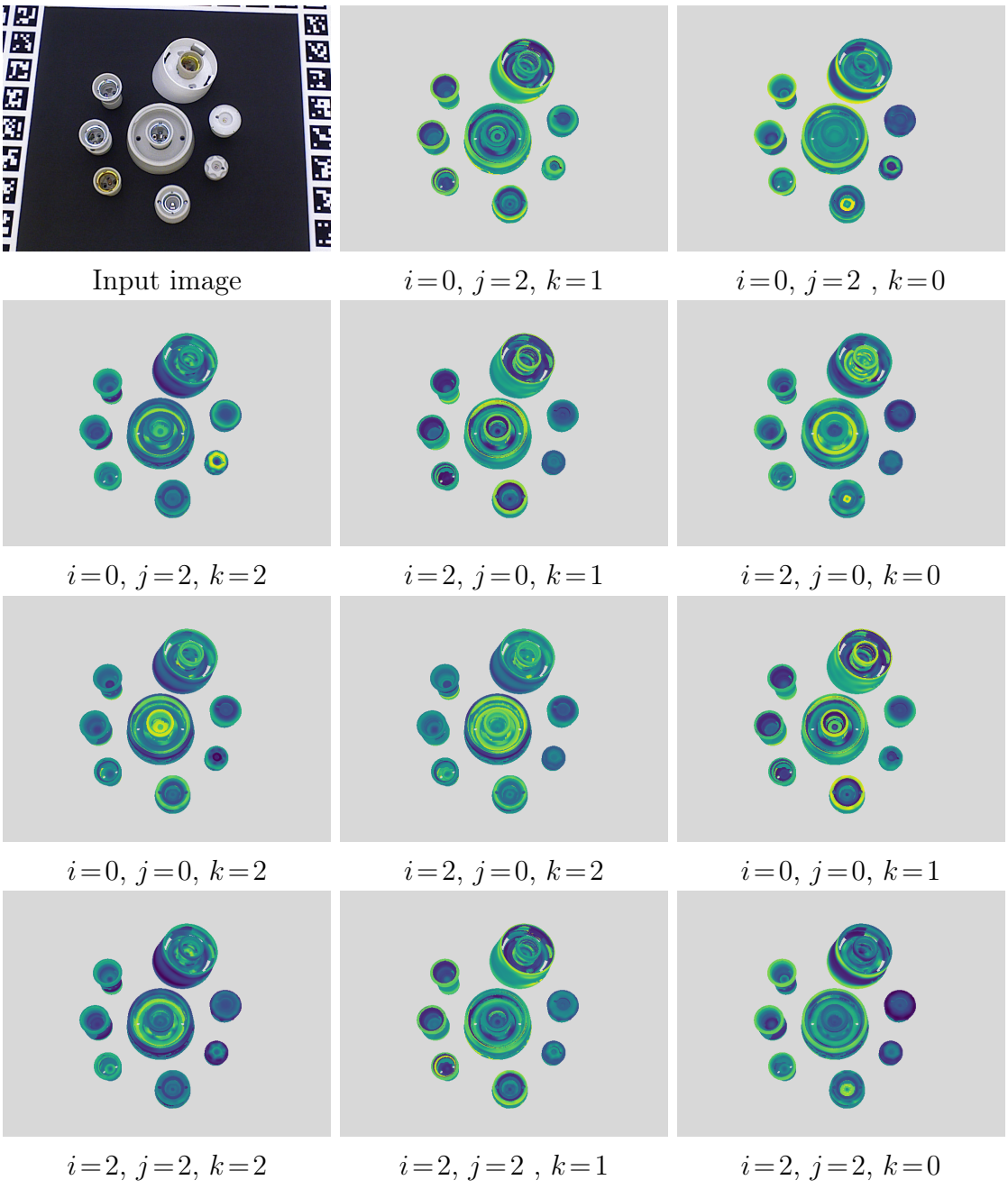
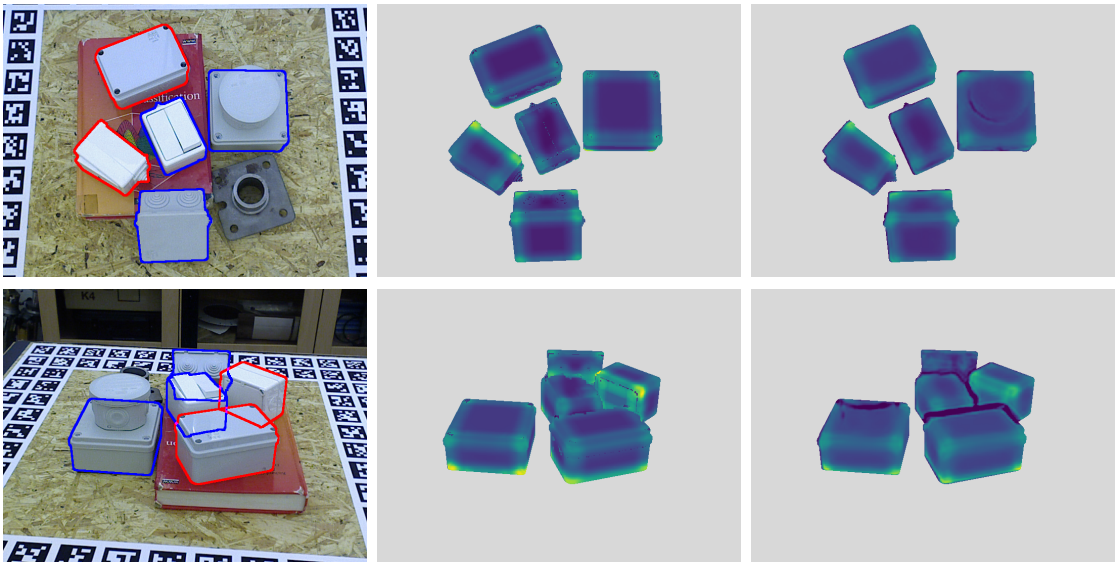


Figure 7.4: Visualization of the 11 coordinates of the LSEs for an example image with a focus on cylindrical and rounded shape objects.

to belong to the same object. Even when objects are not known in advance, it is possible to segment them. To do so, we use Mask R-CNN (He et al., 2017) to predict the masks of the objects. Mask R-CNN architecture is an extension of Faster R-CNN explained in Section 5.3. For a given image, Mask R-CNN, in addition to



(a) (b) (c)

Figure 7.5: Generalization of the LSE prediction network to new objects. (a) Input RGB image with objects seen during the training of the network (blue boundaries) and new objects (red boundaries). The LSE predictions (c) are close to the LSE Ground truth (b) for both the known and new objects.

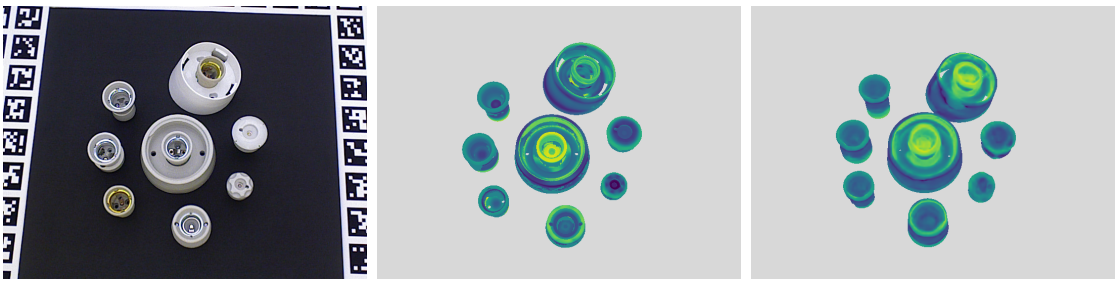


Figure 7.6: LSE prediction network results in the case of rounded shape objects. **Left:** input RGB image. **Center:** LSEs ground truth. **Right:** LSEs prediction.

the class label and bounding box coordinates for each object, will also return the object mask. To do so, a new branch is added to the Faster R-CNN architecture. For each Region of Interest (ROI) that contains an object, this mask branch returns the segmentation mask of size 28×28 which is then scaled up for inference. Another minor detail that differs from Faster R-CNN is the addition of a ROI alignment step that aims to locate the relevant areas of feature map to boost the accuracy.

