# Pose estimation of rigid objects and robots

Yann Labbé

PSL
UNIVERSITÉ PARIS

**THÈSE DE DOCTORAT**
**DE L'UNIVERSITÉ PSL**

Préparée à École Normale Supérieure

# Pose estimation of rigid objects and robots


Soutenue par
**Yann Labbé**
Le 5 juin 2023

École doctorale n°386
**Sciences Mathématiques de Paris Centre**

Spécialité
**Informatique**


Composition du jury :

| | |
|---|---|
| Vincent Lepetit<br>École des Ponts ParisTech | *Rapporteur* |
| Andrew Davison<br>Imperial College London | *Rapporteur* |
| Eric Marchand<br>IRISA, Inria Rennes | *Examinateur* |
| Natalia Neverova<br>Meta AI Research | *Examinateur* |
| Josef Sivic<br>Czech Technical Institute | *Directeur de thèse* |
| Ivan Laptev<br>Inria Paris | *Codirecteur de thèse* |

ENS | PSL

# Acknowledgements

Josef, thank you for making this scientific and human adventure possible. None of the following research would have been written down in this thesis if it weren't for you giving me the opportunity to join the Willow team. I couldn't have hoped for a better mentor to learn how to become a researcher. Your attention to detail, poetry-level writing style, and rigorous attitude have greatly inspired my research. Your caring attitude and continuous support were key to make it through the hard times. Thank you.

Mathieu, thank you for your support and immense technical help. I will forever remember a few of our discussions where your insights were crucial for solving month-running problems in a few minutes.

Justin, thank you for your support and for the collaborative atmosphere you bring to the lab. Our opinions have often diverged, but our numerous interactions helped me grow as a roboticist and mature as a person.

Writing an exhaustive list of everyone who had an impact on me and hence indirectly on this thesis would be far too long and would only expose myself to the risk of forgetting some of you. A smile at the other end of a corridor, a small compliment, a listening ear, a diner conversation, a few messages exchanged once in a while, a career advice, an invitation to a party, a regular lunch partner, a travel advice, climbing partners, a hike, a few hours cycling in your company, days spent together. Ranging from someone I briefly talked with to my dearest friends, all of you positively contributed in one way or another. Thank *you*.

Papa, Maman, merci pour les valeurs humaines et professionelles que vous m'avez transmises. Votre support inconditionnel m'a toujours permis de poursuivre mes rêves.

Romain, May-Lise, Alban et Martin, je vous remercie pour tout ce que vous m'avez apporté, et ce que vous continuez de m'apporter. Je ne serais pas la personne que je suis aujourd'hui sans ces innombrables soirées passés chez vous, sans ces moments passés auprès de mes deux neveux, sans votre soutien infaillible. J'espère qu'Alban et Martin seront heureux un jour de lire la thèse de tonton Python[1], ou du moins d'en lire ces remerciements. Je vous dédie cette thèse.

[1] Python est le langage de programmation avec lequel la plupart des méthodes présentées dans cette thèse ont été implémentées.

# Résumé

L'objectif de cette thèse est de développer des méthodes permettant d'estimer la configuration 3D de scènes contenant des objets rigides et des robots articulés dont les modèles 3D sont connus, en utilisant une ou plusieurs images RGB en entrée. Nous considérons les scènes difficiles et les conditions visuelles suivantes: (i) des objets sans textures et / ou symétriques (ii) des robots avec plusieurs degrés de liberté, (iii) des scènes imagées dans des conditions difficiles (en terme de point de vue ou d'éclairage) et (iv) des objets ou des robots partiellement occlus.

Les principales contributions de cette thèse sont les suivantes. Tout d'abord, nous introduisons une méthode pour identifier un nombre variable d'objets dans l'espace de travail d'un robot et estimer les coordonnées 2D des centroïdes des objets dans le système de coordonnées du robot. Notre approche ne nécessite pas de calibration extrinsèque caméra-robot. Deuxièmement, nous proposons une méthode pour résoudre efficacement le problème de réarrangement. Nous proposons une paramétrisation d'action discrète de ce problème, et appliquons efficacement Monte-Carlo Tree Search (MCTS) pour le résoudre. Troisièmement, nous introduisons une nouvelle méthode basée sur l'apprentissage pour l'estimation 6D de la pose d'objets rigides dont les modèles 3D sont connus. Notre approche repose sur une stratégie de rendu et de comparaison. Nous introduisons des innovations dans le paramétrage de la fonction de coût et de l'orientation de l'objet pour gérer explicitement les symétries d'objets et obtenir un entraînement stable. Nous entraînons notre approche sur des données synthétiques en utilisant des augmentations d'images importantes et montrons l'importance cruciale de l'augmentation des données pour le transfert vers des scènes réelles. Quatrièmement, nous introduisons une approche multi-objet multi-vues pour l'estimation de pose. Nous introduisons une nouvelle stratégie RANSAC au niveau des objets pour estimer conjointement les poses relatives des caméras et trouver des correspondances entre les hypothèses de poses prédites dans chacune des vues indépendamment. Les poses des objets et des caméras sont affinées conjointement en résolvant un problème d'optimisation. Cinquièmement, nous étendons notre troisième contribution pour estimer la pose de nouveaux objets, c'est-à-dire des objets inconnus pendant l'entraînement. Nous introduisons un réseau de notation pour trouver la meilleure estimation initiale parmi un ensemble d'hypothèses grossières, et un réseau pour le raffinement itératif où la forme de l'objet et le système de coordonnées sont implicitement fournis en entrée. Les réseaux sont entraînés sur un nouvel ensemble de données synthétiques à grande échelle affichant des milliers d'objets différents dans des conditions visuelles difficiles. Enfin, nous introduisons une méthode pour estimer la pose 6D et les angles articulaires d'un robot articulé. Nous étendons la stratégie de rendu et de comparaison pour gérer les robots avec plusieurs degrés de liberté. Nous montrons l'importance cruciale du paramétrage du robot dans ce problème, et proposons une stratégie efficace et indépendante du robot.

Les méthodes présentées dans cette thèse font progresser l'état de l'art sur les benchmarks existants pour l'estimation de la pose d'objets et de robots. Pour les objets rigides connus, notre approche CosyPose est la méthode qui a gagné le BOP Challenge 2020. Notre approche pour les objets inconnus pendant l'entraînement, MegaPose, atteint des performances similaires tout en ne nécessitant pas que les objets soient connus à l'avance pour l'entraînement, ouvrant la voie à des applications où le déploiement rapide est crucial.

# Abstract

The goal of this thesis is to develop methods for recovering the 3D configuration of scenes containing rigid objects and articulated robots with known 3D models using one or multiple RGB images as inputs. We consider the following challenging scenes and visual conditions: (i) textureless and/or symmetric objects (ii) robot arms with several degrees of freedom, (iii) scenes imaged under challenging conditions (e.g. viewpoint or illumination) and (iv) objects or robots partially occluded.

The key contributions of this thesis are as follows. First, we introduce a method for identifying a variable number of objects in a robot's workspace and estimate the 2D coordinate of the object's centroids in the robot coordinate frame. Our approach does not require extrinsic camera-to-robot calibration. Second, we propose a method for efficiently solving the planar rearrangement planning problem. We propose a discrete action parametrization of this problem, and efficiently apply Monte-Carlo Tree Search (MCTS) to solve it. Third, we introduce a novel learning-based method for 6D pose estimation of rigid objects with known 3D models. Our approach relies on the render-and-compare strategy. We introduce innovations of the training loss and rotation parametrization to explicitly handle object symmetries and achieve stable training. We train our approach on synthetic data using heavy image augmentations and show the crucial importance of data augmentation for the transfer to real scenes. Fourth, we introduce an approach for multi-view multi-object 6D pose estimation. We introduce a novel object-level RANSAC strategy to jointly estimate relative camera poses and find correspondences between single-view pose hypotheses. Poses of all objects and cameras are jointly refined by solving an object-level bundle adjustment problem. Fifth, we develop an approach to estimate the pose of novel objects, i.e. objects unseen during training, but for which the 3D model is available at test time. We introduce a scoring network for finding the best initial estimate among a set of coarse hypotheses, and design a network for iterative refinement where the object shape and coordinate system are implicitly provided as inputs. The model is trained on a novel large-scale synthetic dataset displaying thousands of different objects in challenging visual conditions. Finally, we introduce a method for estimating the 6D pose and joint angles of an articulated robot. We extend the render-and-compare strategy to handle robots with several degrees of freedom. We show the crucial importance of robot parametrization in this problem, and propose an effective strategy that is independent of the robot.

The methods presented in this thesis advance the state-of-the-art on existing datasets and benchmarks for object and robot pose estimation. For known rigid objects, our single-view approach CosyPose is the winning entry in the BOP Challenge 2020. Our approach for unseen objects, MegaPose, achieves similar performance while not requiring the objects to be known in advance for training, paving the way for real applications where rapid deployment is key.

# Contents

# Chapter 1

# Introduction

## 1.1 Goals

The goal of this thesis is to develop methods for recovering a 3D configuration of a scene using visual sensors. We consider scenes containing rigid objects and articulated robots with known geometric descriptions, i.e. CAD models and kinematic description. Given these descriptions of real-world assets (objects or robots), we consider the following problems: (i) identifying which of the known assets appear in a given scene, and (ii) what are the unknown parameters defining the *state* of these assets within the 3D scene. The state of a rigid object is defined by its 6D pose, composed of 3D rotation and 3D translation with respect to a reference coordinate frame located, e.g. at the camera. The state of a robot is defined by (i) the 6D pose with respect to a reference coordinate frame, and (ii) the joint parameters, e.g. the angle value of a revolute joint.

Existing methods for these tasks suffer from limitations that prevent them from being widely deployed to real-world applications in open and uncontrolled environments. In this thesis, the focus is on methods that (i) can be applied to any type of assets, including texture-less or symmetric objects and robots with high degrees of freedom; (ii) can be applied to cluttered scenes where the observed assets are occluded; (iii) do not rely on costly dedicated sensors (e.g. depth or LIDAR); (iv) do not require tedious calibration of the acquisition system (e.g. camera-to-robot calibration) or other time consuming procedures specific to a given scene or objects; (v) can deal with environment variations during execution on a real robotic system; (vi) are end-to-end learnable and trained entirely on synthetic data without manual annotations; (vii) do not require the CAD models of the objects to be known in advance for training. Examples of scenes obtained by methods developed in this thesis are shown in figures 1-1 and 1-2.

## 1.2 Motivation

Objects play a central role in how humans and robots interact with their environment. Robots are instances of articulated objects which are omnipresent in the context of robot manipulation (e.g. a robot handing an object to a human) where knowledge of the robot configuration within the scene is crucial for planning a task and avoiding collisions. Iden-

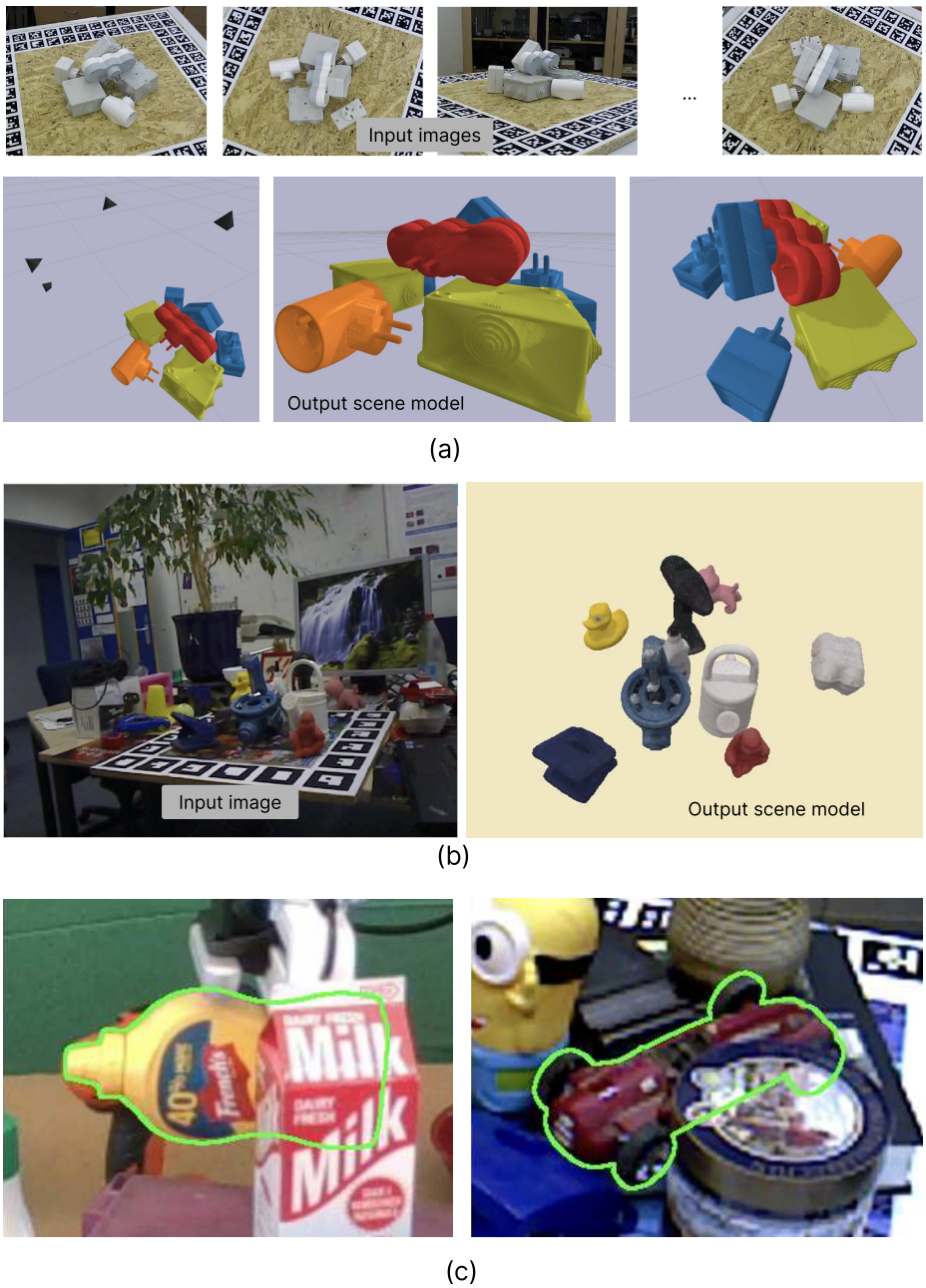
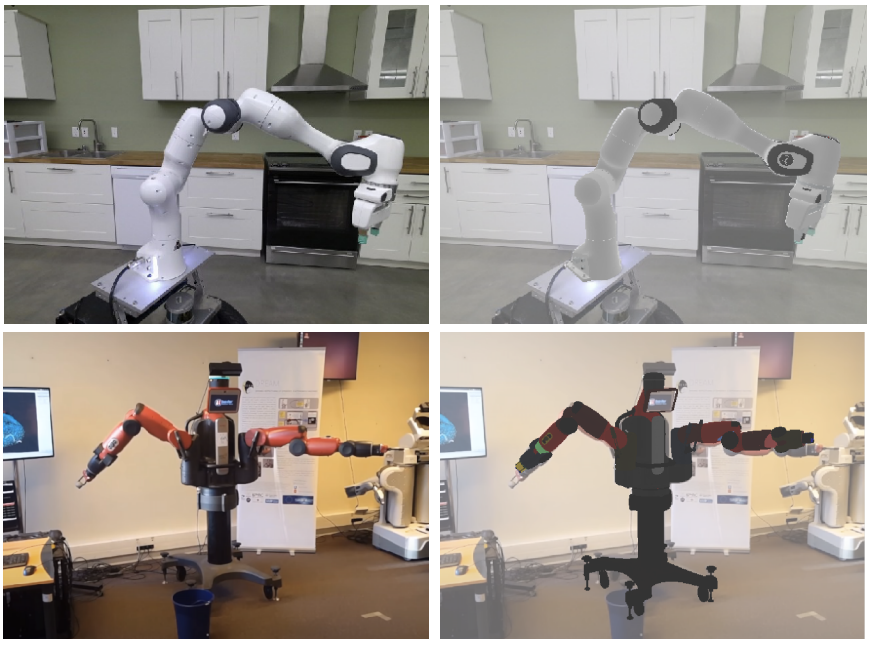
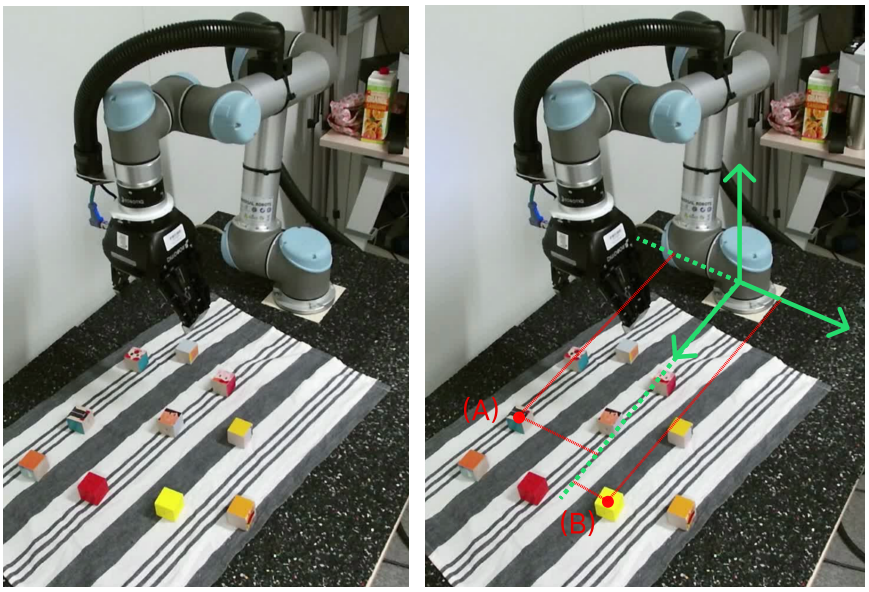



Figure 1-1: **Goal of the thesis: reconstructing the configuration of scenes containing multiple objects.** In (a), multiple RGB images captured from cameras with unknown viewpoints are available as illustrated in the top row of images. The goal is to identify the objects among a set of known 3D models (not displayed here) and recover their poses as well as the poses of the cameras. Output scene model is shown in the bottom row. In (b), the goal is to detect objects and recover their 6D poses using a single RGB image of a cluttered scene. In (c), we show the high accuracy that we aim to achieve in this thesis for objects whose CAD models are known at test time, but not in advance for training. The green contours represent the contours of the object models projected into the image. The results presented in this figure are obtained with the methods presented in chapters 4 and 5.



(a)




(b)

Figure 1-2: **Goal of the thesis: reconstructing the configuration of scenes containing robots.** (a) Given a single RGB image of a known robot (left column), the goal is to recover the state of the robot: it's 6D pose with respect to the camera and the joint angles. The right column shows for each example an overlay of the robot model in the estimated configuration. In (b), a robot and multiple objects are visible in the scene. Given a single RGB image (left), the goal is to estimate the location of each object (colored cubes) on the table. We show the 2D coordinates of two objects (A and B) with repect to the 2D (on the table) coordinate system of the robot placed at the base of the robot. The results presented in this figure are obtained with the methods presented in chapters 3 and 6.

Figure 1-3: **Motivation.** Examples of robotic applications requiring visual object and robot state estimation. In (a) [80], (b) [62], (c) [16] a mobile robot performs household tasks: (a) loading a glass into a dishwasher (b) serving a cup of water (c) storing food items in a drawer. In (d) [75], (e) [162] a robot performs industrial tasks: (d) storing and picking objects from a shelf, (e) bin picking. In (f) [195], two robots collaborate to assemble parts of an Ikea chair.

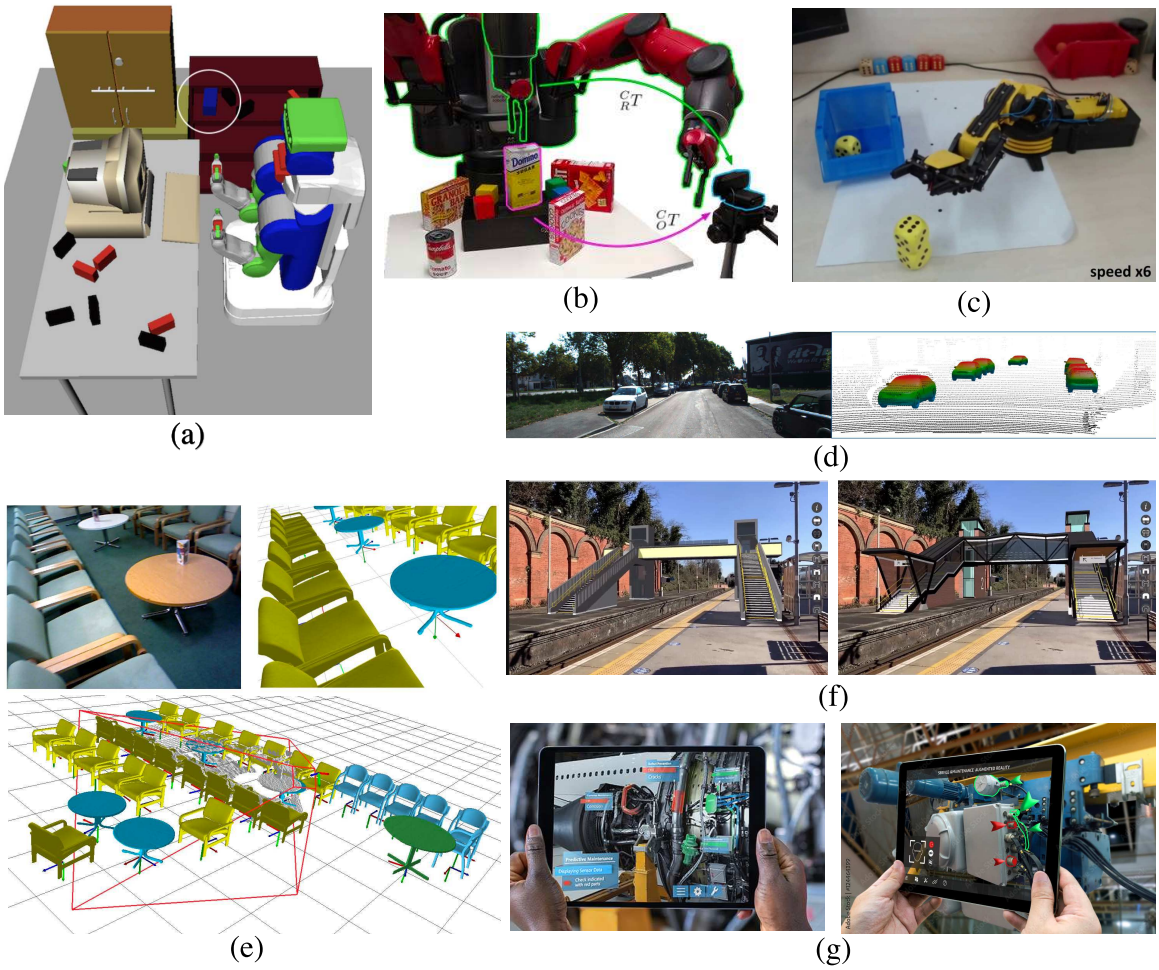



Figure 1-4: **Motivation.** Example applications of estimating the 3D state of objects and robots in a scene. In (a) [88], the object-level reconstruction of a scene allows a robot to plan a motion to grasp an object on a shelf while avoiding collision with other objects in the environment. In (b) [208], estimating the pose of a robot and objects in the camera coordinate frame enables object manipulation without time-consuming camera-to-robot calibration. In (c) [247], an inexpensive robot without rotary encoders on its motors is controlled to stack objects using visual joint angle estimation. In (d) [220] and (e) [178], an object-level SLAM algorithm leverages object detection and pose estimation to jointly create a map of the environment and localize the camera, with applications to (d) autonomous driving and (e) indoor localization. In (f) [4], estimating the pose of an urban building (left) allows an architect to directly visualize variations of the building with augmented reality (right). In (g) [190], pose estimation of industrial objects with respect to the camera of a hand-held consumer tablet provides information to a human operator performing service & maintenance operations.

tifying the objects and their spatial configurations within a scene is a task humans perform seamlessly with their cognitive visual system. Yet, this task remains challenging to perform for machines equipped with visual sensors like cameras. Several important real-world applications would be enabled by the development of algorithms that can quickly process digital images coming from one or multiple cameras into a simple and explicit scene representation: a list of objects associated with their 3D configuration within the observed scene. Some of these applications are discussed below.

**Robotic manipulation.** Deploying autonomous robotic systems that can interact with the environment and each other in open and uncontrolled environments would have a profound impact on society, by helping or replacing humans performing tedious tasks in industrial environments or for personal assistance. Example applications are illustrated in figure 1-3. Current robotic manipulation systems are however typically constrained to specific scenes, where it is possible to design and instrument an entire system for the given environment and the set of objects: e.g by designing a production line to ensure manufactured objects always arrive at the same position with respect to a robot, by placing fiducial markers on objects beforehand or by placing a fixed camera and calibrating it with respect to the robot. The deployment of existing systems to novel environments is thus limited by costly and environment-specific instrumentation and calibration steps. Estimating the accurate position and orientation of all objects within any scene would allow robots to plan, navigate and interact with any novel environment. In this thesis, we make a step towards this vision by developing an approach that can estimate the 6D pose of objects that have a known 3D shape (e.g. in the form of a CAD model) but this CAD model does not need to be known in advance for training, only at test time. The object-level representation of a scene considered in this thesis is particularly suited for modular robotic systems and can be directly combined with existing planning and control algorithms as illustrated in figure 1-4(a)(b).

**Multi-robot interaction.** Estimating the 3D state of a robot from the images of a camera would allow one robot to understand the behavior of another robot, perform collaborative tasks and avoid collisions during manipulation or navigation. An example of two robots performing a collaborative task is illustrated in figure 1-3.

**Low-level visually-guided robotic control.** Estimating values of robot joints from a camera provides visual measurements that can be used to replace robot's proprioceptive sensors (e.g. the rotary encoders mounted on the motors of the robot revolute joints), or increase their accuracy. Knowledge of a robot state is critical for controlling its actuators with a closed-loop feedback loop. Visual state measurements can be useful for controlling inexpensive robots (such as the Owi-535 illustrated in figure 1-4) that have no proprioceptive sensors. For other robots, the proprioceptive sensor measurements alone may be unreliable, for example on a robot with revolute joints composed of one motor and a gear box. The gear box introduces mechanical angular backlash in the articulations due to mechanical flexibilities, which are not measured by the rotary encoders on the motors. Such errors accumulate and can be large for robots with high degrees of freedom like humanoid

robots. Again, using visual measurements coming from an external camera would allow reducing such errors.

**Simultaneous localization and mapping.** Robotic navigation requires a robot to be able to localize itself within a map of the environment. Environments are often composed of objects and can be represented using an object-level description, i.e. a list of objects associated with their poses in the environment. Building such a map in real-time using one or multiple moving cameras while also localizing the cameras in the environment is a problem known as object-level SLAM [178]. Algorithms for solving this problem strongly rely on detection and pose estimation of objects, and their performance correlates with the performance of the detection and pose estimation algorithms, which is the focus of this thesis. Applications of object-level SLAM include autonomous driving of a car or indoor navigation and localization of a mobile robot, as illustrated in figure 1-4.

**Virtual and augmented reality.** A virtual object-level scene reconstruction can be enhanced with other computer-generated visual information. The augmented scene can be visualized by humans in a virtual environment using a virtual reality headset (virtual reality, VR), or superimposed on an image of the real world (augmented reality, AR). Applications of AR and VR are numerous and include art visualization, education, urban design and planning, architecture, or service&maintenance operations. Example applications are illustrated in figure 1-4(f)(g).

**3D scene understanding.** Object-level reconstruction of everyday scenes provides information that can be used to understand how humans interact with their environment. Applications of scene understanding include cognitive sciences and can be used for example to develop artificial intelligence inspired by humans. For example, reconstructing the trajectory of a tennis racket during a swing, or the pose of a bottle with respect to a glass during pouring could be used to extract motion primitives for training robots to perform similar actions.

## 1.3 Challenges

Identifying objects or robots and estimating their state within a scene is important for many applications. There are however several challenges that need to be addressed:

- **Viewpoint, illumination.** The visual appearance of an object depends on its pose with respect to the camera (which we seek to estimate), but also on external factors such as the lighting conditions, as illustrated in figure 1-5, or the presence/absence of visual occluders as illustrated in figure 1-6.
- **Symmetries.** The pose of an object is defined by the object coordinate system attached to its CAD model, which is arbitrary and can be ambiguous. If the object is symmetric, multiple poses are equivalent and correct for an object in a given image, as illustrated in figure 1-7.

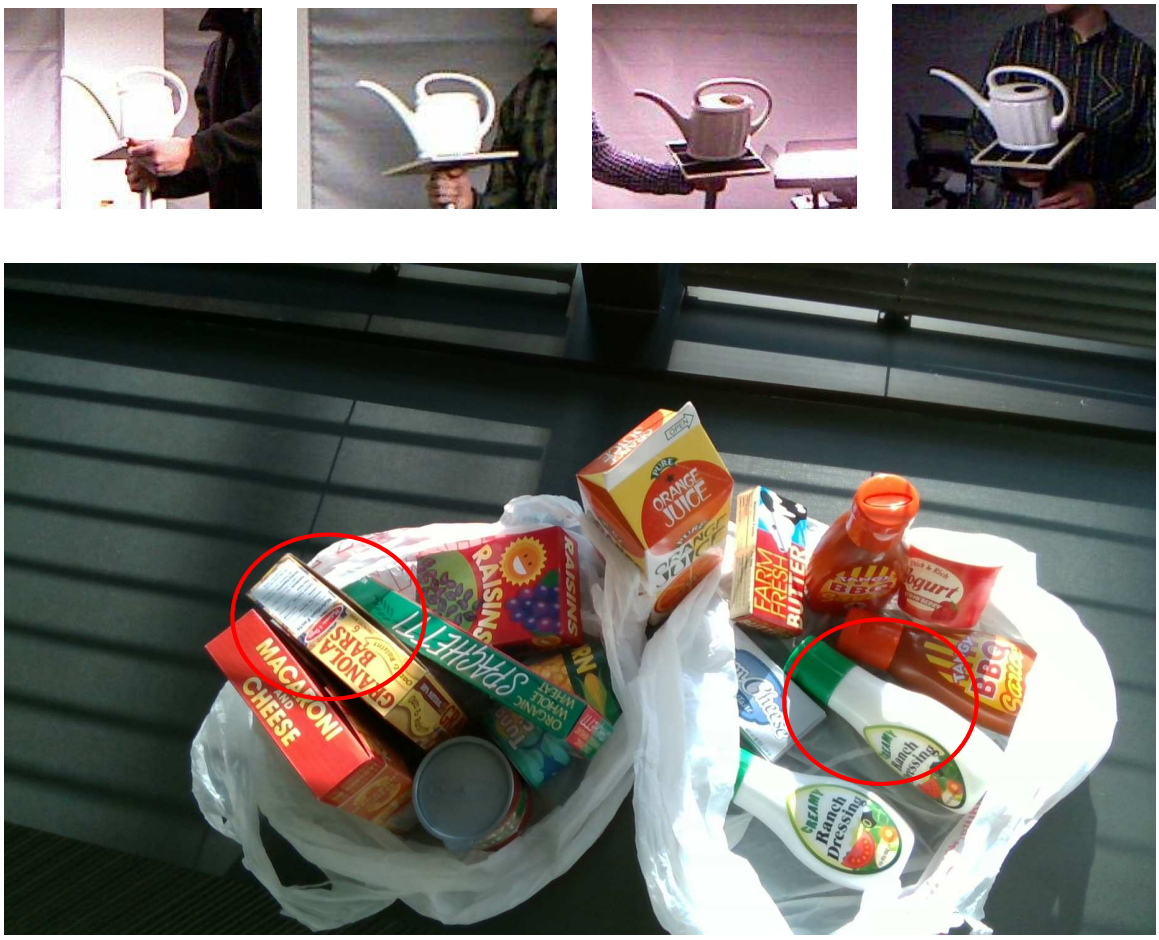
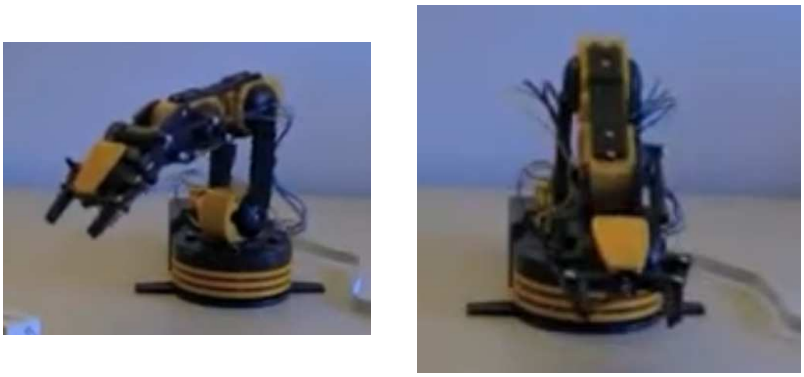



Figure 1-5: **Challenges of pose estimation — illumination variations.** In the top row, we show four different images of the same object under different illumination conditions. Notice how the visual appearance of each object is affected. In the bottom, we show the influence of shadows on the visual appearance of objects (see red ovals). Images are from the TUD-L [70] (top) and HOPE [211] (bottom) datasets.

(a)

(b)



(c)

Figure 1-6: **Challenges of object detection and pose estimation — Occlusions.** (a) illustrates examples of scenes where objects are occluded by another object. Notice how some objects are barely visible from certain viewpoints (right). (b) [125] shows other examples of dynamic occlusions where an object is occluded by an active manipulator, a human hand (right), or a robotic gripper (left). (c) illustrates self-occlusions of articulated robots: parts of the robot are occluded by other parts for certain configurations (right). Images are from the LM-O [66] (a, left), IC-BIN [35] (a, right), DexYCB (b, right), and Owi535-Youtube [247] (c) datasets.

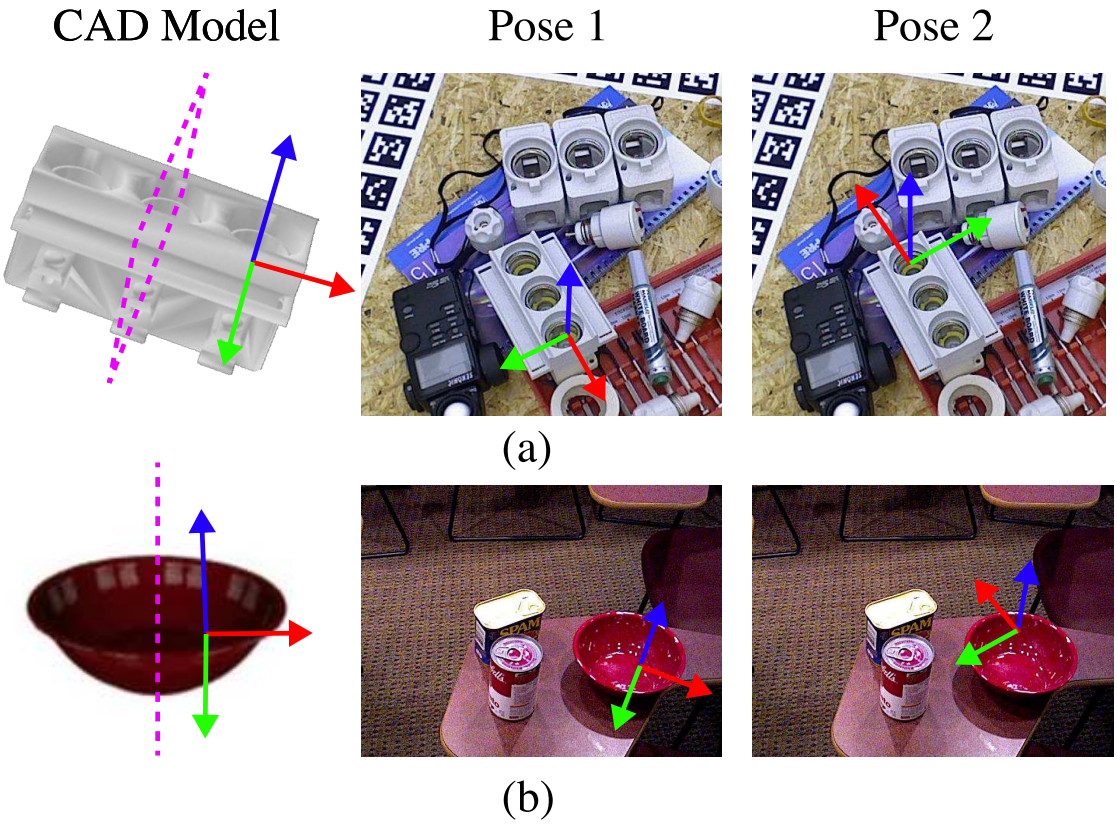
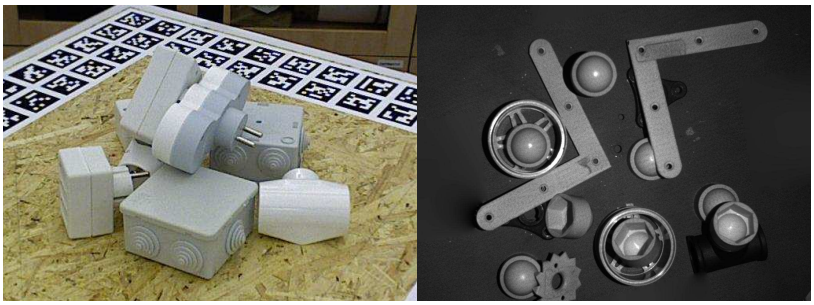


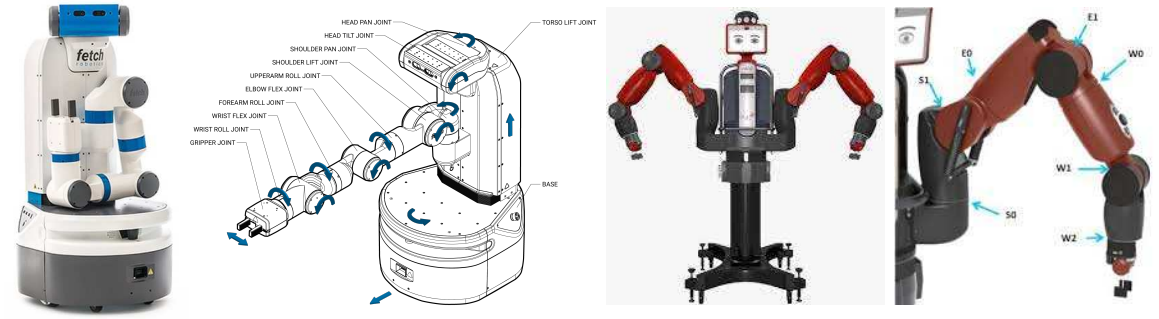

Figure 1-7: **Challenges of pose estimation — Object symmetries.** Two examples of symmetric objects. On the left is the object's CAD model with the coordinate system defining its 6D pose, and the object symmetries: a plane (top row) or an axis of revolution (bottom row). On the right, we show two possible pose estimates for the same observation of each object. Images and objects are from the T-LESS [68] (top) and YCB-V [230] (bottom) datasets.


Textured objects



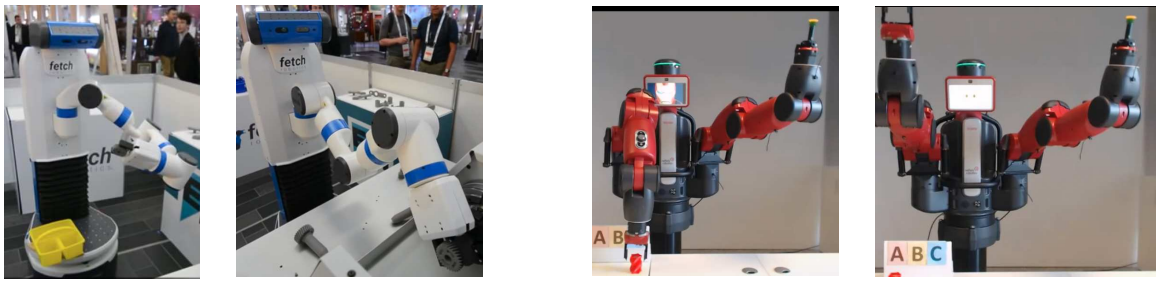
Textureless objects

Figure 1-8: **Challenges of object detection and pose estimation — Presence or absence of textures.** We show examples of textured objects (top) and textureless objects (bottom). The visual details on the textured objects are important to define their pose, while the pose of textureless objects entirely depends on their global shape. Images are from [211] (top-left), [22] (top-right), [68] (bottom-left), and [38] (bottom-right).




Fetch mobile manipulator Rethink Robotics Baxter

(a) Definition of an articulated robot's parts and articulations




(b) Different robot configurations

Figure 1-9: **Challenges of robot pose estimation — Infinite number of possible configurations.** The top row illustrates, for two robots, that robots can have a high number of parts and joints. For example, the robotic arm of the fetch manipulator has 7 degrees of freedom (DoFs), and the bi-arm Baxter robot has 15 DoFs. The number of possible configurations is infinite, and robots can have very different visual appearance when imaged in a different configuration even when observed from similar viewpoints (bottom).

- **Textureless objects.** The pose of an object depends on characteristics specific to each object, such as its shape and texture for textured objects, or its shape only for textureless objects, as illustrated in figure 1-8.
- **Articulations.** Estimating the state of a robot requires estimating the values of its joints. This is challenging because robots can have many degrees of freedom, and each joint affects the position of multiple robot parts, as illustrated in figure 1-9.

## 1.4 Contributions

We now summarize the main contributions of this thesis, which we divide into six main themes.

**(1) Object-robot relative 2D localization.** We develop a method for identifying a variable number of objects in a robot's workspace and estimate the 2D coordinate of the object's centroids in a coordinate frame centered on the robot. This information is sufficient for performing top-down grasps with the robot. The method operates from a single input RGB image and uses the robot as an implicit calibration object. It therefore can be used without any prior intrinsic camera calibration or camera-to-robot extrinsic calibration. The key innovation is to combine (i) a neural network that predicts a dense position field mapping each pixel of the input image to a 2D coordinate in the robot workspace with (ii) an object identification stage relying on a mean-shift clustering algorithm for identifying individual objects. The method is trained entirely on synthetic data generated using domain randomization. In addition, the method does not require knowledge of the CAD models of the objects in the scene. Our method is presented in chapter 3.

**(2) Rearrangement planning.** We propose a method for planar rearrangement planning with many movable objects, i.e. finding a sequence of robot actions to move a set of objects from an initial arrangement to a desired one using overhand grasps. The object arrangements can, for example, be estimated visually. We take inspiration from the success of Monte-Carlo Tree Search (MCTS) for solving complex sequential decision problems, which was for example used to develop artificially intelligent systems capable of beating the best players at the games of chess or Go. The key innovation is a novel discrete action parametrization of the rearrangement planning problem which allows us to efficiently apply MCTS. Our method is presented in chapter 3.

**(3) Learning-based single-view 6D pose estimation of known rigid objects.** We develop *single-view CosyPose*, a method for 6D pose estimation of rigid objects from a single RGB image. We take inspiration from the success of methods based on the iterative render-and-compare strategy. We propose a method that can be trained to perform 6D pose estimation given a CAD model of an object, a detection in the form of a 2D bounding box of that object in an RGB image, and the intrinsic parameters of the camera used to capture it. Key to the success of our method is to leverage recent advances in the design of deep learning models for state estimation used in related areas to achieve stable training. In

particular, (i) we use a training loss that disentangles the effect of the different predictions (i.e. rotation and translations), and (ii) we use a parametrization of the 3D rotation which is less sensitive to regression errors compared to quaternions. We explicitly handle object symmetries in the loss during training to ensure that the gradients used to train the network are not ambiguous for symmetric objects. The learning-based nature of the approach enables us to leverage visual appearance variations of any object in the training data to gain robustness with respect to illumination variations and occlusions. We generate synthetic images of physically plausible scenes using a set of known objects. We apply heavy data augmentation to the images and train the network on these, and show it generalizes to real images of the same known objects. We show this heavy data augmentation is key to the success of the method. Single-view CosyPose is presented in chapter 4.

**(4) Multi-view multi-object detection and 6D pose estimation.** We develop *multi-view CosyPose*, a method for recovering the 6D pose of multiple objects and cameras into a single consistent scene. Our method takes inspiration from dense 3D reconstruction methods but operates at the level of objects instead of operating at the level of low-level primitives like points, lines, or patches. Multi-view CosyPose is designed to address the main limitations inherent to single-view pose estimation such as failures in the case of severe occlusions, or difficulty to estimate depth along the camera $z$ axis. Our first key innovation is a novel object-level RANSAC procedure to (i) match *hypotheses* of object candidates (i.e. single-view pose estimates) across the different views of the same scene, (ii) identify the outliers, (iii) and estimate the relative camera poses. Our second key innovation consists in jointly optimizing the poses of objects and cameras in the scene in order to best explain the single-view inlier candidates by solving an object-level bundle adjustment problem where object symmetries are handled explicitly. Multi-view CosyPose is presented in chapter 4.

**(5) Learning-based single-view 6D pose estimation of novel rigid objects.** We introduce MegaPose, a method for single-view 6D pose estimation of rigid objects which addresses the main drawback of single-view CosyPose. Namely, CosyPose requires the objects of interest to be known in advance to generate training data specific to these objects and train the network. Data generation and training require hours or days, which makes it unusable in open environments where novel objects are encountered frequently. We develop a new method based on the render-and-compare strategy which is agnostic to the specific objects we seek to estimate the 6D pose of. The key innovation is to implicitly provide as input all information required to estimate the pose of an object as input, i.e. its visual shape and information about its coordinate system. Hence, no object-specific information needs to be encoded in the network weights during training. MegaPose is trained using tens of thousands of object CAD models and generalizes to novel objects with CAD models available only at test time. Our MegaPose approach is presented in chapter 5.

**(6) Robot pose and joint angle estimation.** We develop a novel method for estimating the 6D pose and joint angles of a known robot. We take inspiration from the success of approaches based on the render-and-compare strategy which has been successful for pose estimation of rigid objects and develop a new method able to handle the specificities of

articulated robots. One of the key challenges is to generalize to novel robot configurations unseen during training. Our first key innovation is to use the centroid of the robot as a reference point for estimating the robot pose with respect to the camera, which can be visually estimated for novel configurations. Not relying on a specific robot part allows us to gain robustness with respect to external occlusions and self-occlusions. Our second key innovation is a simple automatic solution for choosing the *anchor part*, i.e. the part of the robot whose pose with respect to the camera is independent of the joint angles values. We show this choice of anchor part is important and show our solution can be applied to different robots. RoboPose is presented in chapter 6.

## 1.5 Outline of the thesis

In chapter 2, we review the literature relevant to this thesis in the areas of object pose estimation of rigid and articulated objects.

In chapter 3, we present an integrated robotic system for performing closed-loop online rearrangement planning of objects present on a flat surface in a robot's workspace. We first introduce a method for identifying objects and estimating their 2D location with respect to the robot given a single uncalibrated RGB camera. We then present a method for performing multi-object rearrangement planning in a workspace with no available buffer space. We combine both approaches and present real robot experiments on a UR5 robot demonstrating the robustness of our system with respect to environment variations like background or types of objects as well as external variations like moving the camera or perturbing the object positions during the rearrangement.

Chapter 4 introduces CosyPose, a method for estimating the 6D pose of known rigid objects with known CAD models given one or multiple images captured from cameras with unknown viewpoints. We first present a method based on the render-and-compare strategy for single-view 6D pose estimation of individual objects. Notably, this approach achieves state-of-the-art results on several pose estimation benchmarks, winning five awards in the BOP Challenge 2020. We then present a multi-view framework based on a robust object-level matching strategy and global scene refinement achieved by solving an object-level bundle adjustment problem. This multi-view framework addresses the main limitations inherent to single-view pose estimation and significantly improves the performance of the single-view approach. It is also practical to use in robotic applications because (i) it can be used with any existing single-view approach; (ii) it is relatively fast, taking $300ms$ to estimate the complete state of a scene with 6 objects using 4 cameras; and (iii) it can be used with any cameras with known intrinsic parameters without any additional assumptions.

In chapter 5, we introduce MegaPose. MegaPose is an approach for 6D pose estimation of any novel object with an available CAD model. This approach addresses the main drawbacks of the single-view pose estimation of CosyPose introduced in chapter 4. Namely, CosyPose requires the objects of interest to be known in advance in order to generate synthetic data and train the network, which can take hours or days, and is thus impractical to use in robotic scenarios where new objects are frequently encountered. MegaPose is trained on a large-scale synthetic dataset generated with large databases of CAD models and generalizes to novel objects with available CAD models but unseen during training.

Notably, MegaPose achieves performance competitive with CosyPose while not requiring the objects to be known in advance for training. We experimentally show the method can be used for grasping novel objects on a real robot.

In chapter 6, we consider the problem of estimating the 6D pose and joint angles of a known robot given a single image. We introduce a new deep render-and-compare method for this task trained entirely on synthetic data that generalizes to new unseen robot configurations at test time. The approach relies on a novel parametrization to iteratively update jointly the pose and the joint angles of an observed robot. The proposed parametrization is independent of the robot structure and can be applied to a variety of robots. We evaluate the method on existing benchmarks demonstrating significant improvements over the state-of-the-art for four different robots with up to 15 degrees of freedom and apply the method to videos from Youtube depicting robots in various un-instrumented environments.

Chapter 7 concludes this thesis by summarizing the main results and contributions, presenting recent development in the literature that relies on the presented work, and suggesting multiple avenues for future research.

## 1.6 Publications and software

During the development of this thesis, four papers were presented in major conferences and journals in computer vision and robotics (RAL'2020, ECCV'2020, CVPR'2021, CoRL'2022). In addition, an extended version of a paper is available as a technical report. The following chapters of this thesis present the material from our publications:

- The first version of the work presented in chapter 3 was presented at the *Scalable Learning for integrated Perception and Planning* workshop at RSS 2019. An extended version was then published in the Robotics and Automation Letters (RAL) 2020 [100].
- The work presented in chapter 4 was published at the European Conference on Computer Vision (ECCV) 2020 [96] and an extended version presenting the winning results in the BOP challenge 2020 [71] is available as a technical report [97].
- The work presented in chapter 5 was published at the Conference on Robot Learning (CoRL) 2022 [99].
- The work presented in chapter 6 was published at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021 [98].

All the software, pre-trained models and datasets developed during this thesis are available at https://www.github.com/ylabbe under open-source licenses.

# Chapter 2

# Literature review

This thesis builds on more than 60 years of research from different fields related to computer vision and robotics. The literature review is split into three parts. In section 2.1, we review methods for estimating the 6D camera-to-object pose of a single known rigid object using one view as input. We split the main existing methods for this task into different families. For each family, we show how methods have evolved through the years with general advances in computer vision for extracting image features: edge-based methods, local features, and learning-based methods. We also discuss important aspects of the deployment of pose estimation methods: pose estimation of novel objects, detection and instance recognition as well as tracking. In section 2.2, we review methods that use multiple images for reconstructing scenes and estimating the poses of all camera viewpoints. Finally, section 2.3 reviews methods for reconstructing the state of scenes containing robots.

In all sections, the focus is on methods that can be applied to RGB images but we also discuss the main works that assume depth measurements are available.

## 2.1 Single-view camera-to-object 6D pose estimation

Pose estimation of rigid objects from RGB is one of the oldest problems in computer vision [172]. The camera pose with respect to a known rigid object was recovered by establishing correspondences between (i) two-dimensional points in the image and (ii) three-dimensional points on the object model. 60 years later, many research works still present advances on how to establish such geometric correspondences, and how to recover the object pose given a set of correspondences. We review the literature on correspondence-based methods in section 2.1.1. In [143], an alternative approach was presented. Many *template images* of the object under pre-determined camera viewpoints are captured using an instrumented robotic setup. These images are projected into a lower-dimensional *appearance representation* space. Given a novel image, the object pose is determined by interpolation between the poses of the closest image templates, where similarity is measured using the appearance representation. Several other works have extended this idea, using different approaches for modeling the viewpoint-dependent object appearance. Methods based on object appearance are reviewed in section 2.1.2. In section 2.1.3, we present methods that directly predict the object's rotation and translation through regression or classification with

neural networks without explicit correspondences or object appearance representations. In section 2.1.4, we discuss methods that can be applied directly to novel objects without costly training or pre-processing. Section 2.1.5 presents the main approaches for 6D pose tracking, i.e. estimating the 6D pose of an object in a video sequence by leveraging temporal continuity. Finally, section 2.1.6 discusses how to detect an object in an image and identify it among a set of known 3D models. Object detection and identification are important for reconstructing scenes containing many objects. It can however be considered as a separate problem different than 6D object pose estimation where the goal is to recover an object pose given (i) a region of interest displaying that object in an image and (ii) a model of that object. We also refer the reader to surveys on rigid object pose estimation [43, 72, 130].

### 2.1.1 Correspondence-based methods

In the case of projective cameras, the pose of an object with respect to the camera can be recovered given four or more 2D-3D correspondences. The problem, known as PnP, can be solved with an iterative solution [31] or in closed-form [108]. In practice, estimated correspondences are always noisy and a RANSAC strategy [42] is used. We now review methods for establishing a set of 2D-3D correspondences.

**Edge features.** Early works in object pose estimation [172] model objects as a set of primitives such as line segments or points representing edges on the objects. Primitives are detected in the images by analyzing the intensity gradients, e.g. using Sobel filters. Given a set of detected lines, several different methods were proposed to identify objects in the object database and recover their pose. We illustrate these approaches with [123] in figure 2-1. [6, 13, 77, 78, 123] rely on a "hypothesize and test" strategy [123]: (i) a small set of matches is hypothesized, and (ii) an initial object pose is computed using the hypothesis then tested by measuring the errors between the projection of the 3D model in the hypohtesized pose and the location of the measured primitives. A search over the set of possible matches leads to finding a good initial pose estimate. The pose is then further refined using newton-based optimization with all correspondences. Later works also consider different types of primitives such as faces and distance/angles between faces [50] or curves [40].

**Local image features.** Edge-based methods discard the rich information in image brightness and/or color pattern to identify correspondences. In [175], illustrated in figure 2-2, an alternative approach leveraging this information is proposed. Objects are modeled as a set of 3D patches. Each patch is associated with (i) a local image descriptor invariant under geometric and photometric changes and (ii) the position of the 3D patch within the object coordinate system. Given a novel image of that object, local invariant regions are then detected in an image. The description of each region is then used jointly with spatial relationships to find matches between 2D features and 3D patches. Such methods enable pose estimation for which estimating the pose requires reasoning about the object texture and not only its edges, e.g. for a textured cylindrical object. The same idea is extended in [22,

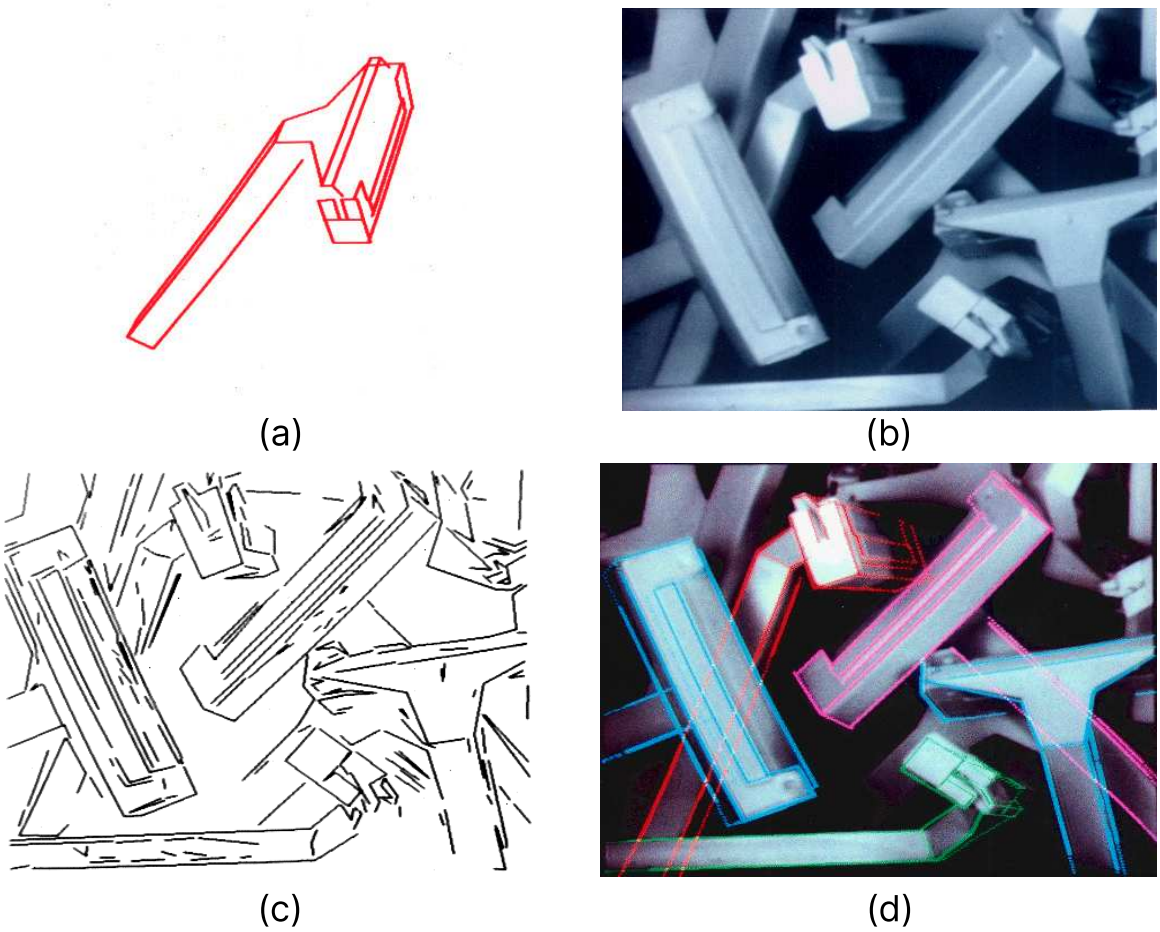



Figure 2-1: **6D pose estimation with edge-based features [123].** An object is represented as a 3D wire-frame model (a). Given an input image (b), image contours are extracted (c). These contours are then matched with the contours of the object's model and the matches are used to detect the objects and recover their 6D poses. (d) shows the 3D model of the object in the estimated poses using the SCERPO [123] system. The system can handle textureless objects occluded in the input image. Figure reproduced from [123].