We fine-tuned it on our synthetic images already used for training the LSE predictor, as described in Section 7.2.2 in a class-agnostic way since we want to generalize to new objects. We found out that it works very well with new objects

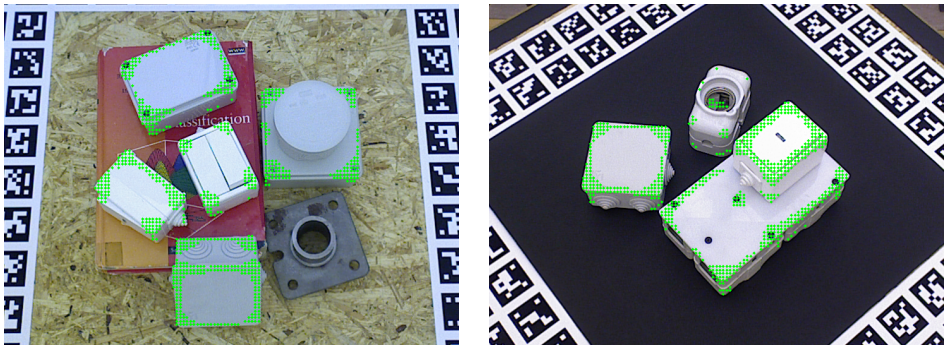


Figure 7.7: Focusing on the most discriminative pixels. In green, pixels with discriminative LSEs. We only consider them for correspondences with the CAD models

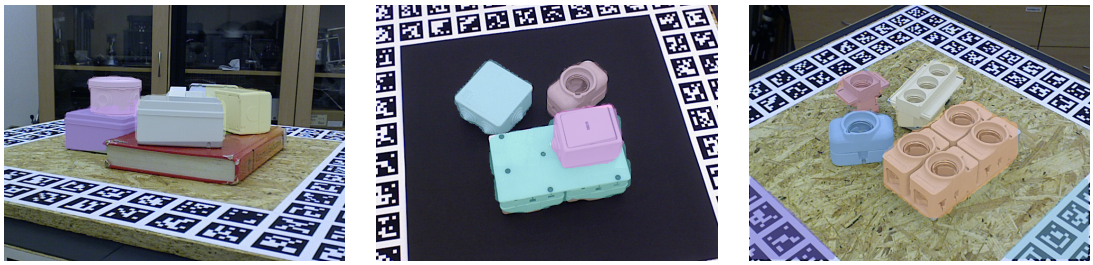


Figure 7.8: Generalization of Mask-RCNN to unknown objects. We train Mask-RCNN in a class-agnostic way on a set of known objects. It generalizes well to new objects, and we use these masks to constrain the pose estimation. Note that we use masks of different colors for visualization only. Mask-RCNN cannot identify the new objects individually as it was not trained on them, it can only detect objects in a class-agnostic way.

even for cluttered backgrounds, as shown in Figure 7.8. This also allows us to easily discard pixels on the background from the possible correspondences.

We match the embeddings predicted for the pixels of the input image against the embeddings computed for the 3D points on the CAD model based on their Euclidean distances. In our implementation, we use the FLANN library (Muja and Lowe, 2009) to efficiently get the k nearest neighbors of a query embedding. In practice, we use $k = 100$. This usually returns points in several clusters, as close points tend to have similar embeddings. We, therefore, go through the list of nearest neighbors sorted by increasing distances. We keep the first 3D point and remove from the list the other points that are also close to this point, and we iterate. This provides for each pixel a list of potential corresponding 3D points separated from each other.

When working on industrial objects like the ones in T-LESS, some pixels can be matched with several 3D points, as shown in Figure 7.9, because of the rotation invariance property of the local LSEs and the similarities between local parts of different objects.

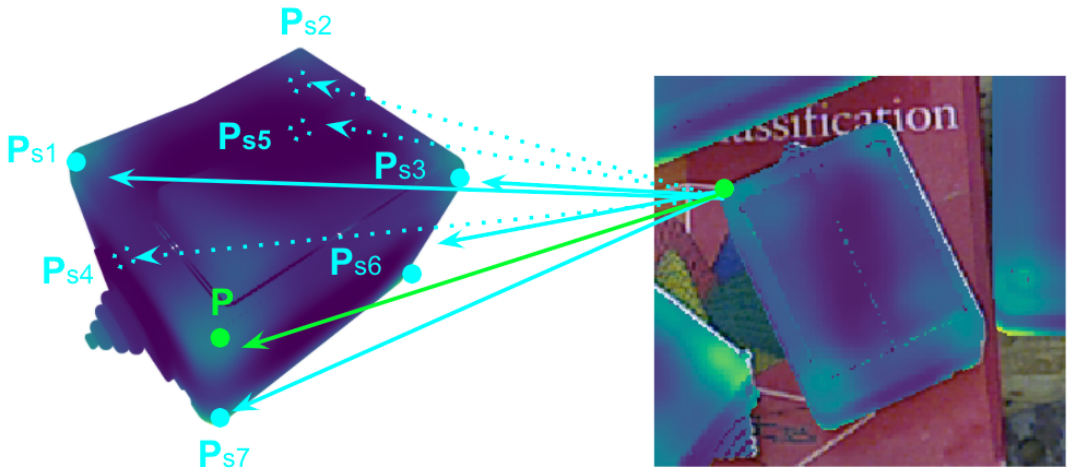


Figure 7.9: A pixel can be matched with multiple 3D points on symmetrical objects because of the rotation invariance property of the LSEs.

We finally use LO-RANSAC (Locally Optimized RANSAC, Chum et al. (2003)) with a PnP algorithm (more specifically, we use Lepetit et al. (2009) followed by a Levenberg-Marquardt optimization) to compute the poses of the visible objects. We take random $n \in [6; 10]$ for each RANSAC sample, where n is the number of 2D-3D correspondences. At each iteration, we compute a score for the predicted pose as a weighted sum of the *Intersection over Union* between the mask from Mask-RCNN and the mask obtained by rendering the model under the estimated pose, and the Euclidean distances between the predicted LSEs and the LSEs for the CAD model after reprojection. We keep the pose with the largest score and refine it using all the inlier correspondences to obtain the final 3D pose.

7.3 Evaluation

In this section, we present and discuss the results of our pose estimation algorithm on the challenging T-LESS dataset.

7.3.1 Dataset

To train our LSE prediction network, we generate synthetic images as in Section 4.6.1 with a slight difference: instead of rendering the CAD models of all the objects from T-LESS, we render only a subset of objects we use for training the network. The exact subset depends on the experiment, and we will detail them below. We used both these synthetic and real images for training the network combined with data augmentation to take care of the domain gap between our synthetic images

Algorithm 2 Pose estimation algorithm.

$\mathcal{C} \leftarrow$ CAD models for the new objects
$\mathcal{E}(C) \leftarrow \text{LSE}_{\text{CAD}}(C)$, the LSEs of 3D points for each CAD model C
$I \leftarrow$ input image
$\mathcal{F} \leftarrow \text{LSE}_{\text{pred}}(I)$, the predicted LSEs for the input image
$\mathcal{O} \leftarrow$ Mask-RCNN(I), the masks predicted by Mask-RCNN
$\mathcal{M} \leftarrow \{m_i\}_i$, the set of 2D-3D matches based on $\mathcal{E}(C)$ and $\mathcal{F}$. Each match m_i is made of an image location $\mathbf{p}$ and 3D points on the CAD models: $(\mathbf{p}, [\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_{m_i}])$

procedure POSE_ESTIMATION_O_C(O, C)
    $s_{\text{best}} \leftarrow 0$
    for iter $\in [0; N_{\text{iter}}]$ **do**
        $n \leftarrow$ random integer in $[6; 10]$
        $M \leftarrow n$ random correspondences $(\mathbf{p}, \mathbf{P})$,
            where $\boldsymbol{p} \in O$ and $\mathbf{P}$ is matched to $\mathbf{p}$ in $\mathcal{M}$
        $pose \leftarrow \text{PNP}(M)$
        $s \leftarrow \text{SCORE}(pose, C, \mathcal{E}(C), \mathcal{F}, O)$
        if $s > s_{\text{best}}$ **then**
            $pose_{\text{best}} \leftarrow pose$
            $s_{\text{best}} \leftarrow s$
    Refine $pose_{\text{best}}$
    return $pose_{\text{best}}$, $\text{SCORE}(pose_{\text{best}}, \mathcal{E}(C), \mathcal{F}, O, C)$

procedure POSE_ESTIMATION
    for each mask $O \in \mathcal{O}$ **do**
        $\triangleright$ $s_{\min}$ is the minimum score for a match with a CAD model:
        $s_{\text{best}}(O) \leftarrow s_{\min}$
        for each CAD model C **do**
            $pose, s \leftarrow$ POSE_ESTIMATION_O_C(O, C)
            if $s > s_{\text{best}}$ **then**
                $s_{\text{best}} \leftarrow s$
                $pose_{\text{best}}(O) \leftarrow pose$
                $C_{\text{best}}(O) \leftarrow C$

and the real test images. More specifically, we use $15K$ synthetic images and $\sim 7K$ real images—all the training images provided by T-LESS for the objects that are used for training the LSE prediction and Mask-RCNN. To create the ground truth embeddings, for each training image, we back project the pixels lying on the objects to obtain their corresponding 3D points and their LSEs. Neither the embedding prediction network nor Mask-RCNN sees the test objects during training.

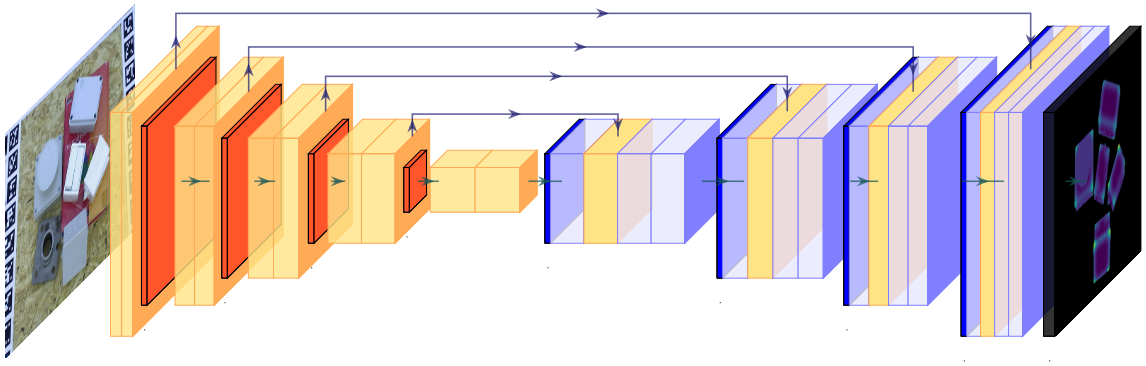


Figure 7.10: U-Net architecture used as LSE prediction network.

7.3.2 LSE prediction network architecture and training

The architecture of the network predicting the LSEs for a given input image is a standard U-Net-like (Ronneberger et al., 2015) encoder-decoder convolutional neural network taking a 720×540 RGB image as input. The U-Net architecture was originally developed for biomedical image segmentation but then it has been used for different tasks. It contains two paths: the first part is an encoder which is used to capture the context of the image. This encoder is a stack of convolutional and max-pooling layers and it is a common practice to take a network pretrained on a bigger dataset. Indeed, as the encoder part we use a 12-layer ResNet-like (He et al., 2016) architecture with the weights pretrained on the ImageNet dataset. After the encoder the image size is reduced while the depth is increased. The encoder function is basically learn "what" the information is rather than "where" it is. The second path, the decoder, is the symmetric expanding path and it upsamples the feature maps up to the original size using bilinear interpolations followed by convolutional layers. The decoder recovers "where" the information is. To get better precise locations, at every step of the decoder there are some skip connections. These connections concatenate the output of the transposed convolution layers in the decoder with the feature maps from the encoder at the same level. U-net is thus an end-to-end fully convolutional network (FCN) and so it does not contain any dense layer and it can accept an image of any size as input.
We train the network with the Adam optimizer and a learning rate set to 10^{-4}. We also use batch normalization to ensure good convergence of the model. Finally, the batch size is set to 8 and we train the network for 150 epochs.

7.3.3 Metrics

We evaluate our method using both the metrics explained in Chapter 5 and 6: the VSD metric (Eq. (5.34)), and the ADD metric (Eq. (6.2)) with its version for symmetrical objects, ADI (Eq. (6.3)). For the ADD metric a pose is considered

correct if the distance is less than 10% of the object's diameter, as reported in the instructions of the BOP challenge.

7.3.4 Results

We compare our method against CorNet, the method we proposed in Chapter 6 and the MP-Encoder (Sundermeyer et al., 2020a), the recent work that considers 3D object detection and pose estimation for unknown objects. We use the same protocols as in these works and report the results from the papers.

Comparison with CorNet (Chapter 6): We use here the same protocol as in CorNet: We split the objects from T-LESS into two sets: One set of known objects (#6, #19, #25, #27, and #28) and one set of unknown objects (#7, #8, #20, #26, and #29), and we compare the 3D detection and pose estimation performance of our method and CorNet for the unknown objects in T-LESS test scenes #02, #03, #04, #06, #08, #10, #11, #12, #13, #14, and #15. We use synthetic and real images of the known objects for training the LSE prediction network. The results are reported in Table 7.1. We outperform CorNet on most of the objects, except on objects #7 and #8 (Figure 7.12). This is because these objects have some 3D points with local geometry very different from the training objects (at the connections of the different parts). As a result, the predicted LSEs for these parts are not very accurate, generating wrong matches. Figure 7.11 shows some qualitative results for the unknown objects in the test images.

Comparison with MP-Encoder Sundermeyer et al. (2020a): We use here the same protocol as in Sundermeyer et al. (2020a): The objects from T-LESS are split into a set of known objects (#1-#18) and one set of unknown objects (#19-#30), and we compare the 3D detection and pose estimation performance of our method and MP-Encoder for the unknown objects in T-LESS test scenes following the BOP benchmark. We use synthetic and real images of the known objects for training the LSE prediction network. Note that we report here the numbers from Table 3 from the Sundermeyer et al. (2020a) paper as the other reported results in this article assume that the ground truth bounding boxes, the ground truth masks, or depth information are provided. The results are reported in Table 7.2. While our method performs slightly better, the performances are close and tell us that both methods are promising. The main difference is that the MP-Encoder relies on an embedding completely learnt by a network while our method incorporates some geometrical meaning that makes our approach more appealing for industrial purposes.

Scene: Obj	$AD\{D\|I\}_{10\%}$	
	CorNet (Chapter 6)	**Ours**
02: 7	**68.3**	61.0
03: 8	**57.9**	44.1
04: 26	28.1	**55.6**
04: 8	21.2	**39.1**
06: 7	36.8	**44.8**
08: 20	10.0	**38.2**
10: 20	27.8	**38.3**
11: 8	**58.8**	40.8
11: 9	-	**46.1**
12: 7	23.1	**41.2**
12: 9	-	**45.8**
13: 20	26.6	**39.5**
15: 29	48.0	**77.0**
15: 26	-	**63.6**
14: 20	10.0	**24.9**
Average	34.7(±18.5)	46.7(±12.4)

Table 7.1: Our quantitative results on T-LESS test Scenes #02, #03, #04, #06, #08, #10, #11, #12, #13, #14, #15 as used in CorNet. We report results also for Objects #9 in Scenes #11 and #12 and for Object #26 in Scene #15 even though we did not do that in Chapter 6. See the text for details.

	VSD recall
MP-Encoder Sundermeyer et al. (2020a)	20.53
Ours	**23.27**

Table 7.2: Mean *Visible Surface Discrepancy* (VSD) recall using the protocol of Sundermeyer et al. (2020a). This metric evaluates the pose error in a way that is invariant to the pose ambiguities due to object symmetries. It is computed from the distance between the estimated and ground truth visible object surfaces.

7.3.5 Textured objects case

We already discussed the difficulty to deal with texture-less objects and our purpose to find solutions for industrial applications where these kinds of objects are very common. However, we are interested to see if our LSEs prediction network can generalize to textured objects as well. Since our LSEs are representations of local geometries of the object, one may think that texture can act like noise or a distractor. To show the robustness of LSEs we made a minor experiment and we generate synthetic images as explained in Section 4.6.1 and, besides, we render the T-LESS objects with random textures. Figure 7.13 shows some examples of the synthetic