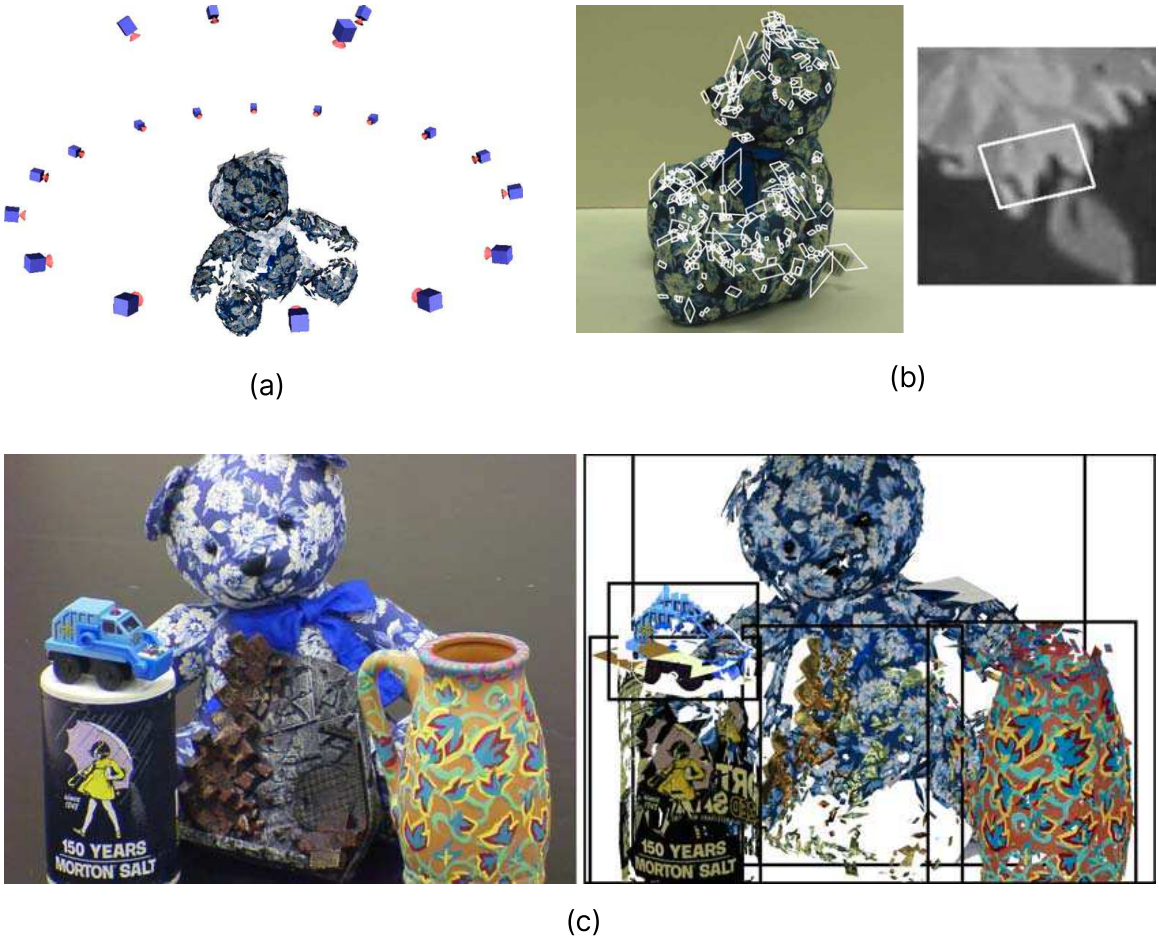



Figure 2-2: **6D pose estimation with local image features [174].** An object is represented using a set of 3D patches described by local features. This object model is created using multiple viewpoints of the object (a). (b) shows a few examples of 3D patches. In (c), the objects detected in the input image (left) are shown on the right in the estimated poses. Figure reproduced from [174].

23] where an efficient implementation is proposed to detect objects and estimate their poses in real time using one or multiple images. This approach can be used with other types of local features such as SIFT [124], SURF [9] or MSER [132], after taking into account their covariance properties. BOLD [204] features can be used for textureless objects.

**3D features.** If depth measurements are available, 3D features can be directly detected in the image and matched with features extracted from object models. Several 3D features have been proposed such as Shot [179], KPQ [136], ISS [244], MeshDog [212]. The most commonly used are Point-Pair features [39] and its variants [1, 11, 20, 37, 67, 89, 92, 213]. We refer to [72] for an overview of the differences between these methods. In this thesis, the focus is on RGB-only pose estimation where depth measurements are not available. If depth measurements are available, the poses predicted using RGB-only images can be refined using a registration technique like Iterative Closest Points (ICP) [243].

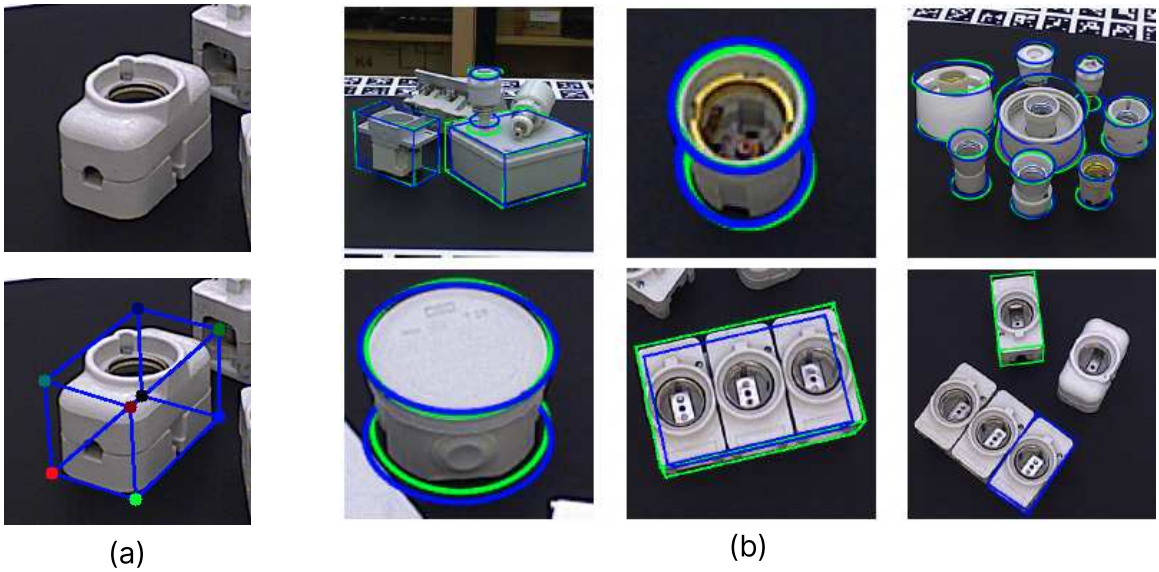



Figure 2-3: **Predictions of sparse keypoints using a deep neural network [169].** The method is illustrated in (a). Given an input image (top), a CNN is used to predict the 2D location of 8 keypoints corresponding to the reprojections of the object's 3D bounding box. Predicted keypoints are shown in the bottom image. Results are shown in (b) for textureless and symmetric objects. Figure reproduced from [169].

**Learning-based correspondences.** In [169], illustrated in figure 2-3, a method based on a deep network is used to predict the location of the 8 corners of the 3D bounding box of an object. In this approach, the network directly predicts the correspondences between the 2D image and the 3D model, and the pose is recovered by solving an EPnP problem. Other learning-based approaches rely on predicting a sparse set of correspondences. [202] extends the approach [169] to handle multiple objects simultaneously without a separate detection stage, and [207] proposes to train a network similar to [169] using synthetic data. [76, 158, 160] rely on semantic object keypoints. To gain robustness to occlusions, dense correspondences can be predicted using a CNN [69, 113, 155, 191, 240], as illustrated in figure 2-4. Estimating the 6D pose can be done using RANSAC and PnP similar to earlier works where dense correspondences were established using local features. While correspondences are established using deep neural networks, these methods rely on non-learning-based PnP and RANSAC stages for pose estimation.

**Fiducial markers.** An alternate approach to all the ones presented above is to place markers on the objects to ease the process of establishing correspondence between the image and the object. The most popular strategy is based on April tags [41, 45, 151]. Such an approach can only be used in instrumented environments where it is possible to precisely place markers on each object. These methods are also not robust to occlusions of the marker.

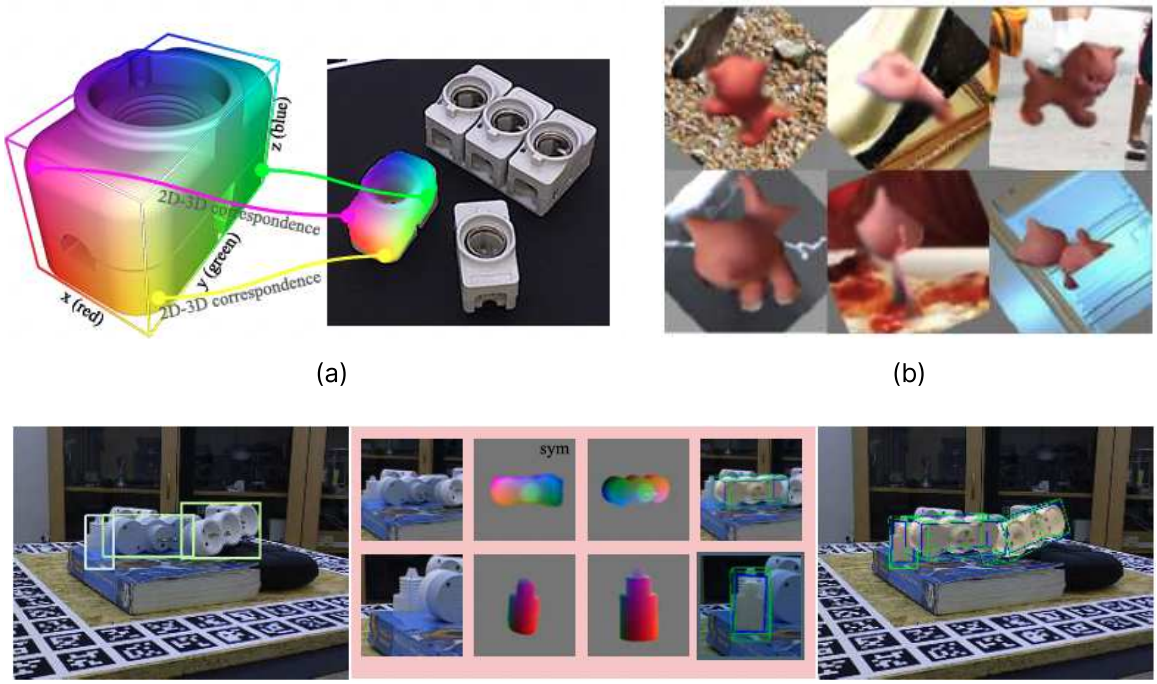



Figure 2-4: **Dense keypoint predictions using a deep neural network [155].** The method is illustrated in (a). Given an input image (right), a CNN is used to map each pixel to a 3D position on the object's model (left). The CNN is trained using a data augmentation strategy that simulates occlusions (b). Example predictions for textureless objects severely occluded are shown in (c). Figure reproduced from [155].

### 2.1.2 Methods based on global appearance

In [143], images of an object are captured under different camera viewpoints with known poses using an instrumented robotic setup with a turntable. This allows building an appearance representation mapping images to a space where the euclidean distance is representative of the similarity between the poses of the object in two different images. During inference, retrieving the closest templates of an image in this appearance space allows for fast estimation of the pose.

**Handcrafted representation.** In [143], illustrated in figure 2-5, the appearance space is created using the raw pixel intensities. The appearance space, or eigenspace, is constructed by computing the most prominent eigenvectors of the set. However this space is not invariant to illumination, and thus the method requires generating a large number of templates with varying illumination. This problem can be addressed by modeling an object's appearance using edges and oriented pixels as in [150]. Edges are typically extracted using a Canny edge detection method, which is sensitive to illumination changes, noise and blur. Images gradient orientations instead of image contours can be used to solve this problem, as proposed in [65]. Surface normals orientations can also be added to gain robustness with respect to background clutter as proposed in [64, 66]. While these handcrafted representations can be applied to any kind of objects without a training procedure, they cannot benefit from being trained on a large amount of data.

**Learning-based features.** A CNN can be used to project an image into a learned appearance representation space as presented in [226]. The CNN is trained using a metric learning loss combining pair-wise and triplet-wise terms that encourage two descriptors of the same object in similar poses to be close in the appearance space, and far in every other case. This method requires annotations of the 3D orientations of the objects and is sensitive to symmetry ambiguities. A representation space can be learned using a CNN without explicit SO(3) orientation using an auto-encoder as done in [199], illustrated in figure 2-6, where the auto-encoder is trained to reconstruct object images augmented with domain randomization.

### 2.1.3 Direct pose estimation

These methods directly estimate the position and orientation of the object given an image. Due to the difficulty of finding explicit relations between raw pixel intensities and an object's 6D pose, these methods are learning-based and rely on neural networks. We make the difference between two regression-based approaches. The first family of methods relies on a single-shot pose estimation network that directly estimates the pose given an input image. The second family of methods iteratively refines the pose using a render-and-compare strategy. The initial estimate can be provided by any method, but typically some method from the first family. Methods presented in this section have an advantage of only relying on learning-based components, allowing to improve their performance by scaling up the training datasets.

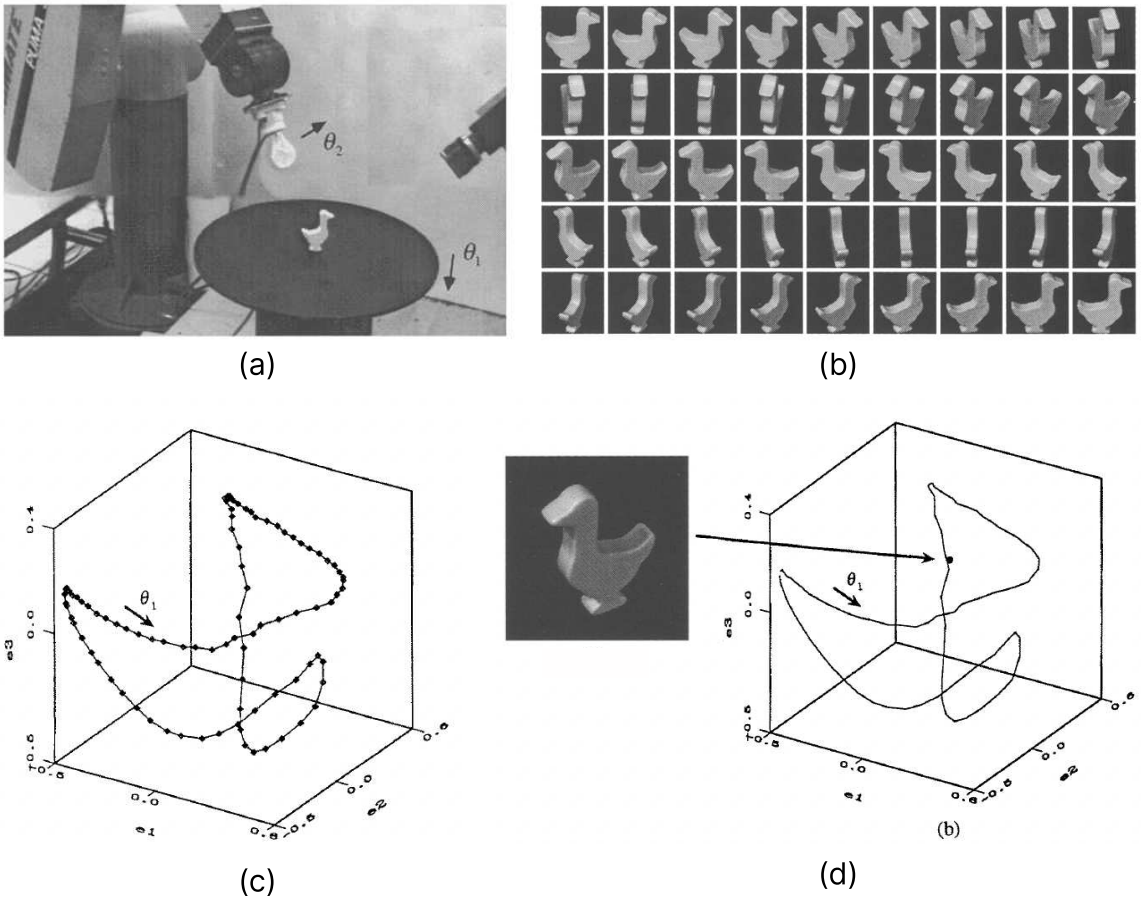



Figure 2-5: **Pose estimation using global appearance and image templates [143].** An instrumented setup composed of a robot and a turntable (a) is used to capture image templates of an object under different viewpoints (b). The image templates annotated are projected in a low-dimensional eigenspace shown in (c), where each point represents an image template. (d) shows the inference strategy. Given an image, the recognition system projects the image to the eigenspace. The exact position of the projection on the manifold determines the object's pose. Figure reproduced from [143].

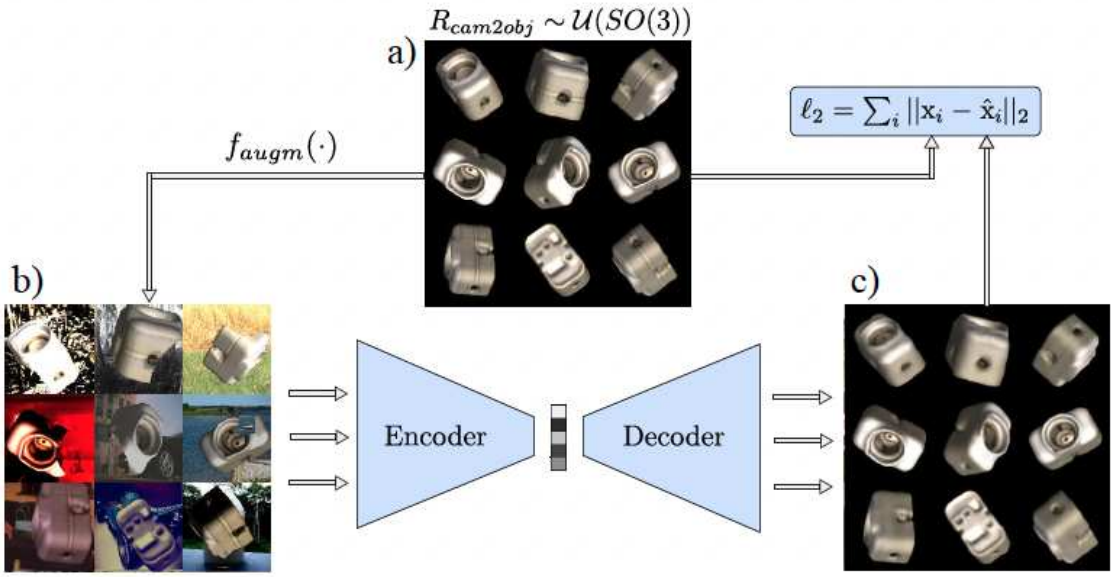



Figure 2-6: **Using a CNN to extract a global appearance representation [199].** Given a set of object models (a), images of the objects are generated with various augmentations (b). An auto-encoder is trained to reconstruct the images without augmentations (c). The latent space of the auto-encoder provides a global appearance representation of an object in the observed pose. Figure reproduced from [199].

**One-shot pose estimation.** SSD6D [87], illustrated in figure 2-7 extends the single-shot object detector SSD [120] to predict a pose for each detection. A classification-based approach is employed, where the space of possible rotations is discretized. The translation of the object is, however, computed using explicit geometric approximations based on the position and size of the predicted object bounding box. The 3D translation can be directly predicted by a network using a tesselation of SE-3 as done in [109], where a classification network is thus employed to directly predict the full 6D pose. The 3D translation can also be predicted using a regression-based network with a differentiable layer for translation estimation as done in PoseCNN [230].

**Iterative regression via render-and-compare.** The pose of a known object can be obtained by iteratively estimating the pose differences between a real observed image and a rendering of the object. The estimated pose of the object corresponds to the pose of the rendered object once the rendering and observed images are closely similar. Similarity between two images and pose gradients can be computed using mutual information (MI) [28, 184], used in [15] for object pose estimation. Later on, CNNs are used to measure similarity and compute pose gradients. A CNN is used to define an energy function between an image of an object and a rendering of the observed object in a given pose in [95]. The pose is estimated by minimizing this energy using a regression-classification random forest. [147, 148] uses a pose update CNN that is trained to directly refine the error between a rendering and an observed image of a hand. The pose is optimized by successively rendering the current estimate and refining the pose using the refinement network multiple times.

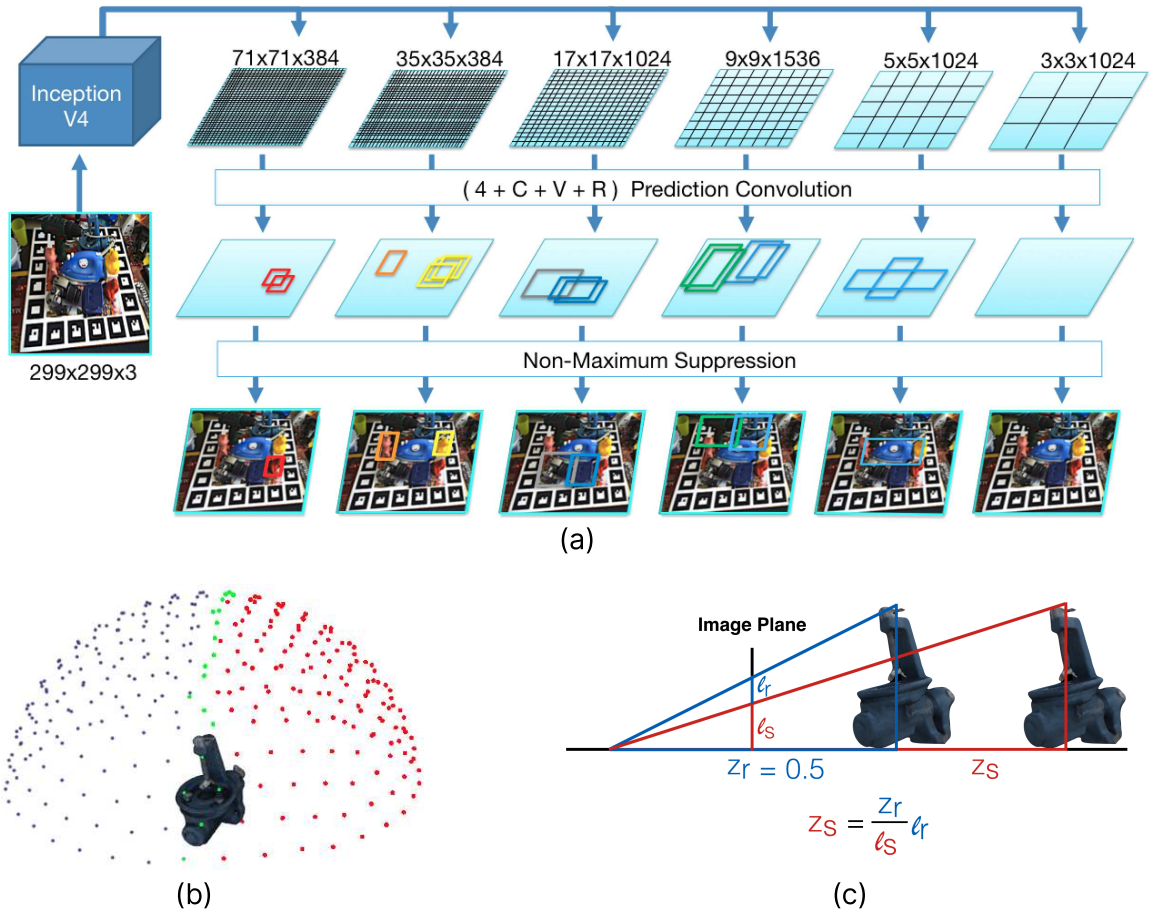



Figure 2-7: **One-shot pose estimation [87].** A CNN (a) is used to directly detect the position and size of objects in the image as well as their orientation. (b) shows the discretization of the space of possible object orientations. (c) illustrates the simple geometric reasoning used to recover the $z$-axis translation of the object pose given the size of the object's 2D bounding box. Figure reproduced from [87].

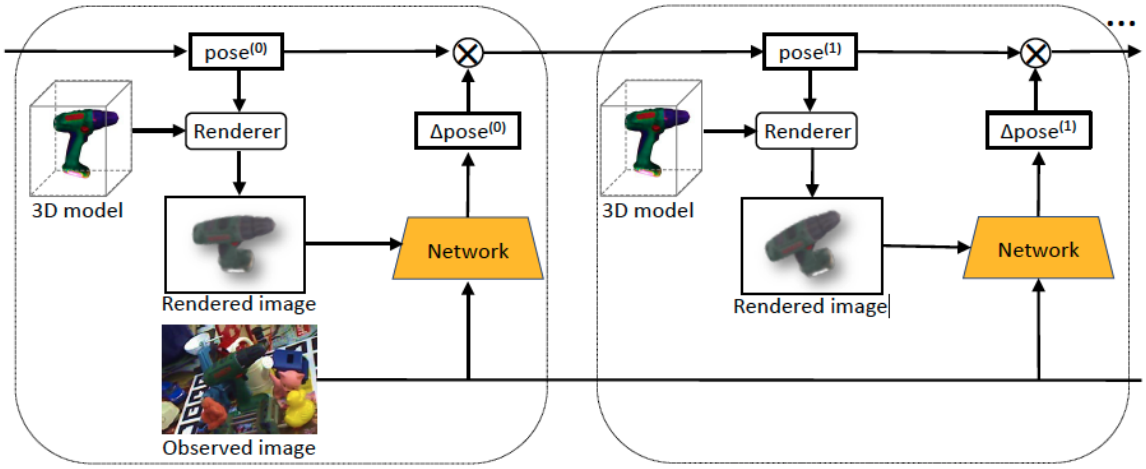



Figure 2-8: **Pose estimation via iterative render-and-compare from [112].** Given an input $\text{pose}^{(0)}$, the object's 3D model is used to render an image of the object in the input pose. A network taking as input the rendered and observed images predicts a pose update $\Delta\text{pose}^{(0)}$. The same strategy is applied multiple times iteratively. Figure reproduced from [112].

The approach is extended to handle objects in [169], where the refinement network iteratively updates 2D reprojection of the corners of the 3D object bounding box. [127] uses a refinement CNN that directly regresses rotation and translation and uses a proxy visual alignment loss for training. In [112], illustrated in figure 2-7, a novel parametrization of the pose update that does not use intermediate keypoints and can be trained end-to-end is introduced. Rotation and translation are disentangled. The rotation update is estimated in the camera coordinate systems to be invariant with respect to the orientation of the object's coordinate system. The translation is updated using intermediate scale and 2D reprojection of the object center in the image, but operations are made differentiable using camera geometry. The refinement network is trained end-to-end using a loss function on the pose. In chapter 4, we present a novel approach based on render-and-compare. Our approach builds on the parametrization introduced DeepIM [112] but we introduce a novel disentangled loss that leads to more stable training and improved performance.

### 2.1.4 Pose estimation of novel objects

Non-learning-based methods require limited efforts to handle novel objects. Edge features of a model can be easily obtained by processing an object's CAD model, and building a representation of the object based on 3D patches can be done using a small number of object views [175]. These methods have been outperformed by learning-based methods [71]. Learning-based methods presented in the previous sections however require obtaining training data and training for the objects of interest. They cannot generalize to novel objects due to object-specific information (appearance and coordinate system) being encoded in the network weights during training. In this section, we review learning-based methods that can be applied to novel objects without re-training.


Figure 2-9: **Category-level pose estimation with normalized coordinates [219].** (a) shows the normalized object coordinate space, a 3D space contained within a unit cube. For a given category, e.g. camera, the orientation of all instances is aligned and their size is normalized in the unit cube. The space is shared across all instances of a given category, allowing for directly predicting 2D-3D correspondences of novel objects of a known category. Example categories are shown in (b): bowls, cameras, cans, laptops and mugs. Figure reproduced from [219].

**Category-level pose estimation.** In category-level pose estimation methods [19, 110, 118, 128, 129, 219, 221], illustrated by [219] in figure 2-9, the CAD models of the test objects are not known, but the objects are assumed to belong to a known category. These methods rely on object properties that are common within categories to define and estimate the object pose, and thus cannot generalize to novel categories. For many types of objects like industrial ones, the notion of a class is often ambiguous and such methods can therefore not be easily applied to estimate the pose of novel objects.

**Pose estimation with available mesh.** Learning-based methods relying on correspondences, appearance representation, and direct estimation have been extended to handle novel objects. [164, 165] uses a keypoint detection network to predict a sparse set of the 2D location and parameters of generic 3D corners. [185] predicts dense correspondences between an image and a rendering of the observed object. [5, 145, 197] present techniques for building learning-based appearance representation spaces that can generalize to novel objects. These methods however still require to generate thousands of template views for novel objects at test time, which take minutes or hours to generate. [234] uses a direct regression method to estimate the orientation of an object, using a shape encoder or multi-view encoder to encode a novel object's appearance and coordinate system.

**Pose estimation without mesh.** In [60, 121, 154, 196, 232], a CAD model of the novel object is not known but a set of reference views with known viewpoints are captured and used to define the object pose. Acquiring such viewpoints is costly in terms of time and requires a preprocessing stage to annotate images with their pose.

In chapter 5, we present MegaPose, an approach for estimating the 6D pose of novel objects, i.e. objects unseen during training. The method only requires knowledge of the CAD model of the object at test time.

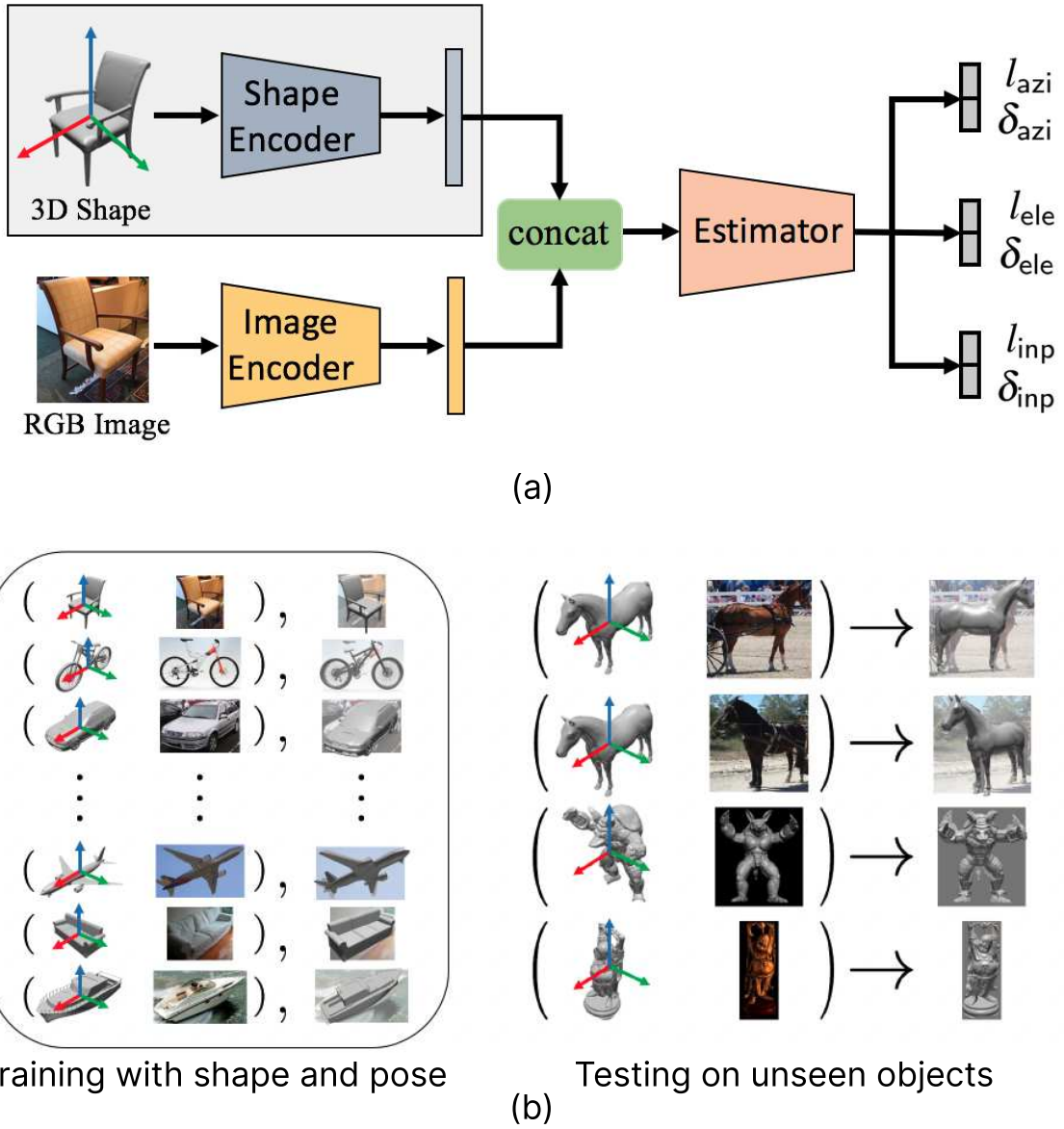



Figure 2-10: **Pose estimation of novel objects using mesh models [234].** A network (a) is used to predict the orientation of a novel object in an RGB image. The 3D shape of the object is provided as input to the network using the shape encoder. (b) illustrates training and testing. The network is trained on a large collection of object shapes (left), e.g. cars, sofas and planes. And can be tested on novel objects such as horses or figurines (right). Figure reproduced from [234].

### 2.1.5 Object tracking

In robotics and augmented reality applications, pose estimation must often be carried out in real-time on a video stream with moving objects and/or cameras. When multiple images are available, it is possible to leverage temporal continuity. We refer to [107] for a survey of methods for tracking pose estimates. These early works mention that a limitation of all tracking approaches is that they may fall into local minima. Nowadays, most works focus on estimating the pose of objects in a single image. Fast and accurate pose estimation in single images allows for tracking-by-detection, e.g. estimating the pose individually in each image. These poses can be temporally smoothed using temporal filters [32, 208]. An advantage of the methods based on the iterative render-and-compare strategy is they can be used for both (i) single-image pose estimation and (ii) tracking by using the pose estimate in the previous frame as an initial guess. Such strategy has been demonstrated to be effective, as shown in [112] and chapters 5 and 6 of this thesis.

### 2.1.6 Object detection and identification

Recovering the state of a scene requires (i) detecting the objects that are in the scene and (ii) estimating the pose of each individual object. The detection and pose estimation problem were initially treated jointly under the term of 3D object recognition [123]. In early non-learning-based works, the search for correspondences between observed features in images and features on the object model was carried out for all keypoints and types of objects in the database of known objects. Later, fast and accurate 2D object detectors based on CNNs [58, 116, 171] were used to detect the objects. The pose estimation problem can be treated jointly by incorporating pose estimation branches in a detection network [87, 230]. The state-of-the-art methods [71, 198] treat the detection and pose estimation problems separately. For each detection, a crop of the entire image around each detected object is used to estimate the object pose with an independent pose estimation method.

## 2.2 Multi-image pose estimation

Multiple images - captured from different distinct cameras or a single moving camera - can be used to reconstruct scenes with improved detection and pose estimation accuracy over single-image methods presented in section 2.1. In this section, we review methods that use multiple input images for recovering the 6D poses of the observed objects as well as estimating all the camera viewpoints. We first present methods that focus on detecting objects and recovering their pose using multiple images of a scene. In a second paragraph, we discuss SLAM methods where the focus is on constructing a map of the environment while localizing a camera in this map in real-time.

**Object pose estimation.** When multiple images of a scene are available, the different cameras (e.g. perspective) can be modeled as a single camera using a generalized multi-camera model [51, 166] which explicitly models geometric constraints between the images. [22, 23] use the constraints between the 2D locations of local features detected in different

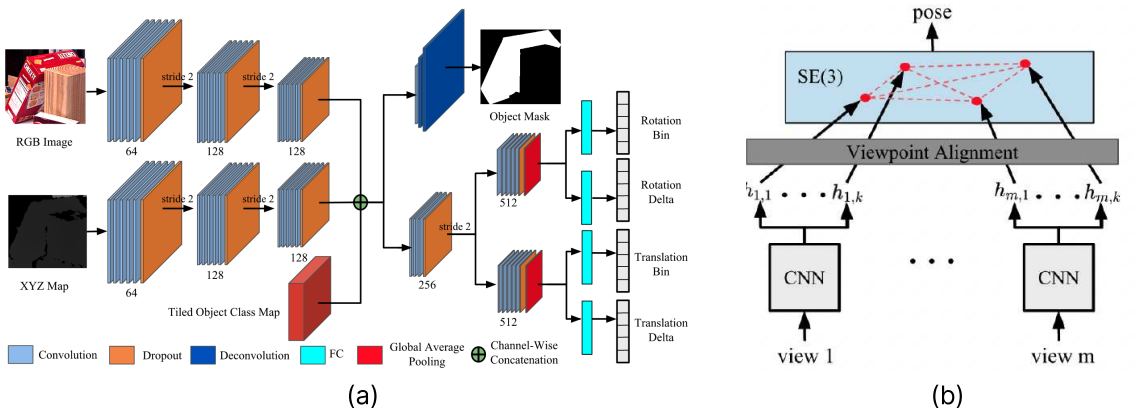



Figure 2-11: **Multi-view object pose estimation [109].** A one-shot pose estimation network is used to predict the pose of an object in a single image (a). The multi-view strategy is illustrated in (b). The same network is applied to multiple views of the same object. The poses are then registered in the same coordinate system using viewpoint alignment based on known relative camera poses. The hypotheses from each view are finally aggregated to predict a single object pose estimate. Figure reproduced from [109].

images in order to estimate the object poses in a common coordinate frame. In [109], illustrated in figure 2-11, a CNN is used to predict an object pose hypothesis in each image. The multiple hypotheses are then registered and aggregated in a common coordinate frame using known relative camera poses, obtained for example through a calibration step. In [7], the relative camera poses are not assumed to be known, and 2D human keypoints detected in different images are used to jointly optimize the pose of a single human and the position of the observing cameras.

**Simultaneous localization and mapping (SLAM).** SLAM systems reconstruct a representation of a scene (the map) while also estimating the 6D pose of a moving camera in real-time. In dense SLAM approaches such as ORB-SLAM [141, 142], no prior knowledge of the structure of the scene is assumed, and a dense reconstruction approach is used (similar to Structure-from-Motion (SfM), e.g. [180, 181]) to recover a dense 3D point cloud of the scene. The semantic and geometric properties associated with known rigid objects are leveraged in [8]. Using object-level information leads to increased camera pose estimation accuracy over SfM algorithms, and using multiple images improves the detection and pose estimation accuracy of the objects in the scene. The method however still relies on local image feature detection and matching. In [163], the representation of the scene only contains objects, but the objects are detected in a 3D point cloud recovered with feature-based ORB-SLAM [141]. In [178], illustrated in figure 2-12, feature points are completely omitted, and an RGB-D method for 3D object recognition and tracking is used to provide camera-object constraints. A database of objects with known 3D models is used for recognition in [178]. The requirement for known 3D object models can be removed. The objects can be reconstructed on the fly by using a 3D reconstruction approach like KinectFusion [144] on an RGB-D point cloud segmented with a CNN, as done in [133]. A deep network can also be used for 3D recognition, as done in [238] where 3D cuboids are fit to observed objects.

Approaches that reconstruct 3D object models [133, 238] can fail to identify individual objects in object stacks, a problem that can be solved by considering physical constraints between objects and using a latent space learned using simulated object stacks to segment new objects as done in [102]. Methods for dense and object-level SLAM can also be combined. A planar surface can be extracted from a 3D point cloud to add a planar constraint between objects and environment[47, 48]. Dense volumetric information can be used to provide context and physical constraints for 6D object pose estimation as done in [216], an approach which has been used for robotic manipulation in challenging scenes [214, 215]. In all SLAM methods, temporal continuity between the images is assumed and the methods are not suitable for images with large baselines. In addition, leveraging temporal continuity makes it challenging to deal with scenes that contain both static and dynamic objects [236, 242]. Our multi-image approach introduced in chapter 4 recovers an object-level representation of a scene without using intermediate keypoints or local image features, is suitable for multiple cameras with large baselines, and can be used with dynamic objects as the estimates do not assume any form of temporal continuity.

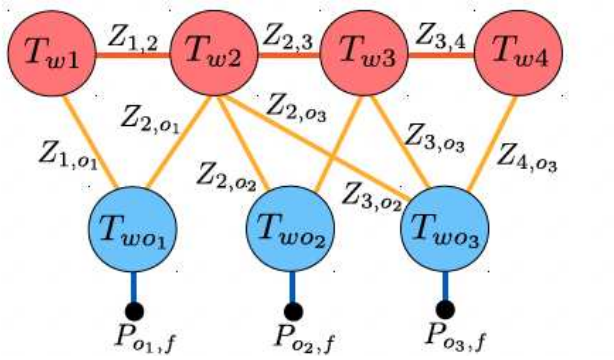
## 2.3 State estimation of scenes with robots

Methods described in previous sections consider scenes containing objects. We now review methods for recovering the state of scenes containing objects and robots. Knowledge of the object poses in a robot's coordinate frame can be used to directly plan robot movements to grasp this object. In section 2.3.1, we present methods that seek to estimate the pose of objects with respect to the robot in order to plan manipulation movements. We show that these methods can be direct or indirect. In the case of indirect methods, the poses of objects and robots are estimated separately in a common coordinate frame - typically the camera coordinate frame - then registered in a single coordinate frame. In this case, the poses of the objects with respect to the camera can be recovered using the methods reviewed in sections 2.1 and 2.2. In section 2.3.2, we review methods for estimating the state of the robot composed of (i) the robot-to-camera pose and (ii) the values of robot's joints.

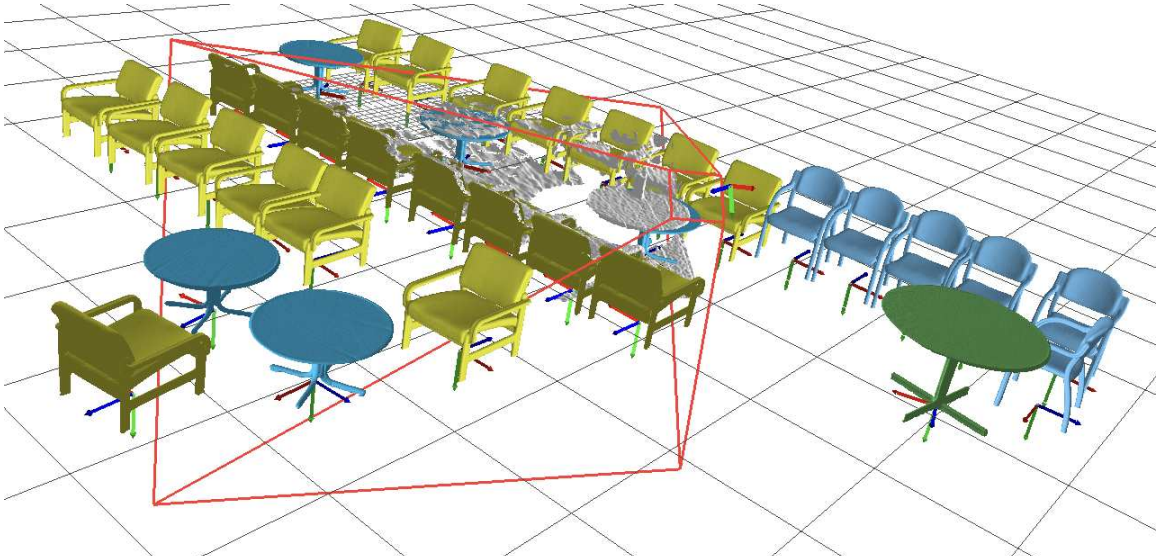
### 2.3.1 Robot-to-object pose estimation

**Direct approaches.** [122] presents a method to directly predict the 2D position and orientation of a single object in the coordinate frame of a known robot. The workspace is discretized in 2D bins and a CNN-based classification network is used to predict a coarse estimate of the localization of the object with respect to the visible robot that serves as the reference for the estimation. A second refinement network uses a finer grid to precisely estimate the position of the object with respect to the gripper when the gripper is close to the object. The prediction from multiple views can be directly combined without a requirement for camera calibration because all predictions are done in the robot or gripper coordinate frame. All networks are trained using thousands of synthetic data displaying the known object and robot. Training data is generated using domain randomization [81, 122, 177, 203] to ensure good sim2real transfer. The main limitation of this work is that it is limited to predicting the 2D location of a single known object with respect to the robot.

(a)




(b)



(c)

Figure 2-12: **Simultaneous localization and mapping of a scene at the level of objects [178].** A single hand-held camera is moved in a scene (a). The pose of visible objects is detected and tracked in each image. Each pose represents a camera-object constraint. (b) shows the scene graph representation. Blue nodes are cameras, red nodes are objects and edges are camera-object constraints. A graph optimization problem is used to recover a unique object-level representation of the scene. The recovered scene is illustrated in (c). Figure reproduced from [178].

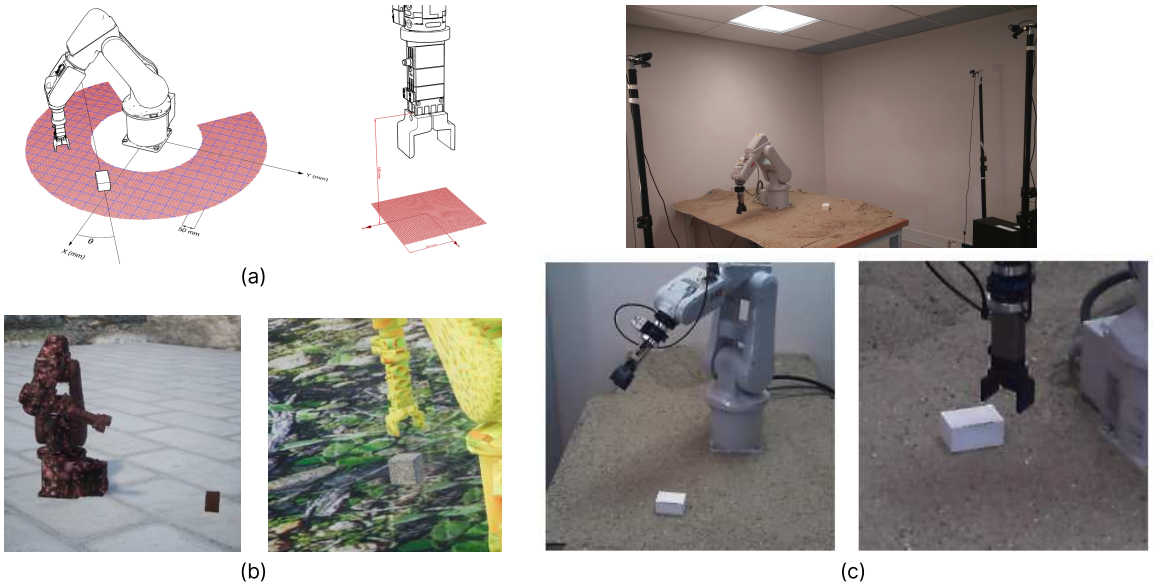



Figure 2-13: **Direct relative object-to-robot localization[122].** The robot (left) and gripper (right) workspaces are discretized using 2D bins. A CNN is trained to predict the location of a block in robot or gripper coordinate systems directly. The CNN is trained on synthetic data generated with domain randomization (b). (c) illustrates a robotic setup where the system is used to precisely estimate the robot-to-object position and grasp the object. Figure reproduced from [122].

In chapter 3, we present a visual system able to directly predict the positions of the 2D centroids of multiple unknown objects in the coordinate system of a robot.

**Indirect approaches.** A method that can estimate the pose of multiple objects with respect to the robot is proposed in [208]. The method separately (i) detects and estimates the pose of the objects using the DOPE [207] detection and pose estimation method, and (ii) detects the pose of a robot with a known internal state using a method for estimating the 6D pose of a robot [106]. Each pose estimate is predicted in the camera coordinate system. The object poses are then expressed with respect to the robot using simple coordinate system transformations. Temporal filters are used to predict smooth pose trajectories by leveraging temporal continuity. The main advantage of this approach is its modularity: the estimation of the object and robot pose is done separately. The main drawback is that errors from the different predictors accumulate when the predictions are expressed in the robot coordinate frame.

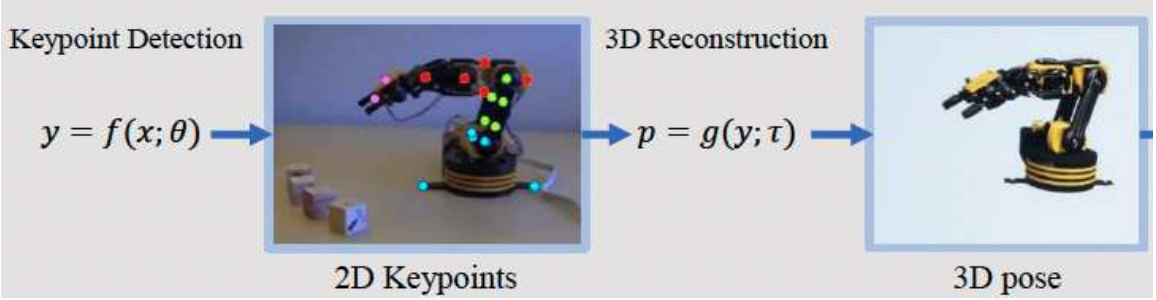
### 2.3.2 Estimating the 6D pose and joint angles of a robot

**Camera-to-robot 6D pose estimation.** Hand-eye calibration [210] is the problem of recovering the relative pose between a robot and a camera. In early works [74, 210], the camera is mounted on the robot gripper. The robot takes multiple pictures of a calibration target, an object with known dimensions easy to detect in the images. For example, edge

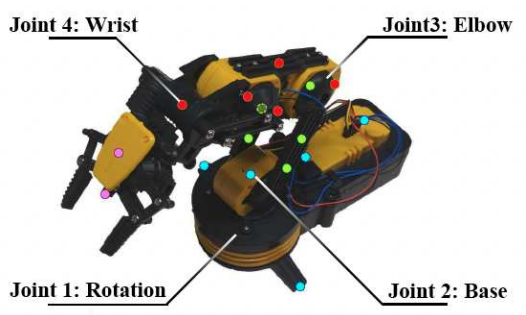

Figure 2-14: **Indirect object-to-robot 6D pose estimation [208].** (a) Two distinct neural networks are used to predict the location of robot keypoints (in red) and object keypoints (corners of the object bounding boxes). (b) Keypoints are used to recover camera-to-robot and camera-to-object poses. Robot-to-object poses are recovered using simple coordinate system transformation and used to manipulate the objects using the robot. Figure reproduced from [208].

detections or fiducial markers presented in section 2.1.1 are used to detect feature points on the calibration target. The robot gripper moves and several cameras are captured. The relative camera-to-robot pose is then obtained by solving an optimization problem [79, 153, 237]. In [61], a SfM technique is used to estimate the camera poses to remove the requirement for using a calibration target. Later, methods are developed for hand-eye calibration of external cameras. When the joint values of the robots are known, recovering the pose of the robot with respect to the camera can be done with the same techniques used for pose estimation of rigid objects. In each image, the robot in a given configuration can be considered as a rigid object. 2D keypoints are detected on the robot using CNNs in [101, 106], and the camera-to-robot pose is recovered using PnP. These methods however assume knowledge of the robot's internal state, i.e. the values of the joint angles.

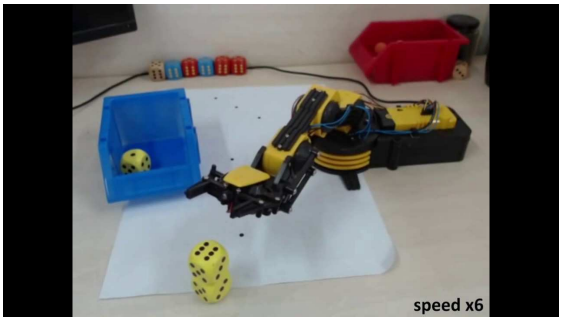
**Articulated objects.** Several works consider reconstructing the entire structure of articulated objects, including the definition of the kinematic structure: the type of joints and their position with respect to the object's parts. In [56, 85, 86, 131], the kinematic structure is discovered through active manipulation of the object parts with a robot. The motion of feature points on each object's parts used to recover the definition of the joints. The same problem is later addressed using learning-based methods [1, 111, 239, 245]. These works are related to this thesis because a robot is an instance of articulated object. In this thesis, we assume the kinematic structure of the robot is known, and only the joint angle and camera-to-robot pose are unknown. [34, 137, 157] make similar assumptions for articulated objects and aim to recover the same unknown parameters as what is done in this thesis. However, all these works consider articulated objects with simple kinematic chains with few degrees of freedom, e.g. laptops or drawers.




(a)




(b)




(c)

Figure 2-15: **Robot pose and joint angle estimation [247].** The method is illustrated in (a). A CNN predicts keypoints on several parts of the known robot shown in (b). The robot pose and joint angles are then recovered by solving a non-linear non-convex optimization problem. The pose and joint angle estimates are used to stack dice (c) using closed-loop control with visual feedback. Figure reproduced from [247].

**Visual robot state estimation.** The pose and joint angles of a robot with four degrees of freedom are estimated using a single RGB image in [247], illustrated in figure 2-15. 2D keypoints are recognized in the images, and the 6D pose and joint angles are recovered by solving a nonlinear non-convex optimization problem. In chapter 6 we present an approach that directly estimates the pose and joint angles of a robot without using an intermediate keypoints representation.

# Chapter 3

# Visually guided multi-object rearrangement planning

In this chapter, we address the problem of visually guided rearrangement planning with many movable objects, i.e., finding a sequence of actions to move a set of objects from an initial arrangement to a desired one, while relying on visual inputs coming from an RGB camera. To do so, we introduce a complete pipeline relying on two key contributions. First, we introduce an efficient and scalable rearrangement planning method, based on a Monte-Carlo Tree Search exploration strategy. We demonstrate that because of its good trade-off between exploration and exploitation our method (i) scales well with the number of objects while (ii) finding solutions which require a smaller number of moves compared to the other state-of-the-art approaches. Note that on the contrary to many approaches, we do not require any buffer space to be available. Second, to precisely localize movable objects in the scene, we develop an integrated approach for robust multi-object workspace state estimation from a single uncalibrated RGB camera using a deep neural network trained only with synthetic data. We validate our multi-object visually guided manipulation pipeline with several experiments on a real UR-5 robotic arm by solving various rearrangement planning instances, requiring only 60 ms to compute the plan to rearrange 25 objects. In addition, we show that our system is insensitive to camera movements and can successfully recover from external perturbations. Video, source code and pre-trained models are available on the project page [168].

## 3.1 Introduction

Using a robot to clean up a room is a dream shared far beyond the robotics community. This would require a robot to both localize and re-arrange many objects. Other industrial scenarios, such as sorting and packing objects on a production line or car assembly tasks, share similar objectives and properties. This chapter presents an integrated approach that makes a step towards the efficiency, scalability and robustness required for solving such rearrangement planning tasks. Figure 3-1 shows an example of the problem we consider, where objects have to be moved from an initial position to a target one. The current and target states are described only by a single image taken from an uncalibrated RGB camera.

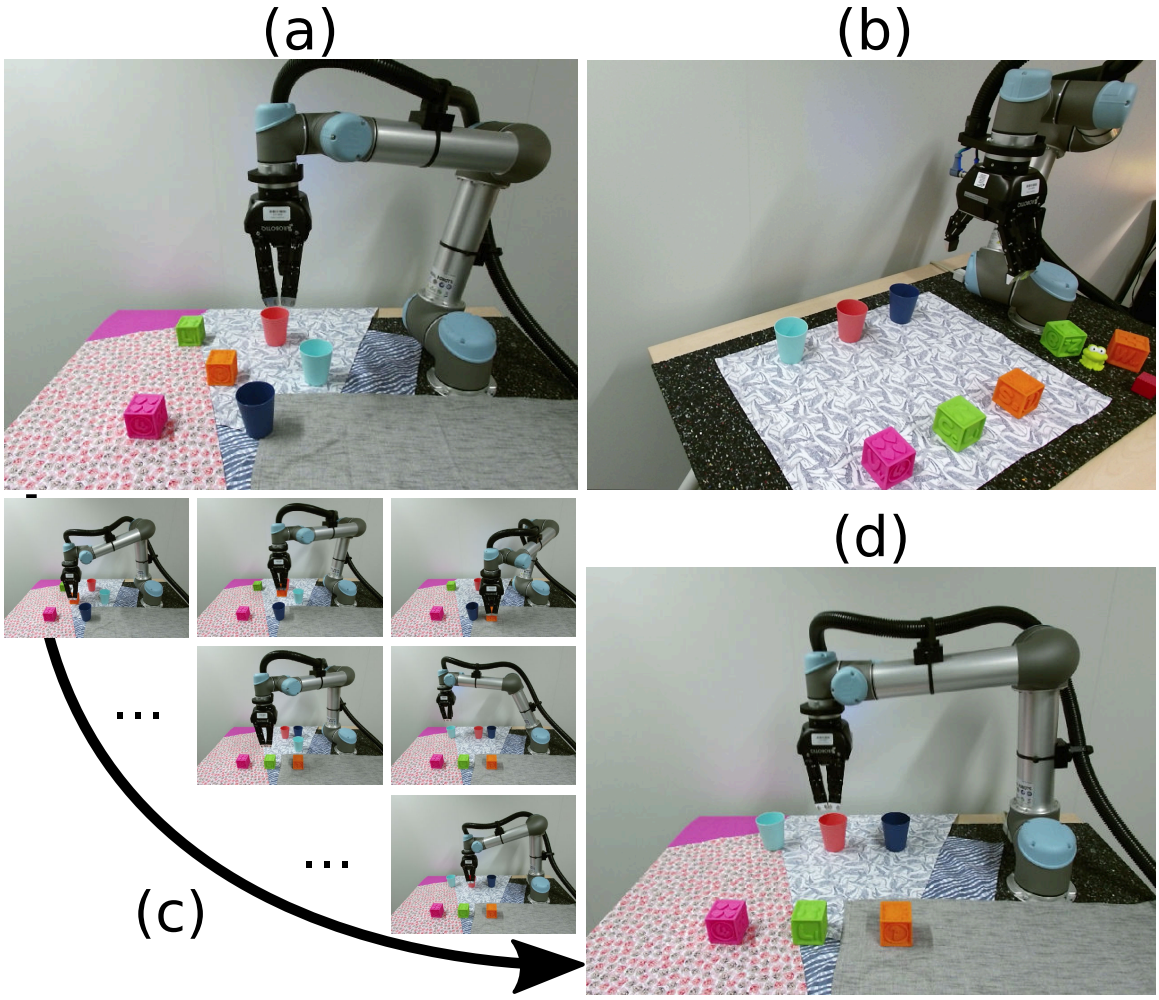



Figure 3-1: **Visually guided rearrangement planning.** Given a source (a) and target (b) RGB images depicting a robot and multiple movable objects, our approach estimates the positions of objects in the scene without the need for explicit camera calibration and efficiently finds a sequence of robot actions (c) to re-arrange the scene into the target scene. Final object configuration after re-arrangement by the robot is shown in (d).

Rearrangement planning has a long history in robotics [3, 83, 103, 104, 188, 194] and remains an active research topic [29, 52, 57, 93] in the motion planing community. The goal is to find a sequence of *transit* and *transfer* motions [3, 103] to move a set of objects from an initial arrangement to a target arrangement, while avoiding collisions with the environment. This leads to a complex sequential decision process, whose complexity depends on the number of objects to move, on the free-space available around the objects, and the robot kinematics.

Several solutions have been proposed in the literature which can be roughly classified into two groups. Methods in the first group [29, 44, 52, 83, 93, 193] rely on the *task* and *motion* planning hierarchy where a high-level task planner is combined with a local motion planner [104]. Methods in the second group [94, 138, 188, 194] aim at solving a single unified formulation of the problem by using classic sample-based algorithms such

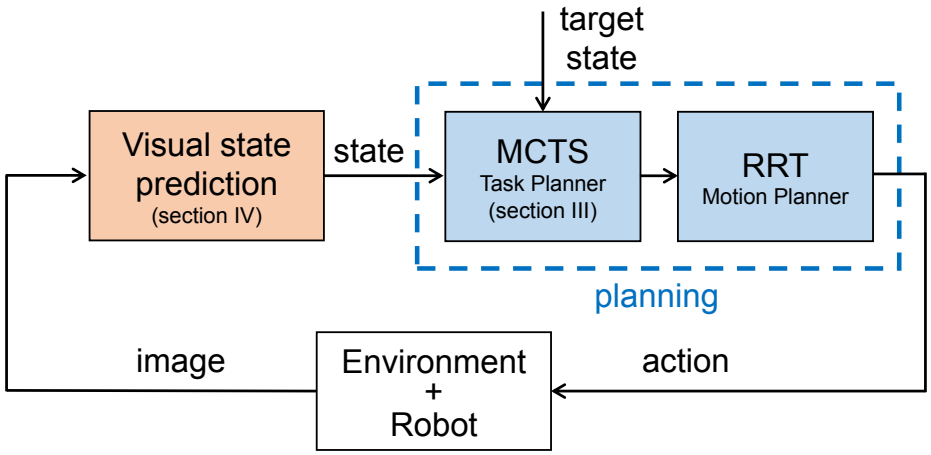



Figure 3-2: **Approach overview.** Given an image of the scene, the visual state prediction module outputs a list of objects and their coordinates in the robot coordinate frame. Together with a target state, these serve as inputs to the task and motion planning module which combines Monte-Carlo Tree Search with a standard robot motion planning algorithm.

as Probabilistic RoadMap (PRM) or Rapidly-Exploring Random Tree (RRT) [104] or use advanced optimization strategies to solve a unique optimization instance [205].

While methods from both groups have been shown to work well in practice with few objects, existing methods do not scale to a large set of objects, because the number of possible action sequences increases exponentially with the number of objects to move. Some recent methods [52, 93, 94] scale better with the number of objects but these methods either only focus on feasibility, producing solutions with sub-optimal number of grasps [93], or are limited to specific constrained scenarios, for example, with explicitly available buffer space [52] or strict constraints of monotony (i.e. an object can be moved only once during the plan).

In this work we describe an efficient and generic approach for rearrangement planning that overcomes these limitations: (i) it scales well with the number of objects, by taking only 60 ms to plan complex rearrangement scenarios for multiple objects, and (ii) it can be applied to the most challenging table-top re-arrangement scenarios, not requiring explicit buffer space. Our approach is based on Monte-Carlo Tree Search [140], which allows us to lower the combinatorial complexity of the decision process and to find an optimistic number of steps to solve the problem, by making a compromise between exploration (using random sampling) and exploitation (biasing the search towards the promising already sampled action sequences to cut off some of the search directions).

To demonstrate the benefits of our planning approach in real scenarios we also introduced a multi-object calibration-free deep neural network architecture for object position estimation. It is trained entirely from synthetic images and, compared to other currently available methods [45, 112, 207, 230], does not use markers or require known CAD models of the specific observed object instances. To the best of our knowledge, the approach we present is the first one able to locate multiple objects in such difficult and generic conditions. This is of high practical interest, since it allows to perform the full task using only

a single non-calibrated, or even hand-held, RGB camera looking at the robot.

Our complete pipeline for visually guided rearrangement planning, illustrated in figure 3-2, is composed of three main stages. The goal of the first stage, visual state prediction (section 3.3), is to estimate the positions of multiple objects relative to a robot given a single non-calibrated image of the scene. The second stage (section 3.4), is our MCTS-based task planner: at each planning step, this module chooses which object to move and computes its new desired position in the workspace. The last stage is a standard RRT-based local motion planner which plans robot movements given the high-level plan computed by the MCTS planner.

## 3.2 Related work

We build our framework on results in robotics, search algorithms and computer vision, which we review below.

**Rearrangement planning** is NP-hard [225]. As a result, standard hierarchical [29, 44, 83, 93] and randomized methods [138, 188, 194] for solving general manipulation planning problems do not scale well with the number of objects. The most efficient and scalable high-level planners only address specific constrained set-ups leveraging the structure of the rearrangement problem [52, 93, 94]. In addition, they often focus on feasibility but do not attempt to find high-quality solutions with a low number of object moves [93, 94]. For instance, some methods [94] only consider the monotone problem instances, where each object can be grasped at most once. In contrast, our method finds high-quality plans but also addresses the more general cases of non-monotone re-arrangement problems, which are known to be significantly harder [194]. Others works have looked at finding optimal plans [52] but address only constrained set-ups that have available buffer space (i.e. space that does not overlap with the union of the initial and target configurations), noting that solving the general case without available buffer space is significantly harder [52]. In this work, we address this more general case and describe an approach that efficiently finds high-quality re-arrangement solutions without requiring any available buffer space. In addition and unlike previous works [52, 93, 94], we also propose a complete system able to operate from real images in closed loop.

**Search algorithms.** The problem of rearrangement planning can be posed as a tree search. Blind tree search algorithms such as Breadth-First search (BFS) [176] can be used to iteratively expand nodes of a tree until a goal node is found, but these methods do not exploit information about the problem (e.g. a cost or reward) to select which nodes to expand first, and typically scale exponentially with the tree depth. Algorithms such as greedy BFS [176] allow to exploit a reward function to drive the exploration of the tree directly towards nodes with high reward, but might get stuck in a local mimima. Other algorithms such as $A^\star$ [53] can better estimate the promising branches using additional hand-crafted heuristic evaluation function. We choose Monte-Carlo Tree Search over others, because it only relies on a reward and iteratively learns a heuristic (the value function) which allows to efficiently balance between exploration and exploitation. It has been used in related areas to solve planning and routing for ground transportation [159] and to guide the tree-search in cooperative manipulation [206]. MCTS is also at the core of AlphaGo, the first system

able to achieve human performance in the game of Go [186], where it was combined with neural networks to speed-up the search. These works directly address problems whose action space is discrete by nature. In contrast, the space of possible object arrangements is infinite. We propose a novel discrete action parameterization of the rearrangement planning problem which allows us to efficiently apply MCTS.

**Vision-based object localization**. In robotics, fiducial markers are commonly used for detecting the objects and predicting their pose relative to the camera [45] but their use limits the type of environments the robot can operate in. This constraint can be removed by using a trainable object detector architecture [73, 112, 207, 228, 230, 241]. However, these methods often require gathering training data for the target objects at hand, which is often time consuming and requires the knowledge of the object (e.g. in the form its 3D model) beforehand. In addition, these methods estimate the pose of the objects in the frame of the camera and using these predictions for robotic manipulation requires calibration of the camera system with respect to the robot. The calibration procedure [61, 74] is time-consuming and must be redone each time the camera is moved. More recently, [122] proposed to directly predict the position of a single object in the robot coordinate frame by training a deep network on hundreds of thousands of synthetically generated images using domain randomization [81, 122, 177, 203]. We build on the work [122] and extend it for predicting the 2D positions of *multiple objects with unknown dimensions* relative to the robot.

## 3.3 Visual scene state prediction with multiple objects

In this section, we detail the visual state prediction stage. Our visual system takes as input a single photograph of a scene taken from an uncalibrated camera and predicts a workspace state that can then be used for rearrangement-planning. More precisely it outputs the 2D positions of a variable number of objects expressed in the coordinate system of the robot. This problem is difficult because the scene can contain a variable number of objects, placed on different backgrounds, in variable illumination conditions, and observed from different viewpoints, as illustrated in figure 3-1. In contrast to [203], we do not assume that the different types of objects are known at training time. In contrast to state-of-the-art pose estimation techniques in RGB images [45, 112], we do not use markers and do not assume the CAD models of the objects are known at test time.

To address these challenges, we design a visual recognition system that does not require explicit camera calibration and outputs accurate 2D positions. Moreover, even if we deploy our system on a real robot, we show that it can be trained entirely from synthetic data using domain randomization [203], avoiding the need for real training images. Also, our system does not require any tedious camera calibration because it is trained to predict positions of objects directly in the coordinate frame of the robot, effectively using the robot itself, which is visible in the image, as an (known) implicit calibration object. This feature is important for applications in unstructured environments such as construction sites containing multiple unknown objects and moving cameras for instance. Our recognition system is summarized in figure 3-3 and in the rest of this section, we present in more details the different components.

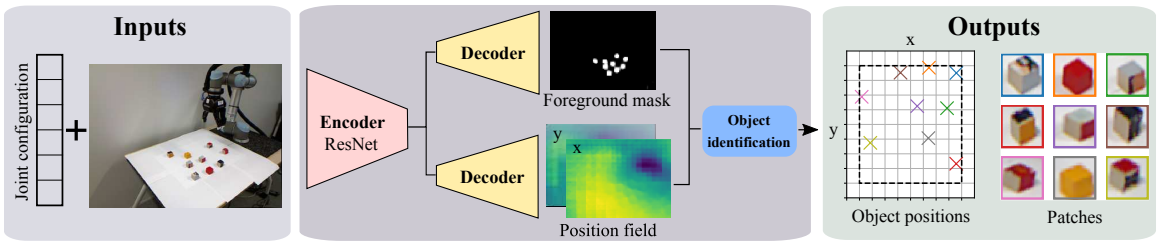



Figure 3-3: **The visual recognition system.** The input is an image of the scene captured by an uncalibrated camera together with the joint configuration vector of the depicted robot. Given this input a convolutional neural network (CNN) predicts the foreground-background object segmentation mask and a dense position field that maps each pixel of the (downsampled) input image to a 2D coordinate in a frame centered on the robot. The estimated masks and position fields are then combined by the object identification module to identify individual object instances. The output is a set of image patches associated with the 2D position of the object in the workspace.

Figure 3-4: **Examples of synthetic training images.** We generate images displaying the robot and a variable number of objects in its workspace. The images are taken by cameras with different viewpoints and depicting large scene appearance variations.

### 3.3.1 Position prediction network

In this section, we give details of the network for predicting a dense 2D position field and an object segmentation mask. The 2D position field maps each input pixel to a 2D coordinate frame of the robot acting as implicit calibration.

**Architecture.** Our architecture is based on ResNet-34 [59]. We remove the average pooling and fully connected layers and replace them by two independent decoders. Both decoders use the same architecture: four transposed convolution layers with batch normalization and leaky ReLU activations in all but the last layer. The resolution of the input image is $320 \times 240$ and the spatial resolution of the output of each head is $85 \times 69$. We add the 6D joint configuration vector of the robot as input to the network by copying it into a tensor of size $320 \times 240 \times 6$, and simply concatenating it with the three channels of the input image. The two heads predict an object mask and a 2D position field which are visualized in figure 3-3. In addition, we found that predicting depth and semantic segmentation during training increased the localization accuracy at test time. These modalities are predicted using two additionnal decoders with the same architecture.

**Synthetic training data.** Following [81, 122, 177, 203], we use domain randomization

for training our network without requiring any real data. We generate two million images displaying the robot and a variable number of objects with various shapes (cubes, cylinders and triangles) in its workspace. In each scene, we randomly sample from 1 up to 12 objects, with various dimensions between 2.5 and 8 cm. Examples of training images are shown in figure 3-4. Randomization parameters include the textures of the robot and objects, the position of the gripper, the position, orientation and field of view of the camera, the positions and intensities of the light sources and their diffuse/ambient/specular coefficients.

**Training procedure.** We train our network by minimizing the following loss: $\mathcal{L} = \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{segm}} + \mathcal{L}_{\text{depth}}$, where the individual terms are explained next. For the position field loss we use $\mathcal{L}_{\text{pos}} = \sum_{i,j} \delta_{i,j} \left[ (\hat{x}_{i,j} - x_{i,j})^2 + (\hat{y}_{i,j} - y_{i,j})^2 \right]$ where $(i, j)$ are the pixel coordinates in the output; $\delta_{i,j}$ is the binary object mask; $x_{i,j}$, $y_{i,j}$ are the ground truth 2D coordinates of the center of the object (that appears at pixel $(i, j)$) and $\hat{x}_{i,j}, \hat{y}_{i,j}$ are the components of the predicted position field. For $\mathcal{L}_{\text{mask}}$, $\mathcal{L}_{\text{segm}}$ and $\mathcal{L}_{\text{depth}}$ we respectively use the following standard losses: binary cross entropy loss, cross entropy loss and mean squared error. These losses are computed pixel-wise. Note that depth is not used to estimate object positions at test time. $L_{\text{segm}}$ and $L_{\text{depth}}$ are auxiliary losses used only for training, similar to [81]. We use the Adam optimizer [90] and train the network for 20 epochs, starting with a learning rate of $10^{-3}$ and decreasing it to $10^{-4}$ after 10 epochs.

### 3.3.2 Identifying individual objects

The model described above predicts a dense 2D position field and an object mask but does not distinguish individual objects in the scene. Hence, we use the following procedure to group pixels belonging to each individual object. Applying a threshold to the predicted mask yields a binary object segmentation. The corresponding pixels of the 2D position field provide a point set in the robot coordinate frame. We use the mean-shift algorithm [25] to cluster the 2D points corresponding to the different objects and obtain an estimate of the position of each object. The resulting clusters then identify pixels belonging to each individual object providing instance segmentation of the input image. We use the resulting instance segmentation to extract patches that describe the appearance of each object in the scene.

### 3.3.3 Source-Target matching

To perform rearrangement, we need to associate each object in the current image to an object in the target configuration. To do so, we use the extracted image patches. We designed a simple procedure to obtain matches robust to the exact position of the object within the patches, their background and some amount of viewpoint variations. We rescale patches to 64×64 pixels and extract conv3 features of an AlexNet network trained for ImageNet classification. We finally run the Hungarian algorithm to find the one-to-one matching between source and target patches maximizing the sum of cosine similarities between the extracted features. We have tried using features from different layers, or from the more recent network ResNet. We found the conv3 features of AlexNet to be the most suited for the task, based on a qualitative evaluation of matches in images coming from

the dataset presented in section 3.5.2. Note that our patch matching strategy assumes all objects are visible in the source and target images.

## 3.4 Rearrangement Planning with Monte-Carlo Tree Search (MCTS)

Given the current and target arrangements, the MCTS task planner has to compute a sequence of pick-and-place actions that transform the current arrangement into the target one. In this section, we first review Monte-Carlo Tree Search and then explain how we adapt it for rearrangement planning.

### 3.4.1 Review of Monte-Carlo Tree Search

The MCTS decision process is represented by a tree, where each node is associated to a state $s$, and each edge represents a discrete action $a = \{1, ..., N\}$. For each node in the tree, a reward function $r(s)$ gives a scalar representing the level of accomplishment of the task to solve. Each node stores the number of times it has been visited $n(s)$ and a cumulative reward $w(s)$. The goal of MCTS is to find the most optimistic path, i.e. the path that maximizes the expected reward, starting from the root node and leading to a leaf node solving the task. MCTS is an iterative algorithm where each iteration is composed of three stages.

During the *selection* stage, an action is selected using the Upper Confidence Bound (UCB) formula:

$$U(s,a) = Q(s,a) + c\sqrt{\frac{2\log n(s)}{n(f(s,a))}}, \tag{3.1}$$

where $f(s,a)$ is the child node of $s$ corresponding to the edge $(s,a)$ and $Q(s,a) = \frac{w(f(s,a))}{n(f(s,a))}$ is the expected value at state $s$ when choosing action $a$. The parameter $c$ controls the trade-off between exploiting states with high expected value (first term in (3.1)) and exploring states with low visit count (second term in (3.1)). We found this trade-off is crucial for finding good solutions in a limited number of iterations as we show in section 3.5.1. The optimistic action selected $a_{\text{selected}}$ is the action that maximizes $U(s,a)$ given by (3.1). Starting from the root node, this stage is repeated until an expandable node (a node that has unvisited children) is visited. Then, a random unvisited child node $s'$ is added to the tree in the *expansion stage*. The reward signal $r(s')$ is then back-propagated towards the root node in the *back-propagation stage*, where the cumulative rewards $w$ and visit counts $n$ of all the parent nodes are updated. The search process is run iteratively until the problem is solved or a maximal number of iterations is reached.

### 3.4.2 Monte-Carlo Tree Search task planner

We limit the scope to tabletop rearrangement planning problems with overhand grasps but our solution may be applied to other contexts. We assume that the complete state of any object is given by its 2D position in the workspace, and this information is sufficient to

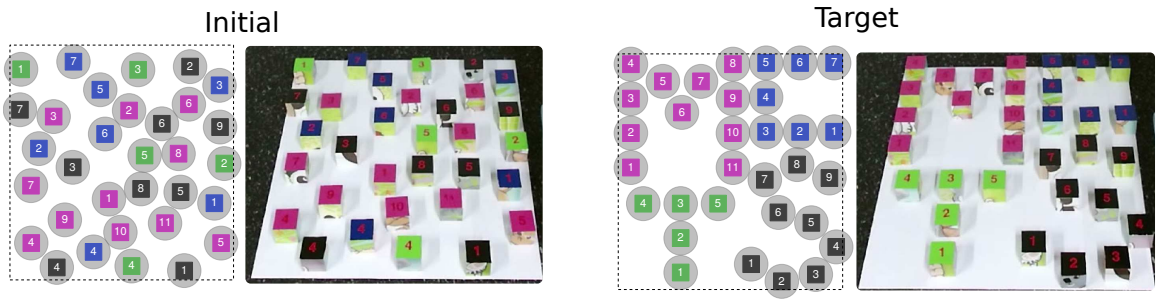



Figure 3-5: Examples of source and target object configurations. The planner has to find a sequence of actions (which object to move and where to displace it inside the workspace), while avoiding collisions with other objects. Workspace limits are shown as dashed lines. Grey circles depict the collision radius for each object. We demonstrate our method solving this complex rearrangement problem in the **video** available on the project page [168]. Here source and target states are known (state estimation network is not used).

grasp it. The movements are constrained by the limited workspace, and actions should not result in collisions between objects. The planner has to compute a sequence of actions which transform the source arrangement into the target arrangement while satisfying these constraints. Each object can be moved multiple times and we do not assume that explicit buffer space is available. An example of source and target configurations is depicted in figure 3-5. We now detail the action parametrization and the reward.

Let $\{C_i\}_{i=1,..,N}$ denote the list of 2D positions that define the current arrangement with $N$ objects, $\{I_i\}_i$ and $\{T_i\}_i$ the initial and target arrangements respectively, which are fixed for a given rearrangement problem. MCTS state corresponds to an arrangement $s = \{C_i\}_i$.

As a reward $r(s)$ we use the number of objects located within a small distance of their target position:

$$r(s) = \sum_{i=1}^{N} R_i \text{ with } R_i = \begin{cases} 1 & \text{if } ||C_i - T_i||_2 \leq \epsilon \\ 0 & \text{otherwise.} \end{cases}, \tag{3.2}$$

where $N$ is the number of objects, $C_i$ is the current location of object $i$, $T_i$ is the target location of object $i$ and $\epsilon$ is a small constant.

We define a discrete action space with $N$ actions where each action corresponds to one pick-and-place motion moving one of the $N$ objects. The action is hence pararametrized by 2D picking and placing positions defined by the function GET_MOTION outlined in detail in Algo. 1. The input to that function is the current state $\{C_i\}_i$, target state $\{T_i\}_i$ and the chosen object $k$ that should be moved. The function proceeds as follows (please see also Algo. 1). First, if possible, the object $k$ is moved directly to it's target $T_k$ (lines 3-4 in Algo. 1), otherwise the obstructing object $j$ which is the closest to $T_k$ (line 7 in Algo. 1) is moved to a position inside the workspace that does not overlap with $T_k$ (lines 8-10 in Algo. 1). The position $P$ where to place $j$ is found using random sampling (line 8). For collision checks, we consider simple object collision models with fixed radiuses as depicted in figure 3-5. If no suitable position is found, no objects are moved (line 11). Note that additional heuristics could be added to the action parametrization to further improve the quality of the resulting solutions and to reduce the number of iterations required. Examples

include (i) avoiding to place $j$ at target positions of other objects and (ii) restricting the search for position $P$ in a neighborhood of $C_j$. The parameters of the pick-and-place motion for a given state and MCTS action are computed only once during the expansion stage and then cached in the tree and recovered once a solution is found.

**Algorithm 1:** Action Parametrization

```
1  function GET_MOTION ({C_i}_i, {T_i}_i, k):
2      /* Check if object k can be moved to T_k */
3      if IS_MOVE_VALID ({C_i}_{i≠k}, C_k, T_k) then
4          return (C_k, T_k)
5      else
6          /* Move obstructing object j to position P*/
7          j = FIND_CLOSEST_OBJECT ({C_i}_i, T_k)
8          found, P = FIND_POSITION ({C_i}_{i≠j} ∪ {T_k})
9          if found then
10             return (C_j, P)
11         return (C_k, C_k)
```

As opposed to games where a complete game must be played before having access to the outcome (win or lose), the reward in our problem is defined in every state. Therefore, we found that using a MCTS simulation stage is not necessary.

The number of MCTS iterations to build the tree is typically a hyper-parameter. In order to have a generic method that works regardless of the number of objects, we adopt the following strategy: we run MCTS until a maximum (large) number of iterations is reached or until a solution is found. We indeed noticed that the first solution provided by MCTS is already sufficiently good compared to the next ones when letting the algorithm run for longer.

The presented approach only considers tabletop rearrangement planning with overhand grasps. The main limitation is that we assume the motion planning algorithm can successfully plan all the pick-and-place motions computed in Algo. 1. This assumption does not hold for more complex environment where some objects are not reachable at any time (e.g. moving objects in a constrained space such as inside a refrigerator). In this case, the function IS_MOVE_VALID can be adapted to check whether the movement can be executed on the robot. Note that we consider simple collision models in FIND_POSITION but meshes of the objects and environment could be used if available.

## 3.5 Experiments

We start by evaluating planning (section 3.5.1) and visual scene state estimation (section 3.5.2) separately, demonstrating that: (i) our MCTS task planner scales well with the number of objects and is efficient enough so that it can be used online (i.e. able to recompute a plan after each movement); (ii) the visual system detects and localizes the objects with an accuracy sufficient for grasping. Finally, in section 3.5.3 we evaluate our full pipeline in challenging setups and demonstrate that it can efficiently perform the task

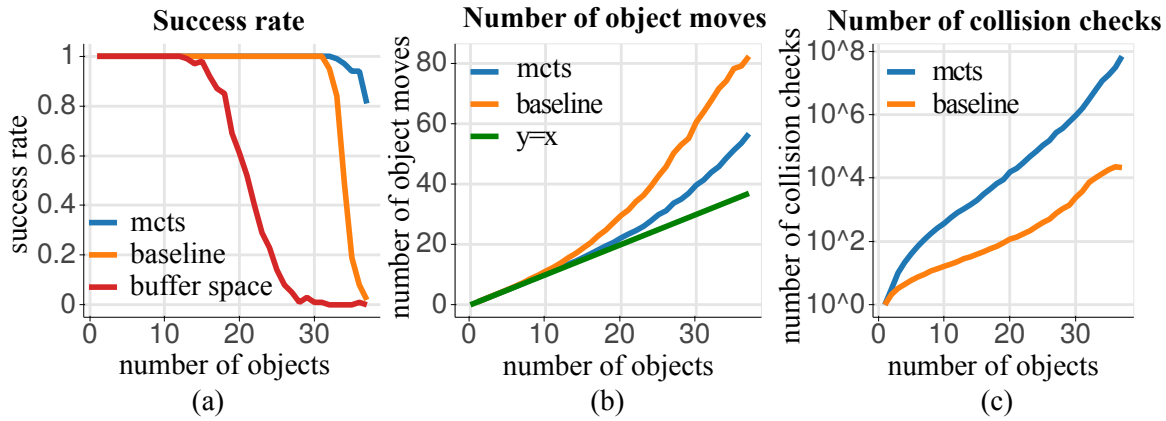



Figure 3-6: Comparison of the proposed MCTS planning approach against the strong baseline heuristic. MCTS is able to solve more complex scenarios (with more objects) in a significantly lower number of steps. MCTS does not require free space to be available or that the problems are monotone.

and can recover from errors and perturbations as also shown in the video available on the project page [168].

### 3.5.1 MCTS planning

**Experimental setup.** To evaluate planning capabilities and efficiency we randomly sample 3700 initial and target configurations for 1 up to 37 objects in the workspace. It is difficult to go beyond 37 objects as it becomes hard to find valid configurations for more due to the workspace constraints.

**Planning performance.** We first want to demonstrate the interest of MCTS-based exploration compared to a simpler solution in term of efficiency and performances. As a baseline, we propose a fast heuristic search method, which simply iterates over all actions once in a random order, trying to complete the rearrangement using the same action space as our MCTS approach, until completion or a time limit is reached. Instead of moving only the closest object that is obstructing a target position $T_k$, we move all the objects that overlap with $T_k$ to their target positions or to a free position inside the workspace that do not overlap with $T_k$. Our MCTS approach is compared with this strong baseline heuristic in figure 3-6. Unless specified otherwise, we use $c = 1$ for MCTS and set the maximum number of MCTS iterations to 100000.

As shown in figure 3-6(a), the baseline is able to solve complex instances but its success rate starts dropping after 33 objects whereas MCTS is able to find plans for 37 objects with 80% success. More importantly, as shown in figure 3-6(b), the number of object movements in the plans found by MCTS is significantly lower. For example, rearranging 30 objects takes only 40 object moves with MCTS compared to 60 with the baseline. This difference corresponds to more than 4 minutes of real operation in our robotics setup. The baseline and MCTS have the same action parametrization but MCTS produces higher quality plans because it is able to take into consideration the future effects of picking-up an object and placing it at a specific location. On a laptop with a single CPU core, MCTS finds plans for

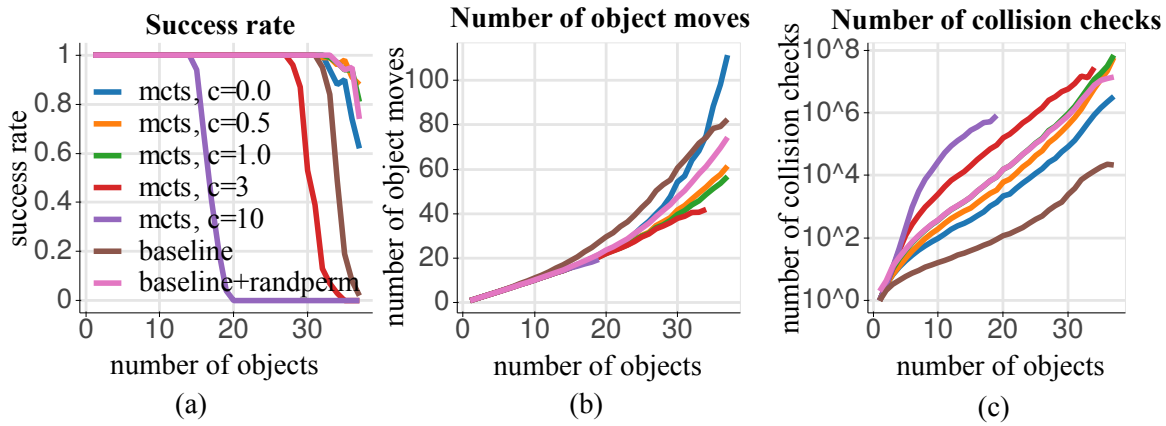



Figure 3-7: Exploration-exploitation tradeoff in MCTS. MCTS performs better than a random baseline heuristic search. Balancing the exploration term of UCB with the parameter $c$ is crucial for finding good solutions while limiting the number of collision checks.

25 objects in 60 ms. This high efficiency allows to replan after each pick-and-place action and perform rearrangement in a closed loop.

**Exploration-exploitation trade-off.** We now want to demonstrate that the benefits of our planning method are due to the quality of the exploration/exploitation trade-off in MCTS. An important metric is the total number of collision checks that the method requires for finding a plan. The collision check (checking whether an object can be placed to a certain location) is indeed one of the most costly operation when planning. figure 3-6(c) shows that MCTS uses more collision checks compared to the baseline because MCTS explores many possible optimistic action sequences while the baseline is only able to find a solution and does not optimize any objective. We propose another method that we refer to as *baseline+randperm* which solves the problem with the baseline augmented with a random search over action sequences: the baseline is run with different random object orders until a number of collision checks similar to MCTS with $c = 1$ is reached and we keep the solution which has the smallest number of object moves. As can be see in figure 3-7, baseline+randperm has a higher success rate and produces higher quality plans with lower number of object moves compared to the baseline (figure 3-7(b)). However, MCTS with $c = 1$, still produces higher quality plans given the same amount of collision checks. The reason is that baseline+randperm only relies on a random exploration of action sequences while MCTS allows to balance the exploration of new actions with the exploitation of promising already sampled partial sequences through the exploration term of UCB (equation 3.1). In figure 3-7, we also study the impact of the exploration parameter $c$. MCTS with no exploration ($c = 0$) finds plans using fewer collision checks compared to $c > 0$ but the plans have high number of object moves. Increasing $c$ leads to higher quality plans while also increasing the number of collision checks. Setting $c$ too high also decreases the success rate (c=3, c=10 in Fig 3-7(a)) because too many nodes are added to the tree and the limit on the number of MCTS iterations is reached before finding a solution.

**Generality of our set-up.** Previous work for finding high-quality solutions to rearrangement problems has been limited to either monotone instances [94] or instances where buffer space is available [52]. The red curve in figure 3-6(a) clearly shows that in our set-up the

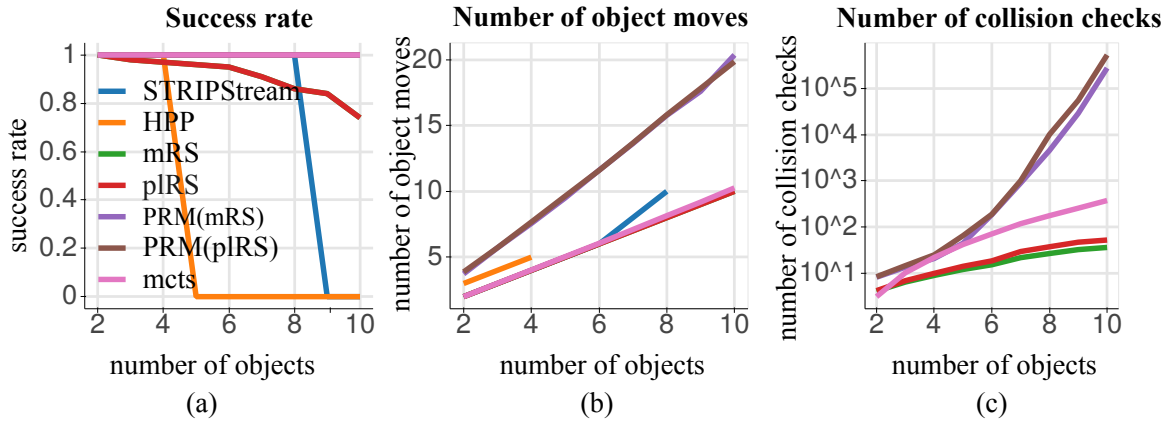
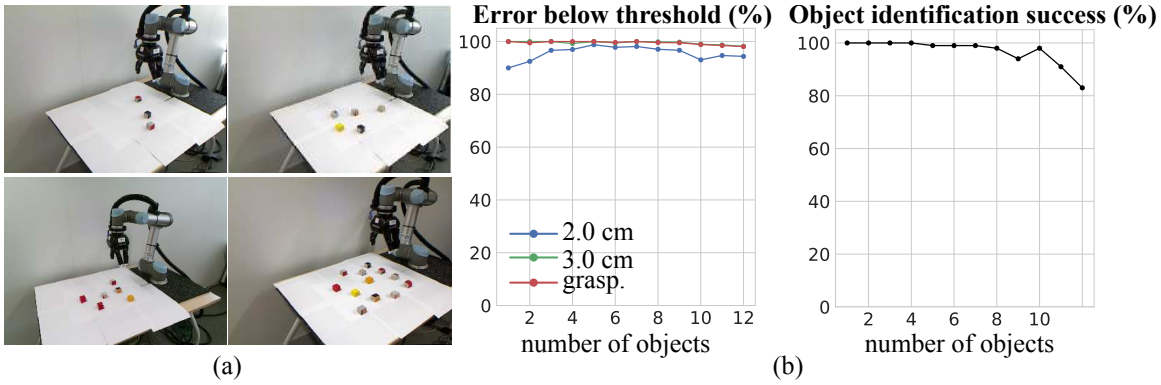



Figure 3-8: Comparison of our MCTS planning approach with several other state-of-the-art methods. MCTS performs better than other methods applied to the rearrangement planning problem. MCTS finds high quality plans (b) using few collisions checks (c) with 100% success rate for up to 10 objects.

number of problems where some buffer space is available for at least one object quickly decreases with the number of objects in the workspace. In other words, the red curve is an upper bound on the success rate of [52], which requires available buffer space. In order to evaluate the performance of our approach on monotone problems, we generated the same number of problems but the target configuration was designed by moving object from the initial configuration one by one, in a random order into free space. This ensures that the instances are monotone and can be solved by moving each object once. Our MCTS-based approach was able to solve 100% of these instances optimally in $N$ steps. Our method can therefore solve the problems considered in [94] while also being able to handle significantly more challenging non-monotonic instances, where objects need to be moved multiple times.

**Comparisons with other methods.** To demonstrate that other planning methods do not scale well when used in a challenging scenario similar to ours, we compared our planner with three other methods of the state of the art: STRIPStream [44], the Humanoid Path-Planner [138], mRS and plRS [93]. Results are presented in figure 3-8 for 900 random rearrangement instances, we limit the experiments to problems with up to 10 objects as evaluating these methods for more complex problems is difficult given a reasonable amount of time (few hours). HPP [138] is the slowest method and could not handle more than 4 objects, taking more than 45 minutes of computation for solving the task with 4 objects. HPP fails to scale because it attempts to solve the combined task and motion planning problem at once using RRT without explicit task/motion planning hierarchy thus computing many unnecessary robot/environment collision checks. The other methods adopt a task/motion planning hierarchy and we compare results for the task planners only. The state-of-the-art task and motion planner for general problems, STRIPSStream [44], is able to solve problems with up to 8 objects in few seconds but do not scale (figure 3-8(a)) when target locations are specified for all objects in the scene as it is the case for rearrangement planning problems. The success rate of specialized rearrangement methods, mRS and plRS, drops when increasing number of objects because these methods cannot handle situations where




(a) (b)

Figure 3-9: (a) Example images from the dataset that we use to evaluate the accuracy of object position estimation as a function of number of objects. (b) Evaluation of our visual system for a variable number of objects. We report the localization accuracy (left) and the percentage of images where all objects are correctly identified (right). Please see the video on the project webpage [168] for examples where our system is used in more challenging visual conditions.

objects are permuted, i.e. placing an object at its target position requires moving another objects first thus requiring longer term planning capability. When used in combination with a PRM, more problems can be addressed but the main drawback is that these methods are slow as they perform a high number of collision checks (figure 3-8(c)). Overall, the main limitation of STRIPSTream, PRM(mRS) and PRM(plRS) comes from the fact that the graph of possible states is sampled randomly whereas MCTS will prioritize the most optimistic branches (exploitation). MCTS and it's tree structure also allows to build the action sequence progressively (moving one object at once) compared to PRM-based approaches that sample entire arrangements and then try to solve them.

### 3.5.2 Visual scene state estimation

**Experimental setup.** To evaluate our approach, we created a dataset of real images with known object configurations. We used 1 to 12 small 3.5 cm cubes, 50 configurations for each number of cubes, and captured images using two cameras for each configuration, leading to a total of 1200 images depicting 600 different configurations. Example images from this evaluation set are shown in figure 3-9(a).

**Single object accuracy.** When a single object is present in the image, the mean-shift algorithm always succeeds and the precision of our object position prediction is $1.1 \pm 0.6$ cm. This is comparable to the results reported in [203] for positioning of a known object with respect to a known table without occlusions and distractors, $1.3 \pm 0.6$ cm, and to results reported in [122] for the coarse alignment of a single object with respect to a robot, $1.7 \pm 3.4$ cm. The strength of our method, however, is that this accuracy remains constant for up to 10 previously unknown objects, a situation that neither [203] nor [122] can deal with.

**Accuracy for multiple objects.** In figure 3-9(b), we report the performance of the object localization and object identification modules as a function of the number of objects. For

localization, we report the percentage of objects localized with errors below 2 cm and 3 cm respectively. For 10 objects, the accuracy is $1.1 \pm 0.6$ cm. The 3 cm accuracy approximately corresponds to the success of grasping, that we evaluate using a simple geometric model of the gripper. Note that grasping success rates are close to 100% for up to 10 objects. As the number of objects increases, the object identification accuracy decreases slightly because the objects start to occlude one each other in the image. This situation is very challenging when the objects are unknown because two objects that are too close can be perceived as a single larger object. Note that we are not aware of another method that would be able to directly predict the workspace position of multiple unseen objects with unknown dimensions using only (uncalibrated) RGB images as input.

**Discussion.** Our experiments demonstrate that our visual predictor is able to scale well up to 10 objects with a constant precision that is sufficient for grasping. We have also observed that our method is able to generalize to objects with shapes not seen during training, such as cups or plastic toys. While we apply our visual predictor to visually guided rearrangement planning, it could be easily extended to other contexts using additional task-specific synthetic training data. Accuracy could be further improved using a refinement similar to [122]. Our approach is limited to 2D predictions for table-top rearrangement planning. Predicting 6DoF pose of unseen objects precise enough for robotic manipulation remains an open problem.

### 3.5.3 Real robot experiments using full pipeline

We evaluated our full pipeline, performing both online visual scene estimation and rearrangement planning by performing 20 rearrangement tasks, each of them composed with 10 objects. In each case, the target configuration was described by an image of a configuration captured from a different viewpoint, with a different table texture and a different type of camera. Despite the very challenging nature of the task, our system succeeded in correctly solving $17/20$ of the experiments. In case of success, our system used on average 12.2 steps. The three failures were due to errors in the visual recognition system (incorrectly estimated number of objects, mismatch of source and target objects). Interestingly, the successful cases were not always perfect runs, in the sense that the re-arrangement strategy was not optimal or that the visual estimation confused two objects at one step of the matching process. However, our system was able to recover robustly from these failures because it is applied in a closed-loop fashion, where then plan is recomputed at each object move.

**The video available on the project page [168] shows additional experiments** including objects other than cubes, different backgrounds, a moving hand-held camera and external perturbations, where an object is moved during the rearrangement. These results demonstrate the robustness of our system. To the best of our knowledge, rearranging a priory unknown number of unseen objects with a robotic arm while relying only on images captured by a moving hand-held camera and dealing with object perturbations has not been demonstrated in prior work.

## 3.6 Conclusion

We have introduced a robust and efficient system for online rearrangement planning, that scales to many objects and recovers from perturbations, without requiring calibrated camera or fiducial markers on objects. To our best knowledge, such a system has not been shown in previous work. At the core of our approach is the idea of applying MCTS to rearrangement planning, which leads to better plans, significant speed-ups and ability to address more general set-ups compared to prior work. While in this work we focus on table-top re-arrangement, the proposed MCTS approach is general and opens-up the possibility for efficient re-arrangement planning in 3D or non-prehensile set-ups.

A limitation of the existing visual system is that it can only be used to predict the 2D position of the centroid of an object in the robot coordinate system. Manipulation is hence limited to overhand grasps of objects relying on a planar 2D surface. To address this limitation, we consider the problem of 6D pose estimation of rigid objects in chapter 4.

# Chapter 4

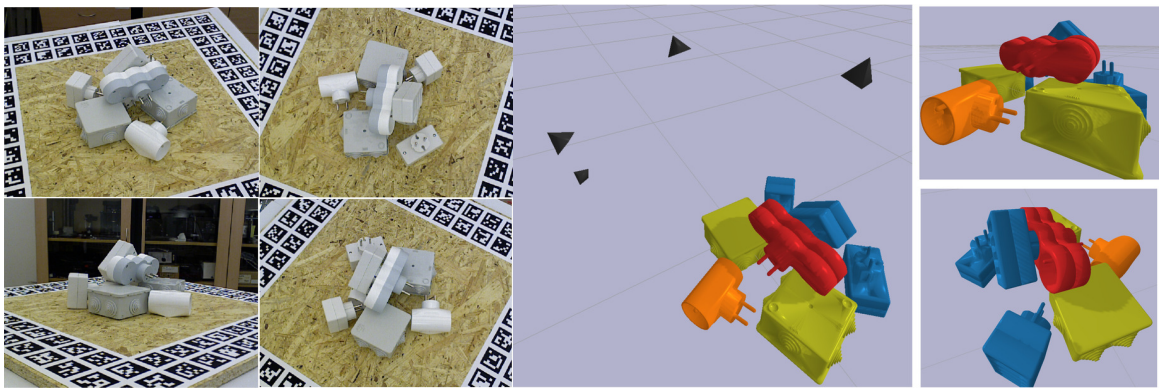
# Single-view and multi-view 6D pose estimation of known rigid objects

In chapter 3, we have presented a robotic system to manipulate objects given a single RGB image of a scene. The visual system was, however, only able to predict the 2D location of the centroid of objects with respect to the robot. In order to perform more complex tasks, we focus on estimating the full 6D pose of an object with a known 3D model. We focus on camera-to-object 6D pose estimation. For manipulating the objects with a robot, knowledge of the robot pose is also required. The problem of camera-to-robot pose estimation will be addressed in chapter 6.

In detail, in this chapter we introduce an approach for recovering the 6D pose of multiple known objects in a scene captured by a set of input images with unknown camera viewpoints. First, we present a single-view single-object 6D pose estimation method based on a render & compare strategy, which we use to generate 6D object pose hypotheses. Second, we develop a robust method for matching individual 6D object pose hypotheses across different input images in order to jointly estimate camera viewpoints and 6D poses of all objects in a *single consistent scene*. Our approach explicitly handles object symmetries, does not require depth measurements, is robust to missing or incorrect object hypotheses, and automatically recovers the number of objects in the scene. Third, we develop a method for global scene refinement given multiple object hypotheses and their correspondences across views. This is achieved by solving an *object-level bundle adjustment* problem that refines the poses of cameras and objects to minimize the reprojection error in all views. We demonstrate that the proposed method, dubbed CosyPose, outperforms current state-of-the-art results for single-view and multi-view 6D object pose estimation by a large margin on several pose estimation benchmarks. Code and pre-trained models are available on the project webpage [26].

## 4.1 Introduction

The goal of this work is to estimate accurate 6D poses of multiple known objects in a 3D scene captured by multiple cameras with unknown positions, as illustrated in figure 4-1. This is a challenging problem because of the texture-less nature of many objects, the pres-



(a) Input: RGB images.

(b) Output: full scene model including objects and camera poses.

Figure 4-1: **CosyPose: 6D object pose estimation optimizing multi-view COnSistencY.** Given (a) a set of RGB images depicting a scene with known objects taken from unknown viewpoints, our method accurately reconstructs the scene, (b) recovering all objects in the scene, their 6D pose and the camera viewpoints. Objects are enlarged for the purpose of visualization.

ence of multiple similar objects, the unknown number and type of objects in the scene, and the unknown positions of cameras. Solving this problem would have, however, important applications in robotics where the knowledge of accurate position and orientation of objects within the scene would allow the robot to plan, navigate and interact with the environment.

Object pose estimation is one of the oldest computer vision problems [123, 124, 172], yet it remains an active area of research [112, 155, 160, 169, 207, 217, 219, 240]. The best performing methods that operate on RGB (no depth) images [69, 112, 113, 155, 199, 217, 240] are based on trainable convolutional neural networks and are able to deal with symmetric or textureless objects, which were challenging for earlier methods relying on local [9, 22, 23, 63, 124] or global [27] gradient-based image features. However, most of these works consider objects independently and estimate their poses using a single input (RGB) image. Yet, in practice, scenes are composed of many objects and multiple images of the scene are often available, e.g. obtained by a single moving camera, or in a multi-camera set-up. In this work, we address these limitations and develop an approach that combines information from *multiple views* and estimates jointly the pose of *multiple objects* to obtain a single consistent scene interpretation.

While the idea of jointly estimating poses of multiple objects from multiple views may seem simple, the following challenges need to be addressed. First, object pose hypotheses made in individual images cannot easily be expressed in a common reference frame when the relative transformations between the cameras are unknown. This is often the case in practical scenarios where camera calibration cannot easily be recovered using local feature registration because the scene lacks texture or the baselines are large. Second, the single-view 6D object pose hypotheses have gross errors in the form of false positives, missed detections, or large pose estimation errors. Third, the candidate 6D object poses estimated from input images suffer from depth ambiguities inherent to single view methods.

In this work, we describe an approach that addresses these challenges. We start from 6D object pose hypotheses that we estimate from each view using a new render-and-compare approach inspired by DeepIM [112]. First, we match individual object pose hypotheses across different views and use the resulting *object-level* correspondences to recover the relative positions between the cameras. Second, gross errors in object detection are addressed using a robust object-level matching procedure based on RANSAC, optimizing the overall scene consistency. Third, noisy single-view object poses are significantly improved using a global *refinement procedure* based on object-level bundle adjustment. The outcome of our approach that optimizes multi-view COnSistencY, hence dubbed CosyPose, is a single consistent reconstruction of the input scene.

Our single-view single-object pose estimation method obtains state-of-the-art results on several 6D pose estimation benchmarks, achieving a significant 34.2% absolute improvement over the state-of-the-art [155] on T-LESS and winning the BOP 2020 6D pose estimation competition [70, 71] in multiple categories. Notably, our method does not rely on an external coarse pose estimation method and unlike prior works we train our networks on all objects of a dataset simultaneously. Our multi-view framework further significantly improves the pose estimation and 6D detection accuracy of our single-view method. Compared to prior works that consider multiple views for 6D pose estimation, our multi-view framework clearly outperforms [109] while not requiring known camera poses and not being limited to a single object of each class per scene. We released the full code, state-of-the-art pre-trained single-view 6D detection and pose estimation models for 7 common object sets, and our multi-view framework which can be combined with any other single-view pose estimator.

## 4.2 Related work

Our work builds on results in single-view and multi-view object 6D pose estimation from RGB images and object-level SLAM.

**Single-view single-object 6D pose estimation.** The object pose estimation problem [123, 124, 172] has been approached either by estimating the pose from 2D-3D correspondences using local invariant features [9, 22, 23, 124], or directly by estimating the object pose using template-matching [63]. However, local features do not work well for texture-less objects and global templates often fail to detect partially occluded objects. Both of these approaches (feature-based and template matching) have been revisited using deep neural networks. A convolutional neural network (CNN) can be used to detect sparse object features in 2D (such as 2D keypoints) [46, 76, 87, 158, 160, 169, 202, 207] or to directly find dense 2D-to-3D correspondences [69, 155, 160, 164, 191, 230, 240]. The resulting 2D-to-3D correspondences are used to recover the camera pose using PnP [108]. Deep approaches have also been used to match implicit pose features, which can be learned without requiring ground truth pose annotations [197, 199]. The estimated 6D pose of the objects can be further refined [112, 127, 147, 169] using an iterative procedure that effectively moves the camera around the object so that the rendered image of the object best matches the input image. Such a refinement step provides important performance improvements and is becoming common practice [217, 240] as a final stage of the estimation process. Our

single-view single-object pose estimation described in section 4.3.2 builds on the ideas of DeepIM [112]. Unlike[112], our method does not rely on a separate method for coarse estimation like PoseCNN [230]. The performance of 6D pose estimation can be further improved using depth sensors [39, 92, 112, 217, 230]. The performance of our pose estimates can be improved using a strategy based on Iterative Closest Points (ICP) alignment [243] if depth is available as shown in section 4.4.3. In the rest of this chapter, we focus on the most challenging scenario where only RGB images are available.

**Multi-view single-object 6D pose estimation.** Multiple views of an object can be used to resolve depth ambiguities and gain robustness with respect to occlusions. Prior work using local invariant features includes [22, 23, 51, 166] and involves some form of feature matching to establish correspondences across views to aggregate information from multiple viewpoints. More recently, the multi-view single-object pose estimation problem has been revisited with a deep neural network that predicts an object pose candidate in each view [109] and aggregates information from multiple views assuming known relative camera poses. In contrast, our work does not assume the camera poses to be known. We experimentally demonstrate that our approach outperforms [109] despite requiring less information.

**Multi-view multi-object 6D pose estimation.** Other works consider all objects in a scene together in order to jointly estimate the state of the scene in the form of a compact representation of the object and camera poses in a common coordinate system. This problem is known as object-level SLAM [178] where a depth-based object pose estimation method [39] is used to recognize objects from a database in individual images and estimate their poses. The individual objects are tracked across frames using depth measurements, assuming the motion of the sensor is continuous. Consecutive depth measurements also enable to produce hypotheses for camera poses using ICP [243] and the poses of objects and cameras are finally refined in a joint optimization procedure. Another approach [35] uses local RGBD patches to generate object hypotheses and find the best view of a scene. All of these methods, however, strongly rely on depth sensors to estimate the 3D structure of the scene while our method only exploits RGB images. In addition, they assume temporal continuity between the views, which is also not required by our approach.

Other works have considered monocular RGB-only object-level SLAM [8, 163, 238]. Related is also [7] where semantic 2D keypoint correspondences across multiple views and local features are used to jointly estimate the pose of a single human and the positions of the observing cameras. All of these works rely on local images features to estimate camera poses. In contrast, our work exploits 6D pose hypotheses generated by a neural network which allows to recover camera poses in situations where feature-based registration fails, as is the case for example for the complex texture-less images of the T-LESS dataset. In addition, [163, 238] do not consider full 6D pose of objects, and [7, 109] only consider scenes with a single instance of each object. In contrast, our method is able to handle scenes with multiple instances of the same object.

This chapter is an extended version of [96]. It provides a significantly extended description of the proposed approach and a new set of results on the seven datasets of the BOP 6D pose estimation challenge [70, 71] together with their analysis.

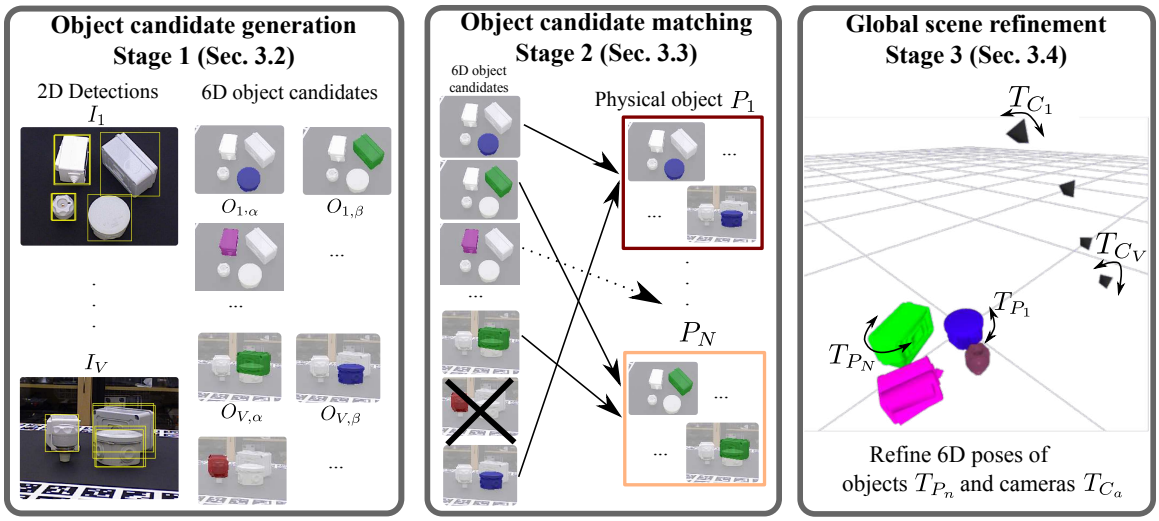



Figure 4-2: **Multi-view multi-object 6D pose estimation.** In the first stage, we obtain initial object candidates in each view separately. In the second stage, we match these object candidates across views to recover a single consistent scene. In the third stage, we globally refine all object and camera poses to minimize multi-view reprojection error.

## 4.3 Multi-view multi-object 6D object pose estimation

In this section, we present our framework for multi-view multi-object pose estimation. We begin with an overview of the approach (section 4.3.1 and figure 4-2), and then detail the three main steps of the approach in the remaining sections.

### 4.3.1 Approach overview

Our goal is to reconstruct a scene composed of multiple objects given a set of RGB images. We assume that we know the 3D models of objects of interest. However, there can be multiple objects of the same type in the scene and no information on the number or type of objects in the scene is available. Furthermore, objects may not be visible in some views, and the relative poses between the cameras are unknown. Our output is a scene model, which includes the number of objects of each type, their 6D poses and the relative poses of the cameras. Our approach is composed of three main stages, summarized in figure 4-2.

In the first stage, we build on the success of recent methods for single-view RGB object detection and 6D pose estimation. Given a set of objects with known 3D models and a single image of a scene, we output a set of candidate detections for each object and for each detection the 6D pose of the object with respect to the camera associated to the image. Note that some of these detections and poses are wrong, and some are missing. We thus consider the poses obtained in this stage as a set of initial *object candidates*, i.e. objects that may be seen in the given view together with an estimate of their pose with respect to this view. This *object candidate generation* process is described in section 4.3.2.

In the second stage, called *object candidate matching* and described in detail in section 4.3.3, we match objects visible in multiple views to obtain a single consistent scene.

This is a difficult problem since object candidates from the first stage typically include many errors due to (i) heavily occluded objects that might be mis-identified or for which the pose estimate might be completely wrong; (ii) confusion between similar objects; and (iii) unusual poses that do not appear in the training set and are not detected correctly. To tackle these challenges, we take inspiration from robust patch matching strategies that have been used in the structure from motion (SfM) literature [54, 200]. In particular, we design a matching strategy similar in spirit to [175] but where we match entire 3D objects across views to obtain a single consistent 3D scene, rather than matching local 2D patches on a single 3D object [175].

The final stage of our approach, described in section 4.3.4, is a global *scene refinement*. We draw inspiration from bundle adjustment [209], but the optimization is performed at the level of objects: the 6D poses of all objects and cameras are refined to minimize a global reprojection error.

### 4.3.2 Stage 1: single-view 6D pose estimation for object candidate generation

Our system takes as input multiple photographs of a scene $\{I_a\}$ and a set of 3D models, each associated to an object label $l$. We assume the intrinsic parameters of camera $C_a$ associated to image $I_a$ are known as is usually the case in single-view pose estimation methods. In each view $I_a$, we obtain a set of object detections using an object detector (e.g. Faster-RCNN [171], RetinaNet [116], or Mask R-CNN [58]), and a set of candidate pose estimates using a single-view single-object pose estimator (e.g. PoseCNN [230], DPOD [240], DeepIM [112]). Each 2D candidate detection in view $I_a$ is identified by an index $\alpha$ and corresponds to an *object candidate* $O_{a,\alpha}$, associated with a predicted object label $l_{a,\alpha}$ and a 6D pose estimate $T_{C_a O_{a,\alpha}}$, composed of a 3D rotation matrix and a 3D translation vector, with respect to camera $C_a$. While our multi-view framework is agnostic to the particular single-view 6D pose estimation method used to generate the candidates, we develop our own single-view single-object pose estimator which improves significantly over state of the art and is described next.

**Single-view 6D pose estimation.** We introduce a method for single-view 6D object pose estimation that builds on the ideas of the iterative render & compare refinement method of DeepIM [112] with some simplifications and technical improvements which are detailed next. The 6D pose estimator takes as input a 2D candidate detection (a bounding box associated with a detected object label) and outputs a 6D pose estimate for this object. The same method is applied to each candidate detection independently. For simplificity, we focus on one view and one detection and drop the subscripts $a$ and $\alpha$ of the estimated pose $T_{C_a O_{a,\alpha}}$.

**Iterative render & compare strategy.** Building on prior work that use render & compare strategy [112, 127, 240] for pose estimation, we use a deep neural network that iteratively refines the 6D pose estimate of the considered object. Our network takes as input two RGB images. The first image is the (real) input image $I$ cropped on the region of the image showing the object, denoted $I^c$ and derived from the object detection bounding box as explained in section 4.6.1 of the appendix. At iteration $k$, the second image is a (synthetic)

rendering of the object with label $l$ rendered in a pose $T_{C,O}^{k-1}$ that corresponds to the object pose estimated at the previous iteration. The network outputs an updated refined pose $T_{C,O}^{k}$, and the same procedure is iterated $K$ times. The initial pose $T_{C,O}^{0}$ can be provided by any coarse 6D pose estimation method. While previous works use different methods for coarse estimation, for example DeepIM [112] relies on PoseCNN [230], we show that the same network architecture can be used for performing coarse pose estimation, as described below. We now give details of our pose estimation network and present the main differences from [112].

**Pose estimation network.** The network takes as input the concatenation of the synthetic and real cropped images. Both images are resized to the input resolution: $320 \times 240$. The backbone is EfficientNet-B3 [201] followed by spatial average pooling. The prediction layer is a single fully connected layer, which outputs 9 values corresponding to one 3-vector $[v_x, v_y, v_z]$ to predict an update of the translation of the input pose, and two 3-vectors $e_1, e_2$ that define the rotation update as explained below. Given these predictions, the pose update is obtained as

$$x^{k+1} = \left( \frac{v_x}{f_x^C} + \frac{x^k}{z^k} \right) z^{k+1}, \tag{4.1}$$

$$y^{k+1} = \left( \frac{v_y}{f_y^C} + \frac{y^k}{z^k} \right) z^{k+1}, \tag{4.2}$$

$$z^{k+1} = v_z z^k, \tag{4.3}$$

$$R^{k+1} = R(e_1, e_2) R^k, \tag{4.4}$$

where $[x^k, y^k, z^k]$ is the 3D translation vector of the relative camera-object pose $T_{CO}^{k}$ at iteration $k$, $R^k$ the rotation matrix of $T_{CO}^{k}$, $f_x^C$ and $f_y^C$ are the (known) focal lengths that correspond to the (virtual) camera associated with the cropped input image $I^C$, and $R(e_1, e_2)$ is a rotation matrix describing the rotation update recovered from $e_1, e_2$ using [246] by orthogonalizing the basis defined by the two predicted rotation vectors $e_1, e_2$ (equations are provided in section 4.6.1 of the appendix). Finally, $[x^{k+1}, y^{k+1}, z^{k+1}]$ and $R^{k+1}$ are, respectively, the translation and rotation components of the output updated pose estimate $T_{CO}^{k+1}$. The main differences from the DeepIM approach [112] are threefold. First, we use the rotation parametrization of [246], which has been shown in [246] to be better suitable for learning compared to quaternions used in [112]. Second, we use EfficientNet-B3[201], which is a more recent backbone compared to FlowNet used in [112] and we do not include auxiliary predictions of flow and mask, which makes the method simpler and easier to train. Third, we use the intrinsics $f_x^C$, $f_y^C$ of the cropped camera associated with the input (cropped) image. DeepIM uses the intrinsics parameters of the non-cropped camera $f_x$, $f_y$ and fixes them to 1 during training because the intrinsic parameters of the input camera are fixed on their datasets. We use the cropped focal lengths instead because cropping and resizing the crop of the input image changes the apparent focal length. Using the cropped focal lengths forces the network to only predict xy translations in pixels and the network can therefore become invariant to the intrinsic parameters of the input (cropped) camera.

**Object symmetries.** Handling object symmetries is a major challenge for object pose estimation since the object pose can only be estimated up to a symmetry. We thus need

to consider symmetries explicitly together with the pose estimates. Each 3D model $l$ is associated to a set of symmetries $S(l)$. Inspired by [165], we define the set of symmetries $S(l)$ as the set of transformations $S$ that leave the appearance of object $l$ unchanged:

$$S(l) = \{S \in \mathrm{SE}(3) \,\mathrm{s.t}\, \forall\, T \in \mathrm{SE}(3), \mathcal{R}(l, T) = \mathcal{R}(l, TS)\}, \tag{4.5}$$

where $\mathcal{R}(l, X)$ is the rendered image of object $l$ captured in pose $X$ and $S$ is the rigid motion associated to the symmetry. We introduce a symmetric distance $D_l$ which measures the distance between two 6D poses represented by transformations $T_1$ and $T_2$ and takes into account the symmetries of the object $l$:

$$D_l(T_1, T_2) = \min_{S \in S(l)} \frac{1}{|\mathcal{X}_l|} \sum_{\mathrm{x} \in \mathcal{X}_l} ||T_1 S\mathrm{x} - T_2\mathrm{x}||_2, \tag{4.6}$$

where $\mathcal{X}_l$ is the set of 3D points of the object $l$ and $|\mathcal{X}_l|$ the number of 3D points. $D_l(T_1, T_2)$ measures the average error between the points transformed with $T_1$ and $T_2$ for the symmetry $S$ that best aligns the (transformed) points. In practice, to compute this distance for objects with axes of symmetries for which $S(l)$ is infinite (e.g. bowls), we discretize $S(l)$ using $64$ rotation angles around each symmetry axis, similar to [219]. The distance $D_l$ is used to compute the loss, which we detail next.