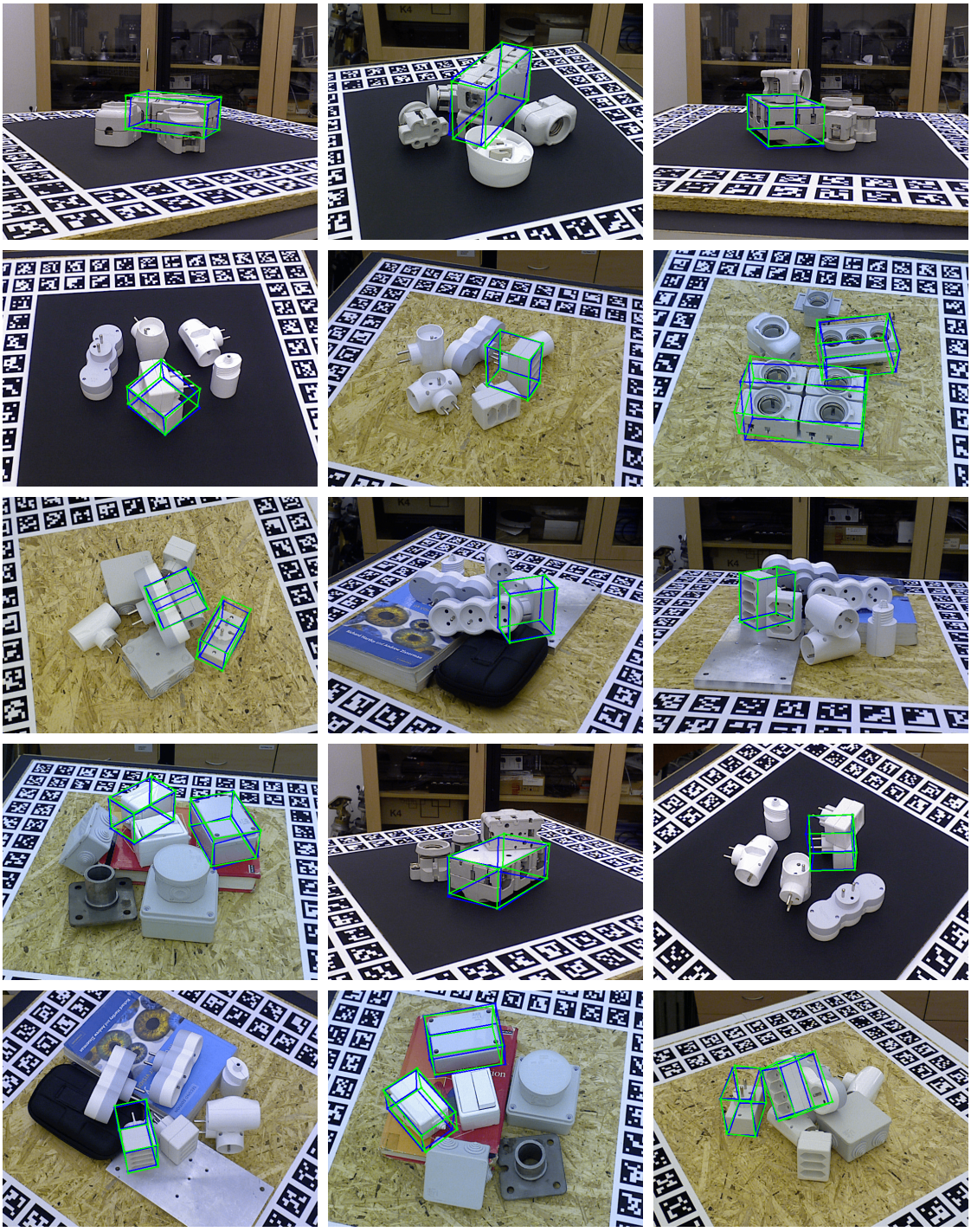


Figure 7.11: Qualitative results on the unknown objects of the test scenes from T-LESS. The green bounding boxes denote ground truth poses, while the blue ones correspond to our predicted poses. Our method is robust to partial occlusions.

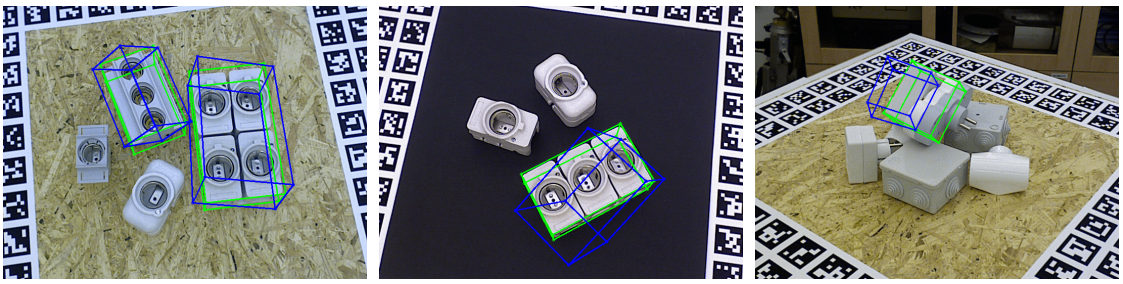
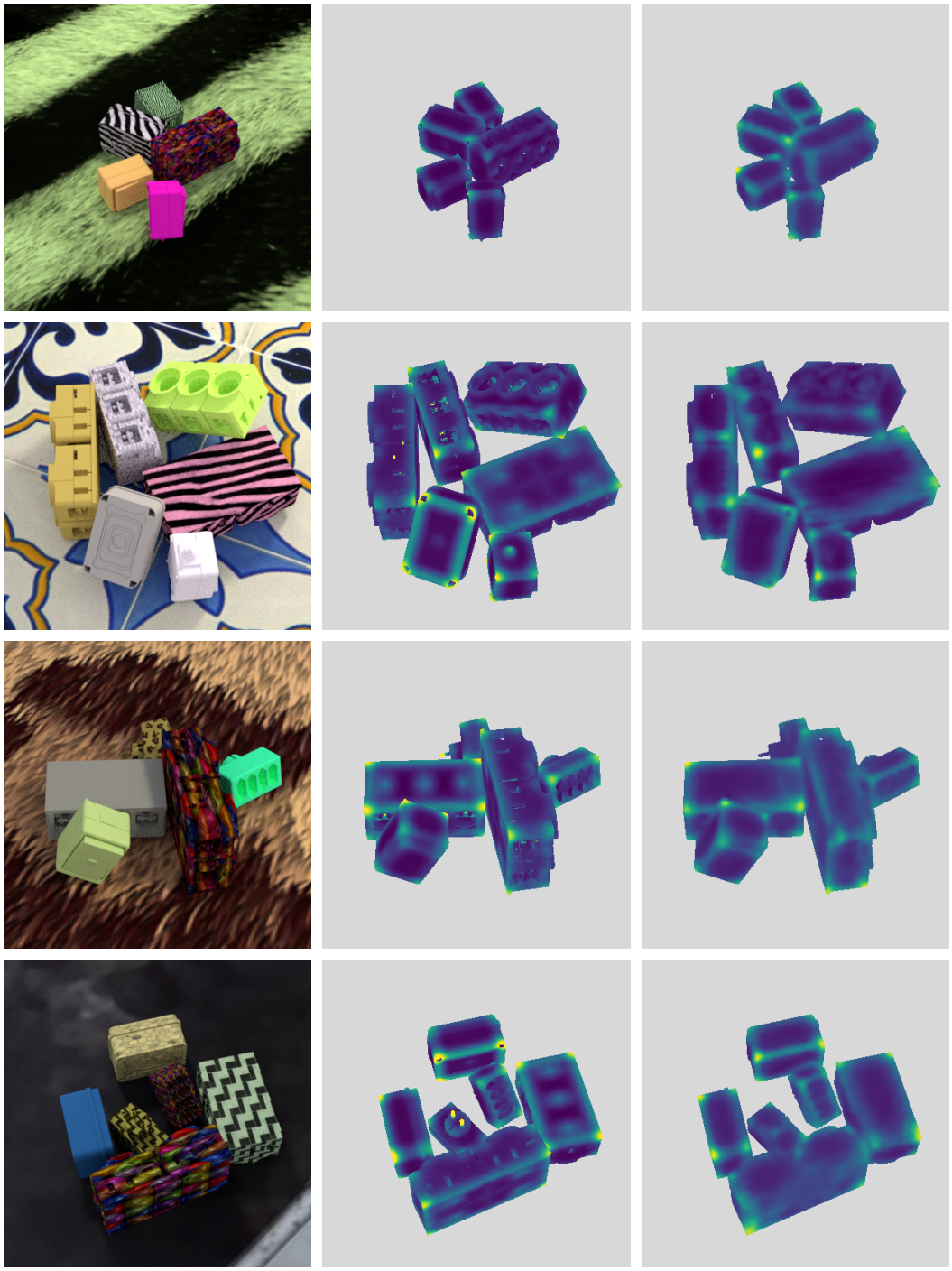


Figure 7.12: Some failure cases. They are due to inaccurate predictions of the LSEs or to large occlusions of the discriminative parts of the object.

Figure 7.13: Synthetic training images use to show the robustness of LSEs to textured objects. We generate synthetic images as explained in Section 4.6.1 and we render the T-LESS objects with random textures.

training images. We kept the same train and test objects sets as in CorNet (Chapter 6) and we test the network trained on synthetic images. The LSEs prediction for some images are shown in Figure 7.14. Not only the network can generalize to new objects but it also able to deal with both texture-less and textured objects.



(a) (b) (c)

Figure 7.14: Our LSE prediction network can generalize to unseen textured objects. (a) Synthetic input images. (b) LSEs ground truths. (c) LSEs predictions.

7.4 Conclusion

In this chapter, we introduced another approach for the detection and the 3D pose estimation of industrial objects in color images. It only requires the CAD models of the objects and no retraining is needed for new objects. We introduce a new type of embedding capturing the local geometry of the 3D points lying on the object surface and we train a network to predict these embeddings per pixel for images of new objects. From these local surface embeddings, we establish correspondences and obtain the pose with a PnP+RANSAC algorithm. Describing the local geometries of the objects allows to generalize to new categories and the rotation invariance of our embeddings makes the method able to solve typical ambiguities that raise with industrial and symmetrical objects. We believe that using local and rotation invariance descriptors is the key to solve the 6D pose of new texture-less objects from color images. In the next chapter, we discuss possible future work to improve the performances of this approach.

Chapter 8

Conclusion

Table of contents

8.1 Conclusion

In this thesis, we focused on the problem of estimating the 3D objects' poses from color images in challenging situations typical of the industrial contexts. Thanks to the advent of Deep Learning, the research community made much progress in 3D pose estimation, but there is still a gap between research scenarios and real industrial applications because of many cited challenges. Only recently more attention has been paid on reducing this gap, and this was the aim of this thesis.

We first proposed a method to handle symmetrical objects that are typical in industrial applications. We explained why symmetrical objects cause ambiguities when aiming to estimate the 3D objects' poses from color images with Machine Learning techniques. We showed what happens if we naively train a Deep Network to learn a mapping from the image (the object appearance) to a pose representation to prove why recent powerful algorithms for 3D pose estimation would fail. We then proposed an analytical solution to handle these objects that can be integrated into any of these 3D pose estimation frameworks that can learn this mapping. We demonstrated the effectiveness of our solution by integrating it in a Faster R-CNN based architecture.