**Disentangled loss.** Our pose update parametrization, detailed in equations (4.1)-(4.4), disentangles the predictions of the 3D translation and the 3D rotation: these two components of the pose update can be predicted independently. However, the camera-object pose $T^k_{C,O}$ and hence the loss are computed using non-linear relations between the parameters predicted by the network, which can lead to unstable training as pointed out by [189]. We follow the recommendations of [189] for better training by disentangling the effects of the relative depth predictions $v_z$, image-space $xy$ translation $v_x, v_y$ and rotation $R$. In order to compute the loss, we define the pose update function $F$ which takes as input the initial estimate of the pose $T^k_{CO}$, the predictions of the neural network $[v_x, v_y, v_z]$ and $R$, and outputs the updated pose:

$$T^{k+1}_{CO} = F(T^k_{CO}, [v_x, v_y, v_z], R), \tag{4.7}$$

where the closed form of function $F$ is expressed in equations (4.1)-(4.4). We also write $[\hat{v}_x, \hat{v}_y, \hat{v}_z]$ and $\hat{R}$ as the target predictions, i.e. the predictions such that

$$\hat{T}_{CO} = F(T^k_{CO}, [\hat{v}_x, \hat{v}_y, \hat{v}_z], \hat{R}), \tag{4.8}$$

where $\hat{T}_{CO}$ is the ground truth camera-object pose. The pose update network is trained using the following loss function:

$$\begin{aligned}
\mathcal{L} = {} & D_l(F(T^k_{CO}, [v_x, v_y, \hat{v}_z], \hat{R}), \hat{T}_{CO}) && (4.9)\\
& + D_l(F(T^k_{CO}, [\hat{v}_x, \hat{v}_y, v_z], \hat{R}), \hat{T}_{CO}) && (4.10)\\
& + D_l(F(T^k_{CO}, [\hat{v}_x, \hat{v}_y, \hat{v}_z], R), \hat{T}_{CO}), && (4.11)
\end{aligned}$$

where $D_l$ is the symmetric distance defined in equation (4.6), with the $L_2$ norm replaced by the $L_1$ norm. The different terms of this loss separate the influence of: $xy$ translation (4.9),

relative depth (4.10) and rotation (4.11).

**Coarse estimation.** The network architecture presented above can be used to iteratively refine the pose of an object given an initial coarse estimate. We use the same strategy to perform coarse pose estimation. This is done by providing as input to a separate coarse pose estimation network a canonical pose that corresponds to the object being rendered at a fixed distance of 1 meter of the camera in the center of the input 2D bounding box with a fixed rotation with the $z$ axis of the object model pointing upwards and the other axes parallel to the camera. The coarse and refinement networks use the same architecture, but the weights are distinct and each network is trained independently. During training, the distribution of rendered input pose is sampled by adding noise to the ground truth for the refinement network, and generated using the initialization strategy described above for the coarse network. At test time, the coarse pose estimation network is only ran for 1 iteration while the refinement network can be run for multiple iterations. While only one network could be used for both coarse and refinement, we observed better results by training two separate networks with the same architecture but different weights obtained by training with different input training distributions.

### 4.3.3 Stage 2: object candidate matching

The output of the first stage is a set of object candidates $\{O_{a,\alpha}\}$ in all views, each associated with a 6D pose estimate $T_{C_a O_{a,\alpha}}$ with respect to camera $C_a$. Given these *single-view* object candidates, our *multi-view* matching module illustrated in figure 4-2 aims at (i) removing the object candidates that are not consistent across views and (ii) matching object candidates that correspond to the same physical object. We solve this problem in two steps detailed below: (A) selection of candidate pairs of objects in all pairs of views, and (B) scene-level matching.

**A. 2-view candidate pair selection.** We first focus on a single pair of views $(I_a, I_b)$ of the scene and find all pairs of object candidates $(O_{a,\alpha}, O_{b,\beta})$, one in each view, which correspond to the same physical object in these two views. To do so, we use a RANSAC procedure where we hypothesize a relative pose between the two cameras and count the number of inliers, i.e. the number of consistent pairs of object candidates in the two views. We then select the solution with the most inliers which gives associations between the object candidates in the two views. In the rest of the section, we describe in more detail how we sample relative camera poses and how we define inlier candidate pairs.

**Sampling of relative camera poses.** Sampling meaningful camera poses is one of the main challenges for our approach. Indeed, directly sampling at random the space of possible camera poses would be inefficient. Instead, as usual in RANSAC, we sample pairs of object candidates (associated to the same object label) in the two views, hypothesize that they correspond to the same physical object and use them to infer a relative camera pose hypothesis. However, since objects can have symmetries, a single pair of candidates is not enough to obtain a relative pose hypothesis without ambiguities and we thus sample two pairs of object candidates, which in most cases is sufficient to disambiguate symmetries.

In detail, we sample two tentative object candidate pairs with pair-wise consistent labels $(O_{a,\alpha}, O_{b,\beta})$ and $(O_{a,\gamma}, O_{b,\delta})$ and use them to build a relative camera pose hypothesis, $T_{C_a C_b}$. We obtain the relative camera pose hypothesis by (i) assuming that $(O_{a,\alpha}, O_{b,\beta})$ cor-

respond to the same physical object and (ii) disambiguating symmetries by assuming that $(O_{a,\gamma}, O_{b,\delta})$ also correspond to the same physical object, and thus selecting the symmetry that minimize their symmetric distance

$$T_{C_aC_b} = T_{C_aO_{a,\alpha}} S^\star T^{-1}_{C_bO_{b,\beta}}, \text{ with} \tag{4.12}$$

$$S^\star = \underset{S \in S(l)}{\operatorname{argmin}} D_l(T_{C_aO_{a,\gamma}}, (T_{C_aO_{a,\alpha}} S T^{-1}_{C_bO_{b,\beta}}) T_{C_bO_{b,\delta}}), \tag{4.13}$$

where $l = l_{a,\alpha} = l_{b,\beta}$ is the object label associated to the first pair, and $S^\star$ is the object symmetry which best aligns the point clouds associated to the second pair of objects ($O_{a,\gamma}$ and $O_{b,\delta}$). This relative camera pose estimation step is illustrated in section 4.6.2 of the appendix. If the union of the two physical objects is symmetric, e.g. two spheres, the pose computed may be incorrect but it would not be verified by a third pair of objects, and the hypothesis would be discarded.

**Counting pairs of inlier candidates.** Let's assume we are given a relative pose hypothesis between the cameras $T_{C_aC_b}$. For each object candidate $O_{a,\alpha}$ in the first view, we find the object candidate in the second view $O_{b,\beta}$ with the same label $l = l_{a,\alpha} = l_{b,\beta}$ that minimizes the symmetric distance $D_l(T_{C_aO_{a,\alpha}}, T_{C_aC_b} T_{C_bO_{b,\beta}})$. In other words, $O_{b,\beta}$ is the object candidate in the second view closest to $O_{a,\alpha}$ under the hypothesized relative pose between the cameras. This pair $(O_{a,\alpha}, O_{b,\beta})$ is considered an inlier if the associated symmetric distance is smaller than a given threshold $C$. The total number of inliers is used to score the relative camera pose $T_{C_aC_b}$. Note that we discard the hypothesis which have fewer than three inliers.

**B. Scene-level matching.** We use the result of the 2-view candidate pair selection applied to each image pair to define a graph between all candidate objects. Each vertex corresponds to an object candidate in one view and edges correspond to pairs selected from 2-view candidate pair selection, i.e. pairs that had sufficient inlier support. We first remove isolated vertices, which correspond to object candidates that have not been validated by other views. Then, we associate to each connected component in the graph a unique physical object, which corresponds to a set of initial object candidates originating from different views. We call these physical objects $P_1, ...P_N$ with $N$ the total number of physical objects, i.e. the number of connected components in the graph. We write $(a, \alpha) \in P_n$ to denote the fact that an object candidate $O_{a,\alpha}$ is in the connected component of object $P_n$. Since all the objects in a connected component share the same object label (they could not have been connected otherwise), we can associate without ambiguity an object label $l_n$ to each physical object $P_n$.

### 4.3.4 Stage 3: scene refinement

After the previous stage, the correspondences between object candidates in the individual images are known, and the non-coherent object candidates have been removed. The final stage aims at recovering a unique and consistent scene model by performing global joint refinement of objects and camera poses.

In detail, the goal of this stage is to estimate poses of physical objects $P_n$, represented by transformations $T_{P_1}, \ldots, T_{P_N}$, and cameras $C_v$, represented by transformations

$T_{C_1}, \ldots, T_{C_V}$, in a common world coordinate frame. This is similar to the standard bundle adjustment problem where the goal is to recover the 3D points of a scene together with the camera poses. This is typically addressed by minimizing a reconstruction loss that measures the 2D discrepancies between the projection of the 3D points and their measurements in the cameras. In our case, instead of working at the level of points as done in the bundle adjustment setting, we introduce a reconstruction loss that operates at the level of objects.

More formally, for each object present in the scene, we introduce an object-candidate reprojection loss accounting for symmetries. We define the loss for a candidate object $O_{a,\alpha}$ associated to a physical object $P_n$ (i.e. $(a, \alpha) \in P_n$) and the estimated candidate object pose $T_{C_a O_{a,\alpha}}$ with respect to $C_a$ as:

$$L\left(T_{P_n}, T_{C_a} | T_{C_a O_{a,\alpha}}\right) = \min_{S \in S(l)} \frac{1}{|\mathcal{X}_l|} \sum_{\mathrm{x} \in \mathcal{X}_l} ||\pi_a(T_{C_a O_{a,\alpha}} S\mathrm{x}) - \pi_a(T_{C_a}^{-1} T_{P_n} \mathrm{x})||, \tag{4.14}$$

where $||\cdot||$ is a truncated $L_2$ loss, $l = l_n$ is the label of the physical object $P_n$, $T_{P_n}$ the 6D pose of object $P_n$ in the world coordinate frame, $T_{C_a}$ the pose of camera $C_a$ in the world coordinate frame, $\mathcal{X}_l$ the set of 3D points associated to the 3D model of object $l$, $S(l)$ the symmetries of the object model $l$, and the operator $\pi_a$ corresponds to the 2D projection of 3D points expressed in the camera frame $C_a$ by the intrinsic calibration matrix of camera $C_a$. The inner sum in Eq. (4.14) is the error between (i) the 3D points x of the object model $l$ projected to the image with the single view estimate of the transformation $T_{C_a O_\alpha}$ that is associated with the physical object (i.e. $(a, \alpha) \in P_n$) (first term, the image measurement) and (ii) the 3D points $T_{P_n}\mathrm{x}$ on the object $P_n$ projected to the image by the global estimate of camera $C_a$ (second term, global estimates).

Recovering the state of the unique scene which best explains the measurements consists in solving the following consensus optimization problem:

$$\min_{T_{P_1}, \ldots, T_{P_N}, T_{C_1}, \ldots, T_{C_V}} \sum_{n=1}^{N} \sum_{(a,\alpha) \in P_n} L\left(T_{P_n}, T_{C_a} | T_{C_a O_{a,\alpha}}\right), \tag{4.15}$$

where the first sum is over all the physical objects $P_n$ and the second one over all object candidates $O_{a,\alpha}$ corresponding to the physical object $P_n$. In other words, we wish to find global estimates of object poses $T_{P_n}$ and camera poses $T_{C_a}$ to match the (inlier) object candidate poses $T_{C_a O_{a,\alpha}}$ obtained in the individual views. The optimization problem is solved using the Levenberg-Marquart algorithm. We provide more details in section 4.6.3 of the appendix.

## 4.4 Results

In this section, we evaluate and analyze our method on several 6D pose estimation benchmarks. We first consider the YCB-Video [230] and T-LESS[68] datasets, which both provide multiple views and ground truth 6D object poses for cluttered scenes with multiple objects. We start by evaluating our single-view 6D pose estimator in section 4.4.1. We

| | AUC of ADD-S | AUC of ADD(-S) |
|---|---|---|
| PoseCNN [230] | - | 61.3 |
| MCN [109] | 75.1 | - |
| PVNet [160] | - | 73.4 |
| DeepIM [112] | 88.1 | 81.9 |
| Ours | **89.8** | **84.5** |

(a) YCB-Video

| | $e_{\text{vsd}} < 0.3$ |
|---|---|
| Implicit [199] | 26.8 |
| Pix2pose [155] | 29.5 |
| Ours | **63.8** |
| w/o loss | 60.1 |
| w/o network | 59.5 |
| w/o rot. | 61.0 |
| w/o data augm. | 37.0 |

(b) T-LESS SiSo task

Table 4.1: **Single-view 6D pose estimation.** Comparisons with state-of-the-art methods on the YCB-Video (a) and T-LESS datasets (b).

notably show that our single-view method already improves state-of-the-art results. In section 4.4.2, we validate our multi-view multi-object framework by demonstrating consistent improvements over the single-view baseline on both datasets and multiple metrics. We then consider the BOP 6D pose estimation challenge [70] which includes seven different datasets in section 4.4.3. We report the results of our approach for single-view 6D detection (2D detection and 6D pose estimation). Our approach was the top performing method in the 2020 edition of the challenge [71] in multiple categories. We further show that these results can be consistently improved using our multi-view framework when multiple images of a scene are available and we present additional multi-view results on the recently released HomebrewDB dataset [84].

### 4.4.1 Single-view single-object experiments

**Implementation details.** *Training data.* Due to the complexity of annotating real data with 6D pose at large scale, most recent methods[112, 199, 240] generate additionnal synthetic training data. All of our models are trained using either only synthetic images, or a combination of real and synthetic images. The synthetic images considered are generated using a standard OpenGL renderer. The synthetic images are generated using domain randomization [100, 122, 203]. We refer to [71] for more details of the synthetic data generation process. In addition to using synthetic images during training, we add data augmentation to all training images (both synthetic and real when available). Data augmentation includes Gaussian blur, contrast, brightness, color and sharpness filters from the Pillow library [21]. We also use images from the Pascal VOC dataset as a background with a probability 0.3, following [112]. The data augmentation was found to be a key ingredient for a successful sim-to-real transfer. We illustrate example of training images with data augmentation applied in figure 4-3.
*Training procedure.* Only two networks are used to represent all objects in each dataset: one network for coarse pose estimation and one network for iterative refinement. All the networks are randomly initialized and are trained with the same procedure. We use synchronous distributed training on 32 GPUs, with 32 images on each GPU for a total batch size of 1024. We use the Adam optimizer [90] with a learning rate of $3.10^{-4}$ and default

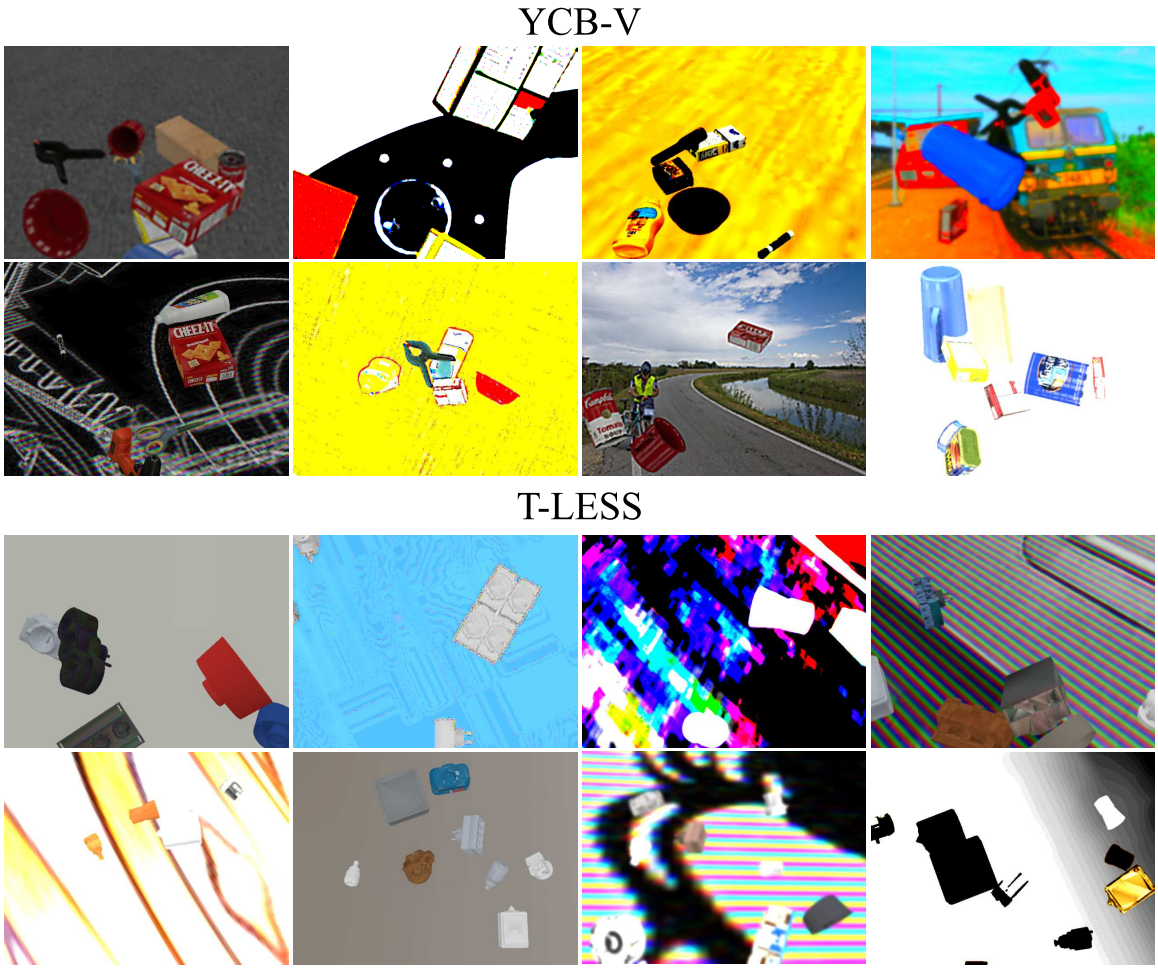



Figure 4-3: **Training images for our single-view pose estimation networks.** Examples of images used for training the networks on T-LESS and YCB-Video. Notice the heavy data augmentation which is a key ingredient for successful sim-to-real transfer.

momentum parameters. First, the network is trained for 80k iterations on synthetic data only. Following [49], we also use a warm-up phase where we progressively increase the learning rate from 0 to $3.10^{-4}$ during the first 5k iterations and drop the learning rate to $3.10^{-5}$ at $50k$ iterations. If real training images are available, the network is trained for another 80k iterations on both real and synthetic training images. In this second phase, the real training images account for around $25\%$ of each batch.

**Evaluation on YCB-Video.** Following [112, 160, 230], we evaluate on a subset of 2949 keyframes from videos of the 12 testing scenes. We use the standard ADD-S and ADD(-S) metrics and their area-under-the-curves [230] (please see section 4.6.4 of the appendix for details on the metrics). We evaluate our refinement method using the same detections and coarse estimates as DeepIM [112], provided by PoseCNN [230]. We ran two iterations of pose refinement network. Results are shown in Table 4.1a. Our method improves over the current-state-of-the-art DeepIM [112], by approximately 2 points on the AUC of ADD-S and ADD(-S) metrics.

**Evaluation on T-LESS.** As explained in section 4.3.2, we use our single-view approach both for coarse pose estimation and refinement. We compare our method against the two recent RGB-only methods Pix2Pose [155] and Implicit [199] which were the top-performing methods on this dataset at the moment of the submission. For a fair comparison of the 6D pose estimation method, we use the detections from the same RetinaNet model as in [155]. We report results on the SiSo task [70] and use the standard visual surface discrepancy (vsd) recall metric with the same parameters as in [155, 199]. Results are presented in Table 4.1b. Using the $e_{\text{vsd}} < 0.3$ metric, our {coarse + refinement} solution achieves a significant $34.2\%$ absolute improvement compared to existing state-of-the-art methods. Note that [112] did not report results on T-LESS. We also evaluate on this dataset the benefits of the key components of our single-view approach compared to the components used in DeepIM[112]. More precisely, we evaluate the importance of the base network (our EfficientNet vs FlowNet pre-trained), loss (our symmetric and disentangled vs. point-matching loss with $L_1$ norm), rotation parametrization (our using [246] vs. quaternions) and data augmentation (our color augmentation, similar to [199] vs. none). Loss, network and rotation parametrization bring a small but clear improvement. Using data augmentation is crucial on the T-LESS dataset where training is performed only on synthetic data and real images of the objects on dark background.

### 4.4.2 Multi-view experiments

As shown above, our single-view method achieves state-of-the-art results on both datasets. We now evaluate the performance of our multi-view approach to estimate 6D poses in scenes with multiple objects and multiples views.

**Implementation details.** On both datasets, we use the same hyper-parameters. In stage 1, we only consider object detections with a score superior to 0.3 to limit the number of detections. In stage 2, we use a RANSAC 3D inlier threshold of $C = 2\,\text{cm}$. This low threshold ensures that no outliers are considered while associating object candidates. We use a maximum number of 2000 RANSAC iterations for each pair of views, but this limit is only reached for the most complex scenes of the T-LESS dataset containing tens of detec-

| | 1 view | 5 views |
|---|---|---|
| [109] | 75.1 | 80.2 |
| Ours | 89.8 | **93.4** |

(a) YCB-Video (AUC of ADD-S)

| | 1 view | 4 views | 8 views |
|---|---|---|---|
| AUC of ADD-S | 72.1 | 76.0 | **78.9** |
| ADD-S $<$ 0.1d | 68.0 | 72.6 | **76.6** |
| $e_{\text{vsd}} < 0.3$ | 62.6 | 67.6 | **71.6** |
| mAP@ADD-S<0.1d | 55.0 | 61.6 | **69.0** |

(b) T-LESS ViVo task (ours, 1000 images)

Table 4.2: **Multi-view multi-object results.** (a) Our approach significantly outperforms [109] on the YCB-Video dataset in both the single view and multi-view scenarios while not requiring known camera poses. (b) Results on the T-LESS dataset. Using multiple views clearly improves our results.

tions. For instance, in the context of two views with six different 6D object candidates in each view, only 15 RANSAC iterations are enough to explore all relative camera pose hypotheses. For the scene refinement (stage 3), we use 100 iterations of Levenberg-Marquart (the optimization typically converges in less than 10 iterations).

**Evaluation details.** In the single-view evaluation, the poses of the objects are expressed with respect to the camera frame. To fairly compare with the single-view baseline, we also evaluate the object poses in the camera frames, that we compute using the absolute object poses and camera placements estimated by our global scene refinement method. Standard metrics for 6D pose estimation strongly penalize methods with low detection recall. To avoid being penalized for removing objects that cannot be verified across several views, we thus add the initial object candidates to the set of predictions but with confidence scores strictly lower than the predictions from our full scene reconstruction.

**Multi-view multi-object quantitative results.** The problem that we consider, recovering the 6D object poses of multiple known objects in a scene captured by several RGB images taken from unknown viewpoints has not, to the best of our knowledge, been addressed by prior work reporting results on the YCB-Video and T-LESS datasets. The closest work is [109], which considers multi-view scenarios on YCB-Video and uses ground truth camera poses to align the viewpoints. In [109], results are provided for prediction using 5 views. We use our approach with the same number of input images but without using ground truth calibration and report results in Table 4.2a. Our method significantly outperforms [109] in both single-view and multi-view scenarios.

We also perform multi-view experiments on T-LESS with a variable number of views. We follow the multi-instance BOP[70] protocol for ADD-S<0.1d and $e_{\text{vsd}} < 0.3$. These metrics only consider the top-$m$ predictions with highest score for each class in each image, where $m$ is the number of ground truth objects of the class in the scene. As a consequence, these metrics do not penalize making incorrect predictions for classes that are not in the scene, which happens for most methods and is problematic for any practical application. We thus propose to analyze precision-recall tradeoff similar to the standard practice in object detection. We consider positive predictions that satisfy ADD-S<0.1d and report mAP@ADD-S<0.1d. Results are shown in Table 4.2b for the ViVo task on 1000 images.

**Benefits of scene refinement.** To demonstrate the benefits of global scene refinement (stage 3), we report in Table 4.3 the average ADD-S errors of the inlier candidates before

| | YCB dataset | T-LESS dataset |
|---|---|---|
| Before refinement | 6.40 | 4.43 |
| After refinement | **5.05** | **3.19** |

Table 4.3: Benefits of the scene refinement stage. We report pose ADD-S errors (in mm) for the inlier object candidates before and after global scene refinement. Scene-refinement improves 6D pose estimation accuracy.

and after solving the optimization problem of Eq.(4.15). We note a clear relative improvement, around 20% on both datasets.

**Relative camera pose estimation.** A key feature of our method is that it does not require camera position to be known and instead robustly estimates it from the 6D object candidates. We investigated alternatives to our joint camera pose estimation. First, we used COLMAP [180, 181], a popular feature-based SfM software, to recover camera poses. On randomly sampled groups of 5 views from the YCB-Video dataset COLMAP outputs camera poses in only $67\%$ of cases compared to $95\%$ for our method. On groups of 8 views from the more difficult T-LESS dataset, COLMAP outputs camera poses only in 4% of cases, compared to 74% for our method. Our method therefore demonstrates a significant interest compared to COLMAP that uses features to recover camera poses, especially for complex textureless scenes like in the T-LESS dataset. Second, instead of estimating camera poses using our approach, we investigated using ground truth camera poses available for the two datasets. We found that the improvements using ground truth camera poses over the camera poses recovered automatically by our method were only minor: within $1\%$ for T-LESS (4 views) and YCB-Video (5 views), and within $3\%$ for T-LESS (8 views). This demonstrates that our approach recovers accurate camera poses even for scenes containing only symmetric objects as in the T-LESS dataset.

**Please see section 4.6.5 of the appendix for additional results**, including a detailed and illustrated discussion of the main limitations of the approach.

**Computational cost.** For a case with 4 views and 6 2D detections per view, our approach takes approximately 320 ms to predict the state of the scene. This timing includes: 190 ms for estimating the 6D poses of all candidates (stage 1, 1 iteration of the coarse and refinement networks), 40 ms for the object candidate association (stage 2) and 90 ms for the scene refinement (stage 3). Further speed-ups towards real-time performance could be achieved, for example, by exploiting temporal continuity in a video sequence.

### 4.4.3 6D Pose estimation challenge

We now consider the BOP 6D pose estimation challenge [70, 71]. This challenge evaluates the performance of 6D detection and pose estimation methods on 7 datasets: LineMOD Occlusion (LM-O) [66], T-LESS [68], TUD-L [70], IC-BIN [35], ITODD [38], HomebrewDB (HB) [84] and YCB-Video (YCB-V) [230] using the $AR$ score, which is an average of multiple metrics over all datasets. We refer to [71] for more details on the metrics and the evaluation protocol. In the following, we first give details of our method used to perform 6D detection and pose estimation on all datasets, and then present our entries in

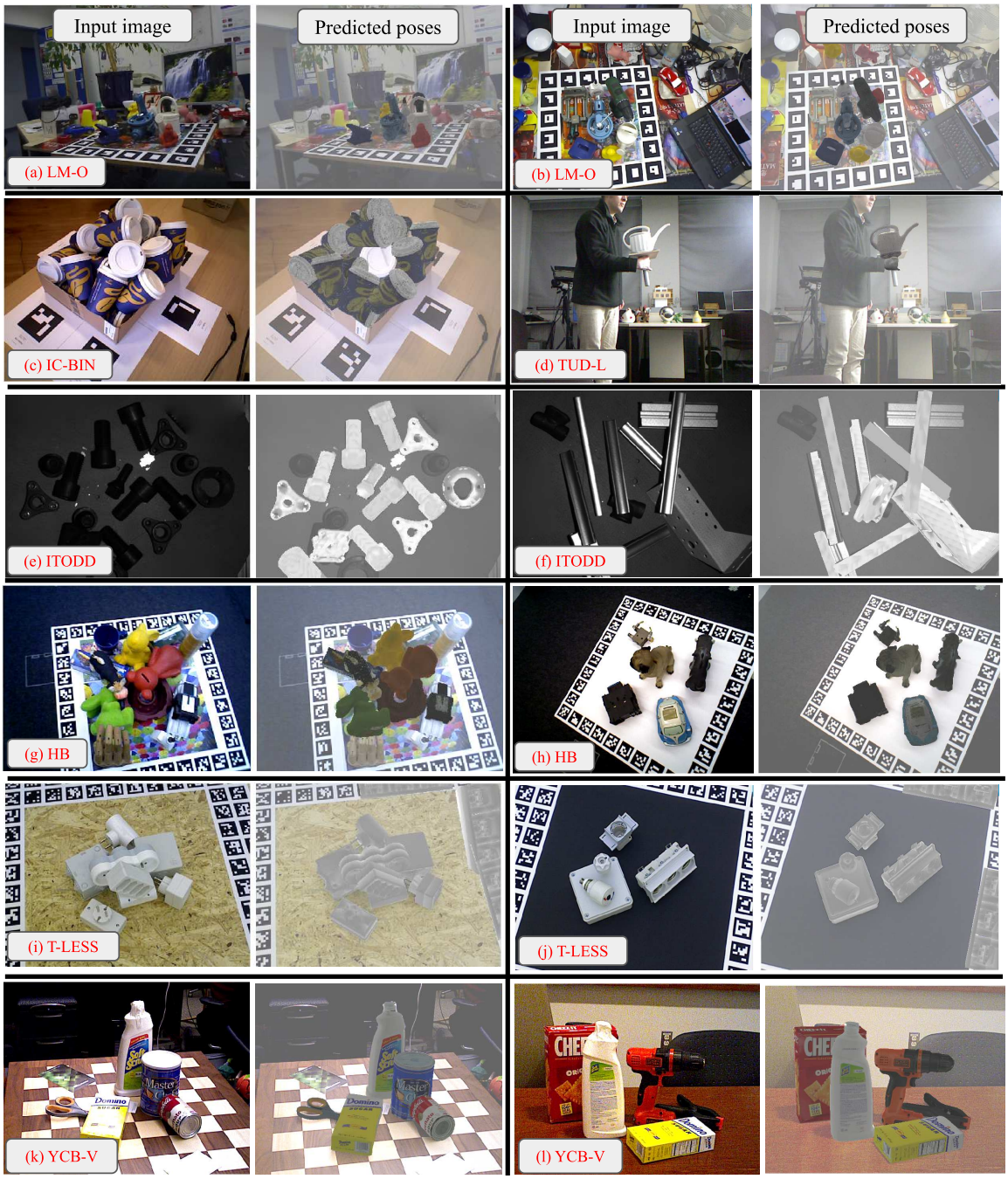



Figure 4-4: **Single-view qualitative results from the 2020 6D object pose estimation (BOP) challenge.** We present twelve examples of predictions by our 2D detector and 6D pose estimation strategy on 7 different datasets. For each example, the first column presents the input RGB image. The second column presents the 6D pose estimates, here illustrated by overlaying the detected object CAD models in the predicted 6D poses over the input image. Notice how our method is robust to partial occlusions (a) (g) (i) (k) and is able to handle complex scenes with many similar (c) (e) and symmetric (c) (i) objects.

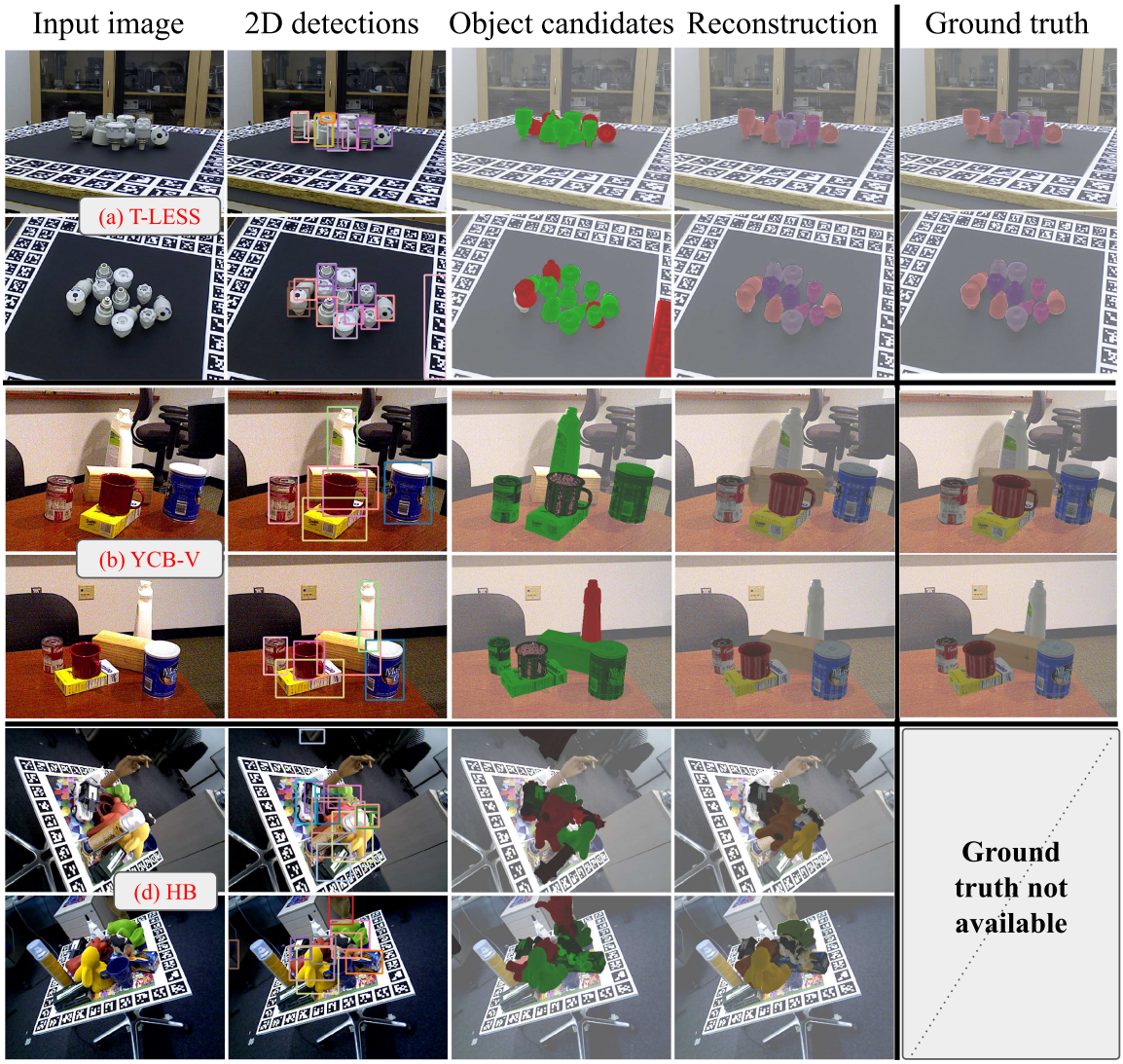



Figure 4-5: **Multi-view qualitative results.** We present three examples of scene reconstructions using our multi-view framework. For each scene, two (out of 8) views that were used to reconstruct the scene are shown as two rows. In each row, the first column shows the input RGB image and the name of the corresponding dataset. The second column shows the 2D detections. The third column shows all object candidates with marked inliers (green) and outliers (red). The fourth column shows the final scene reconstruction. The last column shows the ground truth when available. Notice, for example, in the first example on T-LESS how our method estimates accurate 6D object poses for many objects in challenging scenes containing texture-less and symmetric objects, severe occlusions, and where many objects are similar to each other. **More examples are in section 4.6.5 of the appendix.**

| Method | Real training images | Avg. | LM-O | T-LESS† | TUD-L† | IC-BIN | ITODD | HB | YCB-V† | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| CosyPose-Synt+Real (Ours) | ✓ | **63.7** | **63.3** | **72.8** | **82.3** | **58.3** | **21.6** | 65.6 | **82.1** | **0.449** |
| CDPVNv2 [113] | ✓ | 52.9 | 62.4 | 47.8 | 77.2 | 47.3 | 10.2 | **72.2** | 53.2 | 0.935 |
| Pix2Pose [155] | ✓ | 34.2 | 36.3 | 34.4 | 42.0 | 22.6 | 13.4 | 44.6 | 45.7 | 1.215 |
| CosyPose-PBR (Ours) | × | **57.0** | **63.3** | **64.0** | **68.5** | **58.3** | **21.6** | 65.6 | **57.4** | **0.475** |
| CDPVNv2 [113] | × | 47.2 | 62.4 | 40.7 | 58.8 | 47.3 | 10.2 | **72.2** | 39.0 | 0.978 |
| EPOS [69] | × | 45.7 | 54.7 | 46.7 | 55.8 | 36.3 | 18.6 | 68.0 | 49.9 | 1.874 |

Table 4.4: **Results of the 2020 6D object pose estimation (BOP) challenge for the RGB-only methods.** We report the AR score on each of the 7 datasets considered in the BOP challenge and the average across datasets. † denotes datasets for which real and synthetic training images are available. Only synthetic ones are available for the other datasets.

the different categories of the 2020 edition of the competition.

**Our 2D detector.** The experiments presented in Sections 4.4.1 and 4.4.2 used 2D detections provided by prior works, PoseCNN [230] on YCB-Video and a RetinaNet model trained on images generated by the authors of Pix2pose [155] on T-LESS. Because the challenge requires to use the same method with the same hyper-parameters across all datasets, we trained our own 2D detector on each of the 7 datasets. Our 2D detector is a Mask R-CNN model [58] with a ResNet-50 [59] feature pyramid (FPN) backbone [115]. The network weights are initialized from a network trained on Microsoft COCO [117] and the first ten convolutional layers remained fixed during training while all other layers are fine-tuned using the following procedure. We use distributed training on 32 GPUs (2 images per GPU, global batch size of 64) with SGD using a learning rate of $0.04$, weight decay $0.0001$ and momentum $0.9$. The learning rate is increased progressively from $0$ to $0.04$ during the first 5000 iterations and is divided by $10$ at $40$k iterations. The network is initially fine-tuned using this procedure on synthetic images, and we use the same procedure to train it a second time on both synthetic and real images if real images are available for the considered dataset.

**Pose estimation details.** For estimating the 6D pose of an object given the 2D detection, we use our coarse and refinement networks with $4$ refinement iterations. For the challenge, we improved the input pose in the coarse estimation stage that is used to initialize the alignment. Instead of using a fixed initial guess of the depth $z = 1$, we compute it using the size of the 2D detection such that the reprojection of the object in the canonical pose approximately matches the size and position of the 2D bounding box. This strategy is used for training and testing the coarse estimation network. The details of this pose initialization are given in the section 4.6.1 of the appendix.

**Training data.** For each dataset, we use the $50000$ photorealistic training images generated with the BlenderProc [33] physically-based renderer (PBR) that were provided for the challenge. These PBR training images are used for training the detection and pose estimation networks. On T-LESS, TUD-L and YCB-V datasets, real images are also available.

**Single-view RGB results.** We first consider the RGB-only category, where 6D detection is performed using individual RGB images of a scene. We present two variants of our method. **CosyPose-PBR** trains the detector, coarse estimation and refinement networks on synthetic images only. **CosyPose-Synt+Real** fine-tunes these models on the combination of synthetic PBR images and real images. For this second variant, we also used our syn-

| Method | Depth | AR | Time (s) |
|---|---|---|---|
| CosyPose-Synt+Real-ICP (Ours) | ✓ | **69.8** | 13.74 |
| Koenig Hybrid DL PointPairs [92] | ✓ | 63.9 | 0.633 |
| CosyPose-Synt+Real (Ours) | × | 63.7 | **0.449** |
| Pix2Pose + ICP | ✓ | 59.1 | 4.844 |
| CDPNv2 + ICP | ✓ | 56.8 | 1.462 |

Table 4.5: **Results of the 2020 6D object pose estimation (BOP) challenge for the RGB-D methods.** We report the AR score and running time averaged across the 7 core datasets.

thetic OpenGL images in addition to the PBR ones on T-LESS and YCB-V. We found that using only the PBR images is sufficient and the synthetic OpenGL images can be completely omitted when photorealistic rendered images are available. We present the results of the two variants (**CosyPose-PBR** and **CosyPose-Synt+Real**) and compare them with the closest competitors trained with similar images in Table 4.4. In both scenarios (real training images available or not) our approach outperforms its closest competitor with a margin of around 10%. A more detailed analysis on the impact of the quality of the synthetic images is provided in [71] which notably shows that using photorealistic images is crucial for the performance of the 2D detector, but less important for 6D pose estimation. The BOP challenge also evaluates the running time for 6D detection of all objects in a single-image. In our case, the running time for an image corresponds to computing 2D object detections and predicting the 6D pose of each detection, without using ground truth information of the number of objects in the scene. As shown in Table 4.4, our method is also significantly faster than its closest competitors. We present qualitative results of CosyPose-Synt+Real in figure 4-4 on each of the 7 datasets.

**Single-view RGB-D results.** While our method focuses on the RGB-only setting, images of the 7 core datasets used in the BOP challenge are captured with depth sensors. The results of our single-view method can be improved by running a post-processing refinement step on each 6D pose estimate. This refinement step leverages the measured object 3D points using a simple strategy based on iterative closest point (ICP) refinement. The RGB-only 6D pose estimation of the object is used as an initialization to ICP and the target point cloud of each of the object is extracted from the full depth image using the instance segmentation mask predicted by Mask R-CNN. We report results for this approach, **CosyPose-Synt+Real-ICP**, in Table 4.5 where we compare it with the closest competitors that also use the depth information. Our method achieves a significant 5.9 improvement on the AR metric over the second best RGB-D method. One drawback of our RGB-D approach is its running time which could be significantly improved by using a more efficient ICP implementation like the one used by [92] whose running time remains below a second despite also relying on ICP. Notably, our RGB-only method achieves a score similar to the best RGB-D method while being faster.

**Multi-view RGB results.** While the BOP challenge focuses on single-view pose estimation, we also ran our full multi-view pipeline (2D detection, single-view 6D pose estimation, multi-view multi-object pose estimation) on 3 datasets for which multiple views of each scenes are available: YCB-V, T-LESS and HB. For our multi-view approach, we use

| Method | Avg. | T-LESS | YCB-V | HB | Time (s) per image | Time (s) total |
|---|---|---|---|---|---|---|
| Ours - 1 view | 73.5 | 72.8 | 82.1 | 65.6 | **0.449** | **0.449** |
| Ours - 4 views | 77.9 | 80.1 | 84.0 | 69.6 | 0.518 | 2.07 |
| Ours - 8 views | **82.3** | **83.9** | **85.3** | **74.6** | 0.561 | 4.49 |

Table 4.6: RGB-only multi-view results of our full approach (2D detection + 6D pose estimation + multi-view multi-object reconstruction). We report the AR score on the BOP challenge datasets for which multiple views of each scene are available. The same set of hyper-parameters is used across all datasets.

the hyperparameters presented in section 4.4.2 that are fixed across the 3 datasets. We consider using 4 or 8 views for estimating the state of a scene. Results are presented in Table 4.6. Using the same hyper-parameters as in section 4.4.2 (where different 2D detections and different 6D pose estimation models are used), our multi-view framework again demonstrates significant improvements over the single-view baseline, including on the recent HomebrewDB dataset. We provide examples of recovered 6D object poses in figure 4-5 where we show both object candidates and the final estimated scenes on the three multi-view datasets.

## 4.5 Conclusion

We have developed an approach, dubbed CosyPose, for recovering the 6D pose of multiple known objects viewed by several non-calibrated cameras. Our main contribution is to combine learnable 6D pose estimation with robust multi-view matching and global refinement to reconstruct a single consistent scene. Our approach explicitly handles object symmetries, does not require depth measurements, is robust to missing and incorrect object hypotheses, and automatically recovers the camera poses and the number of objects in the scene. These results make a step towards the robustness and accuracy required for visually driven robotic manipulation in unconstrained scenarios with moving cameras, and open-up the possibility of including object pose estimation in an active visual perception loop.

The main limitation of this approach toward its deployment in real robotic scenarios is that the single-view 6D pose estimation approach must be trained on the objects it will be tested on. Training pose estimation networks for a novel object requires to (i) generate synthetic data and (ii) train the networks for this specific object. Synthetic data generation and training takes hours and requires access to 32 GPUs. We address this limitation in chapter 5, where we design an approach that can directly be applied to novel objects unseen during training.

## 4.6 Appendix

The appendix of this chapter is organized as follows. In section 4.6.1, we give additional details of our single-view single-object 6D object pose estimator. In section 4.6.2 we il-

lustrate the object candidate matching strategy on a simple 2D example. In section 4.6.3, we give additional details about our parametrization and initialization of the object-level bundle adjustment problem, introduced in section 4.3.4. Section 4.6.4 provides additional details on metrics used on YCB-Video and T-LESS. Finally, in section 4.6.5 we present additional qualitative results of our multi-view multi-object 6D pose estimation approach. We discuss in detail some examples to illustrate the key benefits of our method as well as point out the main limitations. Additional qualitative results on the YCB-V and T-LESS datasets are available on the project webpage [26].

### 4.6.1 Our single-view single-object method

We now give additional details on our single-view single-object pose estimation method introduced in section 4.3.2.

**Rotation parametrization.** Given two vectors $e_1$ and $e_2$ (6 values) predicted by the neural network, we recover a rotation parametrization $R$ by following [246]:

$$e'_1 = \frac{e_1}{||e_1||_2} \tag{4.16}$$

$$e'_3 = \frac{e'_1 \times e_2}{||e_2||_2} \tag{4.17}$$

$$e'_2 = e'_3 \times e'_1, \tag{4.18}$$

where $\times$ is the cross product between two 3D vectors. This representation has been shown to be better than quaternions (used by DeepIM [112]) to regress with a neural network [246].

**Cropping strategy.** DeepIM uses (a) the input 2D detections and (b) the bounding box defined by $T^k_{CO}$ and the vertices of the object $l$ to define the size and location of the crop in the real input image during training. Indeed, the ground truth bounding box is known during training. At test time, only (b) is used by DeepIM because ground truth bounding boxes are not available. In our case, we only use (b) while training and testing. The intrinsic parameters of the cropped camera are also used to directly render the cropped synthetic image at a resolution of $320 \times 240$ instead of rendering at a larger resolution followed by cropping.

**Pose initialization in the 6D object pose estimation (BOP) challenge.** Let $u_{det} = (u_{det,x}, u_{det,y})$ and $\Delta u_{det} = (\Delta u_{det,x}, \Delta u_{det,y})$ define the center and the size, respectively, of the approximate 2D detection of the object in the image, provided by the detector. The orientation of the object is set parallel to the axes of the camera with the $z$ axis pointing upwards. The center of the object model $O$ is set to match the center of the bounding box $u_{det}$. We make the first hypothesis of the depth by setting $z^{guess}_{CO} = 1\text{m}$ and use this initial value to estimate the coordinates $x$ and $y$ of the object center in the camera frame:

$$x^{guess}_{CO} = u_{det,x} \frac{z^{guess}_{CO}}{f^C_x}, \tag{4.19}$$

$$y^{guess}_{CO} = u_{det,y} \frac{z^{guess}_{CO}}{f^C_y}, \tag{4.20}$$

where $f_x^C, f_y^C$ denote the known focal lengths of the virtual cropped camera, and $x_{CO}^{guess}, y_{CO}^{guess}, z_{CO}^{guess}$ are the components of an initial guess of the 3D translation of the object with respect to the camera. We then update the depth estimate $z_{CO}^{guess}$ using the following strategy. We project the points of the object using the initial guess. These points define a bounding box with dimensions $\Delta u_{guess,x} = (\Delta u_{guess,x}, \Delta u_{guess,y})$ and the center of the bounding box remains unchanged $u_{guess} = u_d$ by construction. We compute an updated depth of the object such that it's width and height approximately match the size of the 2D detection:

$$z_{CO}^0 = z_{CO}^{guess} \frac{1}{2} \left( f_x \frac{\Delta u_{guess,x}}{\Delta u_{d,x}} + f_y \frac{\Delta u_{guess,y}}{\Delta u_{d,y}} \right) \tag{4.21}$$

and use this new depth to compute $x_{CO}^0$ and $y_{CO}^0$ using equations (4.19) and (4.20). The 3D rotation of the object $R_{CO}^0$ is set such that the object model's $z$ axis points upwards and has all axes parallel to the axes of the camera. The initial 6D pose $T_{CO}^k$ of the object is fully defined by $(x_{CO}^0, y_{CO}^0, z_{CO}^0)$ and $R_{CO}^0$.

### 4.6.2 Object candidate matching: additional illustration

In figure 4-6, we illustrate our method for "Sampling of relative camera poses sampling" described in section 4.3.3 with a simple 2D example.

### 4.6.3 Scene refinement

**Initialization.** There are multiple ways to initialize the optimization problem defined in equation (4.15). We use the following procedure. We start by picking a random camera and setting it's coordinate frame as the world coordinate frame. Then, we iterate over all cameras, trying to initialize each one. In order to initiliaze a camera $a$, we randomly sample another camera $b$ which is already initialized (placed in the world coordinate frame) and use the relative pose between these two cameras $T_{C_aC_b}$ estimated while running RANSAC (relative camera pose sampling in section 4.3.4) to place camera $a$ in the world coordinate frame. Once all the cameras have been initialized, we initalize objects by randomly picking an object $p$ and initializing it using a candidate associated with this physical object from a random view.

**Rotation parametrization.** We use the same rotation parametrization as the one used for our single-view single-object network for which the equations are provided in section 4.6.1 of this appendix.

### 4.6.4 Metrics

In this section, we give some details about the metrics reported in the main chapter. We refer to [66, 70, 230] for more information about these metrics.

The ADD (average distance) metric is introduced in [66] and is typically used to measure the accuracy of pose estimation for non-symmetric objects. Given a label $l$ of an object and following the notation introduced in section 4.3.2, this metric is computed as :

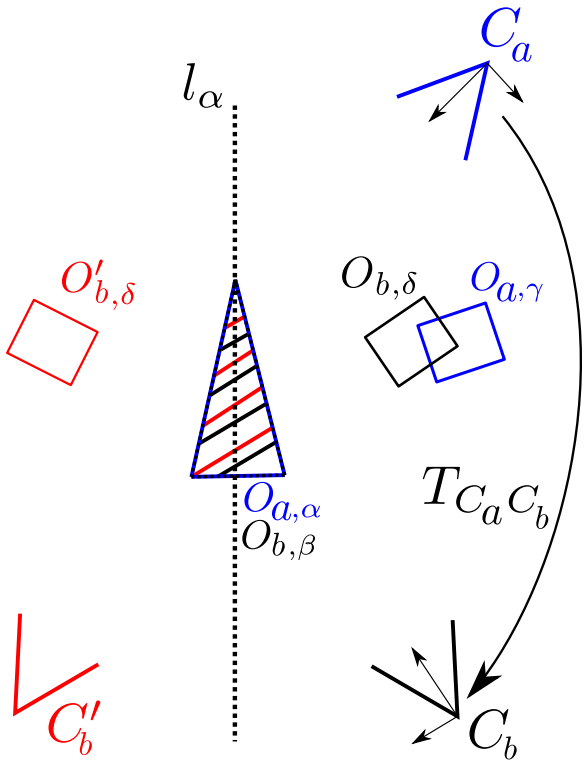



Figure 4-6: **Relative camera pose estimation.** Given two pairs of object candidates $(O_{a,\alpha}, O_{b,\beta})$ and $(O_{a,\gamma}, O_{b,\delta})$, we estimate the relative camera pose $T_{C_aC_b}$ that best aligns candidates $O_{a,\gamma}$, $O_{b,\delta}$. In this example, the red camera pose $C'_b$ is also valid due to the symmetries of the triangular object $l_\alpha$. It is discarded because the error between $O'_{b,\delta}$ and $O_{a,\gamma}$ is bigger than between $O_{b,\delta}$ and $O_{a,\gamma}$.

$$\text{ADD}(l, T, \hat{T}) = \frac{1}{H_l} \sum_h ||\hat{T} X_l^h - T X_l^h||_2, \tag{4.22}$$

where $T$ is the predicted object pose, $\hat{T}$ is the ground truth pose, $X_l^h$ are the vertices of the 3D models and $H_l$ is the number of vertices of the model of the object $l$.

For symmetric objects, the average distance is computed using the closest point distance and noted ADD-S:

$$\text{ADD-S}(l, T, \hat{T}) = \frac{1}{H_l} \sum_h \min_g ||\hat{T} X_l^h - T X_l^g||_2. \tag{4.23}$$

The notation ADD(-S) corresponds to computing ADD for non symmetric objects and ADD-S for symmetric objects. It is also common to report the percentage of objects for which the pose is estimated within a given threshold such as 10% of it's diameter. We use the notations ADD-S $<$ 0.1d and ADD(-S) $<$ 0.1d for this metric and report the mean computed over object types.

The authors of PoseCNN [230] also proposed to report the area under the accurracy-threshold curve for a threshold (on ADD-S, or ADD(-S)) varying between 0 to 10cm. We note this metric as AUC of ADD(-S) or AUC of ADD-S and we use the implementation provided with the evaluation code[1] of YCB-Video.

When evaluating on the T-LESS dataset, we also report the Visual Surface Discrepancy metric (vsd). This metric is invariant to object symmetries and takes into account the visibility of the object. As in [155, 199], the pose is considered correct when the error is less than 0.3 with $\tau = 20mm$ and $\delta = 15mm$. We note this metric $e_{\text{vsd}} < 0.3$ and use the official implementation code of the BOP challenge [70][2]. There are multiple instances of objects in multiple scenes of the T-LESS dataset. When comparing with prior work [155, 199] on all images of the primesense camera, we only evaluate the prediction which has the highest detection score for each class, and only objects visible more than 10% are considered as ground truth targets. This corresponds to the SiSo task.

When evaluating our multi-view method, we follow the more recent 6D localization protocol of the ViVo BOP challenge which considers the top-$m$ predictions with highest score for each class in each image, where $m$ is the number of ground truth objects of the class in the scene.

When computing the mean of ADD-S errors in our scene refinement ablation, we only consider as true positives predictions the ones which have an ADD-S error lower than half of the diameter of the object, to ensure that the prediction is matched to the correct ground truth object. Without limiting the error to this threshold and using only class labels and scores, some predictions may be matched to ground truth objects which are at a very different location in the scene. This tends to increase the errors while not being representative only of the 6D pose accuracy of the predictions.

[1] https://github.com/yuxng/YCB_Video_toolbox
[2] https://github.com/thodan/bop_toolkit

### 4.6.5 Additional multi-view multi-object results

Each scene reconstruction is presented with a dedicated figure and we provide close-ups on various parts of the visualization to illustrate the different aspects in detail. The explanation is provided in the caption of each figure.

**Layout of the figures.** In each figure presented below, four (on T-LESS) or five (on YCB-Video) RGB images were used to reconstruct each scene. In each figure, each row corresponds to results associated with one image and different columns present the results of different stages of our method. The last column shows the ground truth scene. The different columns are described next.

- "Input image" is the (RGB) image used as input to the method.
- "2D detections" shows the detections obtained by the object detector (RetinaNet on T-LESS, PoseCNN on YCB-Video), after removing detections that have scores below $0.3$. The color of each 2D bounding box illustrates the object label predicted for this detection, each color is associated with a unique type of 3D object in the object database. Note that the colors for each type of 3D object are shared for all visualizations corresponding to one scene (one figure) but not shared across the figures because of the high number of objects in the database.
- "Object candidates" illustrates the 6D object poses predicted for each 2D detection. The candidates considered as outliers (those who have not been matched with a candidate from another view and are discarded) are marked with red color and are transparent. The candidates considered inliers are shown in green. Inliers are used in the final scene reconstruction. Note that the red and green colors in this (3rd) column are only used to indicate inliers and outliers and there is no correspondence with red and green colors in the 4th column that denote the different object types.
- "Scene reconstruction" illustrates the scene reconstructed by our method using all the views presented in the figure. Once the scene is reconstructed, we use the recovered 6D poses of physical objects and cameras to render the scene imaged from each of the predicted viewpoints. The renderings are overlaid over the input image.
- "Ground truth" corresponds to the ground truth scene viewed from the ground truth viewpoints. These images are shown to enable visual comparison with the results of our method. The ground truth information (number of objects, types of objects, poses of cameras, poses of objects) is not used by our method.

In the following, we illustrate the main capabilities of our system.

#### 4.6.5.1 Highlights of the capabilities of our system

**Large number of objects, robustness to occlusions, symmetric objects.** Our method is able to recover the state of complex scenes that contain multiple objects, even if parts or the scene are partially or completely occluded in some of the views. The poses of cameras and objects can be correctly recovered even if all objects in the scene are symmetric. An

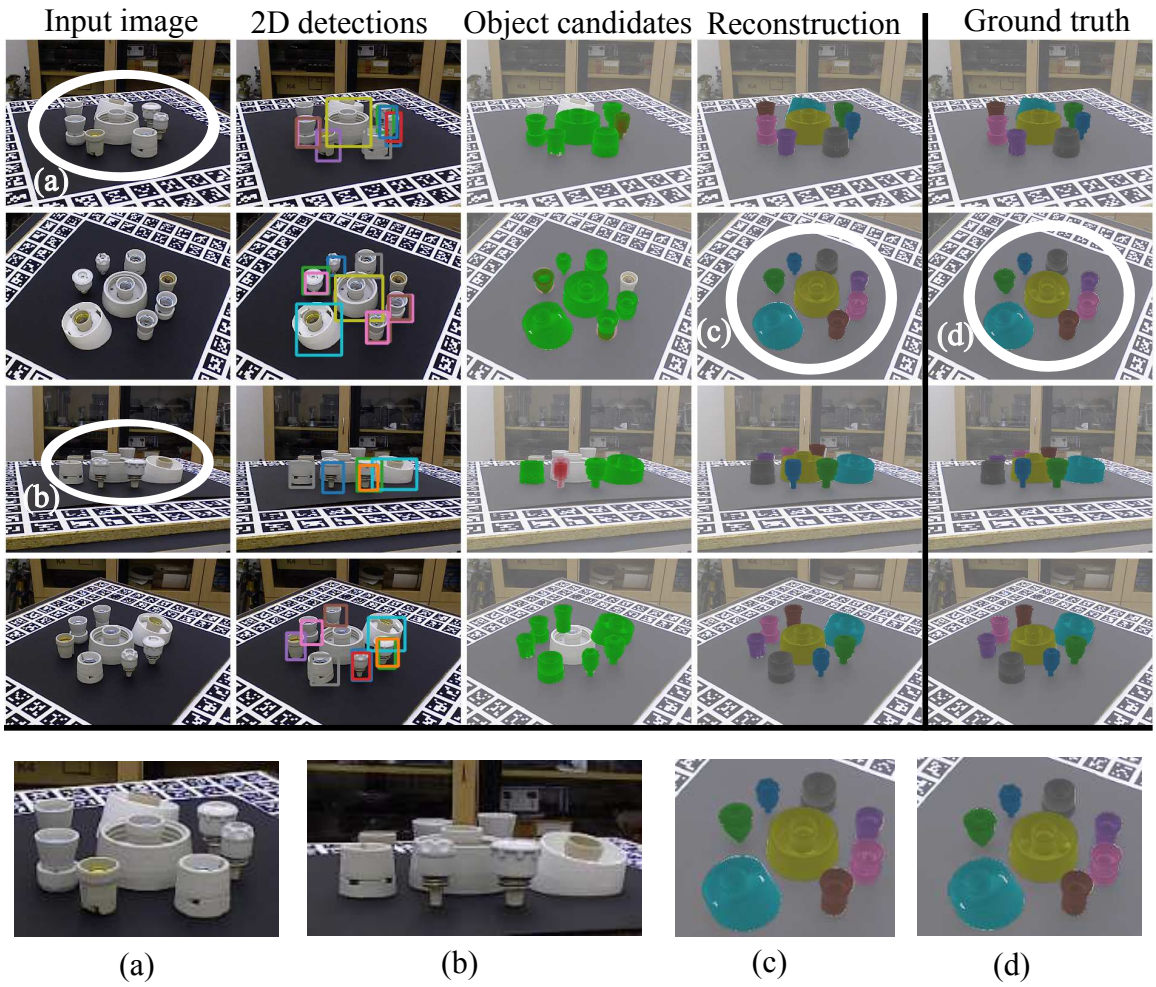



Figure 4-7: **Highlight I: Scene with many symmetric objects and occlusions.** Our method is able to correctly identify and predict the poses of the 8 symmetric objects present in the scene. Please note how object poses and labels/colors are similar in the output of our method, shown in close-up (c), and the ground truth, shown in close-up (d). This is particularly challenging because of the high object density, varying level of occlusions and the fact that all objects of the scene are symmetric, as shown in close-ups (a) and (b).

example is presented in figure 4-7. Note how some objects are missing in each individual view but our method is able to recover correctly all objects.

**Multiple object instances.** Our method is able to successfully identify the correct number of objects and their labels even if there are multiple objects of the same type in the image, objects are partially occluded in some views and multiple types of objects have very similar visual appearance. An example is presented in figure 4-8

**Cluttered scenes with distractors.** Our method is also robust to distractor objects that are not in the database of objects. We present in figure 4-9 a complex example with many distractors where our method is able to successfully recover all objects in the scene, which are in the object database while filtering out the other ones. This is especially important for robotic applications in unstructured environments where the objects of interests are known and should not be confused with other background objects.

**High accuracy.** One of the key components of our approach is scene refinement (section 4.3.4 in the main chapter), which significantly improves the accuracy of pose predictions using information from multiple views. In Fig 4-10, we show an example of a reconstruction that highlights the accuracy that can be reached by our method using only 4 input images.

#### 4.6.5.2 Detailed examples

We now explain in detail few simpler examples that demonstrate how our system works and how it achieves the kind of results presented in the previous section.

**Robustness to missing detections.** In some situations, objects are partially or completely occluded in some of the views. As a result, 2D detections for one physical object are missing in some views. If this physical object is visible in other views, our reconstruction method is able to estimate it's pose with respect to the other objects. If all cameras can be positioned with respect to the rest of the scene using other non-occluded objects, our approach can also position the partially occluded object with respect to all cameras, even if there were initially no candidates corresponding to the object in these views. An example is shown in figure 4-11.

**Robustness to incorrect detections.** In T-LESS, many objects have similar visual appearance. As a result, the 2D detector often makes mistakes, predicting incorrect labels for some of the detections in some views. Our method is able to handle multiple 2D detections that have different labels at the same location in the image. In this case, a pose hypothesis is generated for each of the label hypothesis. If the object candidate cannot be matched with another view - either because the incorrect label is predicted in only one view or because the poses are not consistent - our method is able to discard this object candidate. An example is shown in figure 4-12. Please see the discussion "Duplicate objects" and figure 4-13 for examples where an object is consistently mis-identified across multiple views.

**Duplicate objects.** When multiple objects share the same visual appearance as it is the case in the T-LESS dataset, there are often multiple label hypotheses that are consistent across views for the same physical object. Because these objects look similar to each other and match the observed image, the pose estimation network (which tries to match a rendering with the observed image, regardless of the object type) predicts reasonable poses for each label that are consistent across different views. These candidates are matched across views

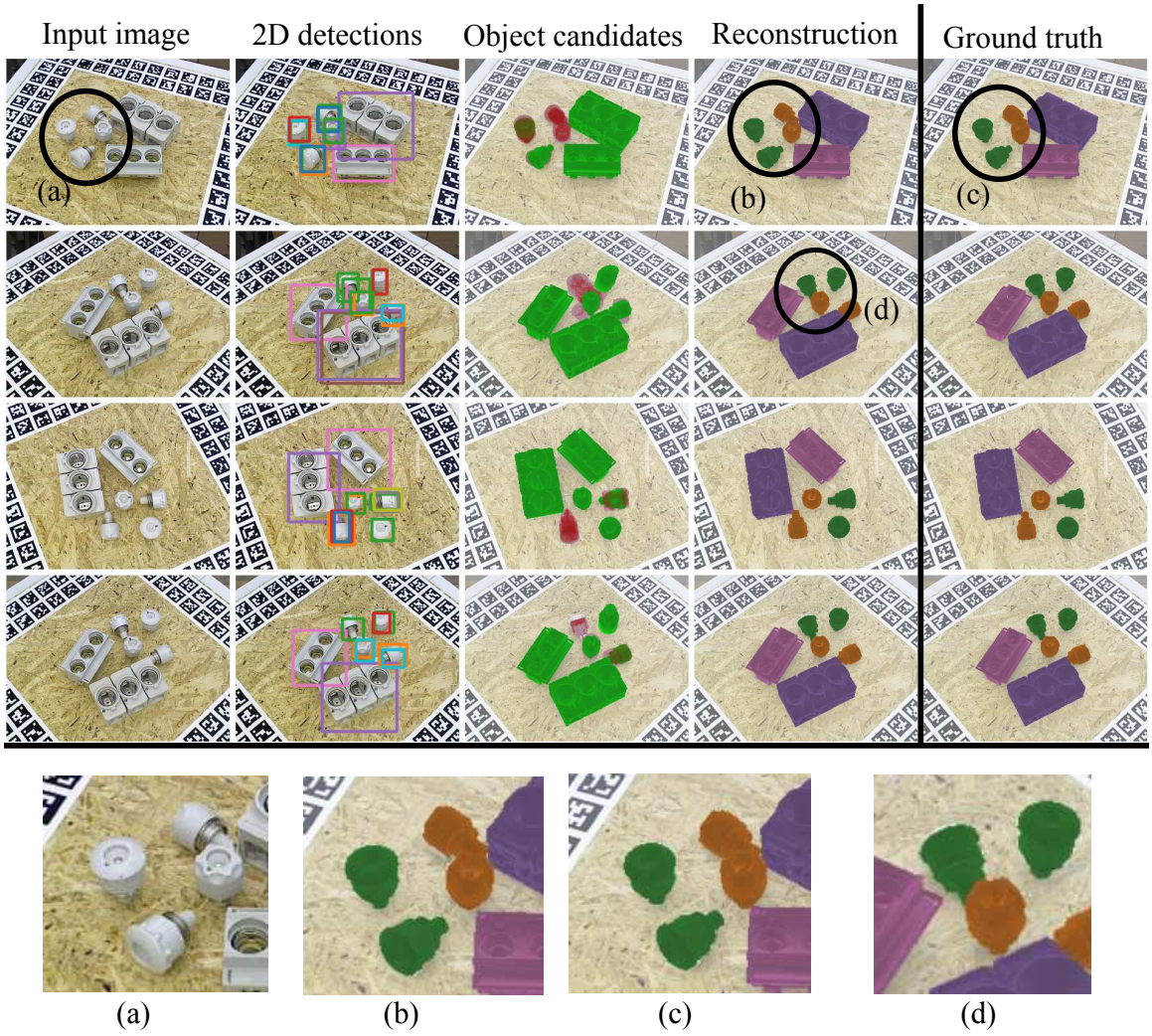



Figure 4-8: **Higlight II: Scene with multiple object instances of the same object type.** Note how our method is able to correctly identify all objects in this challenging scene. Object poses and labels/colors predicted by our method, shown in close-up (b) are very similar to the ground truth, shown in close-up (c). This is particularly challenging because the green and orange objects have similar visual appearance, are close to each other in the scene, and objects are partially occluded in some of the views, as shown in close-ups (a) and (d).

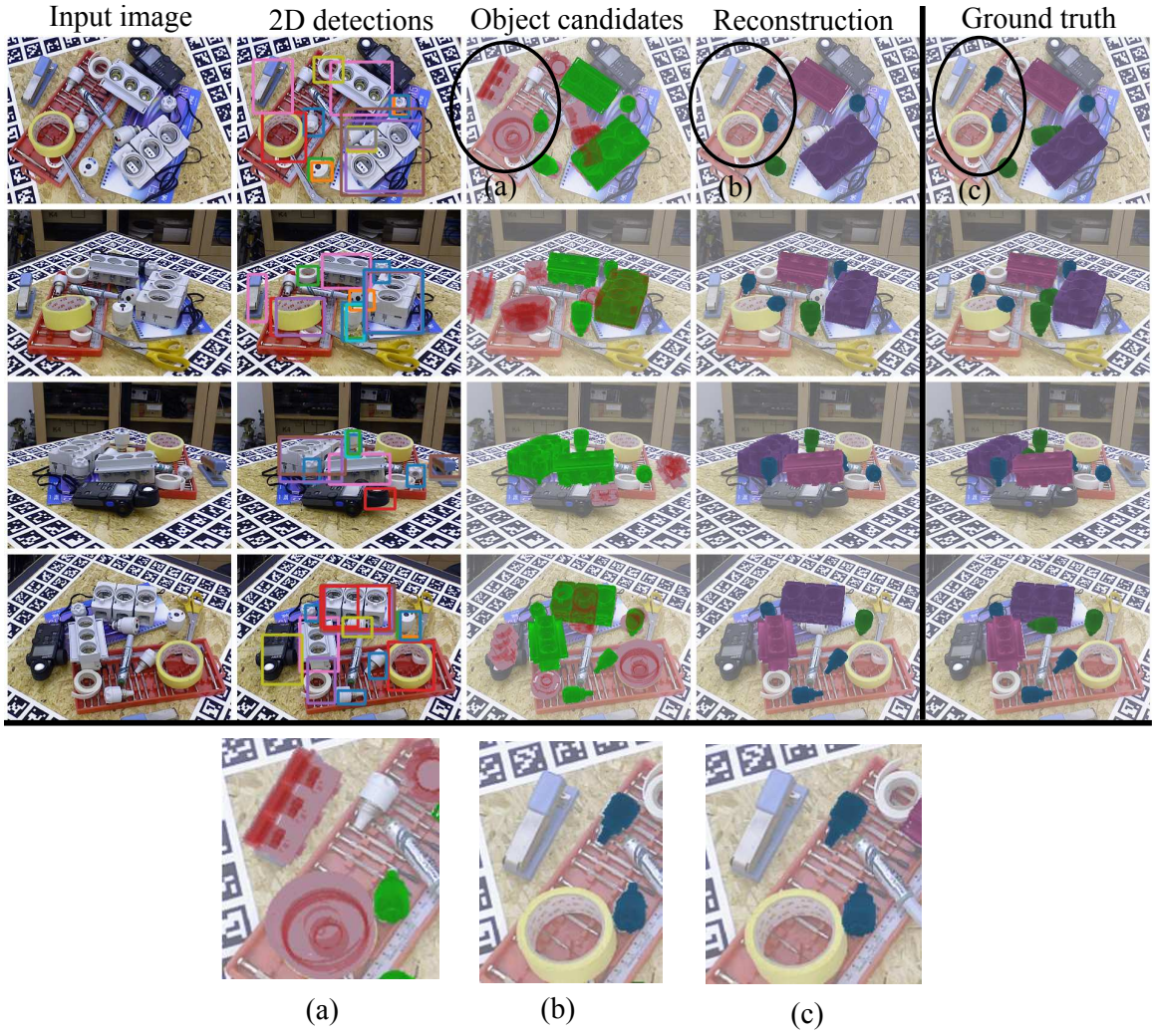



Figure 4-9: **Highlight III: Scene with multiple distractors.** Our method is also robust to distractor objects that are not in the database of objects. Our method correctly localizes and estimates the pose of all databse objects in the scene (cf. our reconstruction (4th column) and the ground truth (5th column)) despite the presence of several distractor objects (objects not colored in the ground truth). A single-view approach (Object candidates, 3rd column) incorrectly detects three of the distractor objects and places them in the scene because they look similar to some objects of the database, as shown in the close-up (a). Our robust multi-view approach is able to filter these outliers: the objects estimated at the positions of the distractors are marked in red in (a). Distractor objects have been filtered in the final reconstruction as shown in the close-up (b) (cf. ground truth close-up (c)).

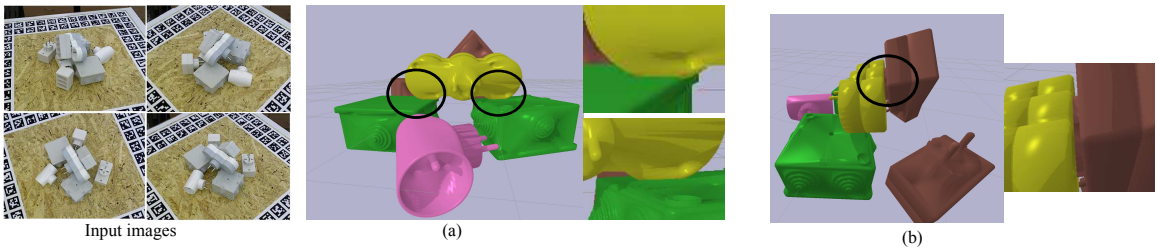



Figure 4-10: **Highlight IV: Accuracy of our approach.** Left: input images. Then (a) and (b) shows the output scene imaged from two viewpoints different from the views used for the reconstruction. Please note in (a) how the yellow object is accurately estimated to only touch the green objects, and in (b) how the brown object is correctly plugged inside the yellow object.

and multiple objects with different labels are predicted in the final scene at the same spatial position. In our visualization, we remove these duplicate objects by using a simple 3D non-maximum suppression (NMS) strategy on the estimated physical objects of the final scene. If multiple objects are too close to each other in the 3D scene, we keep the object with the highest score – the sum of the 2D detection scores of all inlier object candidates that are associated with one physical 3D object. Duplicate objects and 3D non-maximum suppression are illustrated in figure 4-13, including one correct and one incorrect example. The column "Reconstruction" in all figures corresponds to the output of our method *after* the 3D NMS.

**Robustness to distractors and false positives.** The complex scenes in the T-LESS dataset also have background distractor objects that are not in the object database. Some of these distractors look similar to objects in the database and can be incorrectly detected, sometimes in multiple images. In these cases, the pose estimator most often produces 6D pose estimates that are not consistent across views because the input real images are outside of the training distribution (they display objects that are not used to generate the training data). Because these estimates are not consistent across views, our method is able to filter them and mark them as outliers (red), thus gaining robustness with respect to these distractors. An example is shown in figure 4-14.

#### 4.6.5.3 Limitations

We now describe the most challenging scenarios that our method is currently not able to recover from. For each of these, we briefly discuss possible improvements.

**Limitation I: consistent mistakes** If two incorrect 6D object candidates are consistent across at least two views, an (incorrect) object will be present in the reconstructed scene. Such failure case typically happens when two viewpoints are similar to each other. An example is shown in figure 4-15. If two views are very similar, the incorrect candidates will be matched together. Note that this failure mode could be resolved by using a higher number of views, and by only considering physical objects that have a sufficiently high number of associated object candidates.

**Limitation II: Objects missing in the final reconstruction.** Our current approach requires that a candidate in one view is matched with at least one candidate from another

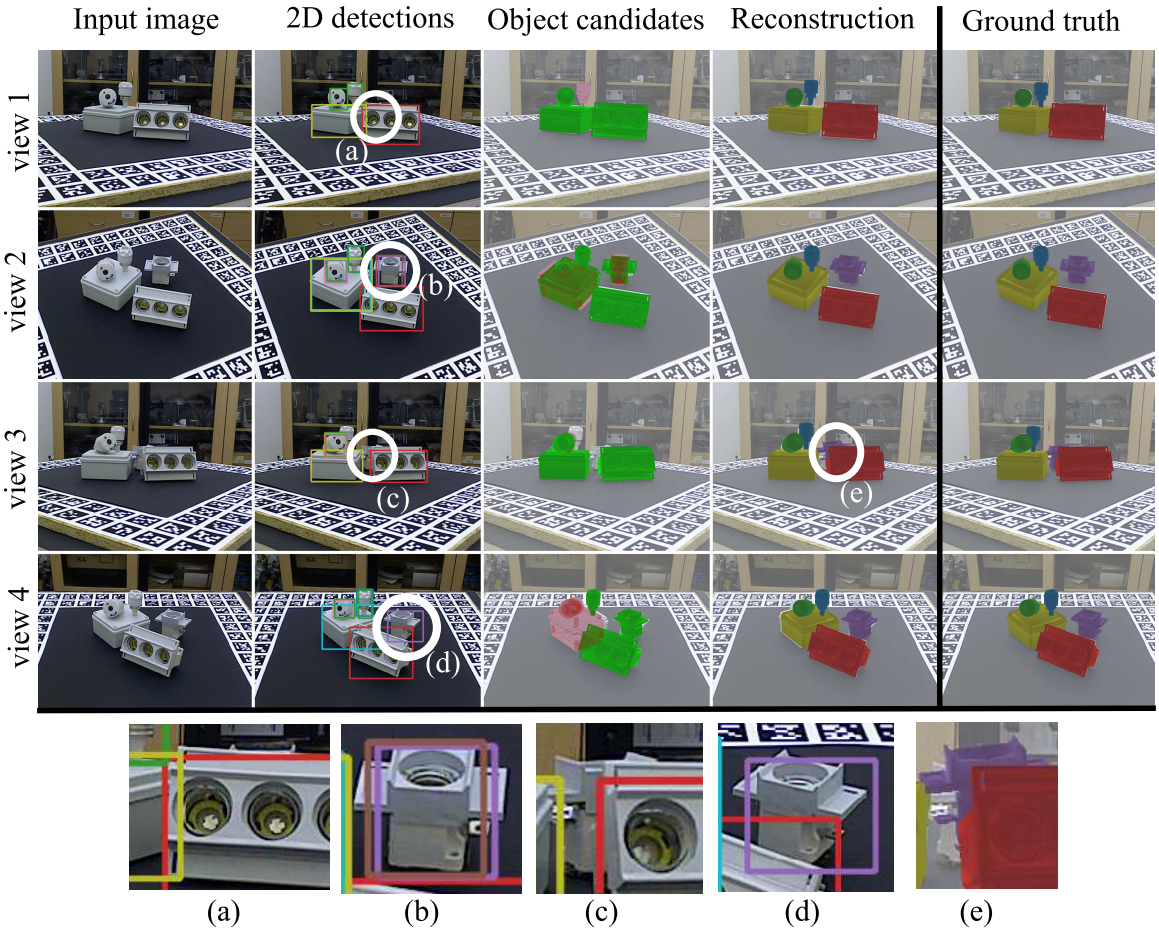



Figure 4-11: **Example I: robustness to missing detections.** One of the objects (marked by purple circle) in the scene is detected in two views (b) (d), but not in the other two views due to partial (c) or complete (a) occlusion. Our method is able to (i) position the views 1 and 3 with respect to the scene using the other visible candidate objects and (ii) position the purple object with respect to these other objects using views 2 and 4, where the purple object is visible. Once the scene is reconstructed, it is also possible to directly recover the pose of the purple object with respect to views, where it was not originally detected, like in (e).

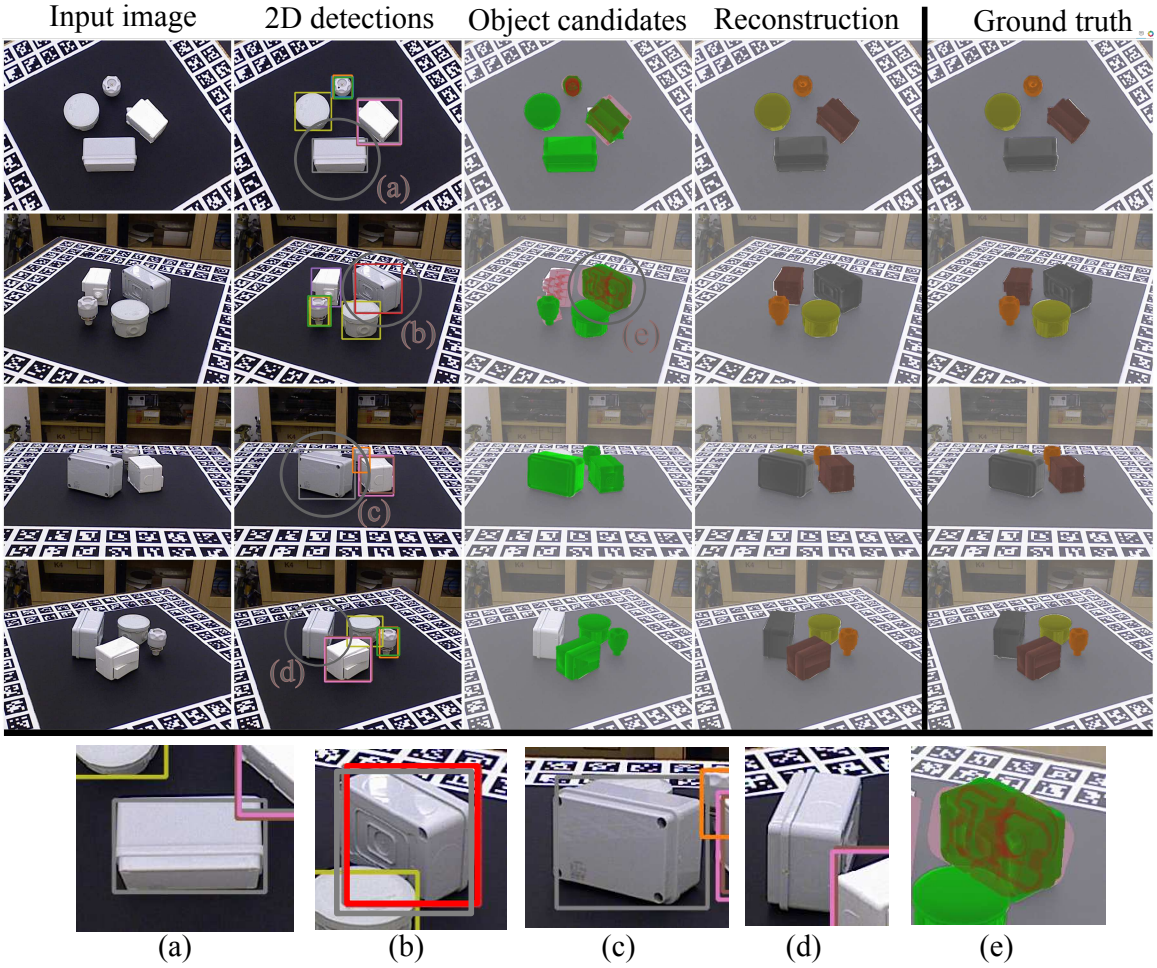



Figure 4-12: **Example II: Robustness to incorrect detection labels.** One of the objects that is correctly identified in two views (a) (c), has two label hypotheses in view (b) and is not detected in view (d). Our method keeps the two hypotheses in (b) and predicts two 6D object candidates (e) but it is able to discard one of them because it's label is not consistent with the other views: one of the two object candidates is marked as an outlier (red) in (e). In our final scene reconstruction, the gray object is correctly recognized (it has the same color (gray) in out output "Reconstruction" and in the "Ground truth").

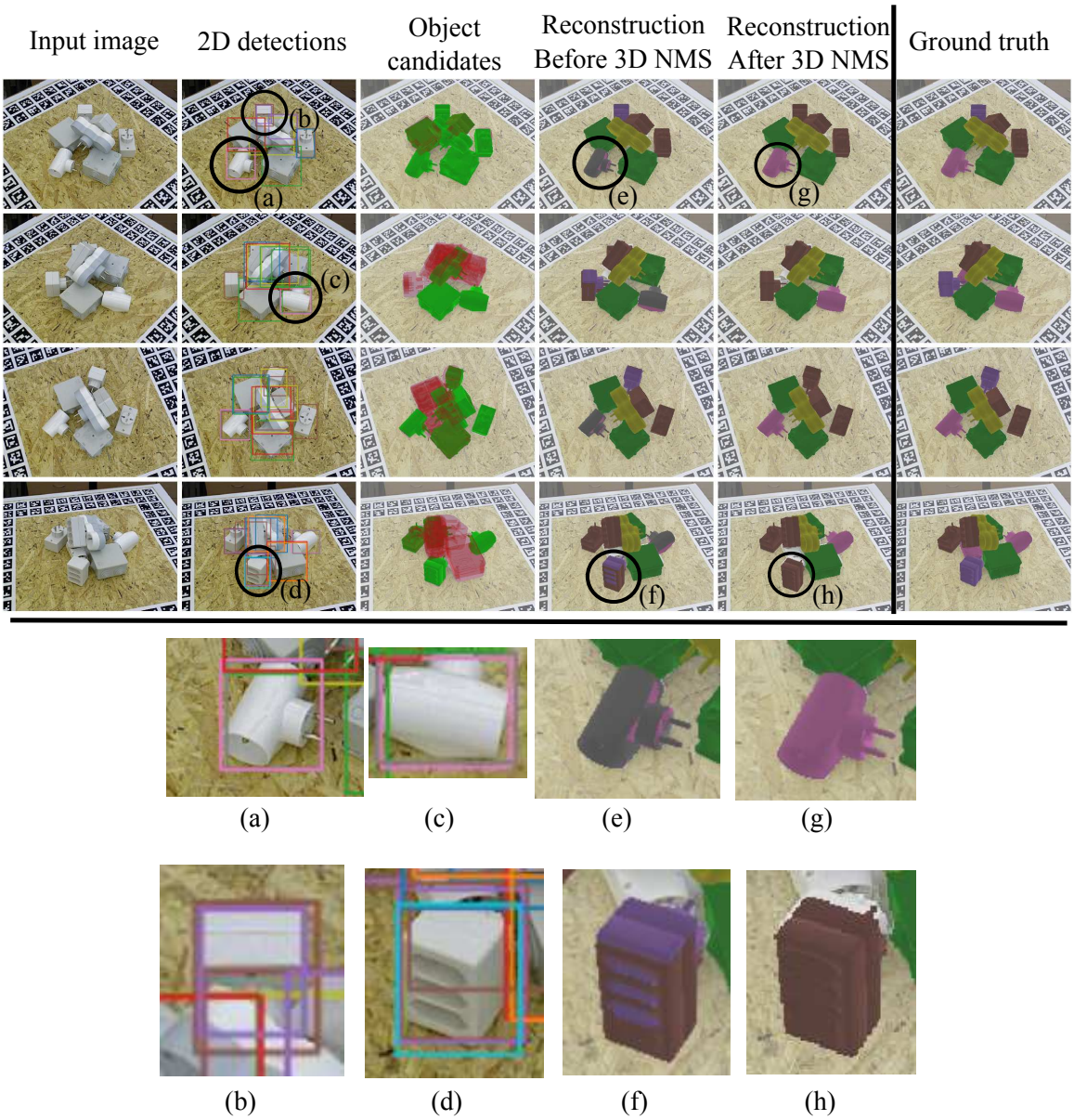



Figure 4-13: **Example III: Duplicate objects.** 2D detections with two different labels (grey and pink) are predicted for the same object consistently across two views, (a) and (c). Because the 3D models of the pink and grey objects are similar, the poses predicted in both views are consistent and thus both pairs of object candidates are associated to separate objects. In the final scene reconstruction, two objects (grey and pink) overlap at the same 3D location (e). We use a 3D non-maximum suppression strategy to retain only a single hypothesis. In the final output (after NMS), the correct object is retained (pink), c.f. the ground truth column. In some cases, incorrectly identified objects are kept as shown in (b), (d), (f), (h).

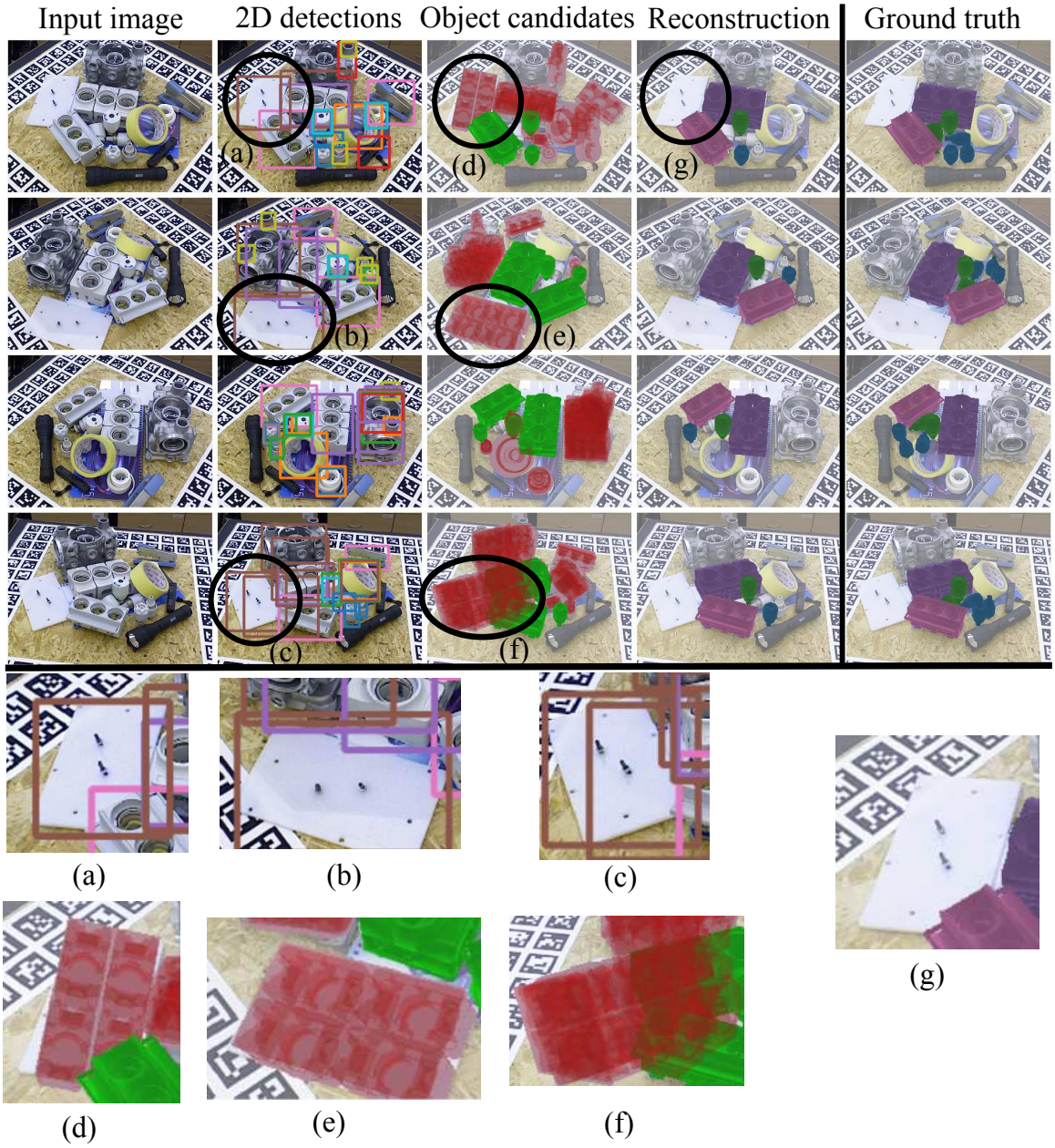



Figure 4-14: **Example IV: Robustness to false positive detections.** One of the distractor objects is incorrectly detected in three views, see close-up (a), (b) and (c), with a consistent label (brown). For each of these detections, a 6D object candidate is generated, see close-ups (d), (e) and (f), but the poses are inconsistent across views because the pose estimation network has not been trained for this object. These candidates are filtered by our robust candidate matching strategy and considered outliers (red), see (d), (e) and (f). Note how this distractor is not present in the final scene reconstruction, as shown in close-up (g).

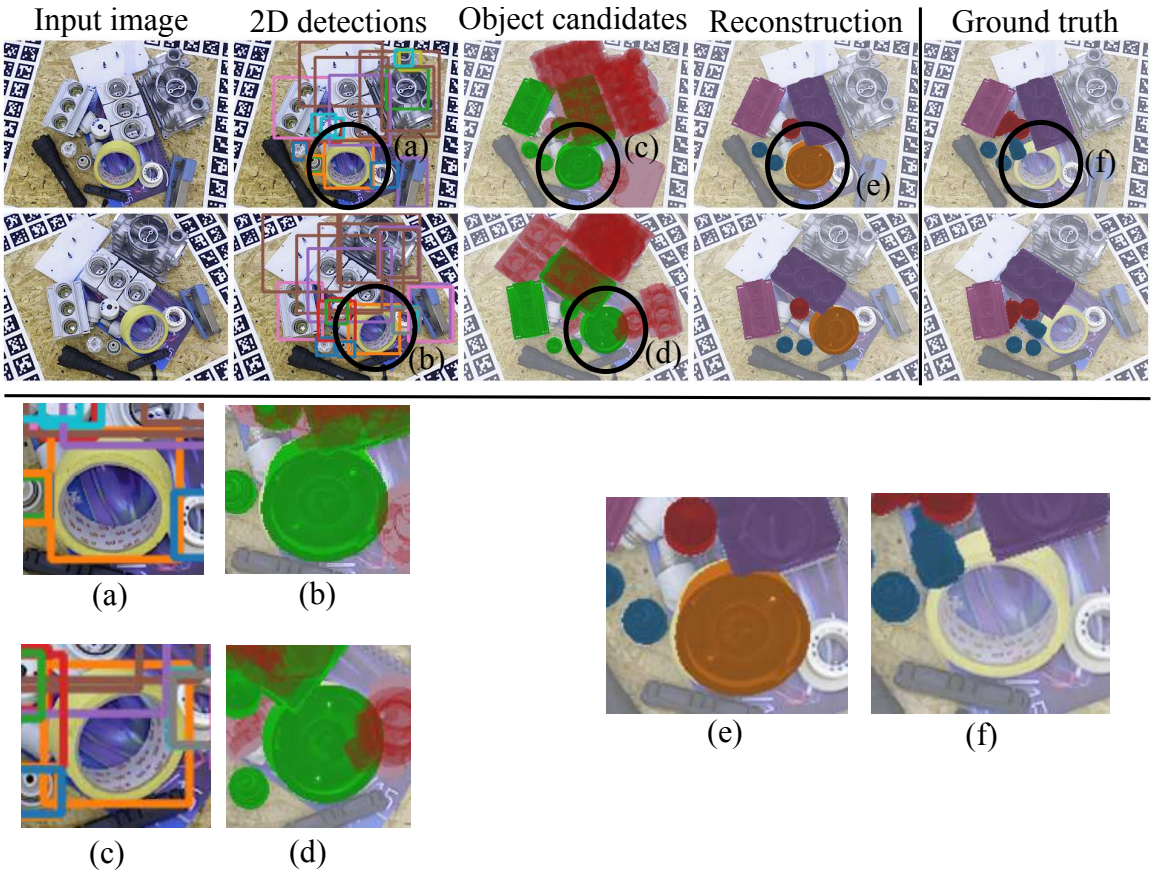



Figure 4-15: **Limitation I: Consistent mistakes.** One of the distractors is incorrectly detected as an orange object (from the object database), as shown in close-ups (a) and (c). The two viewpoints are quite similar and as a result the two estimated object poses are consistent, as shown in (c) and (d). The object is present in the final reconstruction (e) but it does not correspond to the ground truth object (f).

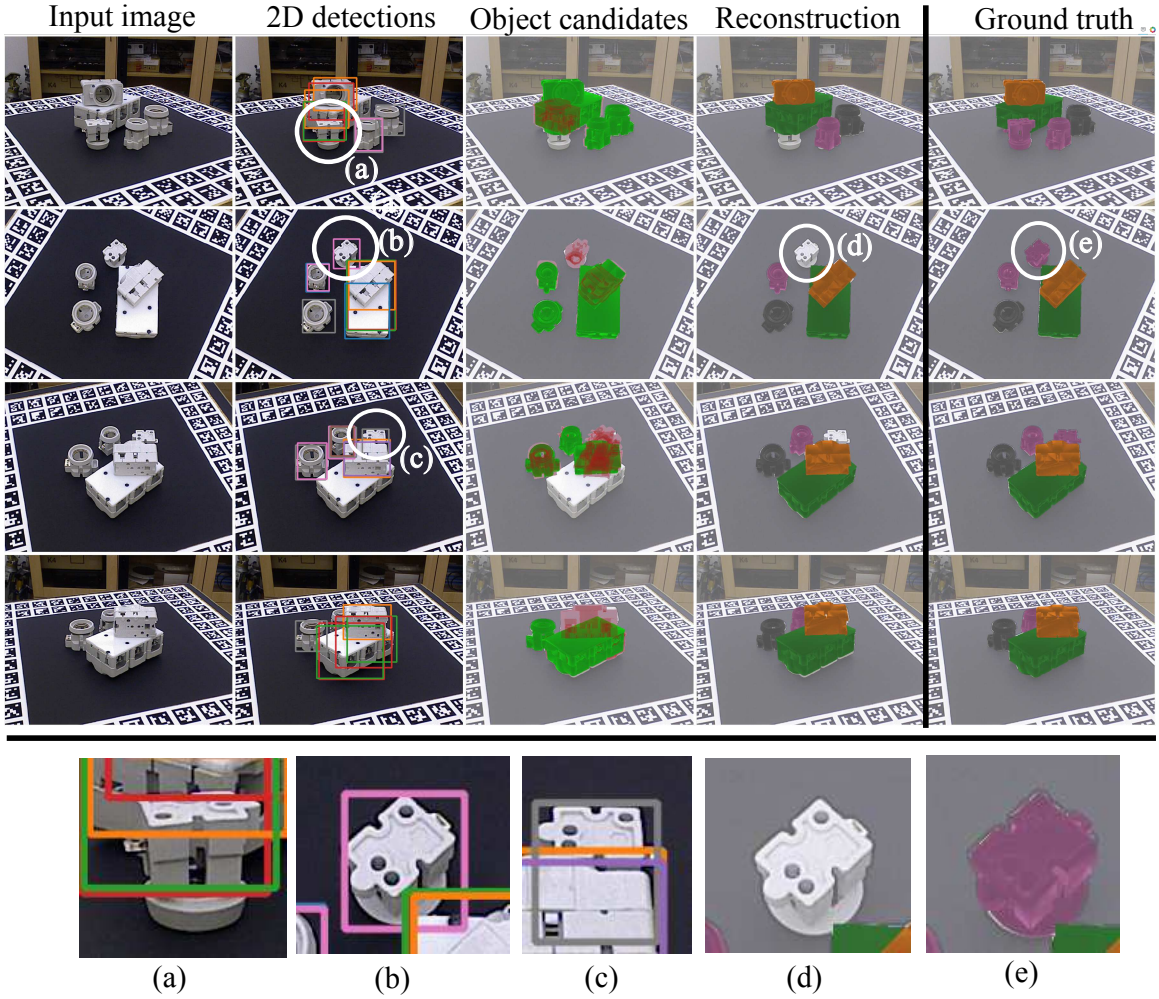



Figure 4-16: **Limitation II: missing objects.** An object is detected correctly in one view as shown in the close-up (b), but the detection is missing in other views, shown in close-up (a), or the detection is incorrect and inconsistent, as shown in close-up (c). The object candidate (b) cannot be matched with another candidate and thus is missing from the final reconstruction, as shown in close-up (d) of the output (cf. ground truth close-up (e)).

view. If a candidate detection and pose estimate is correct in one view but not in any other view, it will be missing from the final reconstruction. An example is presented in figure 4-16. Note that in this case, all camera poses are still estimated correctly. An interesting direction to overcome this problem would be to grow the number of object candidates in each view by reprojecting the detection from other views, as done in guided matching.

**Limitation III: Incorrect estimates of camera pose.** To position the camera with respect to the scene, our method requires that there are at least three object candidate inliers in the view: two for positioning the camera with respect to the scene, and another one to validate the camera pose hypothesis. Sometimes, however, there is an insufficient number of inliers. This typically happens if only two objects are visible, or if there is a small number of objects visible and some of the detections are incorrect. An example is shown in figure 4-17.

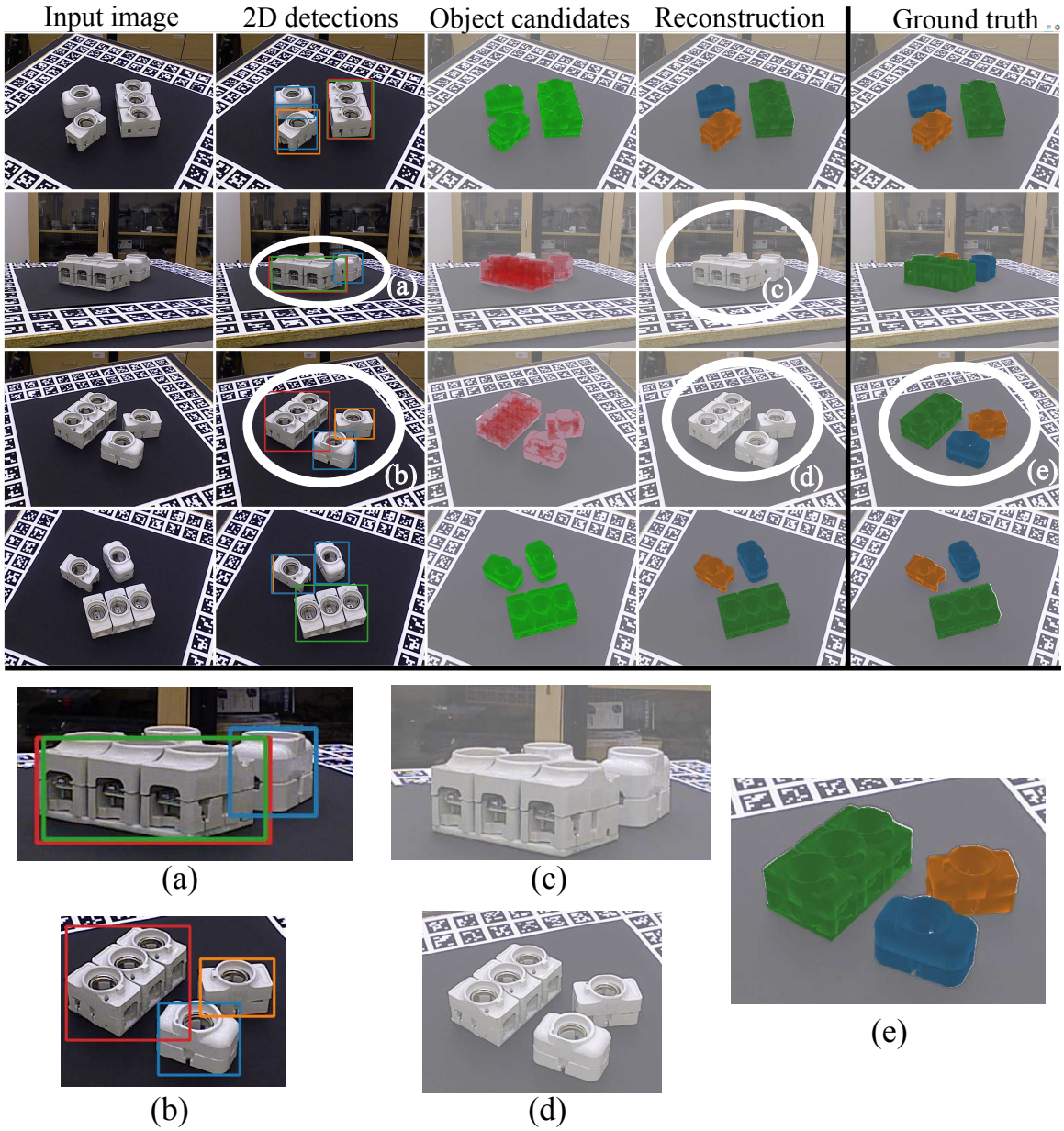



Figure 4-17: **Limitation III: incorrect estimates of camera pose**. If one view has only two visible objects, as shown in close-up (a), the corresponding camera view with respect to the rest of the scene cannot be estimated as it requires at least three correctly estimated objects. As a result the objects are not reprojected in the image (c). This also happens if three candidates are detected in one view, as shown in close-up (b), but one of the object candidates is not consistent with the other views (here red object instead of green object).

# Chapter 5

# Pose estimation of novel objects

In chapter 4, we have introduced an approach for estimating the 6D pose of objects with known CAD models using one multiple image as inputs. The method relies on a learning-based approach for single-view 6D object pose estimation. This method however must be trained on the objects it will be tested on. Deploying this method for novel objects requires costly synthetic data generation and training. In this chapter, we build on the results presented in chapter 4 and design an approach that can directly be applied to novel objects without retraining.

We introduce MegaPose, a method to estimate the 6D pose of novel objects, that is, objects unseen during training. At inference time, the method only assumes knowledge of (i) a region of interest displaying the object in the image and (ii) a CAD model of the observed object. The contributions of this work are threefold. First, we present a 6D pose refiner based on a render-and-compare strategy which can be applied to novel objects. The shape and coordinate system of the novel object are provided as inputs to the network by rendering multiple synthetic views of the object's CAD model. Second, we introduce a novel approach for coarse pose estimation which leverages a network trained to classify whether the pose error between a synthetic rendering and an observed image of the same object can be corrected by the refiner. Third, we introduce a large scale synthetic dataset of photorealistic images of thousands of objects with diverse visual and shape properties and show that this diversity is crucial to obtain good generalization performance on novel objects. We train our approach on this large synthetic dataset and apply it *without retraining* to hundreds of novel objects in real images from several pose estimation benchmarks. Our approach achieves state-of-the-art performance on the ModelNet dataset. An extensive evaluation on the 7 core datasets of the BOP challenge demonstrates that our approach achieves performance competitive with existing approaches that require access to the target objects during training.

## 5.1 Introduction

Accurate 6D object pose estimation is essential for many robotic and augmented reality applications. Current state-of-the-art methods are learning-based [55, 71, 96, 119, 218] and require 3D models of the objects of interest at both training and test time. These

methods require hours (or days) to generate synthetic data for each object and train the pose estimation model. They thus cannot be used in the context of robotic applications where the objects are only known during inference (e.g. CAD models are provided by a manufacturer or reconstructed [36]), and where rapid deployment to novel scenes and objects is key.

The goal of this work is to estimate the 6D pose of novel objects, *i.e.*, objects that are only available at *inference time* and are not known in advance during training. This problem presents the challenge of generalizing to the large variability in shape, texture, lighting conditions, and severe occlusions that can be encountered in real-world applications. Some prior works [19, 110, 118, 128, 129, 219, 221] have considered category-level pose estimation to partially address the challenge of novel objects by developing methods that can generalize to novel object instances of a known class (e.g. mugs or shoes). These methods however do not generalize to object instances outside of training categories. Other methods aim at generalizing to any novel instances regardless of their category [112, 121, 145, 149, 154, 185, 197, 234]. These works present important technical limitations. They rely on non-learning based components for generating pose hypotheses [149] (e.g. PPF [39]), for pose refinement [185] (e.g. PnP [108] and ICP [10, 243]), for computing photometric errors in pixel space [154], or for estimating the object depth [145, 197] (e.g. using only the size of a 2D detection [87]). These components however inherently cannot benefit from being trained on large amount of data to gain robustness with respect to noise, occlusions, or object variability. Learning-based methods also have the potential to improve as the quality and size of the datasets improve.

Pipelines for 6D pose estimation of known (not novel) objects that consist of multiple learned stages [96, 119] have shown excellent performance on several benchmarks [71] with various illumination conditions, textureless objects, cluttered scenes and high levels of occlusions. We take inspiration from [96, 119] which split the problem into three parts: (i) 2D object detection, (ii) coarse pose estimation, and (iii) iterative refinement via render & compare. We aim at extending this approach to novel objects unseen at training time. The detection of novel objects has been addressed by prior works [135, 152, 185, 233] and is outside the scope of this chapter. In this work, we focus on the coarse and refinement networks for 6D pose estimation. Extending the paradigm from [96] presents three major challenges. First, the pose of an object depends heavily on both its visual appearance and choice of coordinate system (defined in the CAD model of the object). In existing refinement networks based on render & compare [96, 112], this information is encoded in the network weights during training, leading to poor generalization results when tested on novel objects. Second, direct regression methods for coarse pose estimation are trained with specific losses for symmetric objects [96], requiring that object symmetries be known in advance. Finally, the diversity of shape and visual properties of the objects that can be encountered in real-world applications is immense. Generalizing to novel objects requires robustness to properties such as object symmetries, variability of object shape, and object textures (or absence of).

**Contributions.** We address these challenges and propose a method for estimating the pose of any novel object in a single RGB or RGB-D image, as illustrated in Figure 5-1. First, we propose a novel approach for 6D pose refinement based on render & compare which enables generalization to novel objects. The shape and coordinate system of the novel object

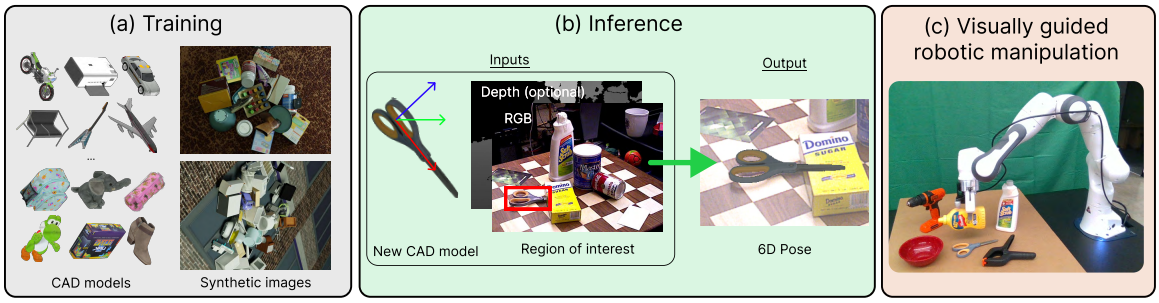



Figure 5-1: **MegaPose** is a 6D pose estimation approach (a) that is trained on millions of synthetic scenes with thousands of different objects and (b) can be applied *without re-training* to estimate the pose of any novel object, given a CAD model and a region of interest displaying the object. It can thus be used to rapidly deploy visually guided robotic manipulation systems in novel scenes containing novel objects (c).

are provided as inputs to the network by rendering multiple synthetic views of the object's CAD model. Second, we propose a novel method for coarse pose estimation which does not require knowledge of the object symmetries during training. The coarse pose estimation is formulated as a classification problem where we compare renderings of random pose hypotheses with an observed image, and predict whether the pose can be corrected by the refiner. Finally, we leverage the availability of large-scale 3D model datasets to generate a highly diverse synthetic dataset consisting of 2 million photorealistic [33] images depicting over 20K models in physically plausible configurations. The code, dataset and trained models are available on the project page [134].

We show that our novel-object pose estimation method trained on our large-scale synthetic dataset achieves state-of-the-art performance on ModelNet [112, 229]. We also perform an extensive evaluation of the approach on hundreds of novel objects from all 7 core datasets of the BOP challenge [71] and demonstrate that our approach achieves performance competitive with existing approaches that require access to the target objects during training.

## 5.2 Related work

In this section, we first review the literature on 6D pose estimation of known rigid objects. We then focus on the practical scenario similar to ours where the objects are not known prior to training.

**6D pose estimation of known objects**. Estimating the 6D pose of rigid objects is a fundamental computer vision problem [123, 124, 172] that was first addressed using correspondences established with locally invariant features [9, 22, 23, 39, 124] or template matching [63, 82]. These have been replaced by learning-based methods with convolutional neural networks that directly regress sets of sparse [76, 87, 158, 160, 169, 202, 207] or dense [55, 155, 160, 191, 230, 240] features. All these approaches use non-learning stages relying on PnP+Ransac [54, 108] to recover the pose from correspondences in RGB images, or variants of the iterative closest point algorithm, ICP [10, 243], when depth is

available. The best performing methods rely on trainable refinement networks [96, 112, 119, 217] based on render & compare [112, 127, 148, 156]. These methods render a single image of the object, which is not sufficient to provide complete information on the shape and coordinate system of a 3D model to the network. This information is thus encoded in the networks weights when training, which leads to poor generalization when tested on novel objects unseen at training. Our approach renders multiple views of an object to provide this 3D information, making the trained network independent of these object-specific properties.

**6D pose estimation of novel objects.** Other works consider a practical scenario where the objects are not known in advance. Category-level 6D pose estimation is a popular problem [19, 110, 118, 128, 129, 219, 221] in which CAD models of test objects are not known, but the objects are assumed to belong to a known category. These methods rely on object properties that are common within categories (*e.g.* handle of a mug) to define and estimate the object pose, and thus cannot generalize to novel categories. Our method requires the 3D model of the novel object instance to be known during inference, but does not rely on any category-level information. Other works address a scenario similar to ours. [5, 145, 197, 231, 233, 234] only estimate the 3D orientation of novel objects by comparing rendered pose hypotheses with the observed image using features extracted by a network. They rely on handcrafted [145, 197] or learning-based DeepIM [234] refiners to recover accurate 6D poses. We instead propose a method that estimates the full 6D pose of the object and show our refinement network significantly outperforms DeepIM [112] when tested on novel object instances. The closest works to ours are OSOP [185] and ZePHyR [149]. OSOP focuses on the coarse estimation by explicitly predicting 2D-2D correspondences between a single rendered view of the object and the observed image, and solves for the pose using PnP or Kabsch [10] which makes inference slower and less robust compared to directly predicting refinement transforms with a network as done in our solution. ZePHyR [149] strongly relies on the depth modality, whereas our approach can also be used in RGB-only images. Finally, [60, 121, 154, 196] investigate using a set of real reference views of the novel object instead of using a CAD model. These approaches have only reported results on datasets with limited or no occlusions. Our use of a deep render & compare network trained on a large-scale synthetic dataset displaying highly occluded object instances enables us to handle highly cluttered scenes with high occlusions like in the LineMOD Occlusion, HomebrewedDB or T-LESS datasets.

## 5.3 Method

In this section we present our framework for pose estimation of novel objects. Our goal is to detect the pose $\mathcal{T}_{CO}$ (the pose of object frame $O$ expressed in camera frame $C$ composed of 3D rotation and 3D translation) of a novel object given an input RGB (or RGBD) image, $I_o$, and a 3D model of the object. Similar to DeepIM [112] and CosyPose [96], our method consists of three components (1) object detection, (2) coarse pose estimation and (3) pose refinement. Compared to these works, our proposed method enables generalization to novel objects not seen during training, requiring novel approaches for the coarse model, the refiner and the training data. Our approach can accept either RGB or RGBD

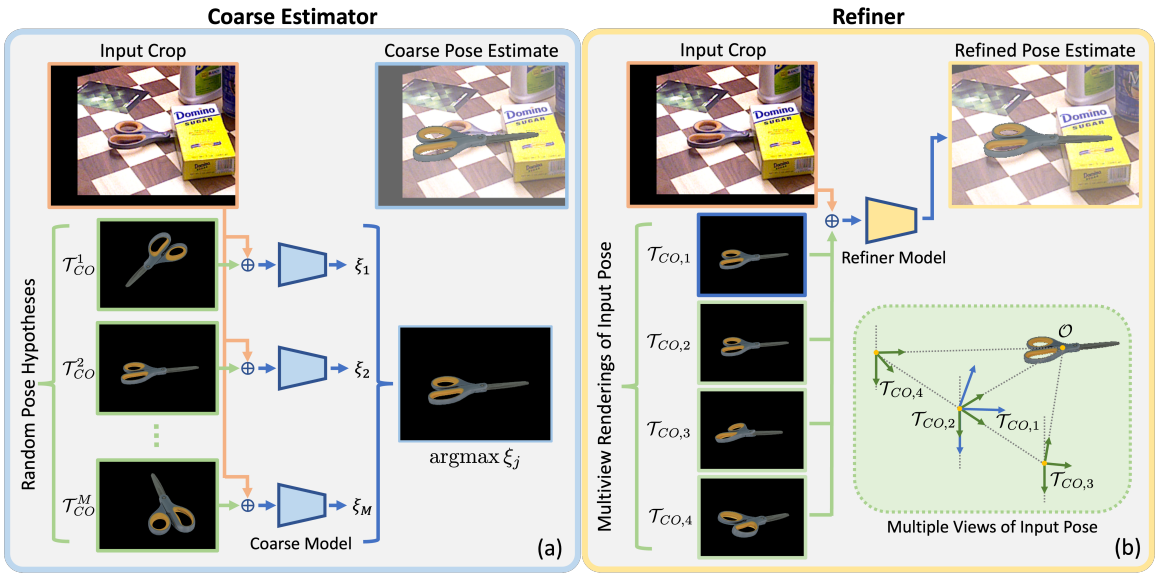



Figure 5-2: $\oplus$ denotes concatenation. **(a) Coarse Estimator:** Given a cropped input image the coarse module renders the object in multiple input poses $\{\mathcal{T}_{CO}^j\}$. The coarse network then classifies which rendered image best matches the observed image. **(b) Refiner:** Given an initial pose estimate $\mathcal{T}_{CO}^k$ the refiner renders the objects at the estimated pose $\mathcal{T}_{CO,1} \coloneqq \mathcal{T}_{CO}^k$ (blue axes) along with 3 additional viewpoints $\{\mathcal{T}_{CO,i}\}_{i=2}^4$ (green axes) defined such that the camera $z$-axis intersects the anchor point $\mathcal{O}$. The refiner network consumes the concatenation of the observed and rendered images and predicts an updated pose estimate $\mathcal{T}_{CO}^{k+1}$.

inputs, if depth is available the RGB and D images are concatenated before being passed into the network. Detection of novel-objects in an image is an interesting problem that has been addressed in prior work [121, 151, 152, 185, 233] but lies outside the scope of this chapter. Thus for our experiments we assume access to an object detector, but emphasize that our method can be coupled with any object detector, including zero-shot methods such as those in [151, 152].

### 5.3.1 Technical Approach

**Coarse pose estimation.** Given an object detection, shown in Figure 5-1(b), the goal of the coarse pose estimator is to provide an initial pose estimate $\mathcal{T}_{CO,\text{coarse}}$ which is sufficiently accurate that it can then be further improved by the refiner. In order to generalize to novel-objects we propose a novel classification based approach that compares observed and rendered images of the object in a variety of poses and selects the rendered image whose object pose best matches the observed object pose.

Figure 5-2(a) gives an overview of the coarse model. At inference time the network consumes the observed image $I_o$ along with rendered images $\{I_r(\mathcal{T}_{CO}^j)\}_{j=1}^M$ of the object in many different poses $\{\mathcal{T}_{CO}^j\}_{j=1}^M$. For each pose $\mathcal{T}_{CO}^j$ the model predicts a score $(I_o, I_r(\mathcal{T}_{CO}^j)) \rightarrow \xi_j$ that classifies whether the pose hypothesis is within the basin of at-

traction of the refiner. The highest scoring pose $\mathcal{T}_{CO}^{j^*}, j^* = \text{argmax}_j \, \xi_j$ is used as the initial pose for the refinement step. Since we are performing classification, our method can implicitly handle object symmetries, as multiple poses can be classified as correct.

**Pose refinement model.** Given an input image and an estimated pose, the refiner predicts an updated pose estimate. Starting from a coarse initial pose estimate $\mathcal{T}_{CO,\text{coarse}}$ we can iteratively apply the refiner to produce an improved pose estimate. Similar to [96, 112] our refiner takes as input observed $I_o$ and rendered images $I_r(\mathcal{T}_{CO}^k)$ and predicts an updated pose estimate $\mathcal{T}_{CO}^{k+1}$, see Figure 5-2 (b), where $k$ refers to the $k^{th}$ iteration of the refiner. Our pose update uses the same parameterization as DeepIM [112] and CosyPose [96] which disentangles rotation and translation prediction. Crucially this pose update $\Delta\mathcal{T}$ depends on the choice of an *anchor point* $\mathcal{O}$, see section 5.6.1 of the appendix for more details. In prior work [96, 112] which trains and tests on the same set of objects, the network can effectively learn the position of the anchor point $\mathcal{O}$ for each object. However in order to generalize to novel objects we must enable the network to infer the anchor point $\mathcal{O}$ at inference time.

In order to provide information about the anchor point to the network we always render images $I_r(\mathcal{T}_{CO}^k)$ such that the anchor point $\mathcal{O}$ projects to the image center. Using rendered images from multiple distinct viewpoints $\{\mathcal{T}_{CO,i}\}_{i=1}^N$ the network can infer the location of the anchor point $\mathcal{O}$ as the intersection point of camera rays that pass through the image center, see Figure 5-2(b).

Additional information about object shape and geometry can be provided to the network by rendering depth and surface normal channels in the rendered image $I_r$. We normalize both input depth (if available) and rendered depth images using the currently estimated pose $\mathcal{T}_{CO}^k$ to assist the network in generalizing across object scales, see section 5.6.3 of the appendix for more details.

**Network architecture.** Both the coarse and refiner networks consists of a ResNet-34 backbone followed by spatial average pooling. The coarse model has a single fully-connected layer that consumes the backbone feature and outputs a classification logit. The refiner network has a single fully-connected layer that consumes the backbone feature and outputs 9 values that specify the translation and rotation for the pose update.

### 5.3.2 Training Procedure

**Training data.** For training, both the coarse and refiner models require RGB(-D)[1] images with ground-truth 6D object pose annotations, along with 3D models for these objects. In order for our approach to generalize to novel-objects we require a large dataset containing diverse objects. All of of our methods are trained purely on synthetic data generated using BlenderProc [33]. We generate a dataset of 2 million images using a combination of ShapeNet [229] (abbreviated as SN) and Google-Scanned-Objects (abbreviated as GSO) [36]. Similar to the BOP [70] synthetic data, we randomly sampled objects from our dataset and dropped them on a plane using a physics simulator. Materials, background textures, lighting and camera positions are randomized. Example images can be seen in Figure 3-1(a) and in the section 5.6.5 of the appendix. Some of our ablations also use the synthetic

[1]Our method can consume either RGB or RGB-D images depending on the input modalities that are available.

training datasets provided by the BOP challenge [70]. We add data augmentation similar to CosyPose [96] to the RGB images which was shown to be a key to successful sim-to-real transfer. We also apply data augmentation to the depth images as explained in section 5.6.5 of the appendix.