We then moved on to another challenge: make deep learning-based algorithms able to generalize to unseen objects. We underlined why it is an important aspect, especially in the industry. Most of the learning-based methods rely on Supervised Machine Learning techniques that means, for each new object, they need to be retrained on many images of this object. These images are often not available and, even if it is possible to create synthetic images, it is often desirable to avoid these training sessions that require time. The first approach we proposed focused on objects with prominent corners, that are common in industrial applications. We learnt to detect corners in the images and predict their 3D poses in the form of a set of 2D reprojections of 3D virtual points previously selected on the CAD models of the objects. We then combined multiple corners to compute the object pose. This method generalizes well to objects with similar corners and it is robust to occlusion. In fact, thanks to the pose representation we used, detecting only 1 or 2 corners is sufficient to estimate the pose of the object. However, this method has some limitations: it requires objects to have specific corners and to offline select corners on the CAD model.

To overcome these limitations, we proposed another method that combines Deep Learning and 3D geometry. We established dense correspondences between the image locations and 3D points on the CAD model. However, we could not predict directly the 3D coordinates of the pixels in the image. This would not generalize to unseen objects. We, therefore, relied on LSEs, embeddings capturing the local geometry of the objects. We computed offline these embeddings for the points of the 3D objects surfaces and we trained a Deep Network to learn to predict them for

each pixel of a color image. We then get 2D-3D correspondences to obtain the 3D object pose. We believe this approach can be a promising starting point for further development to achieve better performances in 3D pose estimation of unseen objects.

8.2 Future Work

The methods we proposed have some limitations, and many different adaptations and experiments can further improve them. In this section we briefly describe some limitations of our proposed methods, and we give some idea of future works that can be exciting research directions to investigate.

Learning Objects Symmetries

The method to handle symmetrical objects exposed in Chapter 5 requires to have prior knowledge on the objects symmetries, that means the axes and the angles of symmetries. An interesting future direction would be to learn the symmetries automatically without having to rely on some prior knowledge. It would be a significant improvement, and it would allow us to apply the method to new objects or to objects which are not symmetrical, but they appear symmetrical only in some particular situations, for example when some occlusions occur. The computer graphics research community has been studying this problem for years and a challenge took place to evaluate different proposed methods (Funk et al., 2017). Recently, Shi et al. (2020) proposes SymmetryNet, a end-to-end deep neural network able to predict both reflectional and rotational symmetries of 3D objects. However, to achieve good performances they need to use an RGB-D input to train the network in a multi-task way and predict, for each 3D point its symmetric counterpart corresponding to a specific predicted symmetry.

Real-Time Performance

All methods that estimate 3D objects' poses through 2D-3D correspondences need to integrate the PnP algorithm with RANSAC to avoid outliers, and it takes time depending on the number of the correspondences that must to be evaluated. In both the approaches proposed in Chapter 6 and 7, we end up with a large number of correspondences that results in a large number of RANSAC iterations to achieve enough good accuracy of the poses that slows down the algorithm. Most of the time is spent in the pose verification part, where the object is rendered with the estimated pose and a score is computed by comparing it with the input image. In particular, the method presented in Chapter 7 is slowed down because of two aspects: its complexity and the time spent in the RANSAC loop. The complexity is due to the nested loops in the pose estimation algorithm. We need to iterate over all the objects in the database associated with their pre-computed LSEs and,

given the most discriminative LSEs of a mask prediction, we need to look for all the possible 2D-3D matches that arise because of the ambiguities and the symmetries of the objects, as discussed in Section 7.2.3.

The RANSAC loop is needed to deal with outliers in the set of 2D-3D correspondences. In particular, we have two types of outliers. The first one happens because of some wrong LSEs predictions while the other because of the ambiguities that raise when objects are symmetrical or when they share some similar local geometries, as already mentioned.

To speed up the pose estimation algorithm we could work on these two aspects. A possible solution to reduce the complexity would be to have less correspondences to evaluate by train a Deep Network to tell us where to look in the image to find the good ones. Finally, to reduce the number of outliers in the RANSAC loop, we need to improve the robustness of our LSE prediction network in order to avoid bad LSEs predictions.

Training on synthetic images

We already discussed the importance of training data in Deep Learning applications and at the same time the difficulty to annotate large datasets for the 3D pose estimation task that cover a lot of different viewpoints. For these reasons, it is necessary to generate synthetic images. However, a network trained on synthetic images is not going to generalize and perform well on real images. Researchers are working to bridge this gap by implementing different techniques. We report here the two techniques we could exploit to improve the performances of our methods.

Domain Randomization Using data augmentation and domain randomization on the RGB images has been demostrated to be crucial to achieve good performances in T-LESS (Sundermeyer et al. (2018), Labbé et al. (2020)). Domain randomization tries to adapt the training domain to another, that in this case would be adapting the synthetic domain to the real one, by applying different data augmentation techniques such as randomize shadow, translation, color tones and texture.

Domain adaptation Another possible way would be using the domain adaptation or feature mapping technique, similar to Rad et al. (2018b) and shown in Fig 8.1. In this case, instead of adapting the training domain, we adapt the features from the network. The idea is that the network $\mathcal{F}$ and $\mathcal{H}$ are composed of the same layer of our LSE prediction network. We can, for example, create the network $\mathcal{F}$ with the encoder and the first layers of the decoder of our LSE network, while the network $\mathcal{H}$ with the remaining layers of the decoder. The model weights are the same and they have been trained as explained in Section 7.3.2.

The upper stream shows what happens at inference time. The network $\mathcal{F}$ takes a RGB image as input and it acts as a feature extractor. These features, instead of

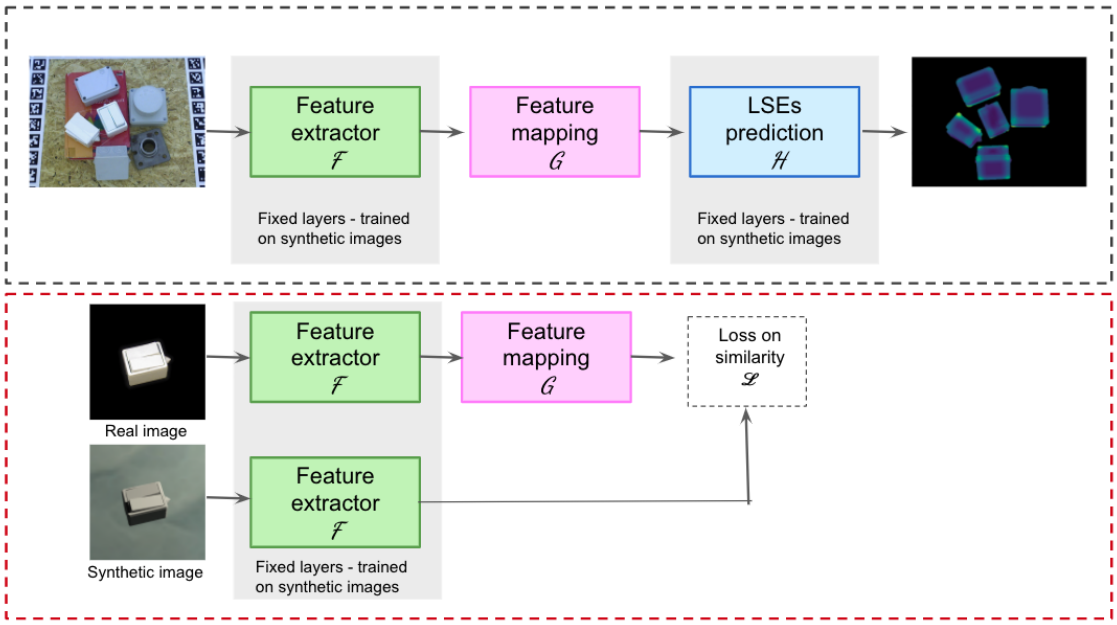


Figure 8.1: Feature mapping pipeline. The upper stream shows what happens at inference time, while the lower stream shows how the network $\mathcal{G}$ is trained.

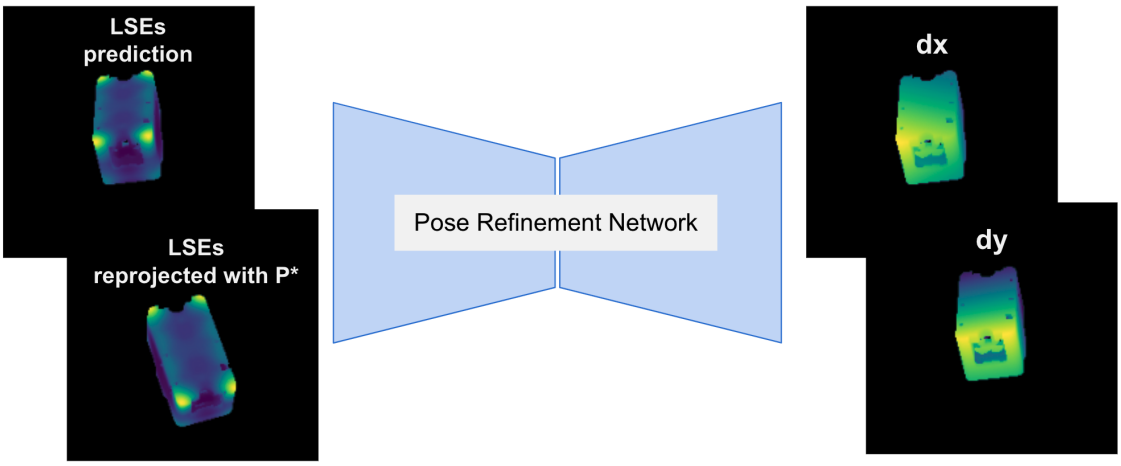
passing directly to the network $\mathcal{H}$ to predict the LSEs are previously transformed through the feature mapping network $\mathcal{G}$.