**Refiner model.** The refiner model is trained similarly to [96]. Given an image with an object $\mathcal{M}$ at ground-truth pose $\mathcal{T}^*_{CO}$ we generate a perturbed pose $\mathcal{T}'_{CO}$ by applying a random translation and rotation to $\mathcal{T}^*_{CO}$. Translation is sampled from a normal distribution with a standard deviations of $(0.02, 0.02, 0.05)$ centimeters and rotation is sampled as random Euler angles with a standard deviation of $15$ degrees in each axis. The network is trained to predict the relative transformation between the initial and target pose. Following [96, 112] we use a loss that disentangles the prediction of depth, $x$-$y$ translation, and rotation. See the section 5.6.2 appendix for more details.

**Coarse model.** Given an input image $I_o$ of an object $\mathcal{M}$ and a pose $\mathcal{T}'_{CO}$ the coarse model is trained to classify whether pose $\mathcal{T}'_{CO}$ is within the basin of attraction of the refiner. In other words, if the refiner were started with the initial pose estimate $\mathcal{T}'_{CO}$ would it be able to estimate the ground-truth pose via iterative refinement? Given a ground-truth pose-annotation $\mathcal{T}^*_{CO}$ we randomly sample poses $\mathcal{T}'_{CO}$ by adding random translation and rotation to $\mathcal{T}^*_{CO}$. The positives are sampled from the same distribution used to generate the perturbed poses the refiner network is trained to correct (see above), and other poses sufficiently distinct to this one (see section 5.6.4 of the appendix for more details) are marked as negatives. The model is then trained with binary cross entropy loss.

## 5.4 Experiments

We evaluate our method for 6D pose estimation of novel objects using the seven challenging datasets of the BOP [70, 71] 6D pose estimation benchmark, and the ModelNet [112] dataset. The dataset and the standard 6D pose estimation metrics we use are detailed in Section 5.4.1. In all our experiments, the objects are considered novel, i.e. they are only available during inference on a new image and they are not used during training. In Section 5.4.2, we evaluate the performance of our approach composed of coarse and refinement networks. Notably, we show that (i) our method is competitive with others that require the object models to be known in advance, and (iii) our refiner outperforms current state-of-the-art on the ModelNet and YCB-V datasets. Section 5.4.3 validates our technical contributions and shows the crucial importance of the training data in the success of our method. Finally, we discuss the limitations in Section 5.4.4.

### 5.4.1 Dataset and metrics

We consider the seven core datasets of the BOP challenge [70, 71]: LineMod Occlusion (LM-O) [66], T-LESS [68], TUD-L [70], IC-BIN [35], ITODD [38], HomebrewedDB (HB) [84] and YCB-Video (YCB-V) [230]. These datasets exhibit 132 different objects in cluttered scenes with occlusions. These objects present many factors of variation: textured or untextured, symmetric or asymmetric, household or industrial (e.g. watcher pitcher, stapler, bowls, multi-socket plug adaptor) which makes them representative of objects that

| | Pose Initialization | | Pose Refinement | | | BOP Datasets | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Method | Novel objects | Method | Novel objects | RGB-D Input | LM-O | T-LESS | TUD-L | IC-BIN | ITODD | HB | YCB-V | Mean |
| 1 | CosyPose [96] | ✗ | CosyPose | ✗ | | 63.3 | 64.0 | 68.5 | 58.3 | 21.6 | 65.6 | 57.4 | 57.0 |
| 2 | SurfEmb [55] | ✗ | BFGS | ✗ | | 66.3 | 73.5 | 71.5 | 58.8 | 41.3 | 79.1 | 64.7 | 65.0 |
| 3 | SurfEmb [55] | ✗ | BFGS+ICP | ✓ | ✓ | *75.8* | *82.8* | *85.4* | *65.6* | *49.8* | *86.7* | *80.6* | *75.2* |
| 4 | OSOP [185] | ✓ | Multi-Hyp. | ✓ | | 31.2 | - | - | - | - | 49.2 | 33.2 | - |
| 5 | OSOP [185] | ✓ | MH+ICP | ✓ | ✓ | 48.2 | - | - | - | - | 60.5 | 57.2 | - |
| 6 | (PPF, Sift) + Zephyr [149] | ✓ | - | ✓ | ✓ | **59.8** | - | - | - | - | - | 51.6 | - |
| 7 | (PPF, Sift) + Our coarse | ✓ | Our refiner | ✓ | ✓ | 57.0 | - | - | - | - | - | 62.3 | - |
| 8 | CosyPose [96] | ✗ | – | | | 53.6 | 52.0 | 57.6 | 53.0 | 13.1 | 33.5 | 33.3 | 42.3 |
| 9 | CosyPose [96] | ✗ | Ours | ✓ | | 65.5 | 72.0 | 70.1 | 57.3 | 28.4 | 67.0 | 56.8 | 59.6 |
| 10 | CosyPose [96] | ✗ | Ours | ✓ | ✓ | 71.2 | 63.8 | 85.0 | 55.1 | 39.9 | 73.2 | 69.2 | 66.0 |
| 11 | Ours | ✓ | – | | | 18.7 | 19.7 | 20.5 | 15.3 | 8.00 | 18.6 | 13.9 | 16.2 |
| 12 | Ours | ✓ | Ours | ✓ | | 53.7 | **62.2** | 58.4 | **43.6** | 30.1 | 72.9 | 60.4 | 54.5 |
| 13 | Ours | ✓ | Ours | ✓ | ✓ | 58.3 | 54.3 | **71.2** | 37.1 | **40.4** | **75.7** | **63.3** | **57.2** |

Table 5.1: **Results on the BOP challenge datasets.** We report the AR score on each of the 7 datasets considered in the BOP challenge and the mean score across datasets. With the exception of Zephyr (row 11), all approaches are trained purely on synthetic data. For each column, we denote the best over result in *italics* and the best novel-object pose estimation method in **bold**.

are typically encountered in robotic scenarios. The ModelNet dataset depicts individual instances of objects from 7 classes of the ModelNet [229] dataset (bathtub, bookshelf, guitar, range hood, sofa, tv stand and wardrobe). We use initial poses provided by adding noise to the ground truth, similar to previous works [112, 154, 197]. The focus is on refining these inital poses. We follow the evaluation protocol of [71] for BOP datasets, and of DeepIM [112] for ModelNet.

### 5.4.2 6D pose estimation of novel objects

**Performance of coarse+refiner.** Table 5.1 reports results of our novel-object pose estimation method on the BOP datasets. We first use the detections and pose hypotheses provided by a combination of PPF and SIFT, similar to the state-of-the-art method Zephyr [149]. For each object detection, these algorithms provide multiple pose hypotheses. We find the best hypothesis using the score of our coarse network, and apply 5 iterations of our refiner. Results are reported in row 7. On YCB-V, our method achieves a +10.7 AR score improvement compared to Zephyr (row 6). Averaged across the YCB-V and LM-O datasets, the AR score of our approach is 59.7 compared to 55.7 for Zephyr (row 6). Next, we provide a complete set of results using the detections from Mask-RCNN networks. Please note that since detection of novel objects is outside the scope of this chapter, we use the networks trained on the synthetic PBR data of the target objects [71] which are publicly available for each dataset. We report the results of our coarse estimation strategy (Table 5.1, row 11), and after running the refiner network, on RGB (row 12) or RGB-D (row 13) images. We observe that (i) our refinement network significantly improves the coarse estimates (+41.0 mean AR score for our RGBD refiner) and (ii) the performance of both models is com-

| Method | RGB-D | Average Recall (5°, 5cm) | ADD (0.1d) | Proj2D (5px) |
|---|---|---|---|---|
| DeepIM [112] | ✓ | 64.3 | 83.6 | 73.3 |
| Multi-Path [197] | | 84.8 | 90.1 | 81.6 |
| LatentFusion[154] | ✓ | 85.5 | 94.3 | 94.7 |
| Ours | | 88.6 | 90.5 | 88.9 |
| Ours | ✓ | **97.6** | **98.9** | **97.5** |

Table 5.2: **Evaluation of the refiner on the ModelNet [112] dataset.** The mean average recall is computed over the seven classes of the dataset.

petitive with the learning-based refiner of CosyPose [96] (row 1) while not requiring to be trained on the test objects. The recent SurfEmb [55] performs better than our approach, but heavily relies on the knowledge of the objects for training and cannot generalize to novel objects.

**Performance of the refiner.** We now focus on the evaluation of our refiner which can be used to refine arbitrary initial poses. Our refiner is the only learning-based approach in Table 5.1 which can be applied to novel objects. In rows 9 and 10, we apply our refiner to the coarse estimates of CosyPose [96] (row 11). Again, we observe that our refiner significantly improves the accuracy of these initial pose estimates (+23.7 in average for the RGB-D model). Notably, the RGB-only method (row 9) performs better than the CosyPose refiner (row 1) on average, while not having seen the BOP objects during training. This is thanks to our large-scale training on thousands of various objects, while CosyPose is only trained on tens of objects for each dataset.

One iteration of our refiner takes approximately 50 milliseconds on a RTX 2080 GPU, making it suitable for use in an online tracking application. Five iterations of our refiner are also 5 times faster than the object-specific refiner of SurfEmb [55] which takes around 1 second per image crop. Finally, we evaluate our refiner on ModelNet and compare it with the state-of-the-art methods MP-AAE [197] and LatentFusion [154]. For this experiment, we remove the ShapeNet categories that overlap with the test ones in ModelNet from our training set in order to provide a fair comparison on novel instances and novel categories similar to [112, 154, 197]. Results reported in Table 5.2 show that our refiner significantly outperforms existing approaches across all metrics.

### 5.4.3 Ablations

In this section we perform ablations of our approach to validate our main contributions. Additional ablations are in section 5.6.6 of the appendix. For these ablations, we consider the RGB-only refiner and re-train several models with different configurations of hyper-parameters and training data.

**Encoding the anchor point and object shape.** As discussed in Section 5.3.1 the refiner must have information about the anchor point $\mathcal{O}$ in order to generalize to novel objects. We accomplish this by using 4 rendered views pointing towards the anchor point, see Figure 5-2(b). Table 5.3(a) shows the performance of the refiner increases as we increase the number

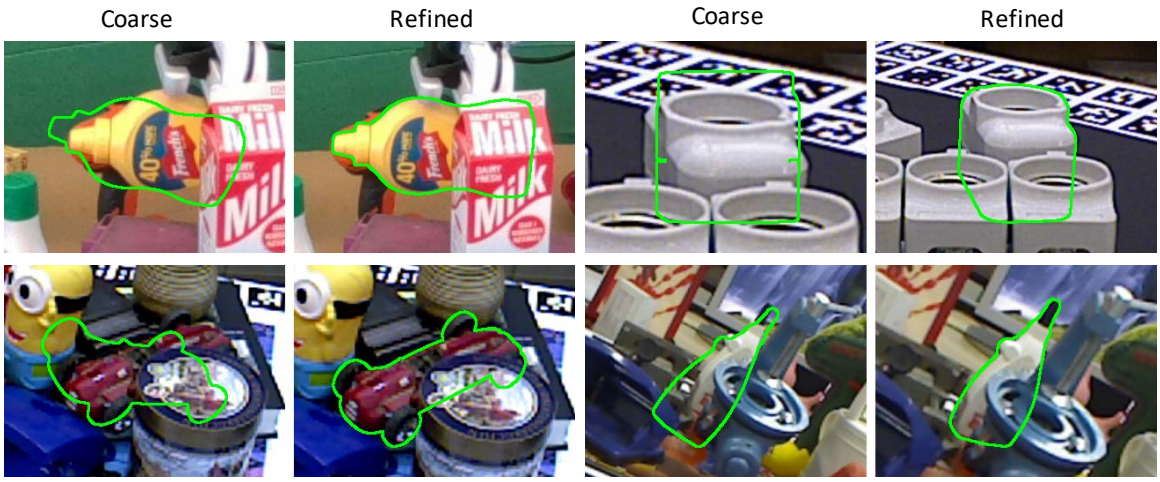



Figure 5-3: **Qualitative results.** For each pair of images, the left image is a visualization of our coarse estimate, and the right image is after applying 5 iterations of our refiner. None of these objects from the YCBV, LMO, HB, or T-LESS datasets were used for training our approach. Please notice the high accuracy of MegaPose despite (i) severe occlusions and (ii) the varying properties of the novel objects (e.g. the texture-less industrial plug in the top-right example, textured mustard bottle in the top-left).

| Rendered views | Rendered normals | BOP5 | ModelNet ADD(0.1d) |
|---|---|---|---|
| 1 | ✓ | 52.0 | 83.3 |
| 2 | ✓ | 59.0 | 90.4 |
| 4 | ✓ | **61.7** | **96.1** |
| 4 | | 59.1 | 83.1 |

(a)

| Training objects | Num. objects | BOP5 | ModelNet ADD(0.1d) |
|---|---|---|---|
| GSO+ShapeNet | 10 + 100 | 47.9 | 28.7 |
| GSO+ShapeNet | 100 + 1000 | 49.3 | 80.3 |
| GSO+ShapeNet | 250 + 2000 | 56.9 | 82.0 |
| GSO+ShapeNet | 500 + 10000 | 59.3 | 89.3 |
| GSO+ShapeNet | 1000 + 20000 | 61.7 | **96.1** |
| GSO | 1000 | 62.2 | 95.7 |
| BOP | 132 | **62.6** | 93.4 |

(b)

Table 5.3: **Ablation study.** We study (a) using multiple rendered object views and normal maps as input to our RGB-only refiner model and (b) training the refiner on different variations of our large-scale synthetic dataset. Average recall is reported on BOP5 (mean of LM-O, T-LESS, TUD-L, IC-BIN and YCB-V) and ModelNet.

of views from 1 to 4, validating our design choice. Multiple views may also help the network to understand the object's appearance from alternate viewpoints, thus potentially helping the refiner to overcome large initial pose errors. We also validate our choice to provide a normal map of the object to the network. This information can help the network use subtle object appearance variations that are only visible under different illumination like the details on the cross of a guitar.

**Number of training objects.** We now show the crucial role of the training data. We report in Table 5.3 (b) the results for our refiner trained on an increasing number of CAD models. The performance steadily increases with the number of objects, which validates that training on a large number of object models is important to generalize to novel ones. These results also suggest that the performance of our approach *could* be improved as more datasets of high-quality CAD models like GSO [36] become available.

**Variety in the training objects.** Next, we restrict the training to different sets of objects. We observe in the bottom of Table 5.3(b) that models from the GoogleScannedObjects are more important to the performance of the method on the BOP dataset compared to using both ShapeNet and GSO. We hypothesize this is due to the presence of high-quality textured objects in the GSO dataset. Finally, we train our model on the 132 objects of the BOP dataset. When testing on the same BOP objects, the performance benefits from knowing these objects during training is small compared to using our GSO+ShapeNet or GSO dataset.

### 5.4.4 Limitations

While MegaPose shows promising results in robot experiments (**please see the video on the project page [134]**) and 6D pose estimation benchmarks, there is still room for improvement. We illustrate the failure modes of our approach in the section 5.6.8 of the appendix. The most common failure mode is due to inaccurate initial pose estimates from the coarse model. The refiner model is trained to correct poses that are within a constrained range but can fail if the initial error is too large. There exist multiple potential approaches to alleviate this problem. We can increase the number of pose hypotheses $M$ at the expense of increased inference time, improve the accuracy of the coarse model, and increase the basin of attraction for refinement model. Another limitation is the runtime of our coarse model. We use $M = 520$ pose hypotheses per object which takes around 2.5 seconds to be rendered and evaluated by our coarse model. In a tracking scenario however, the coarse model is run just once at the initial frame and the object can be tracked using the refiner which runs at 20Hz. Additionally, our refiner could also be coupled with alternate coarse estimation approaches such as [145, 185] to achieve improved runtime performance.

## 5.5 Conclusion

We propose MegaPose, a method for 6D pose estimation of novel objects. Megapose can estimate the 6D pose of novel objects given a CAD model of the object available only at test time. We quantitatively evaluated MegaPose on hundreds of different objects depicted in cluttered scenes, and performed ablation studies to validate our network design choices

and highlight the importance of the training data. We released our models and large-scale synthetic dataset to stimulate the development of novel methods that are practical to use in the context of robotic manipulation where rapid deployment to new scenes with new objects is crucial. While this work focuses on the coarse estimation and fine refinement of an object pose, detecting any unknown object given only a CAD model is still a difficult problem that remains to be solved for having a complete framework for detection and pose estimation of novel objects. Future work will address zero-shot object detection using our large-scale synthetic dataset.

In chapters 4 and 5, we have developed methods for 6D pose estimation of rigid objects with respect to one or multiple cameras. The camera-to-object pose of an object can be used directly for virtual and augmented reality applications. However in the context of robotic applications like the ones considered in chapter 3, knowledge of object-to-robot pose is necessary for visually-guided manipulation. In chapter 6, we seek to recover the camera-to-robot pose which can be combined with the camera-to-object pose to obtain the object-to-robot pose. For controlling the robot arm, knowledge of the internal robot state is also necessary. We consider the most general scenario where this information is not available and must be inferred from images of the robot.

In chapter 6, we address the problem of recovering the full state of a robot state within a scene - the camera-to-robot pose and the joint angles - using a single RGB image, thus enabling 6D manipulation of objects given a single RGB image when combined with the approaches presented in chapters 4 and 5.

## 5.6 Appendix

The appendix of this chapter is organized as follows. In section 5.6.1, we provide the equations of the pose update predicted by the refiner network, and show it depends on the anchor point. In section 5.6.2, we give details on the loss used to train the refiner network. Section 5.6.3 explains the normalization strategy we apply to the observed and rendered depth images of the RGB-D refiner. Section 5.6.4 details the pose hypotheses used during training and inference of the coarse model. In section 5.6.5, we provide examples of training images and give details on the data augmentation and training hardware. In section 5.6.6, we perform additional ablations to validate (i) the contributions of our coarse network, (ii) the choice of hyper-parameter $M$. We also provide details on the robot experiments shown in the video available on the project page [173]. In section 5.6.7, we illustrate qualitatively that our approach is robust to illumination condition variations. Section 5.6.8 illustrates the main failure modes of our approach. Finally, section 5.6.9 investigates the robustness of our approach with respect to an incorrect 3D model.

The **video available on the project webpage [134]** shows predictions of our approach on real images. We apply our approach in *tracking mode* on several videos. Tracking consists in running the coarse estimator on the first frame of a video sequence, and then applying one iteration of the refiner on each new image, using the prediction in the previous image as the pose initialization at the input of refinement network. This approach can process 20 images per second. The video notably demonstrates the method is robust to occlusion and can be used to perform visually guided robotic manipulation of novel objects.

### 5.6.1 Pose update and anchor point

**Pose update.** We use the same pose update as DeepIM [112] and CosyPose [96]. The network predicts 9 values corresponding to one 3-vector $[v_x, v_y, v_z]$ to predict an update of the translation of a 3D anchor point, and two 3-vectors $e_1, e_2$ that define a rotation update explained below. The pose update consists in updating (i) the position of a 3D reference point $\mathcal{O}$ attached on the object, and (ii) the rotation matrix $R_{CO}$ of the object frame expressed in the camera frame (please note the different notations for the anchor point $\mathcal{O}$ and the object frame $O$):

$$x_{\mathcal{O}}^{k+1} = \left(\frac{v_x}{f_x^C} + \frac{x_{\mathcal{O}}^k}{z_{\mathcal{O}}^k}\right) z_{\mathcal{O}}^{k+1}, \tag{5.1}$$

$$y_{\mathcal{O}}^{k+1} = \left(\frac{v_y}{f_y^C} + \frac{y_{\mathcal{O}}^k}{z_{\mathcal{O}}^k}\right) z_{\mathcal{O}}^{k+1}, \tag{5.2}$$

$$z_{\mathcal{O}}^{k+1} = v_z z_{\mathcal{O}}^k, \tag{5.3}$$

$$R_{CO}^{k+1} = R(e_1, e_2) R_{CO}^k, \tag{5.4}$$

where $[x_{\mathcal{O}}^k, y_{\mathcal{O}}^k, z_{\mathcal{O}}^k]$ is the 3D position of the anchor point expressed in camera frame at iteration $k$, $R_{CO}^k$ a rotation matrix describing the objects orientation expressed in camera frame, $f_x^C$ and $f_y^C$ are the (known) focal lengths that correspond to the (virtual) camera associated with the cropped observed image, and $R(e_1, e_2)$ is a rotation matrix describing the rotation update recovered from $e_1, e_2$ using [246] by orthogonalizing the basis defined by the two predicted rotation vectors $e_1, e_2$ similar to [96]. Finally, $[x_{\mathcal{O}}^{k+1}, y_{\mathcal{O}}^{k+1}, z_{\mathcal{O}}^{k+1}]$ and $R_{CO}^{k+1}$ are, respectively, the translation and rotation after applying the pose update. The 3D translation of the anchor point and the rotation matrix $R_{CO}$ are used to define the pose the object.

**Dependency to the anchor point.** We now show that the predictions the network must make to correct a pose error between an initial pose $\mathcal{T}_{CO}^k$ and a target pose $\mathcal{T}_{CO}^{k+1}$ is independent of the choice of the orientation of the objects coordinate frame $O$ but depends on the choice anchor point $\mathcal{O}$. Let us denote $\mathcal{O}^1, \mathcal{O}^2$ two different anchor points, and $R_{CO^1}, R_{CO^2}$ the rotation matrices of the object (expressed in the fixed camera frame) for two different choices of object coordinate frames $O^1$ and $O^2$. We note $t_{\mathcal{O}^1\mathcal{O}^2} = \mathcal{O}^2 - \mathcal{O}^1 = [x_{12}, y_{12}, z_{12}]$ the 3D translation vector between $\mathcal{O}^2$ and $\mathcal{O}^1$; and $R_{O^1O^2} = R_{CO^1}^T R_{CO^2}$ the rotation of coordinate frame $O^2$ expressed in $O^1$. For one choice of anchor point and object frame, e.g. $\mathcal{O}_1$ and $R_{CO^1}$, we derive the predictions the network has to make to correct the

error using equations (5.1),(5.2),(5.3),(5.4):

$$v_x^1 = f_x^c \left( \frac{x_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{x_{O^1}^{k}}{z_{O^1}^{k}} \right) \tag{5.5}$$

$$v_y^1 = f_y^C \left( \frac{y_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{y_{O^k}^{k}}{z_{O^k}^{k}} \right) \tag{5.6}$$

$$v_z^1 = \frac{z_{O^1}^{k+1}}{z_{O^1}^{k}} \tag{5.7}$$

$$R^1 = R_{CO^1}^{k+1} \left( R_{CO^1}^{k} \right)^T, \tag{5.8}$$

and similar for 2 by replacing the superscript. From these equations, we derive:

$$v_x^1 - v_x^2 = f_x^c \left( \frac{x_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{x_{O^1}^{k}}{z_{O^1}^{k}} - \frac{x_{O^2}^{k+1}}{z_{O^2}^{k+1}} + \frac{x_{O^2}^{k}}{z_{O^2}^{k}} \right) \tag{5.9}$$

$$v_y^1 - v_y^2 = f_y^C \left( \frac{y_{O^1}^{k+1}}{z_{O^1}^{k+1}} - \frac{y_{O^1}^{k}}{z_{O^1}^{k}} - \frac{y_{O^2}^{k+1}}{z_{O^2}^{k+1}} + \frac{y_{O^2}^{k}}{z_{O^2}^{k}} \right) \tag{5.10}$$

$$v_z^1 - v_z^2 = \frac{z_{O^1}^{k+1}}{z_{O^1}^{k}} - \frac{z_{O^2}^{k+1}}{z_{O^2}^{k}} \tag{5.11}$$

$$R^1 \left( R^2 \right)^T = R_{CO^1}^{k+1} \left( R_{CO^1}^{k} \right)^T R_{CO^2}^{k} \left( R_{CO^2}^{k+1} \right)^T = R_{CO^1}^{k+1} R_{O^1O^2} R_{O^2C}^{k+1} = Id. \tag{5.12}$$

From eq. (5.12), we have $R^1 \left(R^2\right)^T = Id$. In other words, the rotation matrices that the network must predict to correct the errors in scenarios 1 and 2 are the same. The network predictions for the rotation components thus do not depend on the choice of the choice of object coordinate system. However the other components of the translation cannot be simplified further. For example, derivations of eq. (5.11) leads to $v_z^1 - v_z^2 = \frac{z_{12}\left(z_{O^1}^{k+1} - z_{O^1}^{k}\right)}{z_1^k (z_1^k + z_{12})}$ which is non-zero in the general case where $O^1$ and $O^2$ are different and there is an error between the initial and target poses. This proves that different choices of anchor point leads to different predictions. For the network to generalize to a novel object, the network be able to infer the 3D position of the anchor point on this object. We achieve this by rendering multiple views of the objects in which the anchor point reprojects to the center of each image as explained in section 5.3.

### 5.6.2 Refiner loss

Our refiner network is trained using the same loss as in CosyPose [96], but without using symmetry information on the objects because it is not typically not available for large-scale datasets of CAD models like ShapeNet or GoogleScannedObjects. We first define the distance $D_O(\mathcal{T}_1, \mathcal{T}_2)$ to measure the distance between two 6D poses represented by transformations $\mathcal{T}_1$ and $\mathcal{T}_2$ using the 3D points $\mathcal{X}_O$ of an object O:

$$D_O(\mathcal{T}_1, \mathcal{T}_2) = \frac{1}{|\mathcal{X}_O|} \sum_{x \in \mathcal{X}_O} |\mathcal{T}_1 x - \mathcal{T}_2 x|, \tag{5.13}$$

where $|\cdot|$ is the $L_1$ norm. In practice, we uniformly sample 2000 points on the surface of an object's CAD model to compute this distance. We also define the pose update function $F$ which takes as input the initial estimate of the pose $\mathcal{T}_{CO}^k$, the predictions of the neural network $[v_x, v_y, v_z]$ and $R$, and outputs the updated pose:

$$\mathcal{T}_{CO}^{k+1} = F(\mathcal{T}_{CO}^k, [v_x, v_y, v_z], R), \tag{5.14}$$

where the closed form of function $F$ is expressed in equations (5.1) (5.2) (5.3) (5.4). We also write $[v_x^\star, v_y^\star, v_z^\star]$ and $R^\star$ as the target predictions, i.e. the predictions such that $\mathcal{T}_{CO}^\star = F(\mathcal{T}_{CO}^k, [v_x^\star, v_y^\star, v_z^\star], R^\star)$, where $\mathcal{T}_{CO}^\star$ is the ground truth camera-object pose. The loss used to train the refiner is the following:

$$\mathcal{L} = \sum_{k=1}^{K} D_O(F(\mathcal{T}_{CO}^k, [v_x, v_y, v_z^\star], R^\star), \mathcal{T}_{CO}^\star) \tag{5.15}$$

$$+ D_O(F(\mathcal{T}_{CO}^k, [v_x^\star, v_y^\star, v_z], R^\star), T_{CO}^\star) \tag{5.16}$$

$$+ D_O(F(\mathcal{T}_{CO}^k, [v_x^\star, v_y^\star, v_z^\star], R), \mathcal{T}_{CO}^\star), \tag{5.17}$$

where $D_O$ is the distance defined in eq. (5.13) and $K$ is the number of training iterations. The different terms of this loss separate the influence of: $xy$ translation (5.15), relative depth (5.16) and rotation (5.17). We sum the loss over $K = 3$ refinement iterations to imitate how the refinement algorithm is applied at test time but the error gradients are not backpropagated through rendering and iterations. For simplicity, we write the loss for a single training sample (i.e. a single object in an image), but we sum it over all the samples in the training set.

### 5.6.3 Depth normalization

When depth measurements are available, the observed depth image and depth images of the renderings are concatenated with the images, as mentioned in section 5.3. At test time, the objects may be observed at different depth outside of the training distribution. In order for the network to become invariant to the absolute depth values of the inputs, we normalize both observed and rendered depth. Let us denote $D$ a depth image (rendered or observed are treated similarly). We apply the following operations to $D$. (i) Clipping of the metric depth values of $D$ to lie between 0 and $z_{\mathcal{O}}^k + 1$, where $z_{\mathcal{O}}^k$ is the depth of the anchor point on the object in the input pose at iteration $k$:

$$D \leftarrow \text{clip}(D^k, 0, z_{\mathcal{O}}^k + 1), \tag{5.18}$$

and (ii) centering of the depth values:

$$D \leftarrow \frac{D}{z_{\mathcal{O}}^k} - 1. \tag{5.19}$$

### 5.6.4 Pose hypotheses in the coarse model

**Training hypotheses.** Given the ground truth object pose $\mathcal{T}^{\star}_{CO}$, we generate a perturbed pose $\mathcal{T}'_{CO}$ by applying random translation and rotation to $\mathcal{T}'_{CO}$. The parameters of this (small) perturbation are sampled from the same distribution as the distribution used to sample the perturbed poses the refiner network is trained to correct. The translation is sampled from a normal distribution with a standard deviations of $(0.02, 0.02, 0.05)$ centimeters and rotation is sampled as random Euler angles with a standard deviation of $15$ degrees in each axis.

We then define several poses that depend on $\mathcal{T}'_{CO}$ and cover a large variety of viewing angles of the object. We define a cube of size $2z'_{\mathcal{O}}$, where $z'_{\mathcal{O}}$ is the $z$ component of the 3D translation in the pose $\mathcal{T}'_{CO}$. The CAD model of the object observed under orientation $R'_{CO}$ is placed at the center of the cube. We then place 26 cameras at the locations of each corner, half-side and face centers of the cube. By construction, one of these cameras, which we denote $C^0$, has the same camera-object orientation as $\mathcal{T}'_{CO}$, and all others $\{C^i\}_{i=1..25}$ correspond to cameras observing the object under viewpoints which are sufficiently far from $R_{C^0O}$ and outside the basin of attraction of the refiner by construction. In addition, we apply inplane rotations of $90°$, $180°$ and $270°$ to each camera, which leads to a total of $26 * 4 = 104$ cameras with one positive and 103 negatives.

We mark $\mathcal{T}_{C^0O}$ as a *positive* for the coarse model because the error between $\mathcal{T}_{C^0O}$ and $\mathcal{T}^{\star}_{CO}$ lies within the basin of attraction of the refiner. All other cameras are marked as negatives. During training, the positives account for around $30\%$ of the total numbers of images in a mini-batch.

**Test hypotheses.** At test time, a 2D detection of the object is available. Let $u_{det} = (u_{det,x}, u_{det,y})$ and $(\Delta u_{det} = \Delta u_{det,x}, \Delta u_{det,y})$ define the center and the size of the approximate 2D bounding box of the object in the image. We start by defining a random camera-object orientation $R^p$. The anchor point on the object is set to match the center of the bounding box $u_{det}$. We make a first hypothesis of the depth of the anchor by setting $z^{guess}_{\mathcal{O}^p} = 1\text{m}$ and use this initial value to estimate the coordinates $x_{\mathcal{O}^p}$ and $y_{\mathcal{O}^p}$ of the anchor point in the camera frame:

$$x^{guess}_{\mathcal{O}^p} = u_{det,x} \frac{z^{guess}_{\mathcal{O}^p}}{f_x} \tag{5.20}$$

$$y^{guess}_{\mathcal{O}^p} = u_{det,y} \frac{z^{guess}_{\mathcal{O}^p}}{f_y}, \tag{5.21}$$

where $f_x$ and $f_y$ are the (known) focal lengths of the camera. We then update the depth estimate $z^{guess}_{\mathcal{O}^p}$ using the following simple strategy. We project the points of the object 3D model using $R^p$ and the initial guess of the 3D position of the anchor point we have just defined. These points define a bounding box with dimensions $\Delta u_{guess,x} = (\Delta u_{guess,x}, \Delta u_{guess,y})$ and the center remains unchanged $u_{guess} = u_{det}$ by construction. We compute an updated depth of the anchor point such that its width and height approximately match the size of the 2D detection:

$$z_{\mathcal{O}^p} = z^{guess}_{\mathcal{O}^p} \frac{1}{2} \left( f_x \frac{\Delta u_{guess,x}}{\Delta u_{det,x}} + f_y \frac{\Delta u_{guess,y}}{\Delta u_{det,y}} \right) \tag{5.22}$$

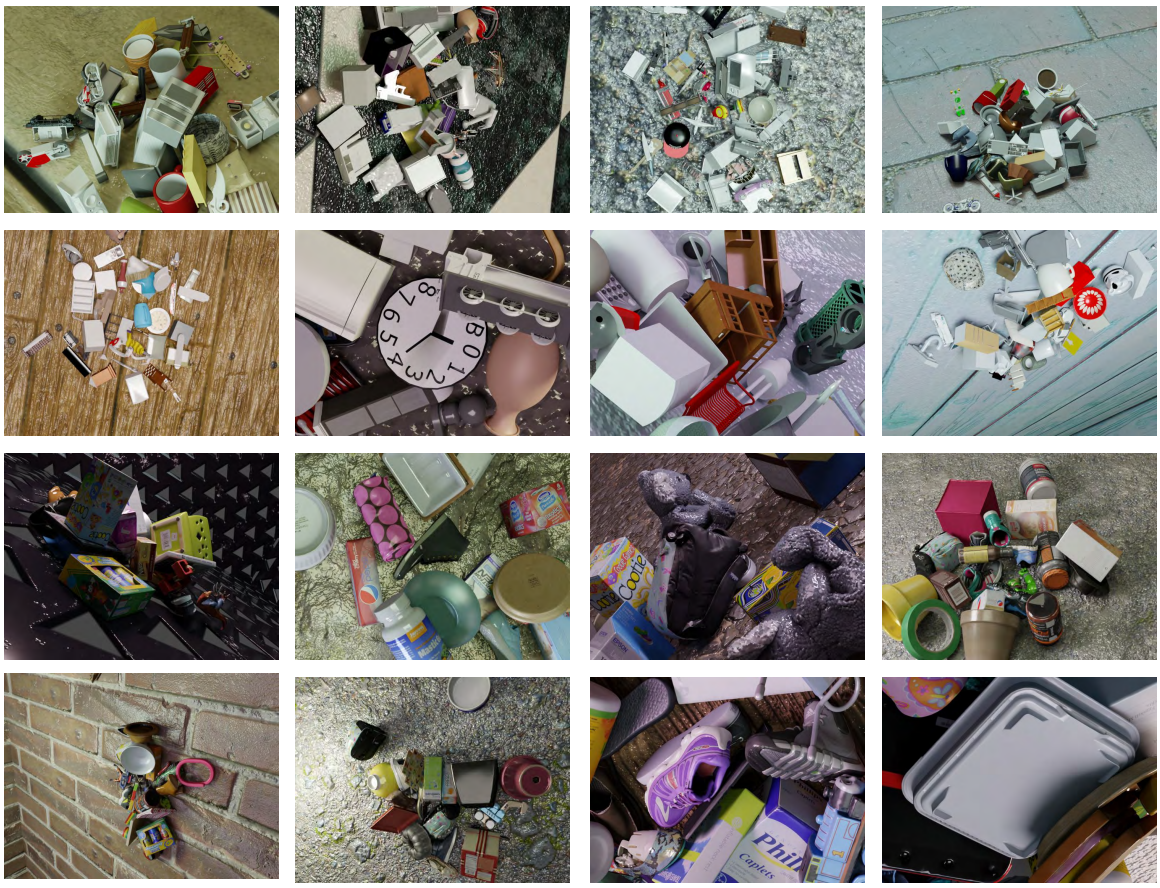


Figure 5-4: **Example of training images**. Randomly sampled images from our large-scale synthetic dataset generated with BlenderProc [33]. CAD models from the ShapeNet [229] and GoogleScannedObjects [36] datasets are used.

and use this new depth to compute $x_{\mathcal{O}^p}$ and $y_{\mathcal{O}^p}$ using equations (5.20) and (5.21) that were used to define $x_{\mathcal{O}^p}^{guess}$ and $y_{\mathcal{O}^p}^{guess}$. The rotation $R^p$ and 3-vector $[x_{\mathcal{O}^p}, y_{\mathcal{O}^p}, z_{\mathcal{O}^p}]$ define the pose of hypothesis $p$. We then use the same strategy used to define the training hypotheses (described above) in order to define 103 additional viewpoints depending on $p$. We repeat the operation $P = 5$ times, for a total of $5 * 104 = 520$ pose hypotheses.

### 5.6.5 Training details

**Training images.** We generate 2 million photorealistic images using BlenderProc [33] as explained in section 5.3.2. Randomly sampled images from the training set are shown in Figure 5-4.

**Data augmentation.** We apply data augmentation to the synthetic images during training. We use the same data augmentation as CosyPose [96] for the RGB images. It includes Gaussian blur, contrast, brightness, colors and shaprness filters from the Pillow library [21]. For the depth images, we take inspiration from the augmentations used in [126, 223, 235]. Augmentations include blur, ellipse dropout, correlated and uncorrelated noise.

**GPU hardware and training time.** Training time is respectively 32 and 48 hours for the

| Pose hypotheses | YCB-V | LM-O |
|---|---|---|
| PPF | 52.7 | 34.4 |
| PPF+Zephyr [149] | 59.8 | 45.8 |
| PPF+Our coarse | **61.6** | **52.1** |

Table 5.4: **Performance of the coarse model.** We compare the performance of our coarse network with Zephyr [149].

coarse and refiner models using 32 V-100 GPUs. This training is performed once, and estimating the pose of novel objects does not require any fine-tuning on the target objects.

### 5.6.6 Additional experiments.

**Coarse network.** In order to evaluate the validate the contributions of our coarse scoring network, we use a set of pose hypotheses generated for novel objects by the commercial Halcon 20.05 Progress software which implements the PPF algorithm described in [22]. Note that these are the same pose hypotheses used in Zephyr [149]. We then find the best hypotheses using the scores of PPF, the scoring network of Zephyr or our coarse network, and report AR results for the LM-O and YCB-V datasets in the table 5.4. On both datasets, our coarse network is better than the two baselines (PPF and Zephyr) for selecting the best poses among a given set of hypotheses.

**Classification-based coarse network.** To validate our classification-based coarse model, we consider a regression-based alternative. We trained a regression-based network similar to the coarse model of CosyPose [96] which takes as input six views of the objects covering viewpoints at the poles of a sphere centered on the object. The network collapsed during training, leading to large errors that cannot be recovered by the refiner and a performance close to zero on the BOP datasets. We hypothesize this failure is due to the presence of symmetric objects in our training set which leads to ambiguous gradients during training. This failure could also be attributed to other factors, such as the difficulty to interpret the full 3D geometry of an object with a CNN given six views of its 3D model captured under distant viewpoints.

**Number of coarse pose hypotheses.** $M$ is an important parameter of our method, which can be used to choose a trade-off between running time and accuracy. The performance significantly improves from M=104 to M=520 (+11.4 AR on BOP5) while keeping the running-time of the coarse model reasonable (1.6 seconds for M=520 compared to 0.3 seconds for M=120). Above M=520, the performance improvement is marginal, e.g. (+0.9 AR) for 4608 hypotheses.

**Robotic grasping experiments.** We performed a qualitative real-robot grasping experiment. For multiple YCB-V objects, we manually annotated one grasp with respect to the object's coordinate frame. We then placed the considered object (e.g. the drill in the supplementary video available on the project page [134]) in a scene among other objects representing visual distractors. The object may be placed on the table or on another object. We then take a single RGB image of the scene using a RealSense D415 camera mounted on the gripper of a Franka Emika Panda robot. We detect the object in 2D using the Mask-

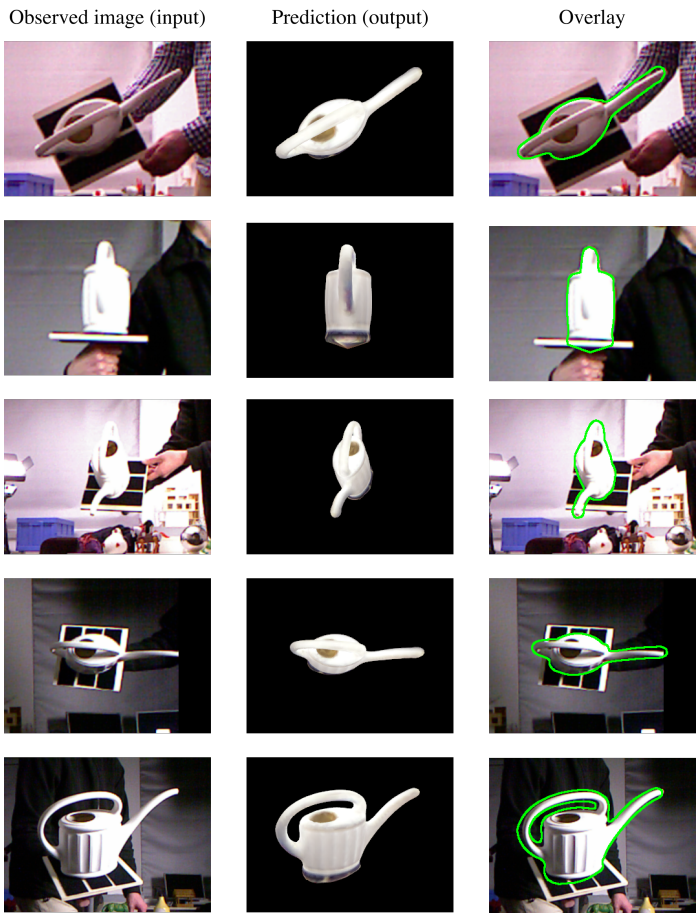



Figure 5-5: Qualitative examples on the TUD-L dataset. Each row presents one example prediction on a real image. The first column is the real observed image, the second column is the prediction of our approach here illustrated using a rendering of the object's CAD model in the predicted pose, and an overlay of the prediction and output is shown on the right.

RCNN detector from CosyPose [96], and run our Megapose approach composed of coarse and refiner modules for estimating the 6D pose of the object with respect to the camera. We then express the 6D pose of the object and grasp with respect to the robot using the known camera-to-robot extrinsic calibration. We then use a motion planner to generate a robot motion that reaches the estimated grasp pose with the gripper and lift the object. This experiment shown in the video available on the project page [134] shows that the pose estimates are of sufficiently high quality to be useful for a robotic manipulation task.

### 5.6.7 Robustness to illumination conditions

In Figure 5-5, we show qualitative predictions of our approach for the watering can on the TUD-L dataset. Please notice the high accuracy of our approach despite challenging illumination conditions.

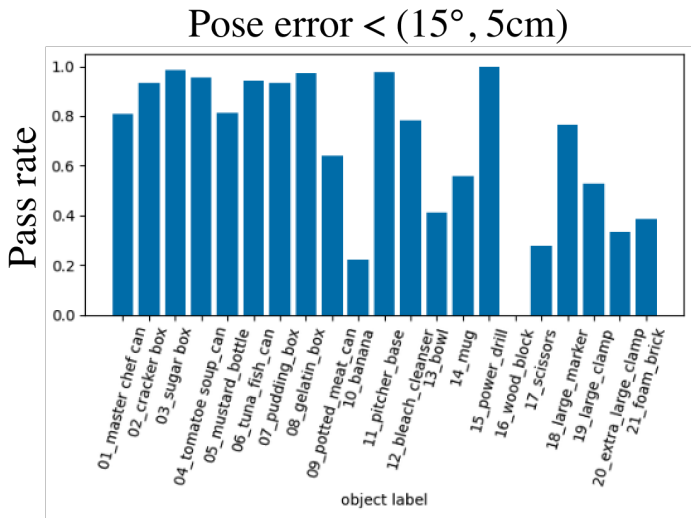



Figure 5-6: Per-object analysis on the YCB-V dataset. For each object, we report the percentage of estimates for which the error between our pose prediction and the ground truth is within 5 centimeters in translation and 15 degrees in rotation.

### 5.6.8 failure modes and performance on specific types of objects

we carry out a per-object analysis of the performance of our approach on the ycb-v dataset. For each of the 21 objects of the dataset, we report the percentage of predictions for which the error with the ground truth is within a threshold of $15^\circ$ in rotation and $5$cm in translation. Results are reported in Figure 5-6.

Next, we illustrate the main failure modes of our approach using a set of objects which have a performance below average on this dataset. Examples of failure cases are presented in Figure 5-7. We observed three main failure modes to our approach. First, we observe the orientation of a novel object may be incorrectly predicted if the object has a similar visual appearance under different viewpoint. We observed this failure mode in particular for textureless objects such as a red bowl that appears similar whether it is standing upside or it is flipped. Second, we observe that our approach may fail to disambiguate the pose of objects that are asymmetric but for which it is necessary to look at fine details on the objects to disambiguate multiple possible poses. An example is a pair of scissors which have left and right handles with slightly different dimensions. In both of these failure modes, we observed that our refiner gets stuck into a local minimal due to an inaccurate coarse estimate outside of the basin of attraction of our refiner model. Finally, using a CAD model with incorrect scale leads to an incorrect estimation of the depth of the object due to the object scale/depth ambiguity in RGB images. We observe for example that the translation estimates of the wooden block of YCB-V have systematically large error despite the rendering of our prediction correctly matching the contours of the object in the observed image. This is because the scale of the CAD model of the wooden block publicly available does not match the correct dimensions of the real object which was used for annotating the ground truth.

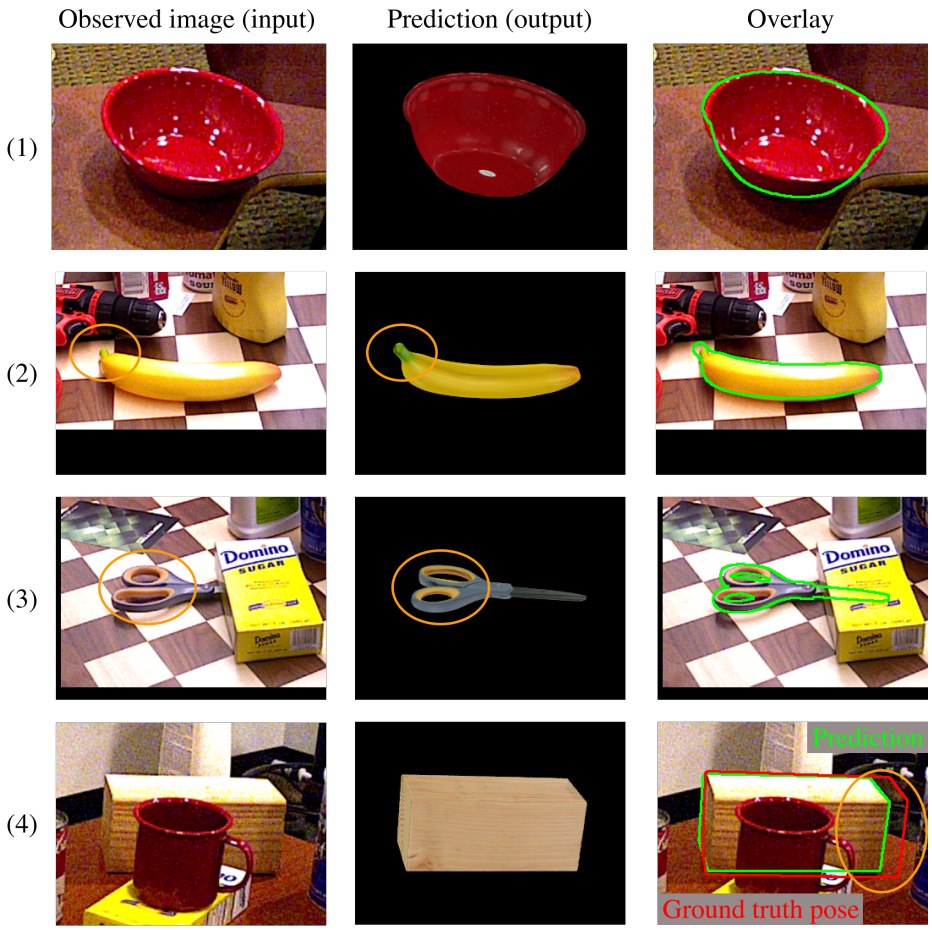



Figure 5-7: Illustration of the main failure modes of our approach. In (1) and (2), the contours of the object in the predicted poses correctly overlay the observed image, but the pose is incorrect because these objects have a similar appearance under different viewpoints. In (3), our approach fails to correctly distinguish the left and right handles with different dimensions in order to disambiguate the orientation of the asymmetric pair of scissors. In (4), our pose prediction does not match the ground truth annotation, because the CAD model of the wooden block we use for pose estimation has different dimensions that do not match the dimensions of the real objects which was used for annotating the ground truth. Please notice in all examples how the contours of the object in the predicted pose are closely aligned with the contours of the object in the input image.

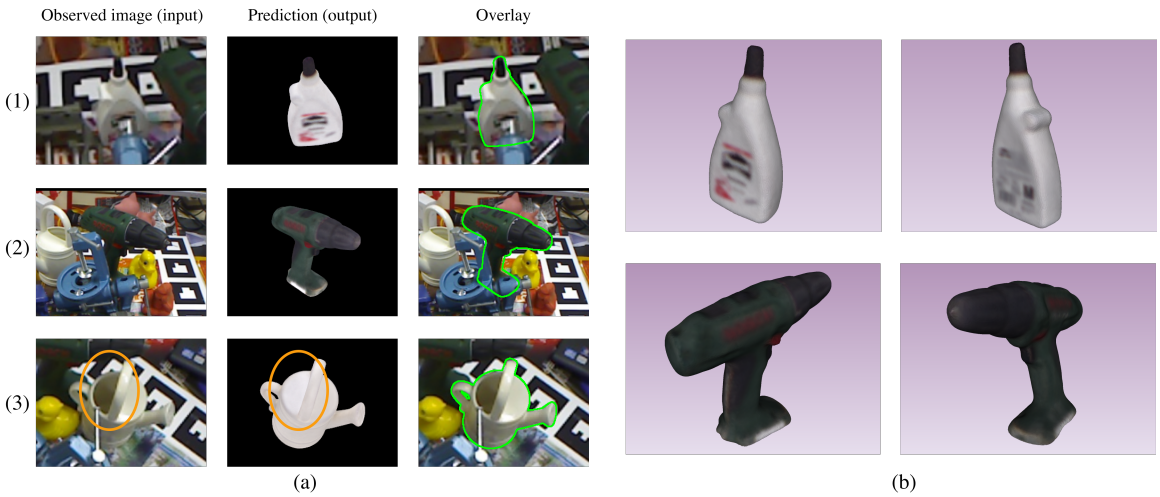



Figure 5-8: Predictions using low-fidelity CAD models. In (a) we show the result of our approach on LineMOD Occlusion for three different objects which have only low-fidelity CAD models available. In (1) and (2), the quality of the mesh and textures is poor as illustrated in (b). Notice for example how the annotations on the glue box or the brand of the drill are not readable on the CAD models. In (3), the hole of the watering can does not appear in the CAD model. Despite these discrepancies between the real object and the CAD model, our approach correctly estimates the pose of each object.

### 5.6.9 3D model quality

Our approach can be applied even if the 3D model of the object does not exactly matches the real object. In figure 5-8, we show examples of correctly estimated poses using low-fidelity CAD models with low-quality textures or geometric discrepancies between the real object and its 3D model.

# Chapter 6

# Robot pose and joint angle estimation

In the previous chapters, 4 and 5, we focus on estimating the 6D of pose of rigid objects. In this chapter we consider articulated objects and focus on robots, which are a type of articulated objects with potentially a large number of degrees of freedom (e.g. up to 15).

We introduce RoboPose, a method to estimate the joint angles and the 6D camera-to-robot pose of a known articulated robot from a single RGB image. This is an important problem to grant mobile and itinerant autonomous systems the ability to interact with other robots using only visual information in non-instrumented environments, especially in the context of collaborative robotics. It is also challenging because robots have many degrees of freedom and infinite space of possible configurations that often result in self-occlusions and depth ambiguities when imaged by a single camera. The contributions of this work are three-fold. First, we introduce a new *render & compare* approach for estimating the 6D pose and joint angles of an articulated robot that can be trained from synthetic data, generalizes to new unseen robot configurations at test time and can be applied to a variety of robots. Second, we experimentally demonstrate the importance of robot parametrization for the iterative pose updates and design a parametrization strategy that is independent of the robot structure. Finally, we show experimental results on existing benchmark datasets for four different robots and demonstrate that our method significantly outperforms the state of the art. Code and pre-trained models are available on the project webpage [173].

## 6.1 Introduction

The goal of this work is to recover the state of a known articulated robot within a 3D scene using a single RGB image. The robot state is defined by (i) its 6D pose, i.e. a 3D translation and a 3D rotation with respect the camera frame, and (ii) the joint angle values of the robot's articulations. The problem set-up is illustrated in figure 6-1. This is an important problem to grant mobile and itinerant autonomous systems the ability to interact with other robots using only visual information in non-instrumented environments. For instance, in the context of collaborative tasks between two or more robots, having knowledge of the pose and the joint angle values of all other robots would allow better distribution of the load between robots involved in the task [14]. The problem is, however, very challenging because robots can have many degrees of freedom (DoF) and an infinite space of admissible

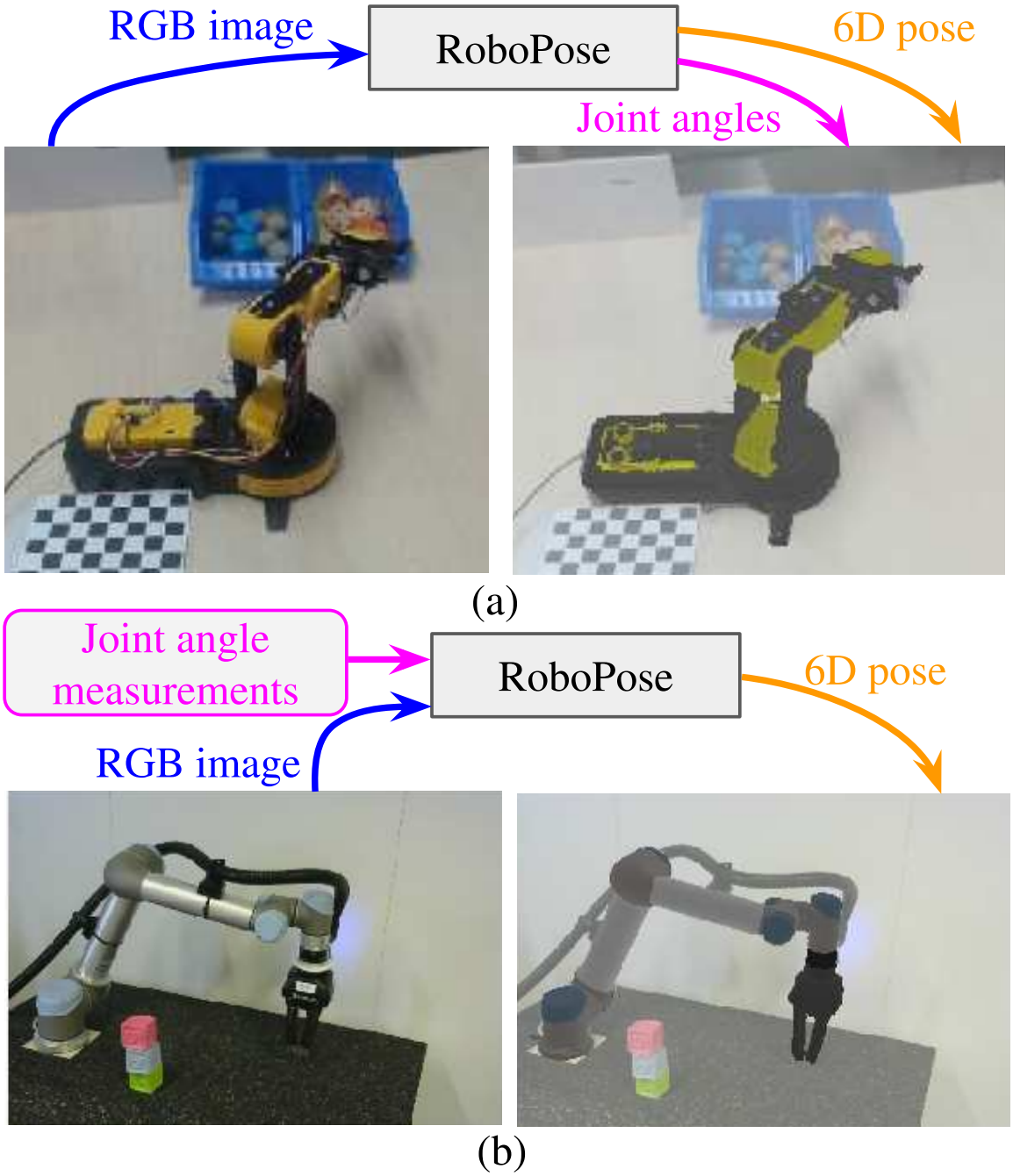



Figure 6-1: **RoboPose.** (a) Given a single RGB image of a known articulated robot in an unknown configuration (left), RoboPose estimates the joint angles and the 6D camera-to-robot pose (rigid translation and rotation) providing the complete state of the robot within the 3D scene, here illustrated by overlaying the articulated CAD model of the robot over the input image (right). (b) When the joint angles are known at test-time (e.g. from internal measurements of the robot), RoboPose can use them as an additional input to estimate the 6D camera-to-robot pose to enable, for example, visually guided manipulation without fiducial markers.

configurations that often result in self-occlusions and depth ambiguities when imaged by a single camera. The current best performing methods for this problem [106, 247] use a deep neural network to localize in the image a fixed number of pre-defined keypoints (typically located at the articulations) and then solve a 2D-to-3D optimization problem to recover the robot 6D pose [106] or pose and configuration [247]. For rigid objects, however, methods based on 2D keypoints [9, 22, 23, 76, 87, 124, 158, 160, 169, 202, 207] have been recently outperformed by *render & compare* methods that forgo explicit detection of 2D keypoints but instead use the entire shape of the object by comparing the rendered view of the 3D model to the input image and iteratively refining the object's 6D pose [96, 112, 169, 240]. Motivated by this success, we investigate how to extend the *render & compare* paradigm for articulated objects. This presents significant challenges. First, we need to estimate many more degrees of freedom than the sole 6D pose. Articulated robots we consider in this work can have up to 15 degrees of freedom in addition to their 6D rigid pose in the environment. Second, the space of configurations is continuous and hence there are infinitely many configurations in which the object can appear. As a result, it is not possible to see all configurations during training and the method has to generalize to unseen configurations at test time. Third, the choice of transformation parametrization plays an important role for 6D pose estimation of rigid objects [112] and finding a good parametrization of pose updates for articulated objects is a key technical challenge.

**Contributions.** To address these challenges, we make the following contributions. First, we introduce a new *render & compare* approach for estimating the 6D pose and joint angles of an articulated robot that can be trained from synthetic data, generalizes to new unseen robot configurations at test time, and can be applied to a large variety of robots (robotic arms, bi-manual robots, etc.). Second, we experimentally demonstrate the importance of the robot pose parametrization for the iterative pose updates and design an effective parametrization strategy that is independent of the robot. Third, we apply the proposed method in two settings: (i) with known joint angles (e.g. provided by internal measurements from the robot such as joint encoders), only predicting the camera-to-robot 6D pose, and (ii) with unknown joint angles, predicting both the joint angles *and* the camera-to-robot 6D pose. We show experimental results on existing benchmark datasets for both settings that include a total of four different robots and demonstrate significant improvements compared to the state of the art.

## 6.2 Related work

**6D pose estimation of rigid objects** from RGB images [123, 124, 172] is one of the oldest problems in computer vision. It has been successfully approached by estimating the pose from 2D-3D correspondences obtained via local invariant features [9, 22, 23, 124], or by template-matching [63]. Both these strategies have been revisited using convolutional neural networks (CNNs). A set of sparse [76, 87, 158, 160, 169, 202, 207] or dense [155, 191, 230, 240] features is detected on the object in the image using a CNN and the resulting 2D-to-3D correspondences are used to recover the camera pose using PnP [108]. The best performing methods for 6D pose estimation from RGB images are now based on variants of the *render & compare* strategy [96, 112, 127, 147, 148, 169, 240] and are approaching

the accuracy of methods using depth as input [70, 71, 96, 112].

**Hand-eye calibration** (HEC) [61, 74] methods recover the 6D pose of the camera with respect to a robot. The most common approach is to detect in the image fiducial markers [41, 45, 151] placed on the robot at known positions. The resulting 3D-to-2D correspondences are then used to recover the camera-to-robot pose using *known* joint angles and the kinematic description of the robot by solving an optimization problem [79, 153, 237]. Recent works have explored using CNNs [101, 106] to perform this task by recognizing 2D keypoints at specific robot parts and using the resulting 3D-to-2D correspondences to recover the hand-eye calibration via PnP. The most recent work in this direction [106] demonstrated that such learning-based approach could replace more standard hand-eye calibration methods [210] to perform online calibration and object manipulation [208]. Our *render & compare* method significantly outperforms [106] and we also demonstrate that our method can achieve a competitive accuracy without requiring known joint angles at test time.