The lower stream shows how the network $\mathcal{G}$ is trained. For each real image, we generate a synthetic one by rendering the same objects with the same poses as the real one. We then pass both these images to the network $\mathcal{F}$. The features coming from the real image are mapped to the synthetic ones through the network $\mathcal{G}$. These real mapped features are compared with the synthetic one and the similarity error is used as a loss function for the optimization step during the training.

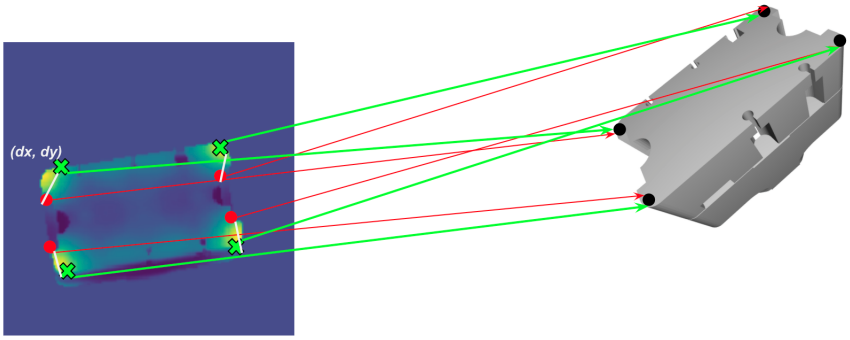
Pose refiner

A natural extension of our work proposed in Chapter 7 would be implementing a pose refiner to improve the accuracy of the pose estimate. This pose refiner should work with color images and no depth information can be used. Recently some pose refiners have been proposed but they were mostly object-dependent and they cannot work with unseen objects. Only DeepIM shows some preliminary results of unseen objects.

A possible way to do that would be as shown in Figure 8.2(a). Our LSE network outputs the LSEs predictions with class-agnostic object mask. Given the pose estimate for an object, we can project its LSEs in 2D and compare them with the predicted pixel-wise and masked LSEs. We train a network with these two inputs to learn the offsets in 2D between the pixels, also referred to as *optical flow* in the literature.



(a)



(b)

Figure 8.2: (a): Pose Refiner pipeline. The network takes as input the LSEs predictions and the LSEs reprojected with the estimated pose $\mathbf{P}^*$ and it outputs the pixel-wise offset (d_x, d_y). (b): These offset predictions are applied to 2D-3D correspondences to improve the 2D locations accuracy and refine the pose.

Given the 2D-3D correspondences we used to estimate the pose, we improve the accuracy of the 2D locations by applying the predicted offsets (δ_x, δ_y), as shown in Figure 8.2(b). We finally can run a RANSAC-like algorithm again on these refined correspondences and refine the pose.

Bibliography

[Armstrong and Zisserman 1995] Armstrong, M. ; Zisserman, A. : Robust Object Tracking. In : *Asian Conference on Computer Vision*, 1995 30

[Balntas et al. 2017] Balntas, V. ; Doumanoglou, A. ; Sahin, C. ; Sock, J. ; Kouskouridas, R. ; Kim, T.-K. : Pose Guided RGBD Feature Learning for 3D Object Pose Estimation. In : *International Conference on Computer Vision*, 2017 42

[Bay et al. 2008] Bay, H. ; Ess, A. ; Tuytelaars, T. ; Goo, L. V. : SURF: Speeded Up Robust Features. In : *Computer Vision and Image Understanding*, 2008 31

[Bousmalis et al. 2017] Bousmalis, K. ; Silberman, N. ; Dohan, D. ; Erhan, D. ; Krishnan, D. : Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. In : *Conference on Computer Vision and Pattern Recognition*, 2017 41, 76

[Bousmalis et al. 2016] Bousmalis, K. ; Trigeorgis, G. ; Silberman, N. ; Krishnan, D. ; Erhan, D. : Domain Separation Networks. In : *Advances in Neural Information Processing Systems*, 2016, pp. 343–351 41

[Brachmann et al. 2014] Brachmann, E. ; Krull, A. ; Michel, F. ; Gumhold, S. ; Scotton, J. ; Rother, C. : Learning 6D Object Pose Estimation using 3D Object Coordinates. In : *European Conference on Computer Vision*, 2014 46

[Brachmann et al. 2016] Brachmann, E. ; Michel, F. ; Krull, A. ; Yang, M. ; Gumhold, S. : Uncertainty-Driven 6D Pose Estimation of Objects and Scene. In : *Conference on Computer Vision and Pattern Recognition*, 2016 9, 92

[Brégier et al. 2018] Brégier, R. ; Devernay, F. ; Leyrit, L. ; Crowley, J. L. : Defining the Pose of Any 3D Rigid Object and an Associated Distance. In : *International Journal of Computer Vision* 126 (2018), June, n. 6, pp. 571–596 39, 40, 57, 60

[Bui et al. 2018] Bui, M. ; Zakharov, S. ; Albarqouni, S. ; Ilic, S. ; Navab, N. : When Regression Meets Manifold Learning for Object Recognition and Pose Estimation. In : *International Conference on Robotics and Automation*, 2018 42

[Calli et al. 2017] Calli, B. ; Singh, A. ; Bruce, J. ; Walsman, A. ; Konolige, K. ; Srinivasa, S. ; Abbeel, P. ; Dollar, A. M. : Yale-Cmu-Berkeley Dataset for Robotic Manipulation Research. In : *International Journal of Robotics Research* 36 (2017), n. 3, pp. 261–268 47

[Chen and Medioni 1991] Chen, Y. ; Medioni, G. : Object modelling by registration of multiple range images. Image and vision computing. In : *Image and Vision Computing* (1991) 14

[Chum et al. 2003] Chum, O. ; Matas, J. ; Kittler, J. : Locally Optimized RANSAC. In : *German Conference on Pattern Recognition*, 2003 101

[Corona et al. 2018] Corona, E. ; Kundu, K. ; Fidler, S. : Pose Estimation for Objects with Rotational Symmetry. In : *International Conference on Intelligent Robots and Systems*, 2018 39

[Crivellaro et al. 2018] Crivellaro, A. ; Rad, M. ; Verdie, Y. ; Yi, K. M. ; Fua, P. ; Lepetit, V. : Robust 3D Object Tracking from Monocular Images Using Stable Parts. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018) 8, 32, 76, 77, 78, 79, 80

[Cybenko 1989] Cybenko, G. : Approximations by Superpositions of Sigmoidal Functions. In : *Mathematics of Control, Signals, and Systems* (1989), n. 4, pp. 303–314 61

[Delobel et al. 2015] Delobel, L. ; Aynaud, C. ; Aufrere, R. ; Debain, C. ; Chapuis, R. ; Chateau, T. ; Bernay-Angeletti, C. : Robust localization using a top-down approach with several LIDAR sensors. In : *International Conference on Robotics and Biomimetics*, 2015 14

[Deng et al. 2018] Deng, H. ; Birdal, T. ; Slobodan, I. : PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors. In : *European Conference on Computer Vision*, 2018 95

[Denninger et al. 2019] Denninger, Maximilian ; Sundermeyer, Martin ; Winkelbauer, Dominik ; Zidan, Youssef ; Olefir, Dmitry ; Elbadrawy, Mohamad ; Lodhi, Ahsan ; Katam, Harinandan : BlenderProc. In : *arXiv preprint arXiv:1911.01911* (2019) 46

[Do et al. 2018] Do, T. ; Cai, M. ; Pham, T. ; Reid, I. : Deep-6DPose: Recovering 6D Object Pose from a Single RGB Image. In : *arXiv Preprint*, 2018 34

[Doumanoglou et al. 2016] Doumanoglou, A. ; Kouskouridas, R. ; Malassiotis, S. ; Kim, T-K. : Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd. In : *Conference on Computer Vision and Pattern Recognition*, 2016 50

[Drost et al. 2017] Drost, B. ; Ulrich, M. ; Bergmann, P. ; Härtinger, P. ; Steger, C. : Introducing MVTec ITODD - A Dataset for 3D Object Recognition in Industry. In : *International Conference on Computer Vision Workshops*, 2017 50

[Drummond and Cipolla 2002] Drummond, T. ; Cipolla, R. : Real-time visual tracking of complex structures. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002 31

[Eggert et al. 1997] Eggert, D. ; Lorusso, A. ; Fisher, R. : Estimating 3D Rigid Body Transformations: A Comparison of Four Major Algorithms. In : *Machine Vision and Applications* 9 (1997), n. 5-6, pp. 272–290 94

[Fischler and Bolles 1981] Fischler, M. ; Bolles, R. : Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In : *Applications to Image Analysis and Automated Cartography*, 1981 25

[Funk et al. 2017] Funk, C. ; Lee, S. ; Oswald, M. R. ; Tsogkas, S. ; Shen, W. ; Cohen, A. ; Dickinson, S. ; Liu, Y. : 2017 ICCV Challenge: Detecting Symmetry in the Wild. In : *International Conference on Computer Vision Workshops*, 2017 113

[Ganin et al. 2016] Ganin, Y. ; Ustinova, E. ; Ajakan, H. ; Germain, P. ; Larochelle, H. ; Laviolette, F. ; Marchand, M. ; Lempitsky, V. : Domain-Adversarial Training of Neural Networks. In : *Journal of Machine Learning Research* (2016) 41, 76

[Geiger et al. 2012] Geiger, A. ; Lenz, P. ; Urtasun, R. : Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In : *Conference on Computer Vision and Pattern Recognition*, 2012 14, 18

[Goodfellow et al. 2014] Goodfellow, I. J. ; Pouget-Abadie, J. ; Mirza, M. ; Xu, B. ; Warde-Farley, D. ; Ozair, S. ; Courville, A. ; Bengio, Y. : Generative Adversarial Nets. In : *Advances in Neural Information Processing Systems*, 2014 41

[Hanin 2017] Hanin, B. : Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations. In : *arXiv Preprint*, 2017 61

[Harris and Stennett 1990] Harris, C. ; Stennett, C. : RAPID - A Video Rate Object Tracker. In : *British Machine Vision Conference*, 1990 30

[He et al. 2017] He, K. ; Gkioxari, G. ; Dollar, P. ; Girshick, R. : Mask R-CNN. In : *International Conference on Computer Vision*, 2017 9, 93, 98

[He et al. 2016] He, K. ; Zhang, X. ; Ren, S. ; Sun, J. : Deep Residual Learning for Image Recognition. In : *Conference on Computer Vision and Pattern Recognition*, 2016 103

[Henderson and Ferrari 2018] Henderson, Paul ; Ferrari, Vittorio : Learning to Generate and Reconstruct 3D Meshes with only 2D Supervision. In : *British Machine Vision Conference (BMVC)*, 2018 40

[Hinterstoisser et al. 2012] Hinterstoisser, S. ; Lepetit, V. ; Ilic, S. ; Holzer, S. ; Bradski, G. ; Konolige, K. ; Navab, N. : Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In : *Asian Conference on Computer Vision*, 2012 31, 41, 46, 58

[Hinterstoisser et al.] Hinterstoisser, S. ; Lepetit, V. ; Wohlhart, P. ; Konolige, K. : On Pre-Trained Image Features and Synthetic Images for Deep Learning 40, 76

[Hodaň et al. 2018] Hodaň, T. ; Michel, F. ; mann, E. ; Kehl, W. ; Buch, A. G. ; Kraft, D. ; Drost, B. ; Vidal, J. ; Ihrke, S. ; Zabulis, X. ; Sahin, C. ; Manhardt, F. ; Tombari, F. ; Kim, T.-K. ; Matas, J. ; Rother, C. : BOP: Benchmark for 6D Object Pose Estimation. In : *European Conference on Computer Vision*, 2018, pp. 19–35 15, 50, 71, 73

[Hodan et al. 2020a] Hodan, T. ; Sundermeyer, M. ; Drost, B. ; Labbé, Y. ; Brachmann, E. ; Michel, F. ; Rother, C. ; Matas, J. : BOP Challenge 2020 on 6D Object Localization. In : *European Conference on Computer Vision Workshops*, 2020 43

[Hodan et al. 2020b] Hodan, Tomas ; Barath, Daniel ; Matas, Jiri : EPOS: Estimating 6D Pose of Objects with Symmetries. In : *arXiv preprint arXiv:2004.00605* (2020) 37, 38, 57

[Hodaň et al. 2017] Hodaň, T. ; Haluza, P. ; Obdržaĺek, S. ; Matas, J. ; Lourakis, M. ; Zabulis, X. : T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects. In : *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017 20, 40, 46, 48, 78

[Hornik 1991] Hornik, K. : Approximation Capabilities of Multilayer Feedforward Networks. In : *Neural Networks* 4 (1991), n. 2, pp. 251–257 61

[Hu et al. 2020] Hu, Y. ; Fua, P ; Wang, W. ; Salzmann, M. : Single-Stage 6D Object Pose Estimation. In : *Conference on Computer Vision and Pattern Recognition*, 2020 38

[Hu et al. 2019] Hu, Yinlin ; Hugonot, Joachim ; Fua, Pascal ; Salzmann, Mathieu : Segmentation-Driven 6D Object Pose Estimation. In : *Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3385–3394 34

[Kaskman et al. 2019] Kaskman, R. ; Zakharov, S. ; Shugurov, I. ; Ilic, S. : HomebrewedDB: RGB-D Dataset for 6D Pose Estimation of 3D Objects. In : *International Conference on Computer Vision Workshops*, 2019 48

[Kehl et al. 2017] Kehl, W. ; Manhardt, F. ; Tombari, F. ; Ilic, S. ; Navab, N. : SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In : *International Conference on Computer Vision*, 2017 32, 40, 76

[Krizhevsky et al. 2012] Krizhevsky, A. ; Sutskever, I. ; Hinton, GE : Imagenet classification with deep convolutional neural networks. In : *Advances in Neural Information Processing Systems*, 2012 12

[Labbé et al. 2020] Labbé, Y. ; Carpentier, J. ; Aubry, M. ; Sivic, J. : CosyPose: Consistent multi-view multi-object 6D pose estimation. In : *European Conference on Computer Vision*, 2020 37, 57, 114

[Lee et al. 2018] Lee, H.-Y. ; Tseng, H.-Y. ; Huang, J.-B. ; Singh, M. ; Yang, M.-H. : Diverse Image-To-Image Translation via Disentangled Representations. In : *European Conference on Computer Vision*, 2018 41, 76

[Lepetit 2020] Lepetit, V. : Recent Advances in 3D Object and Hand Pose Estimation. In : *arXiv Preprint* (2020) 30, 32

[Lepetit and Fua 2005] Lepetit, V. ; Fua, P. : Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. In : *Foundations and Trends in Computer Graphics and Vision* (2005) 30

[Lepetit et al. 2009] Lepetit, V. ; Moreno-Noguer, F. ; Fua, P. : EPnP: An Accurate O(n) Solution to the PnP Problem. In : *International Journal of Computer Vision* (2009) 25, 101

[Li et al. 2018] Li, Yi ; Wang, Gu ; Ji, Xiangyang ; Xiang, Yu ; Fox, Dieter : DeepIM: Deep Iterative Matching for 6D Pose Estimation. In : *European Conference on Computer Vision*, 2018, pp. 683–698 37, 42

[Li et al. 2019] Li, Z. ; Wang, G. ; Ji, X. : CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation. In : *International Conference on Computer Vision*, 2019 36

[Liu et al. 2016] Liu, W. ; Anguelov, D. ; Erhan, D. ; Szegedy, C. ; Reed, S. E. ; Fu, C.-Y. ; Berg, A. C. : SSD: Single Shot Multibox Detector. In : *European Conference on Computer Vision*, 2016 33

[Long et al. 2015] Long, M. ; Cao, Y. ; Wang, J. ; Jordan, M. I. : Learning Transferable Features with Deep Adaptation Networks. In : *International Conference on Machine Learning*, 2015 41, 76

[Lowe 1991] Lowe, D. G. : Fitting Parameterized Three-Dimensional Models to Images. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991 30

[Lowe 2004] Lowe, D. G. : Distinctive Image Features from Scale-Invariant Keypoints. In : *International Journal of Computer Vision*, 2004 31

[Manhardt et al. 2018] Manhardt, F. ; Arroyo, D. M. ; Rupprecht, C. ; Busam, B. ; Navab, N. ; Tombari, F. : Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data. In : *CoRR* abs/1812.00287 (2018) 57

[Manhardt et al. 2019] Manhardt, F. ; Arroyo, D. M. ; Rupprecht, C. ; Busam, B. ; Navab, N. ; Tombari, F. : Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data. In : *arXiv Preprint*, 2019 1, 39

[Marchand et al. 1999] Marchand, E. ; Bouthemy, P. ; Chaumette, F. ; Moreau, V. : Robust real-time visual tracking using a 2D-3D model-based approach. In : *International Conference on Computer Vision*, 1999 31

[Marchand et al. 2016] Marchand, E. ; Uchiyama, H. ; Spindler, F. : Pose Estimation for Augmented Reality: A Hands-On Survey. In : *IEEE Transactions on Visualization and Computer Graphics*, 2016 30

[Muja and Lowe 2009] Muja, M. ; Lowe, D.G. : Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In : *International Conference on Computer Vision*, 2009 100

[Müller et al. 2018] Müller, F. ; Bernard, F. ; Sotnychenko, O. ; Mehta, D. ; Sridhar, S. ; Casas, D. ; Theobalt, C. : Ganerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. In : *Conference on Computer Vision and Pattern Recognition*, 2018 41

[Oberweger et al. 2018] Oberweger, M. ; Rad, M. ; Lepetit, V. : Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. In : *European Conference on Computer Vision*, 2018 34