**Depth-based pose estimation of articulated objects.** Previous work on this problem can be split into three classes. The first class of methods aims at discovering properties of the kinematic chain through active manipulation [56, 85, 86, 131] using depth as input and unlike our approach cannot be applied to a single image. The second class of methods aims at recovering all parameters of the kinematic chain from a single RGBD image, including the joint angles, without knowing the specific articulated object [1, 111, 239, 245]. In contrast, we focus on the set-up with a known 3D model, e.g. a specific type of a robot. The third class of methods, which is closest to our set-up, considers pose and joint angle estimation [34, 137, 157] for known articulated objects but relies on depth as input and only considers relatively simple kinematic chains such as laptops or drawers where the joint parameters only affect the pose of one part. Others recover joint angles of a known articulated robot part [12, 224] but do not recover the 6D pose of the camera and also rely on depth. In contrast, our method accurately estimates the pose and joint angles of a robotic arm with many degrees of freedom from a single RGB image.

**Robot pose and joint angle estimation from an RGB image.** To the best of our knowledge, only [247] has addressed a scenario similar to us where the robot pose and joint angles are estimated together from a single RGB image.

A set of predefined 2D keypoints is recognized in the image and the 6D pose and joint angles are then recovered by solving a nonlinear non-convex optimization problem. Results are shown on a 4 DoF robotic arm. In contrast, we describe a new *render & compare* approach for this problem, demonstrate significantly improvements in 3D accuracy and show results on robots with up to 15 DoF.

## 6.3 Approach

We present our *render & compare* framework to recover the state of a robot within a 3D scene given a single RGB image. We assume the camera intrinsic parameters, the CAD model and kinematic description of the robot are known. We start by formalizing the problem in section 6.3.1. We then present an overview of our approach in section 6.3.2 and explain our training in section 6.3.3. Finally, we detail the key choices in the problem parametrization in section 6.3.4.

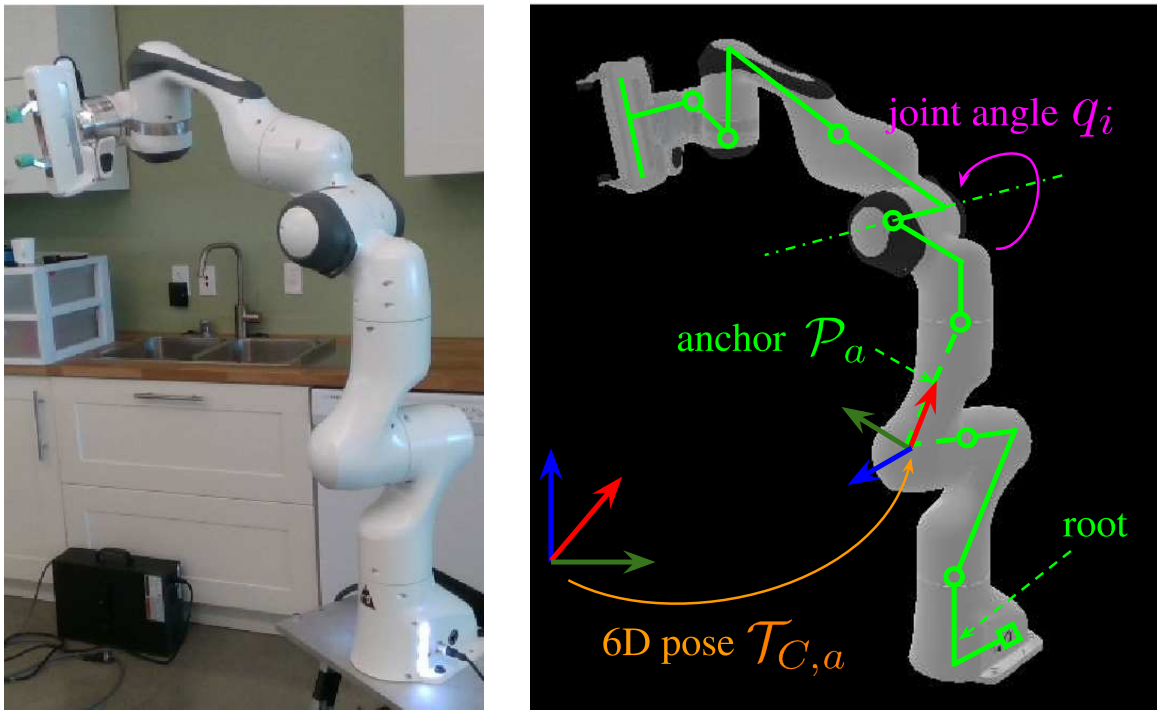



Figure 6-2: **Problem definition.** Given an RGB image (a) of a known robot, the goal is to recover (b) the 6D pose $\mathcal{T}_{C,a}$ of an anchor part $\mathcal{P}_a$ with respect to the camera frame and all the joint angles $q_i$ of the known robot kinematic description (in green).

### 6.3.1 Problem formalization

Our notations are summarized in Figure 6-2. We consider a known robot composed of *rigid parts* $\mathcal{P}_0$,...,$\mathcal{P}_N$ whose 3D models are known. An articulation, or *joint*, connects a parent part to a child part. Given the *joint angle* $q_i$ of the $i$-th joint, we can retrieve the relative 6D transformation between the parent and child reference frames. Note that for simplicity we only consider revolute joints, i.e. joints parametrized by a single scalar angle of rotation $q_i$, but our method is not specific to this type of joints. The rigid parts and the joints define the *kinematic tree* of the robot. This kinematic description can be used to compute the relative 6D pose between any two parts of the robot. In robotics, the full state of a robot $\mathcal{S}$ is defined by the joint angles and the 6D pose of the *root* of the kinematic tree. Defining the 6D pose of the robot with respect to the root (whose pose is independent of the joint angles since it is not a child of any joint) is a crucial choice in the parametrization of the problem, but also arbitrary, since an equivalent kinematic tree could be defined using any part as the root. We instead define the full state of the robot by (i) the selection of an *anchor part* $\mathcal{P}_a$, (ii) the 6D pose of the anchor with respect to the camera $\mathcal{T}_{C,a}$, and (iii) the joint angles $q = (q_1, ..., q_D) \in \mathbb{R}^D$, where $D$ is the number of joints. Note the anchor part can change across iterations of our approach. We discuss the choice of the anchor in section 6.3.4 and experimentally demonstrate it has an important influence on the results.

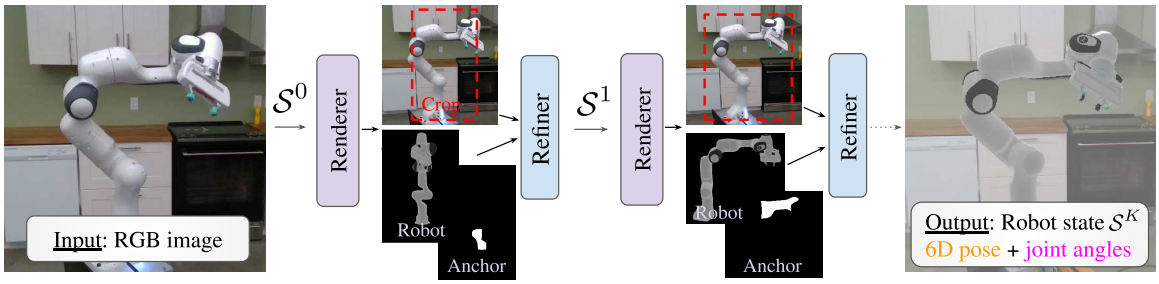



Figure 6-3: **RoboPose overview.** Given a single input RGB image, the state $S$ (6D camera-to-robot pose and joint angles) of the robot is iteratively updated using renderer and refiner modules to match the input image. The refinement module takes as input the cropped observed image and a rendering of the robot as well as the mask of an anchor part. The anchor part is used for updating the rigid 6D pose of the robot while the rest of the parts are updated by changing their joint angles. Note that the anchor part is changing across iterations making the refinement more robust.

### 6.3.2 Render & compare for robot state estimation

We now present our iterative deep *render & compare* framework, illustrated in figure 6-3. We iteratively refine the state estimate as follows. First, given a current estimate of the state $\mathcal{S}^k$ we render an RGB image of the robot $\mathcal{R}(S^k)$ and the mask of the anchor part. We then apply a deep refiner network that takes as input crops of the rendered image and the input RGB image $I$ of the scene. It outputs a new state of the robot $\mathcal{S}^{k+1} = f_\theta(\mathcal{S}^k, I)$ to attempt to match the ground truth state $\mathcal{S}^{gt}$ of the observed robot. Unlike prior works that have used *render & compare* strategies for estimating the 6D pose of rigid objects [96, 112, 240], our method does not require a coarse pose estimate as initialization.

**Image rendering and cropping.** To render the image of the robot we use a fixed focal length (defining an intrinsic camera matrix) during training. The rendering is fully defined by the state of the robot and the camera matrix. Instead of giving to the refiner network the full image and the rendered view, we focus the inputs on the robot by cropping the images as follows. We project the centroid of the rendered robot in the image, consider the smallest bounding box of aspect ratio 4/3 centered on this point that encloses the projected robot and increase its size by 40% (see details in the appendix). This crop depends on the projection of the robot to the input image that varies during training, and thus provides an augmentation of the effective focal length of the virtual cropped cameras. Hence, our method can be applied to cameras with different intrinsics at test time as we show in our experiments.

**Initialization.** We initialize the robot to a state $\mathcal{S}^0$ defined by the joint configuration $q^0$ and the pose $\mathcal{T}^0_{C,a}$ of the anchor part $a$ with respect to the camera $C$. At training time we define $\mathcal{S}^0$ using perturbations of the ground truth state. At test time we initialize the joints to the middle of the joint limits, and the initial pose $\mathcal{T}^0_{C,a}$ so that the frame of the robot base is aligned with the camera frame and the 2D bounding box defined by the projection of the robot model approximately matches the size of image. More details are given in section 6.6.4 of the appendix.

**Refiner and state update.** At iteration $k$, the refiner predicts an update $\Delta q^k$ of the joint

angles $q^k$ (composed of one scalar angle per joint), such that

$$q^{k+1} = q^k + \Delta q^k, \tag{6.1}$$

and an update $\Delta\mathcal{T}^k$ of the current 6D pose $\mathcal{T}^k_{C,a}$ of the anchor part, such that

$$\mathcal{T}^{k+1}_{C,a} = \mathcal{T}^k_{C,a} \circ \Delta\mathcal{T}^k, \tag{6.2}$$

where we follow DeepIM [112]'s parametrization for pose update $\Delta\mathcal{T}^k$. This parametrization disentangles rotation and translation prediction but crucially depends on the choice of a *reference point* we call $O$. In DeepIM this point is simply taken as the center of the reference frame of the rigid object but there is not such a natural choice of reference point for articulated objects, which have multiple moving parts. We discuss several possible choices of the reference point $O$ in section 6.3.4 and demonstrate experimentally it has an important impact on the results. In particular, we show that naively selecting the reference frame of the root part is sub-optimal.

### 6.3.3 Training

In the following, we describe our loss function, synthetic training data, implementation details and discuss how to best use known joint angles if available.

**Loss function.** We train our refiner network using the following loss:

$$\mathcal{L}(\theta) = \sum_{k=0}^{K-1} \mathcal{L}_a(\mathcal{T}^k_{C,a}, \Delta\mathcal{T}^k, \mathcal{T}^{gt}_{C,a}) + \lambda\, \mathcal{L}_q(q^k, \Delta q^k, q^{\text{gt}}), \tag{6.3}$$

where $\theta$ are the parameters of the refiner network, $K$ is the maximum number of iterations of the refinement algorithm, $\mathcal{T}^{gt}_{C,a}$ is the ground truth 6D pose of the anchor, $q^{\text{gt}}$ are the ground truth joint angles and $\lambda$ is a hyper-parameter to balance between the 6D pose loss $\mathcal{L}_a$ and the joint angle loss $\mathcal{L}_q$. The 6D pose loss $\mathcal{L}_a$ measures the distance between the predicted 3D point cloud obtained using $\mathcal{T}^k_{C,a}$ transformed with $\Delta\mathcal{T}^k$ and the ground truth 3D point cloud (obtained using $\mathcal{T}^{gt}_{C,a}$) of the anchor $\mathcal{P}_a$. We use the same loss as [96] that disentangles rotation, depth and image-plane translations [189] (see equations in section 6.6.2 of the appendix). For $\mathcal{L}_q$, we use a simple $L_2$ regression loss, $\mathcal{L}_q = \|q^k + \Delta q^k - q^{gt}\|_2^2$.

Note that the 6D pose loss is measured only on the anchor part $a$ while the alignment of the other parts of the robot is measured by the error on their joint angles (rather than alignment of their 3D point clouds). This disentangles the 6D pose loss $\mathcal{L}_a$ from the joint angle loss $\mathcal{L}_q$ and we found this leads to better convergence. We sum the loss over the refinement iterations $k$ to imitate how the refinement algorithm is applied at test time but the error gradients are not backpropagated through rendering and iterations. Finally, for simplicity the loss (6.3) is written for a single training example, but we sum it over all examples in the training set.

**Training data.** For training the refiner, we use existing datasets [106, 247] provided by prior works for the Kuka, Panda, Baxter, OWI-535 robots. All of these datasets are synthetic, generated using similar procedures based on domain randomization [81, 122, 177,

203]. The joint angles are sampled independently and uniformly within their bounds, without assuming further knowledge about their test time distribution. We add data augmentation similar to [96].

We sample the initial state $\mathcal{S}^0$ by adding noise to the ground truth state in order to simulate errors of the network prediction at the previous state of the refinement as well as the error at the initalization. For the pose, we sample a translation from a centered Gaussian distribution with standard deviation of $10$ cm, and a rotation by sampling three angles from a centered Gaussian distribution of standard deviation $60°$. For the joint angles, we sample an additive noise from a centered Gaussian distribution with a standard deviation equal to $5\%$ of the joint range of motion, which is around $20°$ for most of the joints of the robots we considered.

**Implementation details.** We train separate networks for each robot. We use a standard ResNet-34 architecture [59] as the backbone of the deep refiner. The hyper-parameters are $\lambda = 1$ and $K = 3$ training iterations. Note that at test time we can perform more iterations, and the results we report correspond to 10 iterations. The anchor is sampled randomly among the 5 largest parts of the robot at each iteration. This choice is motivated in section 6.3.4 and other choices are considered in the experiments, section 6.4.3. We initialize the network parameters randomly and perform the optimization using Adam [90], with the procedure described in section 6.6.5 of the appendix for all the networks.

**Known joint angles at test time.** The approach described previously could be used at test time with measured joint angles $q^0 = q^{gt}$ and by ignoring the joint update, but we observed better results by training a separate network which only predicts a pose update for this scenario. In this context where the joint values are known and constant, the full robot is considered as a single and unique anchor. Yet, the problem remains different from classic rigid object 6D object pose estimation because the network must generalize to new joint configurations unseen during training.

### 6.3.4 Parametrization choices

There are two main parametrization choices in our approach: (i) the choice of the reference point $O$ for the parametrization of the pose update $\Delta\mathcal{T}^k$ in equation (6.2) and (ii) the choice of the anchor part to update the 6D pose and measure pose loss in equation (6.3). These choices have a significant impact on the results, as shown in section 6.4.

**Choice of the reference point for the pose update.** Similar to [112], we parametrize the pose update as a rotation around a reference point $O$ and a translation defined as a function of the position of $O$ with respect to the camera. The fact that the rotation is around $O$ is a first obvious influence of this choice on the transformation that needs to be predicted. The impact on the translation update parameters is more complicated: they are defined by a multiplicative update on the depth of $O$ and by an equivalent update in pixels in the image, which is also related to the real update by the depth of $O$ (see equations in section 6.6.1 of the appendix).

A seemingly natural choice for reference point $O$ would be a physical point on the robot, for example the center of the base of the robot or the anchor part. However, on the contrary to the rigid object case, if that part is not visible or is partially occluded, the network cannot infer the position of the reference $O$ precisely, and thus cannot predict a relevant translation

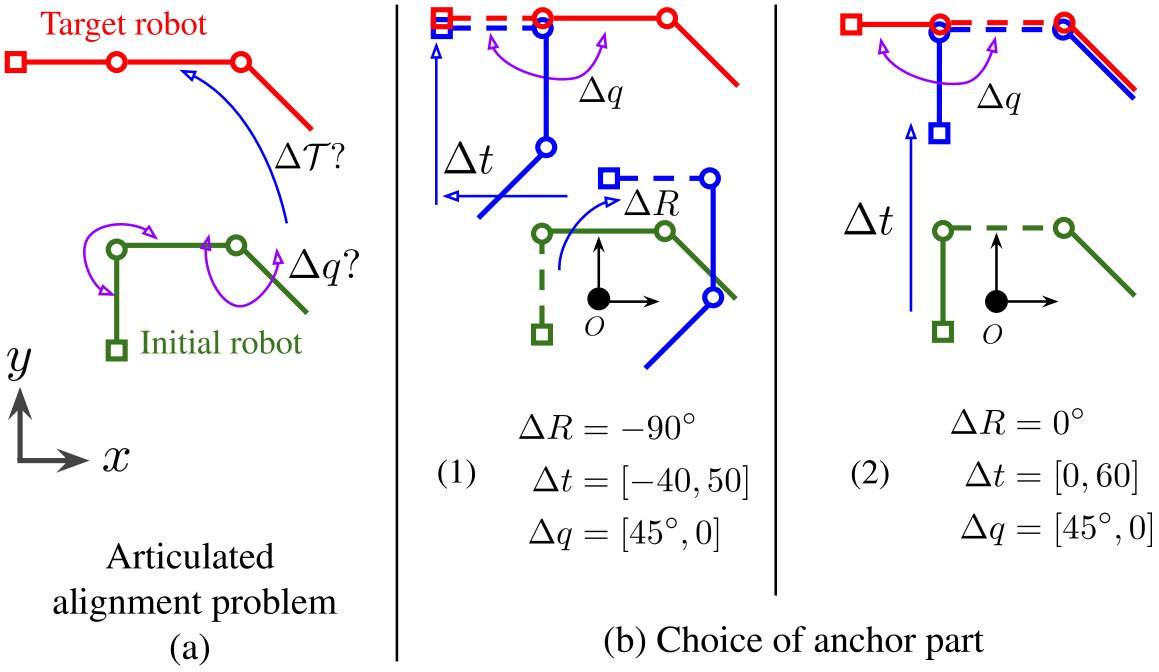



Figure 6-4: **Choice of the anchor part. (a)** We analyze how the choice of the anchor part affects the complexity of the rigid alignment $\Delta\mathcal{T}$ and the joint angle update $\Delta q$ to align an initial state of the robot (green) with the target state of the robot (red). **(b)** We show the required rigid pose update (composed of a rotation and a translation) and the required joint update for two different choices of the anchor part (shown using a dashed line). In (1), the required pose update of the anchor part consists of successively applying rotation $\Delta R$ and translation $\Delta t$ along $x$ and $y$ axes (in blue). In (2), the anchor part is aligned using only a translation along the $y$ axis resulting in a simpler solution compared to (1). These examples illustrate that the choice of the anchor can have a significant impact on the complexity of the alignment problem.

and rotation update. In experiments, we show it is better to use as $O$ the centroid of the estimated robot state, which takes into account the estimated joint configuration, and can be more reliably estimated.

**Choice of the anchor part.** The impact of the choice of the anchor part $\mathcal{P}_a$ used for computing the 6D pose loss in equation (6.3), is illustrated in figure 6-4. We explore several choices of anchor part in our experiments, and show that this choice has a significant impact on the results. Since the optimal choice depends on the robot, and the observed pose, we introduce a strategy where we randomly select the anchor among the largest parts of the robot, during both training and refinement, and show that on average it performs similarly or slightly better than the optimal oracle choice of a single unique anchor on the test set.

## 6.4 Experiments

We evaluate our method on recent benchmarks for the following two tasks: (i) camera-to-robot 6D pose estimation for three widely used manipulators (Kuka iiwa7, Rethink robotic

| Robot | Dataset informations | | | | | DREAM [106] | DREAM [106] | DREAM [106] | Ours | Ours |
|---|---|---|---|---|---|---|---|---|---|---|
| | Robot (DoF) | Real | # images | # 6D poses | cam. | VGG19-F | VGG19-Q | ResNet101-H | ResNet34 | Unknown angles |
| Baxter DR | Baxter (15) | × | 5982 | 5982 | GL | - | 75.47 | - | **86.59** | **32.66** |
| Kuka DR | Kuka (7) | × | 5997 | 5997 | GL | - | - | 73.30 | **89.62** | **80.17** |
| Kuka Photo | Kuka (7) | × | 5999 | 5999 | GL | - | - | 72.14 | **86.87** | **73.23** |
| Panda DR | Panda (8) | × | 5998 | 5998 | GL | 81.33 | 77.82 | 82.89 | **92.70** | **82.85** |
| Panda Photo | Panda (8) | × | 5997 | 5997 | GL | 79.53 | 74.30 | 81.09 | **89.89** | **79.72** |
| Panda 3CAM-AK | Panda (8) | ✓ | 6394 | 1 | AK | 68.91 | 52.38 | 60.52 | **76.54** | **70.37** |
| Panda 3CAM-XK | Panda (8) | ✓ | 4966 | 1 | XK | 24.36 | 37.47 | 64.01 | **85.97** | **77.61** |
| Panda 3CAM-RS | Panda (8) | ✓ | 5944 | 1 | RS | 76.13 | 77.98 | **78.83** | 76.90 | **74.31** |
| Panda ORB | Panda (8) | ✓ | 32315 | 27 | RS | 61.93 | 57.09 | 69.05 | **80.54** | **70.39** |

Table 6.1: Comparison of RoboPose (ours) with the state-of-the-art approach DREAM [106] for the camera-to-robot 6D pose estimation task using the 3D reconstruction ADD metric (higher is better). The robot joint configuration is assumed to be known (results in black) and is different in each of the image in the dataset, but the pose of the camera with respect to the robot can be fixed (# number of 6D poses). Multiple cameras are considered to capture the input RGB images: synthetic rendering (GL), and real Microsoft Azure (AK), Microsoft Kinect360 (XK) and Intel RealSense (RS), which all have different intrinsic parameters. Our results in blue do not use ground truth joint angles (see section 6.4.2) and the accuracy of the robot 3D reconstruction is evaluated using both the estimated 6D pose and the joint angles.

Baxter, Franka Emika Panda) [106], and (ii) full state estimation of the low-cost 4 DoF robotic arm OWI-535 [247]. In section 6.4.1, we consider the first task, where an image of a robot with fixed known joint angles is used to estimate the 6D camera-to-robot pose. We show that our approach outperforms the state-of-the-art DREAM method [106]. In section 6.4.2, we evaluate our full approach where both the 6D pose and joint angles are unknown. We show our method outperforms the state-of-the-art method [247] for this problem on their dataset depicting the low-cost 4 DoF robot and that it can recover the 6D pose *and* joint angles of more complex robotic manipulators. Finally, section 6.4.3 analyzes the parametrization choices discussed in section 6.3.4.

### 6.4.1 6D pose estimation with known joint angles

**Datasets and metrics.** We focus on the datasets annotated with 6D pose and joint angle measurements recently introduced by the state-of-the-art method for single-view camera-to-robot calibration, DREAM [106]. We use the provided training datasets with 100k images generated with domain randomization. Test splits are available as well as photorealistic synthetic test images (Photo). For the Panda robot, real datasets are also available. The Panda 3CAM datasets display the fixed robot performing various motions captured by 3 fixed different cameras with different focal lengths and resolution, all of which are different than the focal length used during training. The largest dataset with the more varied viewpoints is Panda-ORB with 32,315 real images in a kitchen environnement captured from 27 viewpoints with different joint angles in each image.

We use the 3D reconstruction ADD metric which directly measures the pose estimation accuracy, comparing distances between 3D keypoints defined at joint locations of the robot in the ground truth and predicted pose. We refer to the section 6.6.6 of the appendix for

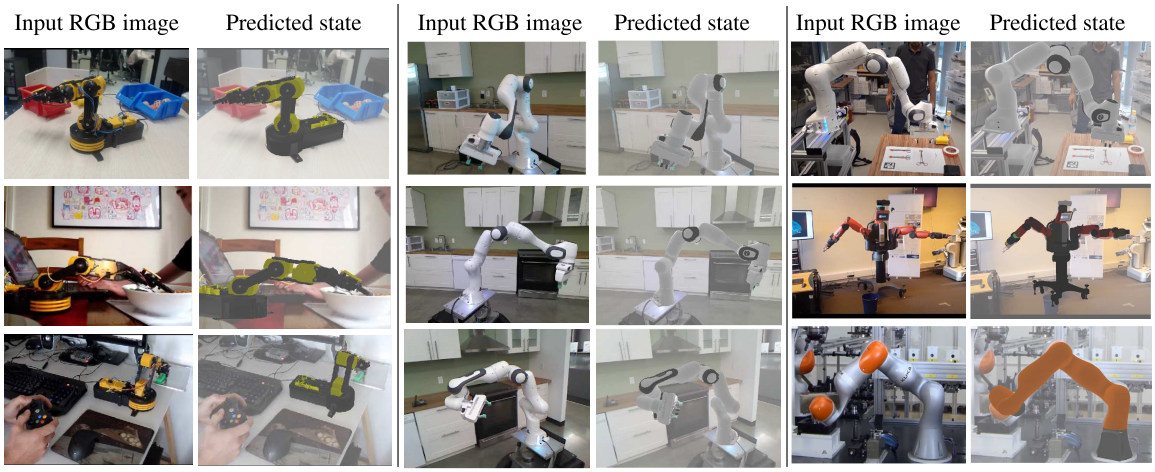



Figure 6-5: Qualitative results of RoboPose 6D pose and joint angle estimation for four different robots. (a) The OWI-535 robot from the CRAVES-lab (first row) and CRAVES-youtube (second and third row) datasets, (b) the Panda robot from the Panda 3CAM dataset and (c) the Panda, Baxter and Kuka robots on example images from the Internet. **Please see additional results in section 6.6.9 of the appendix and in the video on the project webpage [173].**

exact details on the evaluation protocol of our comparison with DREAM [106].

**Comparison with DREAM [106].** We train one network for each robot using the same synthetic datasets as [106] and report our results in Table 6.1. Our method achieves significant improvements across datasets and robots except on Panda 3CAM-RS where the performance of [106] with ResNet101-H variant is similar to ours. On the Panda 3CAM-AK and Panda 3CAM-XK datasets, the performance of our method is significantly higher than the ResNet101-H model of [106] (e.g. $+21.96$ on 3CAM-XK), which suggests that the approach of [106] based on 2D keypoints is more sensitive to some viewpoints or camera parameters. Note that our method trained with the synthetic GL camera can be applied to different real cameras with different intrinsics at test time thanks to our cropping strategy which provides an augmentation of the effective focal length during training.

On Panda-ORB, the largest real dataset that covers multiple camera viewpoints, our method achieves a large improvement of $11.5$ points. Our performance on the synthetic datasets for the Kuka and Baxter robots is also significantly higher than [106]. We believe the main reason for this large improvements is the fact that our Render & Compare approach can directly use the shape of the entire robot rendered in the observed configuration for estimating the pose rather than detecting a small number of keypoints.

**Running time.** In Table 6.2 we report the running time of our method on the Panda-ORB dataset which consists of robot motion videos captured from 27 different viewpoints. The first observation is that the accuracy increases with the number of refinement iterations $K$ used at test-time, and the most significant improvements are during the first 3 iterations. The importance of using multiple network iterations during training is further discussed in section 6.6.7 of the appendix. We also report an online version of our approach that leverages temporal continuity. It runs the refiner with $K = 10$ iterations on the first frame

| | Ours (individual frames) | | | | | Ours (online) | DREAM [106] |
|---|---|---|---|---|---|---|---|
| | K=1 | K=2 | K=3 | K=5 | K=10 | K=1 | ResNet101-H |
| ADD | 28.5 | 72.8 | 79.1 | 80.4 | **80.7** | 80.6 | 69.1 |
| FPS | 16 | 8 | 4 | 2 | 1 | 16 | **30** |

Table 6.2: Benefits of iterative refinement and running time on Panda-ORB video sequence of robot trajectories. We report ADD and running time (frames per second, FPS) for a varying number of refiner iterations $K$. The frames are either considered individually, or the estimate is used to initialize the refiner in the subsequent frames (online) without additionnal temporal filtering.

| | CRAVES [247] synt | CRAVES [247] synt+real* | | ours synt | |
|---|---|---|---|---|---|
| PCK@0.2 | 95.66 | **99.55** | | 99.20 | |
| Error | | all | top 50% | all | top 50% |
| Joints (degrees) | | 11.3 | 4.74 | **5.49** | **3.22** |
| Trans xyz. (cm) | | 10.1 | 5.52 | **0.61** | **0.42** |
| Trans norm. (cm) | | 19.6 | 10.5 | **1.31** | **0.90** |
| Rot. (degrees) | | 10.3 | 5.29 | **4.12** | **2.91** |

Table 6.3: Results on the CRAVES-lab [247] dataset with unknown joint angles. We report average errors on *all* the images of the dataset, or on the *top 50%* images selected according to the best joint angle accuracy with respect to the ground truth. Networks are trained on synthetic data only (synt) or also using non-annotated real images of the robot (synt+real*).

of the video and then uses the output pose as the initialization for the next frame and so on, without any additional temporal filtering of the resulting 6D poses. This version runs at 16 frames per second (FPS) and achieves a similar performance as the full approach that considers each frame independently and runs at 1 FPS.

### 6.4.2 6D pose and joint angle estimation

We now evaluate the performance of our method in the more challenging scenario where the robot joint angles are unknown and need to be estimated jointly with the 6D pose from a single RGB image. Qualitative results on the considered datasets as well as on real images crawled from the web are shown in figure 6-5. Please see section 6.6.9 of the appendix for additional qualitative examples, and the project page [173] for a movie showing our predictions on several videos.

**Comparison with CRAVES [247].** CRAVES [247] is the state-of-the-art approach for this task. We consider the two datasets used in [247]. CRAVES-lab displays the OWI-535 4DoF in a lab environment and contains 20,000 RGB images of which 428 key frames are annotated with 2D robot keypoints, ground truth joint angles (not used by our method) and camera intrinsics. CRAVES-youtube is the second dataset containing real-world images crawled from YouTube depicting large variations in viewpoint, illumination conditions and

| CRAVES synt [247] | CRAVES synt+real* [247] | Ours, synt f=500 | f=1000 | f=1500 | f=2000 | f=best |
|---|---|---|---|---|---|---|
| 81.61 | 88.89 | 87.34 | **88.97** | 87.37 | 85.49 | 92.91 |

Table 6.4: PCK@0.2 on the CRAVES-Youtube dataset [247].

robot appearance. It contains 275 frames annotated with 2D keypoints but no camera intrinsic parameters, 6D pose or joint angle ground truth. In addition to metrics that measure the 6D pose and joint angle estimates, we report a 2D keypoint metric, PCK (percentage of keypoints), following [247]. We refer to section 6.6.6 of the appendix for details of the metrics and the evaluation protocol.

We compare with two variants of CRAVES, one trained only on synthetic images (synt), and one that also requires real non-annotated images (synt+real*). Our method is trained only using the $5,000$ provided synthetic images. We report results on CRAVES-lab in Table 6.3. To compare with the 2D keypoint metric PCK@0.2, we project in the image the 3D keypoints of our estimated robot state. On this metric, our method outperforms CRAVES trained only on synthetic images and achieves a near-perfect score, similar to their approach trained with real images. More importantly, we achieve much better results when comparing with the 3D metrics (joint angles error and translation/rotation error). CRAVES achieves high average errors when all images of the datasets are considered, which is due to the complexity of solving the nonlinear nonconvex 2D-to-3D optimization problem for recovering 6D pose and joint angles given 2D keypoint positions. Our method trained to directly predict the 6D pose and joint angles, achieves large improvements in precision. We reduce the translation error by a factor of 10, demonstrating robustness to depth ambiguities.

We also evaluated our method on CRAVES-youtube. On this dataset, the camera intrinisic parameters are unknown and cannot be used for projecting the estimated robot pose into the 2D image. We therefore report results for different hypothesis of (fixed) focal lengths for all the images of the dataset, as well as using an oracle (f=best) which selects the best focal length for each image. Results are reported in Table 6.4. For 2D keypoints, our method for $f = 1000$ achieves results superior to CRAVES while not requiring real training images. Our method also clearly outperforms CRAVES when selecting the best focal length. 3D ground truth is not available, but similar to CRAVES-lab we could expect large improvements in 3D accuracy.

**Experiments on 7DoF+ robots.** We also train our method for jointly predicting 6D pose and joint angles for the robots considered in section 6.4.1. We evaluate the 6D pose *and* joint angles accuracy using ADD. Results are reported in Table 6.1 in blue (last column). For the 7DoF robotic arms (Kuka and Panda), these results demonstrate a competitive or superior ADD accuracy compared to [106] for inferring the 3D geometry of a known robot, but our method does not require known joint angles. The more complex 15 DoF Baxter robot remains challenging although our qualitative results often show reasonable alignments. We discuss the failure modes of our approach in section 6.6.10 of the appendix.

| Reference point | ADD |
|---|---|
| on Root $\mathcal{P}_0$ | 75.02 |
| on Middle $\mathcal{P}_4$ | 79.45 |
| on Hand $\mathcal{P}_7$ | 00.00 |
| Centroid (ours) | **80.54** |

(a)

| Reference point | Volume ($cm^3$) | ADD |
|---|---|---|
| on $\mathcal{P}_5$ | 3092 | 74.40 |
| on $\mathcal{P}_2$ | 2812 | 75.06 |
| on $\mathcal{P}_1$ | 2763 | 74.89 |
| on $\mathcal{P}_0$ | 2660 | 75.02 |
| on $\mathcal{P}_4$ | 2198 | 79.45 |
| Centroid (ours) | - | **80.54** |

(b)

Table 6.5: **Analysis of the choice of reference point $O$.** Networks are trained and evaluated with known joint angles as in section 6.4.1. The reference point is placed on (a) a naively chosen part and (b) on one of the 5 largest parts. Our strategy of using the centroid of the imaged robot performs the best.

### 6.4.3 Analysis of parametrization choices

We analyze our method on the Panda-ORB dataset: it is the largest real dataset containing significant variations in joint angles and camera viewpoints and the Panda robot has a long kinematic chain with 8 DoF. We study the choice of reference point $O$ for the 6D pose update and the choice of the anchor part (see section 6.3.4).

**Reference point.** We train different networks with the reference point at the origin of the root $\mathcal{P}_0$, the part in the middle of the kinematic chain $\mathcal{P}_4$ and at the end of the kinematic chain $\mathcal{P}_7$. Results are reported in Table 6.5(a). We observe that the performance indeed depends on the choice of the reference point and our approach of using the centroid of the robot as the reference point performs the best. The network trained with the "Hand" part ($\mathcal{P}_7$, the end effector) as a reference point fails to converge because this part is often difficult to identify in the training images and its pose cannot be inferred from any other part because the robot is not a rigid object. We investigate picking the reference point on one of the five largest parts (measured by their 3D volume which is correlated with 2D visibility) in Table 6.5(b) again demonstrating our approach of using the centroid of the robot performs better than any of these specific parts.

**Choice of the anchor part.** Table 6.6 reports results using different strategies for chosing the anchor part during training and testing. First, in 6.6(a) we show that choosing different parts as one (fixed) anchor results in significant variation in the resulting performance. To mitigate this issue we consider in 6.6(b) a strategy where the anchor is picked randomly among the robot parts at each iteration (both during training and testing). This strategy performs better than always naively selecting the root $\mathcal{P}_0$ as anchor. By restricting the sampled anchors to the largest parts, our automatic strategy can also perform better than the best performing part $\mathcal{P}_4$.

| Anchor | Volume ($cm^3$) | ADD |
|---|---|---|
| $\mathcal{P}_5$ | 3092 | 68.01 |
| $\mathcal{P}_2$ | 2812 | 65.56 |
| $\mathcal{P}_1$ | 2763 | 60.40 |
| $\mathcal{P}_0$ | 2660 | 57.44 |
| $\mathcal{P}_4$ | 2198 | **69.54** |
| $\mathcal{P}_7$ | 637 | 63.40 |

(a)

| Anchor | ADD |
|---|---|
| Root part $\mathcal{P}_0$ | 57.44 |
| Middle part $\mathcal{P}_4$ | 69.54 |
| Hand part $\mathcal{P}_7$ | 63.40 |
| Random (all) | 64.28 |
| Random (5 largest) | 70.39 |
| Random (3 largest) | **71.36** |

(b)

Table 6.6: **Analysis of the choice of the anchor part.** Networks are trained and evaluated with unknown joint angles as in section 6.4.2. (a) Results when one fixed anchor part is used during training and testing. (b) Randomly selecting the anchor part among a given set of largest robot parts during refinement in both training and testing.

## 6.5 Conclusion

We have introduced a new *render & compare* approach to estimate the joint angles and the 6D camera-to-robot pose of an articulated robot from a single image demonstrating significant improvements over prior state-of-the-art for this problem. These results open-up exciting applications in visually guided manipulation or collaborative robotics without fiducial markers or time-consuming hand-eye calibration. To stimulate these applications, we released the training code as well as the pre-trained models for commonly used robots.

## 6.6 Appendix

The appendix of this chapter is organized as follows. In section 6.6.1, we provide the complete set of equations for the pose update and its relation to the reference point introduced in section 6.3.4. In section 6.6.2, we give the details of the pose loss $\mathcal{L}_a$ used in equation (6.3) in the main chapter. Sections 6.6.3, 6.6.4, 6.6.5 and 6.6.6 give details of the cropping strategy, the state initialization, training strategy and evaluation, respectively. In section 6.6.7, we provide additional experiments showing the benefits of the iterative formulation and studying the importance of running the refiner network for multiple iterations during training. In section 6.6.8, we discuss the applicability of our approach to real robotic problems. Section 6.6.9 presents additional qualitative examples randomly selected from the images in the datasets introduced in section 6.4. Finally, we discuss and illustrate the main failure modes of our approach in section 6.6.10. Additional examples are in the video available on the project page [173].

### 6.6.1 Pose update

Any rigid motion can be modeled by a transformation in $SE(3)$. The 6D pose of the anchor part of the robot at iteration $k+1$ is therefore defined by the composition of its current transformation w.r.t. to the camera coordinate system $\mathcal{T}_{C,a}^{k}$ at iteration $k$ composed

with a pose update transformation $\Delta\mathcal{T} \in SE(3)$:

$$\mathcal{T}_{C,a}^{k+1} = \mathcal{T}_{C,a}^{k} \circ \Delta\mathcal{T}, \tag{6.4}$$

as introduced in section 6.3.2 in the main chapter. In the following we present the equations that define this pose update. We follow the parametrization of DeepIM's [112] pose update and explicitly parametrize the pose update using a 3D reference point we call $O^k$. We choose to use the centroid of the estimated robot at the $k$-th iteration as $O^k$, but other choices are also possible. Our neural network refiner predicts parameters $v_x$, $v_y$, $v_z$ that are used to compute the 3D translation $\Delta t$. The refiner also predicts the 3D rotation $\Delta R$ that together with $\Delta t$ define the pose update $\Delta\mathcal{T}$ in (6.4). The parameters $v_x$ and $v_y$ correspond to the displacement in the image plane (in pixels) of the reference point $O^k$ at iteration $k$ and $v_z$ corresponds to a relative update of the depth of $O^k$, again relative to the camera coordinate system:

$$v_x = u_{O^k,x}^{k+1} - u_{O^k,x}^{k} = f_x^c \left( \frac{x_{O^k}^{k+1}}{z_{O^k}^{k+1}} - \frac{x_{O^k}^{k}}{z_{O^k}^{k}} \right) \tag{6.5}$$

$$v_y = u_{O^k,y}^{k+1} - u_{O^k,y}^{k} = f_y^C \left( \frac{y_{O^k}^{k+1}}{z_{O^k}^{k+1}} - \frac{y_{O^k}^{k}}{z_{O^k}^{k}} \right) \tag{6.6}$$

$$v_z = \frac{z_{O^k}^{k+1}}{z_{O^k}^{k}}, \tag{6.7}$$

where $u_{O^k}^{k} = (u_{O^k,x}^{k}, u_{O^k,y}^{k})$ is the 2D projection onto the image plane of the 3D point $O^k$ before applying the pose update, $u_{O^k}^{k+1}$ the 2D reprojection onto the image plane of $O^k$ after applying the pose update, $\left[x_{O^k}^{k}, y_{O^k}^{k}, z_{O^k}^{k}\right]$ are the 3D coordinates of the reference point $O^k$ before the pose update expressed in the camera frame, and $\left[x_{O^k}^{k+1}, y_{O^k}^{k+1}, z_{O^k}^{k+1}\right]$ are the 3D coordinates of the reference point $O^k$ after the pose update expressed in the camera frame. $f_x^C$ and $f_y^C$ are the intrinsic parameters of the virtual cropped camera that are assumed known and fixed.

From these equations, we can derive the update of the 3D translation of the 3D point $O^k$:

$$t_{O^k}^{k+1} = t_{O^k}^{k} + \Delta t_{O^k}, \tag{6.8}$$

with the $x$, $y$ and $z$ components of translation update $\Delta t_{O^k}$ defined as :

$$\Delta t_{O^k,x} = x_{O^k}^{k+1} - x_{O^k}^{k} = \frac{1}{f_x^C} v_x v_z z_{O^k}^{k} + x_{O^k}^{k}(v_z - 1) \tag{6.9}$$

$$\Delta t_{O^k,y} = y_{O^k}^{k+1} - y_{O^k}^{k} = \frac{1}{f_y^C} v_y v_z z_{O^k}^{k} + y_{O^k}^{k}(v_z - 1) \tag{6.10}$$

$$\Delta t_{O^k,z} = z_{O^k}^{k+1} - z_{O^k}^{k} = z_{O^k}^{k}(v_z - 1). \tag{6.11}$$

Note that $\Delta t_{O^k}$ depends on the 3D point $O^k$. Hence the network has to learn to internally infer this point to predict consistent transformation updates. The rotation update $\Delta R$ (parametrized by three angles) predicted by the refiner network is applied around the 3D

reference point $O^k$. This defines the transformation of any 3D point on the anchor. Let's consider a point $a$ on the anchor part in position $t_a^k$ at iteration $k$, its position at iteration $k+1$ is given by:

$$t_a^{k+1} = \Delta R(t_a^k - t_{O^k}^k) + t_{O^k}^k + \Delta t_{O^k}, \tag{6.12}$$

where $\Delta R$ is the rotation matrix predicted by the network using the same rotation parametrization used in [96, 246]. The equation (6.12) can also be used to define the rotation matrix of the anchor part with respect to the camera:

$$R_{C,a}^{k+1} = \Delta R R_{C,a}^k, \tag{6.13}$$

where $R_{C,a}^k$ defines the rotation of the anchor part with respect to the camera before the pose update, $R_{C,a}^{k+1}$ defines the rotation of the anchor part with respect to the camera after the pose update and $\Delta R$ is the rotation matrix predicted by the network. The equations (6.12) and (6.13) fully define the pose update given by equation (6.4) as a function of the network predictions $v_x$, $v_y$, $v_z$ and $\Delta R$, and the reference point $O^k$. For computing the loss in the next section, we write the pose $\mathcal{T}_{C,a}^{k+1}$ of the anchor part $a$ in the camera coordinate system $C$ at iteration $k+1$ (after the pose update) as

$$\mathcal{T}_{C,a}^{k+1} = \mathcal{U}([v_x, v_y, v_z], \Delta R), \tag{6.14}$$

where $\mathcal{U}$ expresses the updated pose of the anchor as a function of all network predictions $v_x$, $v_y$, $v_z$, $\Delta R$, and can be computed in a closed form using the equations derived above.

### 6.6.2 Pose loss

We define the following distance $D_a(\mathcal{T}_1, \mathcal{T}_2)$ to measure the distance between two transformations $\mathcal{T}_1$ and $\mathcal{T}_2$ using the 3D points $\mathcal{X}_a$ on the anchor part $a$ :

$$D_a(\mathcal{T}_1, \mathcal{T}_2) = \frac{1}{|\mathcal{X}_a|} \sum_{x \in \mathcal{X}_a} |\mathcal{T}_1 x - \mathcal{T}_2 x|, \tag{6.15}$$

where $|\cdot|$ is the $L_1$ norm. The pose loss $\mathcal{L}_a$ in equation (6.3) in the main chapter follows [96] and is written as:

$$\mathcal{L}_{pose} = D_a(\mathcal{U}([v_x, v_y, \hat{v}_z], \Delta\hat{R}), \hat{T}_{C,a}) \tag{6.16}$$

$$+ D_a(\mathcal{U}([\hat{v}_x, \hat{v}_y, v_z], \Delta\hat{R}), \hat{T}_{C,a}) \tag{6.17}$$

$$+ D_a(\mathcal{U}([\hat{v}_x, \hat{v}_y, \hat{v}_z], \Delta R), \hat{T}_{C,a}), \tag{6.18}$$

where $\mathcal{U}$ defines the pose update as explained in (6.14) in section 6.6.1, $\hat{T}_{C,A}$ is the ground truth 6D pose of the anchor with repect to the camera, $v_x, v_y, v_z, \Delta R$ are the parameters of the pose update predicted by the network, and $\hat{v}_x, \hat{v}_y, \hat{v}_z, \Delta\hat{R}$ are the values of these updates that would lead to the ground truth pose. The different terms of this loss separate (or disentangle) the influence of different subsets of parameters on the loss. In detail, the

first term (6.16) of the loss considers only the $xy$ translation of the anchor part, where the rest of the parameters are fixed to their ground truth values, $\hat{v}_z$ and $\Delta\hat{R}$ . The second term (6.17) of the loss considers only the relative depth of the anchor part where the rest of the parameters are fixed to their ground truth values. Finally, the last term (6.18) of the loss considers only the rotation update $\Delta R$ (6.18) where the rest of the parameters are fixed to their ground truth values.

### 6.6.3 Cropping strategy

Let $u_O = (u_{O,x}, u_{O,y})$ be the 2D projection of the 3D centroid $O$ of the robot by the camera with intrinsics $K$. We use a cropping strategy similar to DeepIM but using only the projection of the 3D points of the model in the current pose estimate to define the bounding box of the crop. Let $u_1 = (u_{1,x}, u_{1,y})$, $u_2 = (u_{2,x}, u_{2,y})$ be the coordinates of the upper left and lower right corners of the bounding box defined by the robot projection in the image, respectively. We define:

$$\Delta u_x = max(u_{dist,x}, u_{dist,y}/r) \cdot 2\lambda, \tag{6.19}$$
$$\Delta u_y = max(u_{dist,x}/r, u_{dist,y}) \cdot 2\lambda, \tag{6.20}$$

where

$$u_{dist,x} = max(|u_{1,x} - u_{O,x}|, |u_{2,x} - u_{O,x}|), \tag{6.21}$$
$$u_{dist,y} = max(|u_{1,y} - u_{O,y}|, |u_{2,y} - u_{O,y}|), \tag{6.22}$$

$r = 4/3$ is the aspect ratio of the crop and $\lambda = 1.4$ following DeepIM [112]. The crop is centered at $u_O$, of width $\Delta u_x$ and height $\Delta u_y$. An example of a crop is given in figure 6-6.

### 6.6.4 State initialization

Let $u_{det} = (u_{det,x}, u_{det,y})$ and $(\Delta u_{det} = \Delta u_{det,x}, \Delta u_{det,y})$ define the center and the size of the approximate 2D bounding box of the robot in the image. In our experiments, there is only one robot per image and thus we use the entire image as the 2D bounding box and denote the bounding box $u_{det}$. The orientation of the base of the robot $\mathcal{P}_0$ is set parallel to the axes of the camera with the $z$ axis pointing upwards. The centroid of the robot $O$ is set to match the center of the bounding box $u_{det}$. We make the first hypothesis of the depth of the centroid by setting $z_{C,O}^{guess} = 1\text{m}$ and use this initial value to estimate the coordinates $x$ and $y$ of the centroid in the camera frame:

$$x_{C,O}^{guess} = u_{det,x} \frac{z_{C,O}^{guess}}{f_x} \tag{6.23}$$
$$y_{C,O}^{guess} = u_{det,y} \frac{z_{C,O}^{guess}}{f_y}. \tag{6.24}$$

We then update the depth estimate $z_{C,O}^{guess}$ using the following simple strategy. We project the points of the robot using the initial guess we have just defined. These points define a

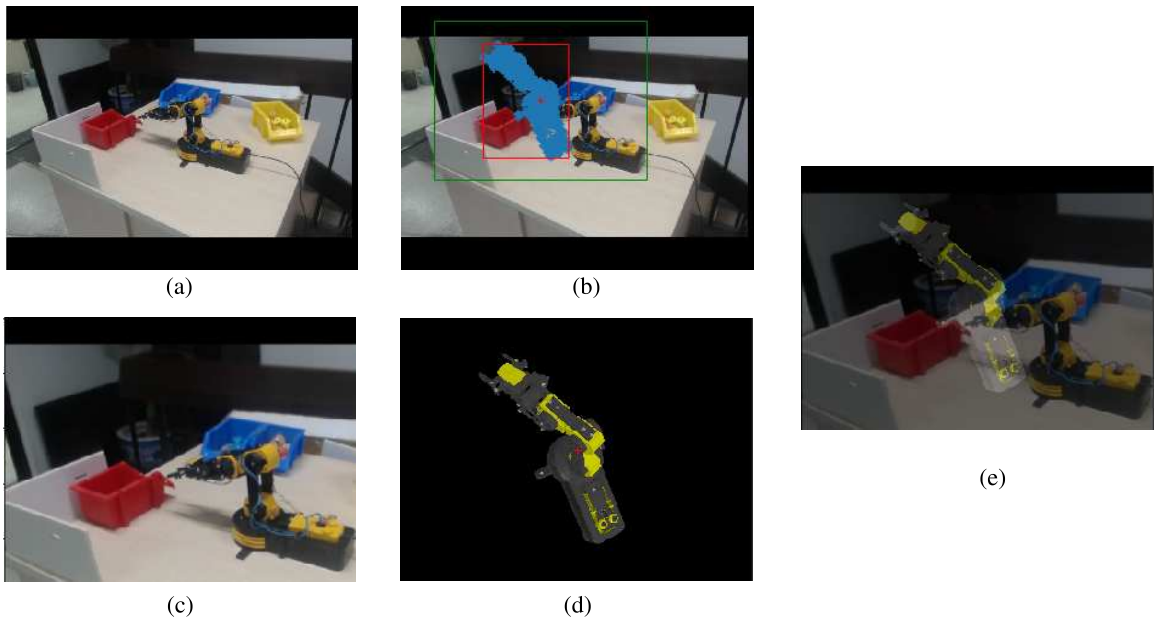



Figure 6-6: Cropping strategy. (a) input RGB image, (b) illustration of the cropping strategy: bounding box $[u_1, u_2]$ (shown in red) defined by the reprojection of the robot in the input state to the image (blue points). Bounding box defining the crop is shown in green. (c) The cropped input image, (d) Cropped rendered image, (e) The crop and the input image are overlaid. These two images are given as input to our network (the anchor mask is not displayed).

bounding box with dimensions $\Delta u_{guess,x} = (\Delta u_{guess,x}, \Delta u_{guess,y})$ and the center remains unchanged $u_{guess} = u_{det}$ by construction. We compute an updated depth of the centroid such that its width and height approximately match the size of the 2D detection:

$$z^0_{C,M} = z^{guess}_{C,O} \frac{1}{2} \left( f_x \frac{\Delta u_{guess,x}}{\Delta u_{det,x}} + f_y \frac{\Delta u_{guess,y}}{\Delta u_{det,y}} \right) \tag{6.25}$$

and use this new depth to compute $x^0_{C,O}$ and $y^0_{C,O}$ using equations (6.23) and (6.24) that were used to define $x^{guess}_{C,O}$ and $y^{guess}_{C,O}$. The initial state $\mathcal{S}^0$ of the robot is defined by $(x^0_{C,M}, y^0_{C,M}, z^0_{C,M})$, $R^0_{C,O} = Id$, and the initial joint angles are set (when they are not measured) to $q^0 = \frac{q^+ + q^-}{2}$, where $q^+$ and $q^-$ define the interval of the robot articulation angles. An example of initialization is shown in figure 6-7.

In the experiments of section 6.4 of the main chapter, we use the entire image as the 2D bounding box because the test images show a single robot. Please note however that it would be straightforward to process multiple bounding boxes like [96] for multiple rigid objects. If multiple robots are in the input bounding box (e.g. as in some of the examples in the results video on the project webpage [173]), our iterative refinement typically converges to the largest robot in the image.

(a) (b) (c)

Figure 6-7: Example of initialization. (a) Input RGB image, (b) robot in the initial state, (c) output of RoboPose (pose and joint angles are predicted) after $K = 10$ refinement iterations.

### 6.6.5 Training strategy

The training of the neural networks is performed with Adam [90] using a learning rate of $0.003$ and batches of 1408 images, split on 44 GPUs. The networks are trained for 60k iterations. Following recommendations from [49] for distributed training, we use a warm up-phase: the learning rate is linearly increased from to $0.$ to $0.003$ during the first 5k iterations. The learning rate is reduced to 0.0003 at 45k iterations. In order to speed up the training, we start by training with $K = 1$ refinement iterations and add other iterations $K = 2$, $K = 3$ at 15k and 30k iterations respectively. We found that starting to train with $K = 1$ and increasing the number of refinement iterations during training has no effect on the results, but allows to train the refiner network faster.

### 6.6.6 Evaluation details

Next, we give more details about the evaluation metric and the evaluation protocol of our comparison with DREAM [106] and CRAVES [247].

**Comparison with DREAM.** The average distance (ADD) metric measures the distance between 3D keypoints on the robot transformed with the predicted and ground truth 6D poses. The 3D keypoints are the same as in [106] for the Kuka, Panda and Baxter robots and are defined at the locations of robot joints. The ADD errors of the 3D keypoints for the robot depicted in the image are averaged, and we report the area-under-the-curve of the ADD pass rate vs. threshold following [106]. We refer to [106, 230] for more details about this metric. A slight difference with the evaluation protocol of [106] is that all images are considered as possible evaluation targets, whereas [106] discards images where there are fewer than 4 robot keypoints visible in the ground truth because PnP cannot be solved in this situation. This favors methods that rely on keypoints and does not fully consider the difficulty of estimating the pose of robots in situations with high self-occlusions, a situation that our method can handle. We thus evaluate ADD on all images, even those with fewer than four visible robot keypoints. The models from DREAM [106] were re-evaluated with this protocol and in practice the difference is only minor compared to the results reported in DREAM [106], less than $2\%$ for all datasets. For reproducing DREAM results reported

in Table 6.1 of the main chapter, we used the provided code and pre-trained models[1] using the instructions to reproduce the results from the paper, e.g. using the "shrink-and-crop" cropping strategy. Note that while evaluating these models, we found some results to be higher than reported in the original paper, e.g. the ResNet101-H is the best model on the largest real dataset Panda-ORB (69.05) but only the VGG19-F (61.93) was reported in [106]. We reported results for all models of DREAM [106] in Table 6.1 of the main chapter.

When evaluating our method that predicts joint angles, ADD is computed using 3D keypoints on the robot that are computed using both the estimated 6D pose and the predicted joint angles using the robot forward kinematics. This provides a principled way to measure the accuracy of the 3D reconstruction of the robot without giving arbitrary weights to rotation, translation or joint angle errors.

**Comparison with CRAVES.** For the comparison with CRAVES [247], we based our evaluation on the code provided with the paper[2]. The 3D metrics reported in Table 3 for CRAVES-lab are:

- Joints (degrees): Errors of the joint angle predictions over the 4 articulations $\frac{1}{4}\sum_{i=1}^{4}|q_i^{pred} - q_i^{gt}|$, where $q_i^{pred}$ is the predicted value of the joint angle for joint $i$ and $q_i^{gt}$ is the ground truth value of the joint angle for joint $i$.

- Trans xyz. (cm): Translation error for the base averaged over the $x, y, z$ axes: $\frac{1}{3}(|x^{pred} - x^{gt}| + |y^{pred} - y^{gt}| + |z^{pred} - z^{gt}|)$, where "pred" denotes the predicted and "gt" the ground truth values.

- Trans norm. (cm): Translation error for the base computed using the $L_2$ norm: $||t^{pred} - t^{gt}||_2$.

- Rot. (degrees): Errors between the Euler angles that define the rotation of the base with respect to the camera $\frac{1}{3}(|\theta_x^{pred} - \theta_x^{gt}| + |\theta_y^{pred} - \theta_y^{gt}| + |\theta_z^{pred} - \theta_z^{gt}|)$, where again "pred" denotes the predicted and "gt" the ground truth values.

We found that the "Joints", "Trans xyz." and "Rot." metrics reported in [247] under "3D pose estimation errors" were an average computed only over the top 50% images with the best predictions according to the joint angle errors. We also report the errors averaged over *all* images of the dataset (all) in Table 6.3 of the main chapter as our method can successfully recover the 3D configuration of the robot in all images of the dataset because it does not rely on solving the 2D-to-3D nonlinear, nonconvex optimization problem that is difficult to solve in some situations.

On CRAVES-Youtube only the visible 2D keypoints are considered as estimation targets to provide a fair comparison with "Youtube-vis" of [247]. The results that we reproduced with the provided pre-trained models were slightly lower (around 0.5% lower) than reported in the paper for the PCK@0.2 metric, but we kept the (higher) results reported in the paper when comparing with their method.

[1] https://github.com/NVlabs/DREAM
[2] https://github.com/zuoym15/craves.ai

| $K_{train}$ \ $K_{test}$ | 1 | 2 | 3 | 5 | 10 |
|---|---|---|---|---|---|
| 1 | 0.43 | 24.31 | 34.72 | 39.77 | 41.25 |
| 2 | 1.70 | 32.22 | 52.08 | 62.76 | 65.73 |
| 3 | 1.14 | 25.51 | 49.22 | 65.64 | 70.39 |
| 5 | 0.61 | 15.80 | 36.81 | 61.08 | **70.77** |

Table 6.7: **Benefits of the iterative refinement.** We study the influence of the number of training and testing iterations, denoted $K_{train}$ and $K_{test}$, respectively. We report the $ADD$ metric (higher is better) on the Panda ORB dataset with unknown joint angles. The best performing set-up is shown in **bold**.

### 6.6.7 Benefits of the iterative formulation

In this section, we investigate the benefits of the number of refiner iterations at training and test time. In the following, we denote as $K_{train}$ the number of refiner network iterations used during training (denoted $K$ in equation (6.3) of the main chapter) and as $K_{test}$ the number of iterations used at test time. We experimentally evaluate the influence of $K_{train}$ and $K_{test}$ using the ADD metric on the Panda-ORB dataset with unknown joint angles, similar to the experimental set-up reported in section 6.4.2. We trained four networks with $K_{train} = 1/2/3/5$ and report the results for a varying number of iterations at test time for each network. Results are reported in Table 6.7.

First, we observe that for each refiner network (each row in the table), the accuracy is significantly improved by using multiple iterations at test time as opposed to running the network for a single iteration ($K_{test} = 1$) in a single-shot regression fashion. These results demonstrate that the iterative formulation is crucial to the success of the approach.

Second, we observe that running the network for multiple iterations during training significantly improves the results. For example, for $K_{test} = 10$, the ADD results are improved from $41.25$ for $K_{train} = 1$ to $70.77$ for $K_{train} = 5$. Using multiple training iterations without backpropagating through the renderer augments the distribution of errors between the input state and the ground truth state with actual errors the refiner makes. These results show the importance of training the refiner network to correct both the large errors (between the initial state $\mathcal{S}^0$ and the ground truth state $\mathcal{S}^{gt}$) as well as smaller errors (between the prediction at iteration k, $\mathcal{S}^k$, and $\mathcal{S}^{gt}$) to simulate what the network is going to see at test-time where the network is run for multiple iterations. In our experiments in chapter 6, we used $K_{train} = 3$ as it is a good trade-off between performance and training speed.