[Park et al. 2020] Park, K. ; Patten, T. ; Vincze m. : Neural Object Learning for 6D Pose Estimation Using a Few Cluttered Images. In : *European Conference on Computer Vision*, 2020 41

[Park et al. 2019] Park, Kiru ; Patten, Timothy ; Vincze, Markus : Pix2pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. In : *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7668–7677 9, 35, 36, 92

[Peng et al. 2019a] Peng, S. ; Liu, Y. ; Huang, Q. ; Zhou, X. ; Bao, H. : PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In : *Conference on Computer Vision and Pattern Recognition*, 2019 66

[Peng et al. 2019b] Peng, Sida ; Liu, Yuan ; Huang, Qixing ; Zhou, Xiaowei ; Bao, Hujun : PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. In : *Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570 34, 35

[Pitteri et al. 2020] Pitteri, G. ; Bugeau, A. ; Ilic, S. ; Lepetit, V. : 3D Object Detection and Pose Estimation of Unseen Objects in Color Images with Local Surface Embeddings. In : *Asian Conference on Computer Vision*, 2020 94

[Pitteri et al. 2019] Pitteri, G. ; Ilic, S. ; Lepetit, V. : CorNet: Generic 3D Corners for 3D Pose Estimation of New Objects without Retraining. In : *International Conference on Computer Vision Workshops*, 2019 77

[Pitteri et al.] Pitteri, G. ; Ramamonjisoa, M. ; Ilic, S. ; Lepetit, V. : On Object Symmetries and 3D Pose Estimation from Images. In : *3* 58

[R. Haralick and Nolle 1991] R. Haralick, K. O. ; Nolle, M. : Analysis and Solutions of the Three Point Perspective Pose Estimation Problem. In : *Conference on Computer Vision and Pattern Recognition*, 1991 25

[Rad and Lepetit 2017] Rad, M. ; Lepetit, V. : BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects Without Using Depth. In : *International Conference on Computer Vision*, 2017, pp. 3848–3856 32, 57, 66, 84

[Rad et al. 2018a] Rad, M. ; Oberweger, M. ; Lepetit, V. : Domain Transfer for 3D Pose Estimation from Color Images Without Manual Annotations. In : *Asian Conference on Computer Vision*, 2018 41, 76

[Rad et al. 2018b] Rad, M. ; Oberweger, M. ; Lepetit, V. : Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images. In : *Conference on Computer Vision and Pattern Recognition*, 2018 114

[Rad et al. 2017] Rad, M. ; Roth, P. ; Lepetit, V. : Adaptive Local Contrast Normalization for Robust Object Detection and 3D Pose Estimation . In : *British Machine Vision Conference*, 2017 20

[Redmon et al. 2016] Redmon, J. ; Divvala, S. ; Girshick, R. ; Farhadi, A. : You Only Look Once: Unified, Real-Time Object Detection. In : *Conference on Computer Vision and Pattern Recognition*, 2016 33

[Ren et al. 2015] Ren, S. ; He, K. ; Girshick, R. ; Sun, J. : Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In : *Advances in Neural Information Processing Systems*, 2015 66, 67, 80

[Rennie et al.] Rennie, C. ; Shome, R. ; Bekris, K. E. ; Souza, A. F. D. : A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place 50

[Ronneberger et al. 2015] Ronneberger, O. ; Fischer, P. ; Brox, T. : U-Net: Convolutional Networks for Biomedical Image Segmentation. In : *Conference on Medical Image Computing and Computer Assisted Intervention*, 2015 96, 103

[Rublee et al. 2011] Rublee, E. ; Rabaud, V. ; Konolidge, K. ; Bradski, G. : ORB: An Efficient Alternative to SIFT or SURF. In : *International Conference on Computer Vision*, 2011 31

[Sahin et al. 2020] Sahin, C. ; G.G Hernando, J. S. ; Kim, T.K. : A Review on Object Pose Recovery: from 3D Bounding Box Detectors to Full 6D Pose Estimators. In : *Image and Vision Computing* (2020) 30

[Shi et al. 2020] Shi, Y. ; Huang, J. ; Zhang, H. ; Xu, X. ; Rusinkiewicz, S. ; Xu, K. : SymmetryNet: Learning to Predict Reflectional and Rotational Symmetries of 3D Shapes from Single-View RGB-D Images. In : *ACM SIGGRAPH* (2020) 113

[Simonelli et al. 2019] Simonelli, A. ; Bulò, S.R. ; Porzi, L. ; Antequera, M. L. òpez ; Kontschieder, P. : Disentangling monocular 3d object detection. In : *International Conference on Computer Vision*, 2019 38

[Song et al. 2020] Song, C. ; Song, J. ; Huang, Q. : HybridPose: 6D Object Pose Estimation under Hybrid Representations. In : *Conference on Computer Vision and Pattern Recognition*, 2020 35

[Sundermeyer et al. 2020a] Sundermeyer, M. ; Durner, M. ; Puang, E. Y. ; Marton, Z. C. ; Vaskevicius, N. ; Arras, K. O. ; Triebel, R. : Multi-path Learning for Object Pose Estimation Across Domains. In : *Conference on Computer Vision and Pattern Recognition*, 2020 3, 37, 42, 92, 94, 104, 105

[Sundermeyer et al. 2018] Sundermeyer, M. ; Marton, Z.C. ; Durner, M. ; Brucker, M. ; Triebel, R. : Implicit 3D Orientation Learning for 6D Object Detection from RGB Images. In : *European Conference on Computer Vision*, 2018 39, 41, 42, 57, 71, 74, 76, 85, 114

[Sundermeyer et al. 2020b] Sundermeyer, Martin ; Marton, Zoltan-Csaba ; Durner, Maximilian ; Triebel, Rudolph : Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection. In : *International Journal of Computer Vision* 128 (2020), n. 3, pp. 714–729 36, 37

[Tejani et al. 2014] Tejani, A. ; Tang, D. ; Kouskouridas, R. ; Kim, T-K. : Latent-Class Hough Forests for 3D Object Detection and Pose Estimation. In : *European Conference on Computer Vision*, 2014 50

[Tekin et al. 2018] Tekin, B. ; Sinha, S. N. ; Fua, P. : Real-Time Seamless Single Shot 6D Object Pose Prediction. In : *Conference on Computer Vision and Pattern Recognition*, 2018 33, 66

[Tobin et al. 2017] Tobin, J. ; Fong, R. ; Ray, A. ; Schneider, J. ; Zaremba, W. ; Abbeel, P. : Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In : *International Conference on Intelligent Robots and Systems*, 2017 41

[Tremblay et al. 2018] Tremblay, J. ; To, T. ; Sundaralingam, B. ; Xiang, Y. ; Fox, D. ; Birchfield, S. : Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. In : *Conference on Robot Learning (CoRL)*, 2018 66

[Tzeng et al. 2015] Tzeng, E. ; Hoffman, J. ; Darrell, T. ; Saenko, K. : Simultaneous Deep Transfer Across Domains and Tasks. In : *International Conference on Computer Vision*, 2015 41, 76

[Wang et al. 2019] Wang, H. ; Sridhar, S. ; Valentin, J. Huang J. ; Song, S. ; Guibas, L. J. : Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In : *Conference on Computer Vision and Pattern Recognition*, 2019 42, 57

[Wohlhart and Lepetit 2015] Wohlhart, P. ; Lepetit, V. : Learning Descriptors for Object Recognition and 3D Pose Estimation. In : *Conference on Computer Vision and Pattern Recognition*, 2015 42

[Xiang et al. 2018] Xiang, Y. ; Schmidt, T. ; Narayanan, V. ; Fox, D. : PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In : *Robotics: Science and Systems Conference* (2018) 33, 46, 47, 81

[Zakharov et al. 2018] Zakharov, S. ; Planche, B. ; Wu, Z. ; Hutter, A. ; Kosch, H. ; Ilic, S. : Keep It Unreal: Bridging the Realism Gap for 2.5D Recognition with Geometry Priors Only. In : *International Conference on 3D Vision*, 2018 41, 76

[Zakharov et al. 2019] Zakharov, S. ; Shugurov, I. ; Ilic, S. : DPOD: Dense 6D Pose Object Detector in RGB Images. In : *arXiv Preprint*, 2019 9, 35, 92

[Zakharov et al. 2017] Zakharov, S. ; Wadim, K. ; Planche, B. ; Hutter, A. ; Ilic, S. : 3D Object Instance Recognition and Pose Estimation Using Triplet Loss with Dynamic Margin. In : *International Conference on Intelligent Robots and Systems*, 2017 42

[Zhang 2017] Zhang, Z. : Iterative point matching for registration of free-form curves and surfaces. In : *International Journal of Computer Vision*, 2017 14

[Zhou et al. 2019] Zhou, Y. ; Barnes, C. ; Lu, J. ; Yang, J. ; Li, H. : On the continuity of rotation representations in neural networks. In : *Conference on Computer Vision and Pattern Recognition*, 2019 37

[Zhu et al. 2017] Zhu, J.-Y. ; Park, T. ; Isola, P. ; Efros, A. A. : Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In : *International Conference on Computer Vision*, 2017 41, 76