### 6.6.8 Applications in robotic set-ups

Similar to related work [106, 247], our approach can be applied in real robotic set-ups. For example, the offline hand-eye calibration approach DREAM [106] has been used in [208] for a manipulation task on a real robot, and CRAVES [247] demonstrates visually guided closed-loop control of a low-cost robot without reliable joint angle measurements. Our approach significantly improves over the accuracy of DREAM [247] and CRAVES [106], and therefore we expect to also improve the capabilities of robotic systems similar to [208,

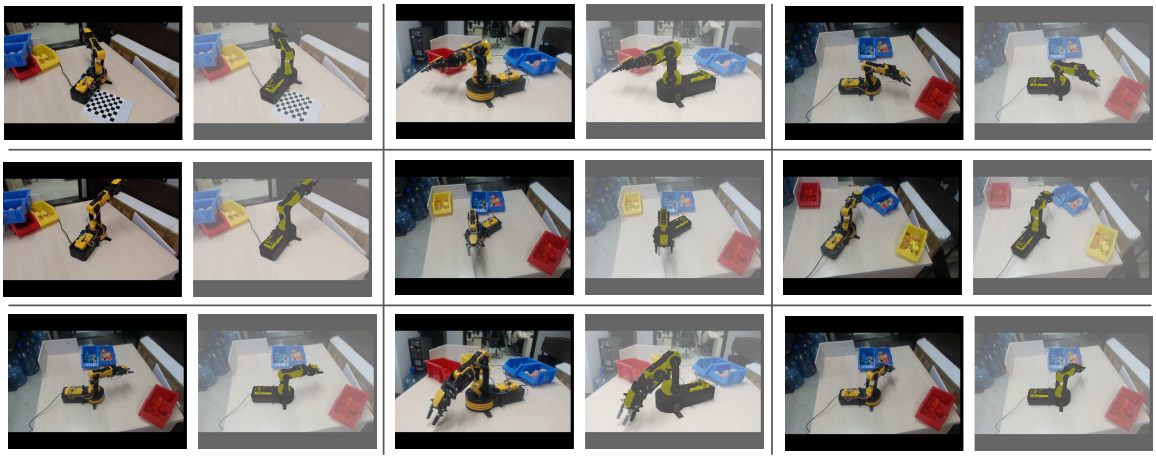


Figure 6-8: **Random selection of examples on the CRAVES-lab datasets**. Both joint angles and the 6D pose of the robot are predicted from the input image. For each of the 9 examples, we show the input RGB image (left) and the predicted state of the robot within the 3D scene (right). We illustrate the predicted state (right) by overlaying the articulated CAD model of the robot in the predicted state over the input image.

247]. Please note that while this chapter focuses on the robot state estimation from a single image, it is also suited for real-time and online applications as explained in section 6.4.1. In this scenario (also illustrated in the video on the project webpage [173]), our predictions could be temporally smoothed to reduce the jitter in the predictions by applying a simple temporal low-pass filter similar to [208].

### 6.6.9 Qualitative examples

**Random examples.** Here we provide more qualitative examples on the datasets where we have shown the quantitative evaluation. All the presented results consider the most challenging scenario where *the joint angles are unknown and are predicted* together with the robot's 6D pose. For each dataset, we have randomly sampled 9 images without any further manual selection and show the input RGB image and the output state predicted by our method. Results are presented for the CRAVES-lab (figure 6-8), CRAVES-youtube (figure 6-9), Panda ORB (figure 6-10), Kuka Photo (figure 6-11) and Baxter DR (figure 6-12) datasets.

### 6.6.10 Failure modes

There are four main failure modes and limitations of our approach in the most challenging set-up with unknown joint angles, illustrated in figure 6-13. First, while our method is tolerant to some amount of self-occlusion, it can fail in situations where multiple parts of the robot are occluded by external objects. Second, in some in-the-wild images (where camera intrinsics are also unknown), our iterative alignment can get stuck in a local minima. This could be improved by (i) using a better initialization (e.g. by using a separate external coarse estimation method) or by (ii) trying multiple initializations and selecting the best

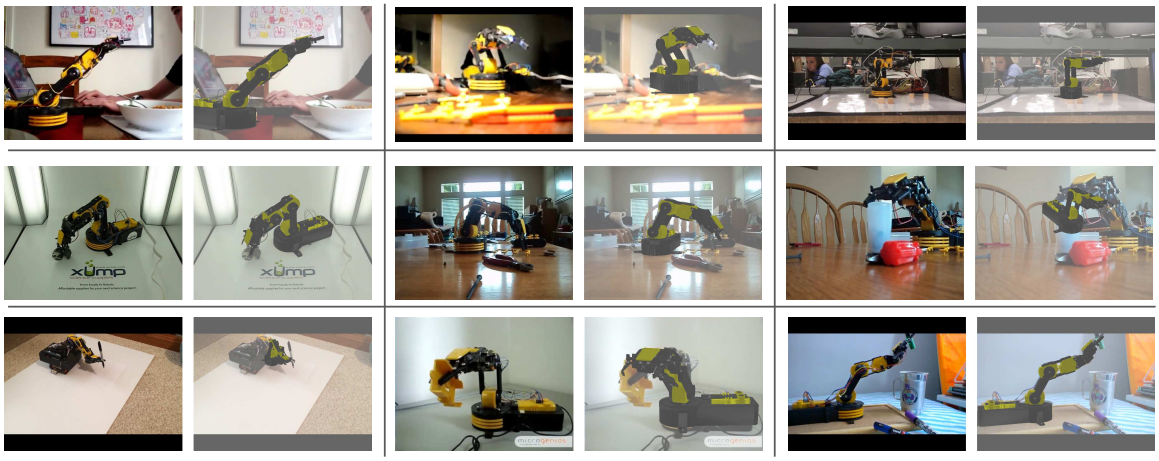


Figure 6-9: **Random selection of examples on the CRAVES-youtube dataset**. Both joint angles and the 6D pose of the robot are predicted from the input image. For each of the 9 examples, we show the input RGB image (left) and the predicted state of the robot within the 3D scene (right). We illustrate the predicted state (right) by overlaying the articulated CAD model of the robot in the predicted state over the input image.

result. The third failure mode is related to the symmetry of individual robot parts or of entire robot configurations that are not taken into account in our method. Finally, robots with many degrees of freedom such as the 15 DoF Baxter remain challenging (please see the quantitative evaluation in section 6.4.2 in the main chapter), although our qualitative results often show reasonable alignment, as illustrated in figure 6-12.

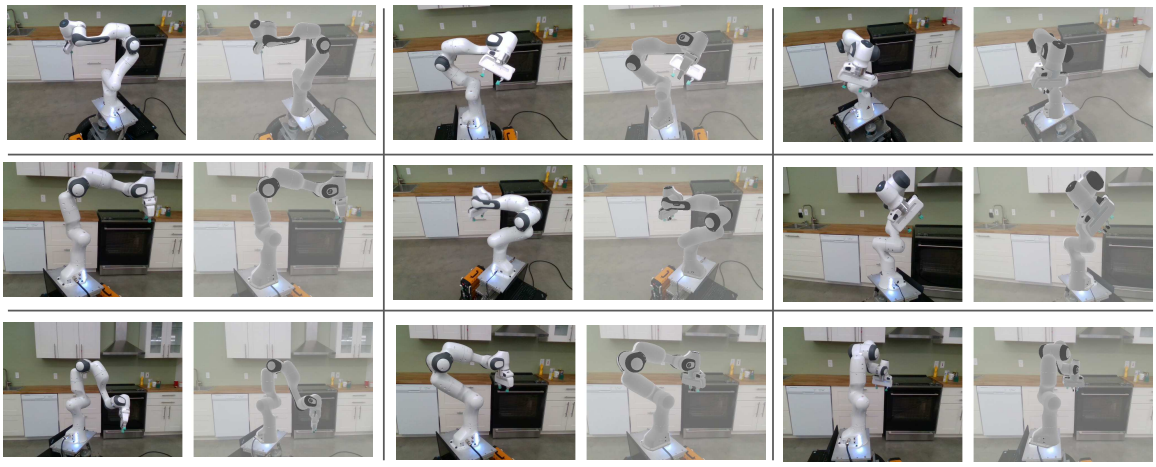


Figure 6-10: **Random selection of examples on the Panda ORB dataset**. Both joint angles and the 6D pose of the robot are predicted from the input image. For each of the 9 examples, we show the input RGB image (left) and the predicted state of the robot within the 3D scene (right). We illustrate the predicted state (right) by overlaying the articulated CAD model of the robot in the predicted state over the input image.

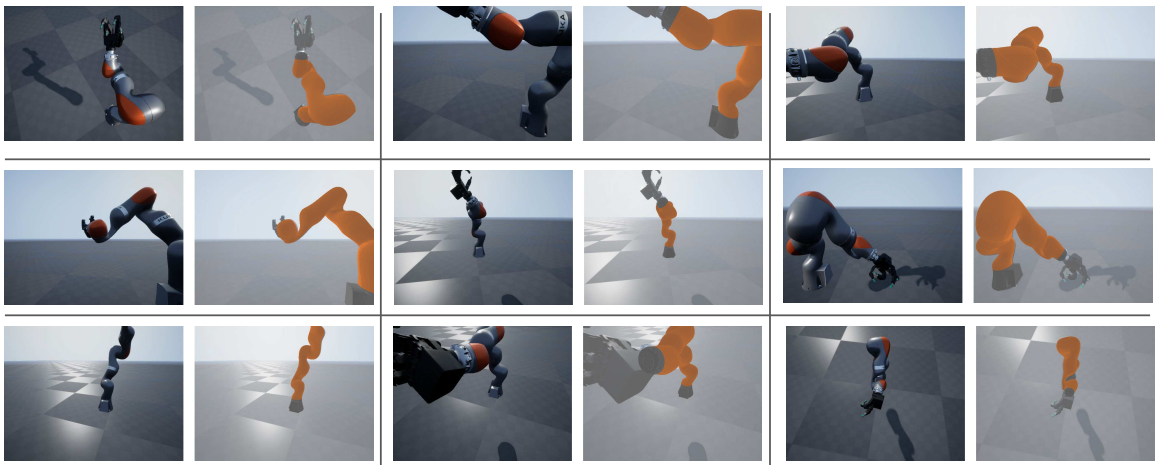


Figure 6-11: **Random selection of examples on the Kuka Photo dataset**. Both joint angles and the 6D pose of the robot are predicted from the input image. For each of the 9 examples, we show the input RGB image (on the left) and the predicted state of the robot within the 3D scene (on the right). We illustrate the predicted state (on the right) by overlaying the articulated CAD model of the robot in the predicted state over the input image.

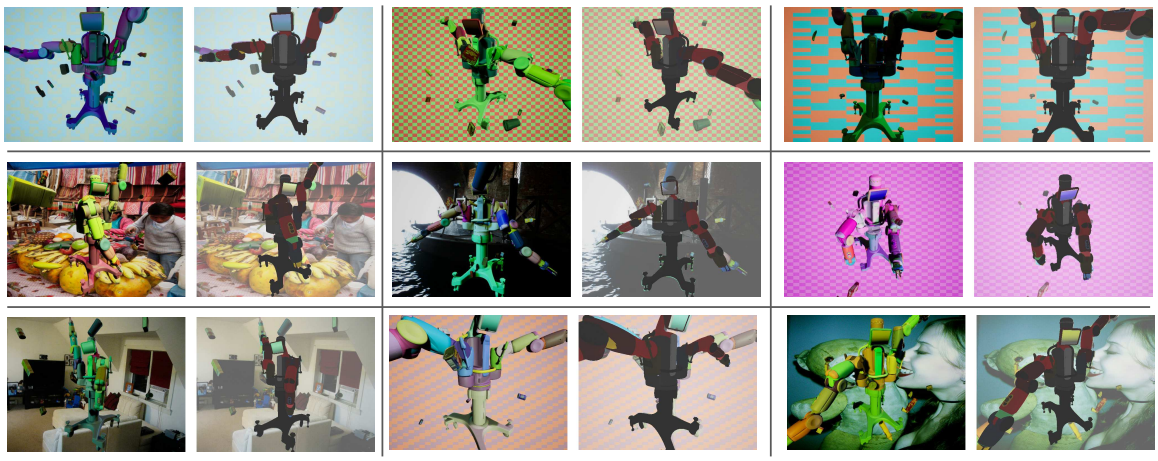


Figure 6-12: **Random selection of examples on the Baxter DR dataset**. Both joint angles and the 6D pose of the robot are predicted from the input image. For each of the 9 examples, we show the input RGB image (left) and the predicted state of the robot within the 3D scene (right). We illustrate the predicted state (right) by overlaying the articulated CAD model of the robot in the predicted state over the input image.

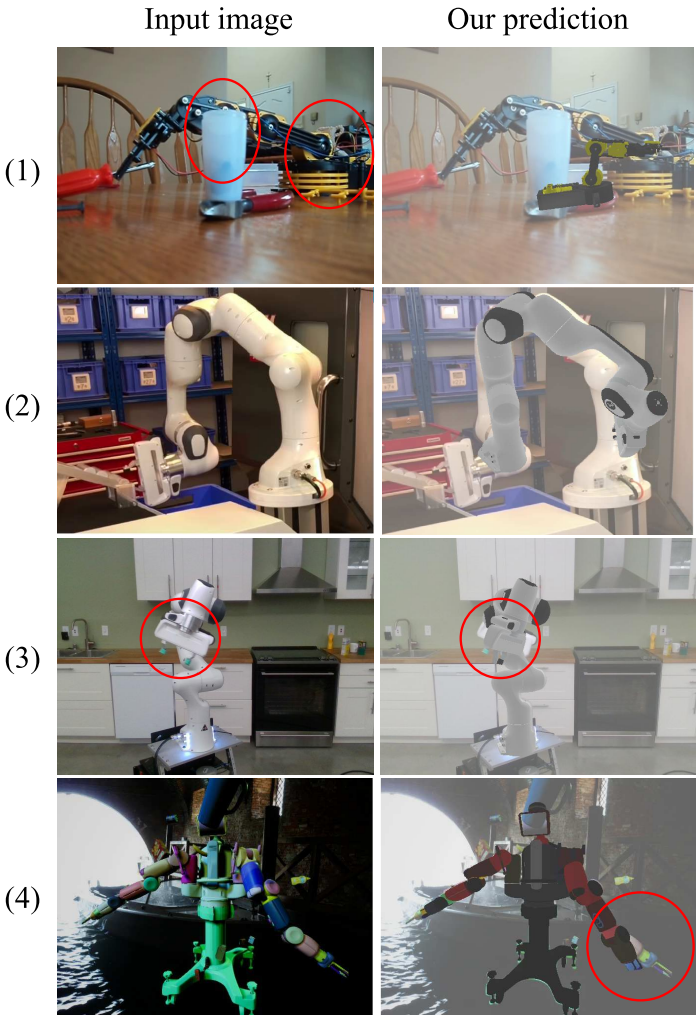



Figure 6-13: **Main failure modes.** We illustrate the four main failure modes of our approach. In (1), the severe occlusions (circled in red) of the base of the robot and of one of the parts in the middle of the kinematic chain lead to an incorrect prediction. In (2), the state of the robot is incorrectly estimated because our iterative procedure gets stuck in a local minimum. In (3), the gripper of the robot (circled in red) is a symmetric part, which leads to an incorrect estimate of the joint angle between the gripper and the rest of the robot. In (4), we illustrate an example of incorrect alignment of parts (circled in red) for the complex 15 DoF Baxter robot.

# Chapter 7

# Conclusions

In this chapter, we summarize the main results and contributions of the thesis, present recent developments in the literature that rely on the presented work, and suggest multiple avenues for future research.

## 7.1 Contributions

In this thesis, we have proposed methods for identifying the state of assets with known descriptions within a scene using visual sensors. In particular, we focused on estimating the pose of rigid objects as well as the pose and joint angles of articulated robots. The methods proposed in this thesis can be applied to a wide variety of assets, including textureless or symmetric objects, and robots with many degrees of freedom. We proposed learning-based methods which can be trained entirely on synthetic data without manual annotations and can be deployed in real environments. The approaches were evaluated on real challenging scenes depicting clutter and various illumination conditions, demonstrating significant improvements over previous state-of-the-art on several benchmarks. The methods are also fast and practical. They can be used online during execution on a real robotic system and can be deployed in un-instrumented environments using cheap RGB sensors, without a requirement for fiducial markers or time-consuming extrinsic camera calibration.

In chapter 3, we have proposed a multi-object calibration-free deep neural network for estimating object positions within a robot workspace. The network directly maps image pixels to 2D coordinates in the robot's coordinate frame, using the (known) robot as an implicit calibration target. We have also developed an efficient algorithm for planar rearrangement planning of multiple objects based on Monte-Carlo Tree Search. Visual state estimation and planning modules were combined to demonstrate a full system on a real robot that can operate from a hand-held camera and is robust to external perturbations of the object positions during execution.

In chapter 4, we have developed methods for single-view and multi-view 6D pose estimation of known rigid objects. We first proposed a new method based on the iterative render-and-compare strategy for single-view pose estimation. We introduced a new training loss for pose estimation that handles symmetric objects and disentangles the effect of the different components of the pose predictions. Our learning-based method leverages heavy

data augmentation and advances in deep neural network architecture design for extracting visual features and predicting a 3D rotation. It achieves significant improvements over the previous state-of-the-art methods, winning the BOP challenge 2020. We then proposed a multi-view method that addresses the failure modes inherent to single-view pose estimation. We developed (i) an object-level RANSAC procedure to match 6D object detections across different camera viewpoints and filter-out single-view outliers; and (ii) a multi-view refinement strategy that recovers a single consistent scene by solving an object-level bundle adjustment problem.

In chapter 5, we have addressed an important limitation of most learning-based methods for 6D pose estimation of rigid objects. Namely, the best-performing methods require the mesh model of the objects to be known in advance for generating synthetic data and training, thus making them unusable in many robotic scenarios where rapid deployment to novel objects is key. We introduced MegaPose, a learning-based method trained on a large-scale synthetic dataset of thousands of objects that generalizes to novel objects given their mesh model only available during inference. We first presented a 6D pose refiner based on the render-and-compare strategy which can be applied to novel objects. The shape and coordinate system of the novel object are provided as inputs to the network by rendering multiple synthetic views of the object's CAD model. We then introduced a novel approach for coarse pose estimation which leverages a network trained to classify whether a pose error between a synthetic rendering and an observed image of the same object can be corrected by the refiner. We demonstrate the performance of our approach is competitive with other methods that require the object models to be known in advance and show it can be used to grasp novel objects with a real robot.

In chapter 6 we have presented a method to recover the full state of a robot within a scene given an RGB image. We present an approach based on the render-and-compare strategy that iteratively updates the 6D pose of the robot and its joint angles. We highlight multiple choices of update parametrization are possible because one robot part attached to others can be moved in 3D space by the 6D transformation of the anchor part, or by joint angle displacements. We show the importance of the selection of the anchor part, and propose an automatic strategy where the anchor part is randomly sampled among the largest robot parts at each iteration of the iterative pose estimation procedure. We demonstrate our approach significantly outperforms existing state-of-the-art and we notably present results on robots with up to 15 degrees of freedom.

## 7.2 Recent developments and applications

In this section, we present examples of recent works in the literature that build on the methods introduced in this thesis. The examples are illustrated in figure 7-1

**Focal length and pose estimation.** The approach introduced in chapters 4, 5 and 6 assume the intrinsic parameters of the camera are known. For many "in-the-wild" images such as internet pictures or archival photographs, these parameters are unknown. This problem was addressed in [167] where a novel method for jointly estimating the pose of a known object and the focal length of the camera is presented. FocalPose - the method introduced in [167] - builds on CosyPose which we introduced in chapter 4 and takes inspiration from

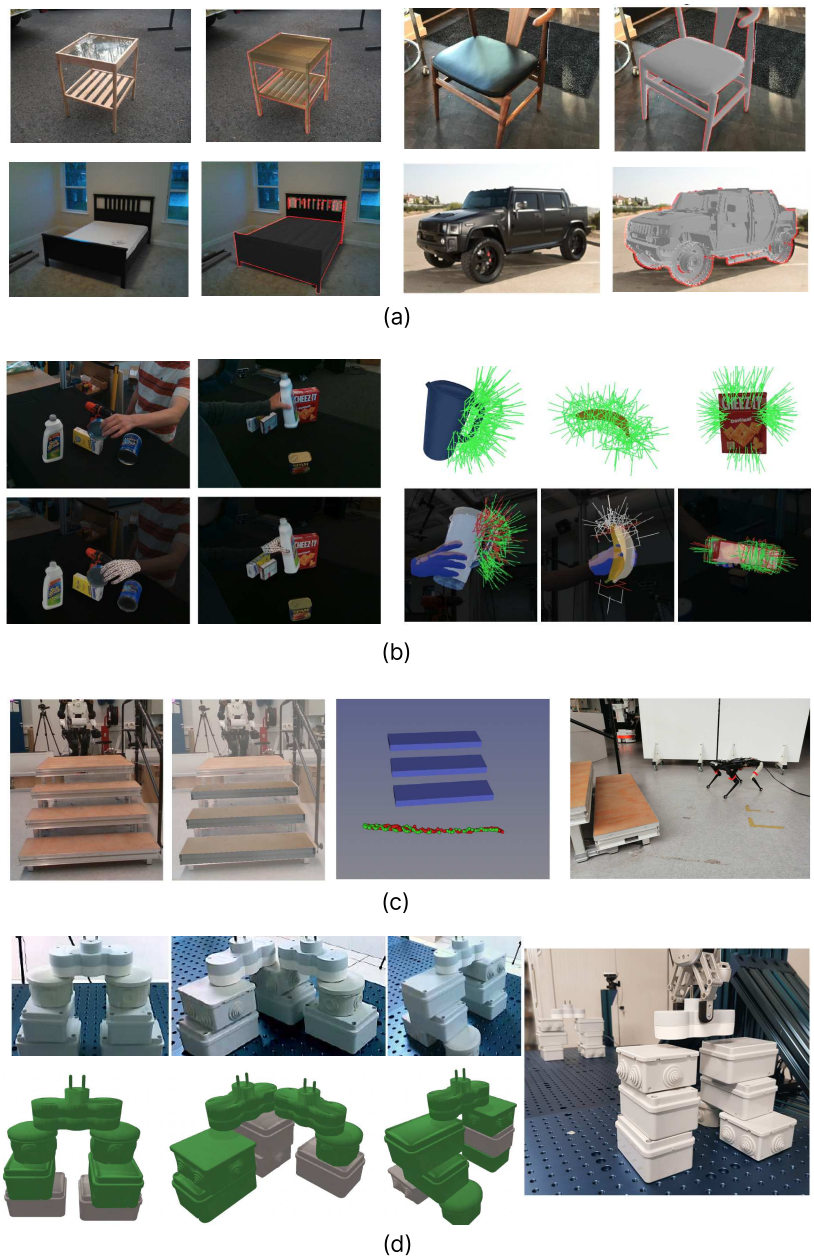



Figure 7-1: **Recent works in the literature building on the methods introduced in this thesis.** In (a) [167], FocalPose [167] builds on CosyPose (chapter 4) to predict jointly the focal length of the camera and an object pose. Results are presented for "in-the-wild" images of furniture like sofas, beds, tables; and large objects like cars. In (b)[18], CosyPose (chapter 4) is used to predict the pose of YCB-V object during active human manipulation (left). The poses are then used to generate robot grasps for human-robot handover (right). In (c)[30], CosyPose (chapter 4) is used to predict stair steps (left). The information is combined with the IMU of a quadruped robot to recover a robot trajectory in real-time (right). In (d)[17], CosyPose is combined with physical constraints and task-and-motion planning tools to perform assembly planning with textureless objects on a real robotic platform.

the pose update and disentangled loss of CosyPose to address the scenario where the focal length of the camera is unknown. In this work, the CAD model of an observed object is used as an implicit calibration target, which draws similarities with our work presented in chapter 3 where the known robot is used as implicit calibration for performing object position estimation without assuming the focal length of the camera is known.

**Reconstructing hand-object interaction.** In [18], CosyPose is used to estimate the 6D pose of objects during active manipulation by a human. The estimated poses are used to generate robotic grasps and experiments on a real robot demonstrate the estimated poses can be used to perform human-to-robot handover.

**Object-level SLAM.** In [30], CosyPose is trained to estimate the poses of stair steps using the onboard camera of a quadruped robot. An object-level SLAM method is presented to fuse the predicted poses with inertial measurements to reliably estimate the pose of the robot in real-time. The output is the trajectory of the robot with respect to a set of candidate contact surfaces, which could be combined with model predictive control (MPC) to allow the robot to walk on the stairs.

**Assembly planning of textureless objects.** In [17], CosyPose is used to estimate the pose of textureless industrial objects to perform assembly planning. Instead of using multiple views for correcting failure modes inherent to single-view pose estimation, this method proposes to use a combination of physical stability constraints, convex optimization, and Monte-Carlo Tree search. The proposed approach is robust to detection errors and experiments presented on a real robot demonstrate the accuracy of CosyPose is sufficient to grasp textureless objects.

These examples of recent works building on CosyPose illustrate that the works introduced in this thesis open-up applications in the domains of robotics and augmented reality.

## 7.3 Future research directions

In this section we outline several directions for future research.

### 7.3.1 Motion planning

- In chapter 3, we propose a formulation of the planar rearrangement planning problem that can be solved with Monte-Carlo Tree Search (MCTS). An interesting direction is to consider other planning problems such as planning the motion of a robotic arm in presence of obstacles.

- The efficiency and speed of the MCTS solver could be improved using learned action policy and value function introduced in Alpha Go [187]. For example [192] uses a deep-learning based action policy for non-prehensile rearrangement planning.

### 7.3.2 Single-view 6D object detection

- The ablation study of section 5.4.3 on the number of objects used to train MegaPose suggests the performance could be improved with a larger number of training objects.

Additional meshes could be obtained by considering other datasets such as Abc [91], Amazon Berkeley Objects (ABO) [24], or Common Objects in 3D (CO3D) [170].

- The MegaPose method could be used for estimating the 6D pose of a camera outdoor given 3D models of buildings, or to localize an indoor camera in a 3D-scanned building.

- The running time of MegaPose could be improved with a more efficient rendering pipeline. In the current implementation, 70% of the running time is spent rendering images and transferring them back and forth between GPU and CPU due to technical incompatibilities between renderer based on OpenGL and Pytorch. Running time could be improved by rendering images in batch and converting buffers directly to pytorch tensors.

- The coarse estimation strategy used in MegaPose relies on a network that predicts a similarity score between an object in a specific pose in the rendered and observed images. This network could be used to score pose hypotheses for sensor fusion, e.g. for a SLAM application. Another interesting direction is to investigate if this score is correlated with discrepancies between the mesh of the rendered object and the real observed object. A score discriminative of both object pose and appearance could be used for fine-grained object classification and pose verification.

### 7.3.3 Multi-view pose estimation

- The multi-view approach presented in chapter 4 could be extended to leverage temporal continuity. For example, the candidate matching stage could be run only once. The object hypotheses would be tracked with a single-view tracker (e.g. CosyPose or MegaPose) in each separate view and refined globally by solving object-level bundle adjustment in real-time.

- The running-time performance of the bundle adjustment solver could be improved by using an efficient C++ solver such as CERES [2].

- Physical constraints could be leveraged to further improve the reconstruction accuracy. For example, a term penalizing penetration between the objects could be added to the loss of the global object-level bundle adjustment problem. This term could be made differentiable using recent developments in differentiable contact detection [139].

### 7.3.4 Beyond 6D pose estimation with render-and-compare

- In RoboPose introduced in chapter 6, different networks are used for each robot. The robot kinematics are encoded in the weights of the networks and learned during training. Future work should consider developing a single network that can be applied to novel robots without re-training. The robot kinematics could be provided as input to the network by showing multiple images of the robot in different configurations,

or the robot kinematics could be integrated explicitly inside the network using the (differentiable) forward kinematics.

- In MegaPose introduced in chapter 5, the information on a novel object's coordinate system is implicitly provided as input to the network by providing multiple views of the object. The same idea could be used to provide information about the basis of an arbitrary transformation. For example, polyhedrons could be deformed iteratively by a render-and-compare network to match parts of the environment. The visual effects of network predictions (e.g. a 3-axis relative scale deformation of the polyhedron) would be provided as input to the network. Such an approach could be used to reconstruct walls, floors, or contact surfaces for a walking robot, and directly be used for motion planning. A network based on render&compare could also be used to iteratively refine deformable object models [146, 182, 183].

- The render&compare networks used in this thesis predict a 6D pose that aligns two images, the input image and a rendering. A similar network could be used to predict 6D motion between multiple real images with applications in computational photography, e.g. super-resolution or HDR from burst images [105, 227].

### 7.3.5 Predicting SE-3 transforms for robotic manipulation

- The scope of the methods presented in this thesis is limited to an object-level reconstruction of a scene. Future work should explore how to use reconstructed scenes for robotic manipulation. MegaPose could for example be used to extract the trajectory of specific object poses during active manipulation, e.g. bottle and mugs during pouring. Extracting the object-to-object pose trajectory on many human demonstrations could be used to extract object-centric motions [114, 161, 222] that could be transferred to a robot.

- The networks (e.g. in MegaPose) presented in this thesis present effective ways for predicting 6D pose updates given 2D RGB images. The information learned in this network could be re-used for other tasks that bypass the explicit object-level representation of a scene. For example, the networks trained for 6D pose estimation could be used as pre-training for policies with actions in the form of a rigid gripper motion.

# Bibliography


[1] Ben Abbatematteo, Stefanie Tellex, and George Konidaris. "Learning to Generalize Kinematic Models to Novel Objects". In: *CoRL*. 2020.

[2] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver*. 2022.

[3] Rachid Alami, Thierry Simeon, and Jean-Paul Laumond. "A Geometrical Approach to Planning Manipulation Tasks. The Case of Discrete Placements and Grasps". In: *ISRR*. 1990.

[4] *Apple ARKit*. URL: https://developer.apple.com/augmented-reality/arkit/.

[5] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. "Seeing 3D Chairs: Exemplar Part-Based 2D-3D Alignment Using a Large Dataset of CAD Models". In: *CVPR*. 2014.

[6] N. Ayache and O. D. Faugeras. "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects". In: *TPAMI* (1986).

[7] Roman Bachmann, Jörg Spörri, Pascal Fua, and Helge Rhodin. "Motion Capture from Pan-Tilt Cameras with Unknown Orientation". In: *3DV*. 2019.

[8] Sid Yingze Bao and Silvio Savarese. "Semantic Structure from Motion". In: *CVPR*. 2011.

[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *ECCV*. 2006.

[10] P.J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *TPAMI* (1992).

[11] Tolga Birdal and Slobodan Ilic. "Point Pair Features Based Object Detection and Pose Estimation Revisited". In: *3DV*. 2015.

[12] Jeannette Bohg, Javier Romero, Alexander Herzog, and Stefan Schaal. "Robot Arm Pose Estimation through Pixel-Wise Part Classification". In: *ICRA*. 2014.

[13] Rodney A. Brooks. "Symbolic Reasoning among 3-D Models and 2-D Images". In: *Artificial Intelligence* (1981).

[14] Fabrizio Caccavale and Masaru Uchiyama. "Cooperative Manipulation". In: *Springer Handbook of Robotics*. 2016.

[15] Guillaume Caron, Amaury Dame, and Eric Marchand. “Direct Model Based Visual Tracking and Pose Estimation Using Mutual Information”. In: *Image and Vision Computing* (2014).

[16] *Celebrating Robotics in Seattle: NVIDIA Opens New AI Research Lab*. URL: https://www.youtube.com/watch?v=JT2viTz_0jU.

[17] Thomas Chabal, Robin Strudel, Etienne Arlaud, Jean Ponce, and Cordelia Schmid. “Assembly Planning from Observations under Physical Constraints”. In: *IROS*. 2022.

[18] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. “DexYCB: A Benchmark for Capturing Hand Grasping of Objects”. In: *CVPR*. 2021.

[19] Xu Chen, Zijian Dong, Jie Song, Andreas Geiger, and Otmar Hilliges. “Category Level Object Pose Estimation via Neural Analysis-by-Synthesis”. In: *ECCV*. 2020.

[20] Changhyun Choi and Henrik I. Christensen. “3D Pose Estimation of Daily Objects Using an RGB-D Camera”. In: *IROS*. 2012.

[21] Alex Clark. *Pillow (PIL Fork)*. 2015.

[22] Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. “The MOPED Framework: Object Recognition and Pose Estimation for Manipulation”. In: *IJRR* (2011).

[23] Alvaro Collet and Siddhartha S. Srinivasa. “Efficient Multi-View Object Recognition and Full Pose Estimation”. In: *ICRA*. 2010.

[24] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. “ABO: Dataset and Benchmarks for Real-World 3D Object Understanding”. In: *CVPR*. 2022.

[25] Dorin Comaniciu and Peter Meer. “Mean Shift: A Robust Approach Toward Feature Space Analysis”. In: *TPAMI* (2002).

[26] *CosyPose Project Page*. URL: https://www.di.ens.fr/willow/research/cosypose/.

[27] N Dalal and B Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *CVPR*. 2005.

[28] Amaury Dame and Eric Marchand. “Mutual Information-Based Visual Servoing”. In: *TRO* (2011).

[29] Neil Dantam, Zachary Kingston, Swarat Chaudhuri, and Lydia Kavraki. “Incremental Task and Motion Planning: A Constraint-Based Approach”. In: *RSS*. 2016.

[30] César Debeunne, Médéric Fourmy, Yann Labbé, Pierre-Alexandre Léziart, Guilhem Saurel, Joan Solà, and Nicolas Mansard. “CosySlam: Investigating Object-Level SLAM for Detecting Locomotion Surfaces”. 2022.

[31] Daniel F. Dementhon and Larry S. Davis. “Model-Based Object Pose in 25 Lines of Code”. In: *IJCV* (1995).

[32] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. “PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking”. In: *RSS*. 2019.

[33] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. “BlenderProc”. 2019. arXiv: arXiv:1911.01911.

[34] Karthik Desingh, Shiyang Lu, Anthony Opipari, and Odest Chadwicke Jenkins. “Factored Pose Estimation of Articulated Objects Using Efficient Nonparametric Belief Propagation”. In: *ICRA*. 2019.

[35] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. “Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd”. In: *CVPR*. 2016.

[36] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. “Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items”. In: *ICRA*. 2022.

[37] Bertram Drost and Slobodan Ilic. “3D Object Detection and Localization Using Multimodal Point Pair Features”. In: *3DV*. 2012.

[38] Bertram Drost, Markus Ulrich, Paul Bergmann, Philipp Härtinger, and Carsten Steger. “Introducing MVTec ITODD — A Dataset for 3D Object Recognition in Industry”. In: *ICCV Workshops*. 2017.

[39] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. “Model Globally, Match Locally: Efficient and Robust 3D Object Recognition”. In: *CVPR*. 2010.

[40] Olivier Faugeras and M. Hebert. “The Representation, Recognition, and Locating of 3-D Objects”. In: *IJRR* (1986).

[41] Mark Fiala. “ARTag, a Fiducial Marker System Using Digital Techniques”. In: *CVPR*. 2005.

[42] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* (1981).

[43] Pascal Fua and Vincent Lepetit. “Vision Based 3D Tracking and Pose Estimation for Mixed Reality”. In: *Emerging Technologies of Augmented Reality: Interfaces and Design*. 2007.

[44] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. “PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning”. In: *ICAPS*. 2020.

[45] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco Madrid-Cuevas, and Manuel J. Marín-Jiménez. “Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion”. In: *Pattern Recognition* (2014).

[46] Mathieu Gonzalez, Amine Kacete, Albert Murienne, and Eric Marchand. “L6DNet: Light 6 DoF Network for Robust and Precise Object Pose Estimation With Small Datasets”. In: *RAL* (2021).

[47] Mathieu Gonzalez, Eric Marchand, Amine Kacete, and Jérôme Royan. “Twist-SLAM: Constrained SLAM in Dynamic Environment”. In: *IEEE Robotics and Automation Letters* (2022).

[48] Mathieu Gonzalez, Eric Marchand, Amine Kacete, and Jerome Royan. “S3LAM: Structured Scene SLAM”. In: *IROS*. 2022.

[49] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. 2018. arXiv: 1706.02677.

[50] W. Eric L. Grimson and Tomas Dideriksen. “Localizing Overlapping Parts by Searching the Interpretation Tree”. In: *TPAMI* (1987).

[51] Michael D. Grossberg and Shree K. Nayar. “A General Imaging Model and a Method for Finding Its Parameters”. In: *ICCV*. 2001.

[52] Shuai D Han, Nicholas M Stiffler, Athanasios Krontiris, Kostas E. Bekris, and Jingjin Yu. “High-Quality Tabletop Rearrangement with Overhand Grasps: Hardness Results and Fast Methods”. In: *RSS*. 2017.

[53] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE transactions on Systems Science and Cybernetics* (1968).

[54] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[55] Rasmus Laurvig Haugaard and Anders Glent Buch. “SurfEmb: Dense and Continuous Correspondence Distributions for Object Pose Estimation with Learnt Surface Embeddings”. In: *CVPR*. 2022.

[56] Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S. Sukhatme. “Active Articulation Model Estimation through Interactive Perception”. In: *ICRA*. 2015.

[57] Giray Havur, Guchan Ozbilgin, Esra Erdem, and Volkan Patoglu. “Geometric Rearrangement of Multiple Movable Objects on Cluttered Surfaces: A Hybrid Reasoning Approach”. In: *ICRA*. 2014.

[58] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask R-CNN”. In: *ICCV*. 2017.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CVPR*. 2016.

[60] Yisheng He, Yao Wang, Haoqiang Fan, Jian Sun, and Qifeng Chen. “FS6D: Few-Shot 6D Pose Estimation of Novel Objects”. In: *CVPR*. 2022.

[61] J Heller, M Havlena, A Sugimoto, and T Pajdla. “Structure-from-Motion Based Hand-Eye Calibration Using L$\infty$minimization”. In: *CVPR*. 2011.

[62] *Helper Robots We Desperately Need: the HRP-2 Dishbot*. URL: https://www.popsci.com/gadgets/article/2010-04/dishbot-cleans-house-learns-live-uncertainty/.

[63] S Hinterstoisser, S Holzer, C Cagniart, S Ilic, K Konolige, N Navab, and V Lepetit. "Multimodal Templates for Real-Time Detection of Texture-Less Objects in Heavily Cluttered Scenes". In: *ICCV*. 2011.

[64] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. "Gradient Response Maps for Real-Time Detection of Textureless Objects". In: *TPAMI* (2012).

[65] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. "Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects". In: *CVPR*. 2010.

[66] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes". In: *ACCV*. 2012.

[67] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. "Going Further with Point Pair Features". In: *ECCV*. 2016.

[68] T Hodan, P Haluza, Š Obdržálek, J Matas, M Lourakis, and X Zabulis. "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-Less Objects". In: *WACV*. 2017.

[69] Tomas Hodan, Daniel Barath, and Jiri Matas. "EPOS: Estimating 6D Pose of Objects with Symmetries". In: *CVPR*. 2020.

[70] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, and Xenophon Zabulis. "Bop: Benchmark for 6d Object Pose Estimation". In: *ECCV*. 2018.

[71] Tomas Hodan, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiri Matas. "BOP Challenge 2020 on 6D Object Localization". In: *ECCV Worskhops*. 2020.

[72] Tomáš Hodaň. "Pose Estimation of Specific Rigid Objects". PhD thesis. Czech Technical University, 2021.

[73] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. "On Evaluation of 6D Object Pose Estimation". In: *ECCV Workshops*. 2016.

[74] Radu Horaud and Fadi Dornaika. "Hand-Eye Calibration". In: *IJRR* (1995).

[75] *https://newatlas.com/amazon-picking-challenge-icra-2015/37773/*. URL: https://newatlas.com/amazon-picking-challenge-icra-2015/37773/.

[76] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. "Segmentation-Driven 6D Object Pose Estimation". In: *CVPR*. 2019.

[77] Daniel P. Huttenlocher and Shimon Ullman. “Recognizing Rigid Objects by Aligning Them with an Image”. In: *IJCV* (1987).

[78] Daniel P. Huttenlocher and Shimon Ullman. “Recognizing Solid Objects by Alignment with an Image”. In: *IJCV* (1990).

[79] Jarmo Ilonen and Ville Kyrki. “Robust Robot-Camera Calibration”. In: *ICAR*. 2011.

[80] *Introducing Spot*. URL: https://www.youtube.com/watch?v=tf7IEVTDjng.

[81] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. “Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks”. In: *CVPR*. 2019.

[82] Frédéric Jurie and Michel Dhome. “A Simple and Efficient Template Matching Algorithm”. In: *ICCV*. 2001.

[83] Leslie Pack Kaelbling and Tomás Lozano-Pérez. “Hierarchical Task and Motion Planning in the Now”. In: *ICRA*. 2011.

[84] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. “Homebreweddb: Rgb-d Dataset for 6d Pose Estimation of 3d Objects”. In: *ICCV Workshops*. 2019.

[85] Dov Katz and Oliver Brock. “Manipulating Articulated Objects with Interactive Perception”. In: *ICRA*. 2008.

[86] Dov Katz, Moslem Kazemi, J. Andrew Bagnell, and Anthony Stentz. “Interactive Segmentation, Tracking, and Kinematic Modeling of Unknown 3D Articulated Objects.” In: *ICRA*. 2013.

[87] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. “SSD-6D: Making RGB-based 3D Detection and 6D Pose Estimation Great Again”. In: *ICCV*. 2017.

[88] Beomjoon Kim, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. “Learning to Guide Task and Motion Planning Using Score-Space Representation”. In: *ICRA*. 2017.

[89] Eunyoung Kim and Gerard Medioni. “3D Object Recognition in Range Images Using Visibility Context”. In: *IROS*. 2011.

[90] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR*. 2015.

[91] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. “ABC: A Big CAD Model Dataset For Geometric Deep Learning”. In: *CVPR*. 2019.

[92] Rebecca König and Bertram Drost. “A Hybrid Approach for 6DoF Pose Estimation”. In: *ECCV Workshops*. 2020.

[93] Athanasios Krontiris and Kostas E. Bekris. “Dealing with Difficult Instances of Object Rearrangement”. In: *RSS*. 2015.

[94] Athanasios Krontiris and Kostas E. Bekris. “Efficiently Solving General Rearrangement Tasks: A Fast Extension Primitive for an Incremental Sampling-Based Planner”. In: *ICRA*. 2016.

[95] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. “Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images”. In: *ICCV*. 2015.

[96] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. “CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation”. In: *ECCV*. 2020.

[97] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. *Extended version. CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation*. 2021. URL: https://drive.google.com/file/d/1YVemqAEb_aL174XPYD-wag-DW7mUnvzC/view.

[98] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. “Single-View Robot Pose and Joint Angle Estimation via Render & Compare”. In: *CVPR*. 2021.

[99] Yann Labbé, Lucas Manuelli, Arsalan Mousavian, Stephen Tyree, Stan Birchfield, Jonathan Tremblay, Justin Carpentier, Mathieu Aubry, Dieter Fox, and Josef Sivic. “MegaPose: 6D Pose Estimation of Novel Objects via Render & Compare”. In: *CoRL*. 2022.

[100] Yann Labbé, Sergey Zagoruyko, Igor Kalevatykh, Ivan Laptev, Justin Carpentier, Mathieu Aubry, and Josef Sivic. “Monte-Carlo Tree Search for Efficient Visually Guided Rearrangement Planning”. In: *IEEE Robotics and Automation Letters* (2020).

[101] Jens Lambrecht and Linh Kästner. “Towards the Usage of Synthetic Data for Marker-Less Pose Estimation of Articulated Robots in RGB Images”. In: *ICAR*. 2019.

[102] Zoe Landgraf, Raluca Scona, Tristan Laidlow, Stephen James, Stefan Leutenegger, and Andrew J. Davison. “SIMstack: A Generative Shape and Instance Model for Unordered Object Stacks”. In: *ICCV*. 2021.

[103] Jean-Claude Latombe. *Robot Motion Planning*. Springer Science & Business Media, 2012.

[104] Steven M. LaValle. *Planning Algorithms*. Cambridge university press, 2006.

[105] Bruno Lecouat, Thomas Eboli, Jean Ponce, and Julien Mairal. “High Dynamic Range and Super-Resolution from Raw Image Bursts”. In: *SIGGRAPH*. 2022.

[106] Timothy E. Lee, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Oliver Kroemer, Dieter Fox, and Stan Birchfield. “Camera-to-Robot Pose Estimation from a Single Image”. In: *ICRA*. 2020.

[107] Vincent Lepetit and Pascal Fua. “Monocular Model-Based 3d Tracking of Rigid Objects: A Survey”. In: *Foundations and Trends® in Computer Graphics and Vision* (2005).

[108] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “EPnP: An Accurate O(n) Solution to the PnP Problem”. In: *IJCV* (2009).

[109] Chi Li, Jin Bai, and Gregory D Hager. “A Unified Framework for Multi-View Multi-Class Object Pose Estimation”. In: *ECCV*. 2018.

[110] Fu Li, Ivan Shugurov, Benjamin Busam, Minglong Li, Shaowu Yang, and Slobodan Ilic. “PolarMesh: A Star-Convex 3D Shape Approximation for Object Pose Estimation”. In: *IEEE Robotics and Automation Letters* (2022).

[111] Xiaolong Li, He Wang, Li Yi, Leonidas J. Guibas, A. Lynn Abbott, and Shuran Song. “Category-Level Articulated Object Pose Estimation”. In: *CVPR*. 2020.

[112] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. “DeepIM: Deep Iterative Matching for 6D Pose Estimation”. In: *ECCV*. 2018.

[113] Zhigang Li, Gu Wang, and Xiangyang Ji. “Cdpn: Coordinates-based Disentangled Pose Network for Real-Time Rgb-Based 6-Dof Object Pose Estimation”. In: *ICCV*. 2019.

[114] Zongmian Li, Jiri Sedlar, Justin Carpentier, Ivan Laptev, Nicolas Mansard, and Josef Sivic. “Estimating 3D Motion and Forces of Human-Object Interactions from Internet Videos”. In: *CVPR*. 2019.

[115] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. “Feature Pyramid Networks for Object Detection”. In: *CVPR*. 2017.

[116] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. “Focal Loss for Dense Object Detection”. In: *ICCV*. 2017.

[117] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. “Microsoft COCO: Common Objects in Context”. In: *ECCV*. 2014.

[118] Yunzhi Lin, Jonathan Tremblay, Stephen Tyree, Patricio A. Vela, and Stan Birchfield. “Single-Stage Keypoint-Based Category-Level Object Pose Estimation from an RGB Image”. In: *ICRA*. 2022.

[119] Lahav Lipson, Zachary Teed, Ankit Goyal, and Jia Deng. “Coupled Iterative Refinement for 6D Multi-Object Pose Estimation”. In: *CVPR*. 2022.

[120] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector”. In: *ECCV*. 2016.

[121] Yuan Liu, Yilin Wen, Sida Peng, Cheng Lin, Xiaoxiao Long, Taku Komura, and Wenping Wang. “Gen6D: Generalizable Model-Free 6-DoF Object Pose Estimation from RGB Images”. In: *ECCV*. 2022.

[122] Vianney Loing, Renaud Marlet, and Mathieu Aubry. “Virtual Training for a Real Application: Accurate Object-Robot Relative Localization Without Calibration”. In: *IJCV* (2018).

[123] David G Lowe. “Three-Dimensional Object Recognition from Single Two-Dimensional Images”. In: *Artificial Intelligence* (1987).

[124] David G. Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *ICCV*. 1999.

[125] Qingkai Lu and Tucker Hermans. “Modeling Grasp Type Improves Learning-Based Grasp Planning”. In: *RAL* (2019).

[126] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. “Dex-Net 3.0: Computing Robust Robot Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning”. In: *ICRA*. 2018.

[127] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. “Deep Model-Based 6D Pose Refinement in RGB”. In: *ECCV*. 2018.

[128] Fabian Manhardt, Gu Wang, Benjamin Busam, Manuel Nickel, Sven Meier, Luca Minciullo, Xiangyang Ji, and Nassir Navab. *CPS++: Improving Class-level 6D Pose and Shape Estimation From Monocular Images With Self-Supervised Learning*. 2020. arXiv: 2003.05848.

[129] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. “KPAM: KeyPoint Affordances for Category-Level Robotic Manipulation”. In: *ISRR*. 2019.

[130] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. “Pose Estimation for Augmented Reality: A Hands-On Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* (2016).

[131] Roberto Martín-Martín, Sebastian Höfer, and Oliver Brock. “An Integrated Approach to Visual Perception of Articulated Objects”. In: *ICRA*. 2016.

[132] J Matas, O Chum, M Urban, and T Pajdla. “Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions”. In: *BMCV*. 2002.

[133] John Mccormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. “Fusion++: Volumetric Object-Level SLAM”. In: *3DV*. 2018.

[134] *MegaPose Project Page*. URL: https://megapose6d.github.io/.

[135] Jean-Philippe Mercier, Mathieu Garon, Philippe Giguère, and Jean-François Lalonde. “Deep Template-based Object Instance Detection”. In: *WACV*. 2021.

[136] A. Mian, M. Bennamoun, and R. Owens. “On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes”. In: *IJCV* (2010).

[137] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. “Pose Estimation of Kinematic Chain Instances via Object Coordinate Regression”. In: *BMVC*. 2015.

[138] J Mirabel, S Tonneau, P Fernbach, A Seppälä, M Campana, N Mansard, and F Lamiraux. “HPP: A New Software for Constrained Motion Planning”. In: *IROS*. 2016.

[139] Louis Montaut, Quentin Le Lidec, Vladimir Petrik, Josef Sivic, and Justin Carpentier. “Collision Detection Accelerated: An Optimization Perspective”. In: *RSS*. 2022.

[140] Rémi Munos. “From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning”. In: *Foundations and Trends® in Machine Learning* (2014).

[141] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *TRO* (2015).

[142] Raúl Mur-Artal and Juan D. Tardós. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”. In: *TRO* (2017).

[143] Hiroshi Murase and Shree K. Nayar. “Visual Learning and Recognition of 3-d Objects from Appearance”. In: *IJCV* (1995).

[144] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. “Kinectfusion: Real-time Dense Surface Mapping and Tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 2011.

[145] Van Nguyen Nguyen, Yinlin Hu, Yang Xiao, Mathieu Salzmann, and Vincent Lepetit. “Templates for 3D Object Pose Estimation Revisited: Generalization to New Objects and Robustness to Occlusions”. In: *CVPR*. 2022.

[146] David Novotny, Ignacio Rocco, Samarth Sinha, Alexandre Carlier, Gael Kerchenbaum, Roman Shapovalov, Nikita Smetanin, Natalia Neverova, Benjamin Graham, and Andrea Vedaldi. “KeyTr: Keypoint Transporter for 3D Reconstruction of Deformable Objects in Videos”. In: *CVPR*. 2022.

[147] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. “Generalized Feedback Loop for Joint Hand-Object Pose Estimation”. In: *TPAMI* (2019).

[148] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. “Training a Feedback Loop for Hand Pose Estimation”. In: *ICCV*. 2015.

[149] Brian Okorn, Qiao Gu, Martial Hebert, and David Held. “ZePHyR: Zero-shot Pose Hypothesis Rating”. In: *ICRA*. 2021.

[150] Clark F. Olson and Daniel P. Huttenlocher. “Automatic Target Recognition by Matching Oriented Edge Pixels”. In: *IEEE Transactions on Image Processing* (1997).

[151] Edwin Olson. “AprilTag: A Robust and Flexible Visual Fiducial System”. In: *ICRA*. 2011.

[152] Anton Osokin, Denis Sumin, and Vasily Lomakin. “OS2D: One-Stage One-Shot Object Detection by Matching Anchor Features”. In: *ECCV*. 2020.

[153] Frank C. Park and Bryan J. Martin. “Robot Sensor Calibration: Solving AX=XB on the Euclidean Group”. In: *TRO* (1994).

[154] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. “LatentFusion: End-to-End Differentiable Reconstruction and Rendering for Unseen Object Pose Estimation”. In: *CVPR*. 2020.

[155] Kiru Park, Timothy Patten, and Markus Vincze. “Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation”. In: *ICCV*. 2019.

[156] Karl Pauwels and Danica Kragic. “SimTrack: A Simulation-Based Framework for Scalable Real-Time Object Pose Detection and Tracking”. In: *IROS*. 2015.

[157] Karl Pauwels, Leonardo Rubio, and Eduardo Ros. “Real-Time Model-Based Articulated Object Pose Detection and Tracking with Variable Rigidity Constraints”. In: *CVPR*. 2014.

[158] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G. Derpanis, and Kostas Daniilidis. “6-DoF Object Pose from Semantic Keypoints”. In: *ICRA*. 2017.

[159] Chris Paxton, Vasumathi Raman, Gregory D Hager, and Marin Kobilarov. “Combining Neural Networks and Tree Search for Task and Motion Planning in Challenging Environments”. In: *IROS*. 2017.

[160] Sida Peng, Yuan Liu, Qixing Huang, Hujun Bao, and Xiaowei Zhou. “PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation”. In: *CVPR*. 2019.

[161] Vladimír Petrík, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. “Learning Object Manipulation Skills via Approximate State Estimation from Real Videos”. In: *CoRL*. 2020.

[162] *Photo by Universal Robots USA*.

[163] Sudeep Pillai and John Leonard. “Monocular SLAM Supported Object Recognition”. In: *RSS*. 2015.

[164] Giorgia Pitteri, Slobodan Ilic, and Vincent Lepetit. “CorNet: Generic 3D Corners for 6D Pose Estimation of New Objects without Retraining”. In: *ICCV Workshops*. 2019.

[165] Giorgia Pitteri, Michaël Ramamonjisoa, Slobodan Ilic, and Vincent Lepetit. “On Object Symmetries and 6D Pose Estimation from Images”. In: *3DV*. 2019.

[166] Robert Pless. “Using Many Cameras as One”. In: *CVPR*. 2003.

[167] Georgy Ponimatkin, Yann Labbé, Bryan Russell, Mathieu Aubry, and Josef Sivic. “Focal Length and Object Pose Estimation via Render and Compare”. In: *CVPR*. 2022.

[168] *Project Page: Monte-Carlo Tree Search for Efficient Visually Guided Rearrangement Planning*. URL: https://ylabbe.github.io/rearrangement-planning/.

[169] Mahdi Rad and Vincent Lepetit. “BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth”. In: *ICCV*. 2017.

[170] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. “Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction”. In: *ICCV*. 2021.

[171] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *NeurIPS*. 2016.

[172] Lawrence G Roberts. “Machine Perception of Three-Dimensional Solids”. Massachusetts Institute of Technology, 1963.

[173] *RoboPose Project Page*. URL: https://di.ens.fr/willow/robopose.

[174] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. "3D Object Modeling and Recognition Using Affine-Invariant Patches and Multi-View Spatial Constraints". In: *CVPR*. 2003.

[175] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints". In: *IJCV* (2006).

[176] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[177] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. "Sim2Real View Invariant Visual Servoing by Recurrent Control". In: *CVPR*. 2018.

[178] R F Salas-Moreno, R A Newcombe, H Strasdat, P H J Kelly, and A J Davison. "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects". In: *CVPR*. 2013.

[179] Samuele Salti, Federico Tombari, and Luigi Di Stefano. "SHOT: Unique Signatures of Histograms for Surface and Texture Description". In: *Computer Vision and Image Understanding* (2014).

[180] Johannes L. Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *CVPR*. 2016.

[181] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *ECCV*. 2016.

[182] Agniva Sengupta, Alexandre Krupa, and Eric Marchand. "Tracking of Non-Rigid Objects Using RGB-D Camera". In: *SMC*. 2019.

[183] Agniva Sengupta, Alexandre Krupa, and Eric Marchand. "Visual Tracking of Deforming Objects Using Physics-based Models". In: *ICRA*. 2021.

[184] C. E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* (1948).

[185] Ivan Shugurov, Fu Li, Benjamin Busam, and Slobodan Ilic. "OSOP: A Multi-Stage One Shot Object Pose Estimation Framework". In: *OSOP*. 2022.

[186] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* (2016).

[187] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. "Mastering the Game of Go without Human Knowledge". In: *Nature* (2017).

[188] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. “Manipulation Planning with Probabilistic Roadmaps”. In: *IJRR* (2004).

[189] Andrea Simonelli, Samuel Rota Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. “Disentangling Monocular 3D Object Detection”. In: *ICCV*. 2019.

[190] *Smart Maintenance: La maintenance industrielle 4.0*. URL: https://www.audace-digital-learning.fr/smart-maintenance-la-maintenance-industrielle-4-0/.

[191] Chen Song, Jiaru Song, and Qixing Huang. “HybridPose: 6D Object Pose Estimation under Hybrid Representations”. In: *CVPR*. 2020.

[192] Haoran Song, Joshua A. Haustein, Weihao Yuan, Kaiyu Hang, Michael Yu Wang, Danica Kragic, and Johannes A. Stork. “Multi-Object Rearrangement with Monte Carlo Tree Search: A Case Study on Planar Nonprehensile Sorting”. In: *IROS*. 2020.

[193] S Srivastava, E Fang, L Riano, R Chitnis, S Russell, and P Abbeel. “Combined Task and Motion Planning through an Extensible Planner-Independent Interface Layer”. In: *ICRA*. 2014.

[194] Mike Stilman, Jan-Ullrich Schamburek, James Kuffner, and Tamim Asfour. “Manipulation Planning Among Movable Obstacles”. In: *ICRA*. 2007.

[195] Francisco Suárez-Ruiz, Xian Zhou, and Quang-Cuong Pham. “Can Robots Assemble an IKEA Chair?” In: *Science Robotics* (2018).

[196] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. “OnePose: One-shot Object Pose Estimation without CAD Models”. In: *CVPR*. 2022.

[197] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, Narunas Vaskevicius, Kai O. Arras, and Rudolph Triebel. “Multi-Path Learning for Object Pose Estimation Across Domains”. In: *CVPR*. 2020.

[198] Martin Sundermeyer, Tomas Hodan, Yann Labbé, Gu Wang, Eric Brachmann, Bertram Drost, Carsten Rother, and Jiri Matas. *BOP Challenge 2022 on Detection, Segmentation and Pose Estimation of Specific Rigid Objects*. 2023. arXiv: 2302.13075.

[199] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. “Implicit 3D Orientation Learning for 6D Object Detection from RGB Images”. In: *ECCV*. 2018.

[200] Richard Szeliski and Sing Bing Kang. “Recovering 3D Shape and Motion from Image Streams Using Nonlinear Least Squares”. In: *Journal of Visual Communication and Image Representation* (1994).

[201] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *ICML*. 2019.

[202] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. “Real-Time Seamless Single Shot 6D Object Pose Prediction”. In: *CVPR*. 2018.

[203] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". In: *IROS*. 2017.

[204] Federico Tombari, Alessandro Franchi, and Luigi Di. "BOLD Features to Detect Texture-less Objects". In: *ICCV*. 2013.

[205] Marc Toussaint. "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning". In: *IJCAI*. 2015.

[206] Marc Toussaint and Manuel Lopes. "Multi-Bound Tree Search for Logic-Geometric Programming in Cooperative Manipulation Domains". In: *ICRA*. 2017.

[207] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects". In: *CoRL*. 2018.

[208] Jonathan Tremblay, Stephen Tyree, Terry Mosier, and Stan Birchfield. "Indirect Object-to-Robot Pose Estimation from an External Monocular RGB Camera". In: *IROS*. 2020.

[209] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. "Bundle Adjustment — A Modern Synthesis". In: *Vision Algorithms: Theory and Practice*. 2000.

[210] R.Y. Tsai and R.K. Lenz. "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration". In: *TRO* (1989).

[211] Stephen Tyree, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Jeffrey Smith, and Stan Birchfield. "6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark". In: *IROS*. 2022.

[212] Ranjith Unnikrishnan and Martial Hebert. "Multi-Scale Interest Regions from Unorganized Point Clouds". In: *CVPR Workshops*. 2008.

[213] Joel Vidal, Chyi-Yeu Lin, Xavier Lladó, and Robert Martí. "A Method for 6D Pose Estimation of Free-Form Rigid Objects Using Point Pair Features on Range Data". In: *Sensors* (2018).

[214] Kentaro Wada, Stephen James, and Andrew J. Davison. "ReorientBot: Learning Object Reorientation for Specific-Posed Placement". In: *ICRA*. 2022.

[215] Kentaro Wada, Stephen James, and Andrew J. Davison. "SafePicking: Learning Safe Object Extraction via Object-Level Mapping". In: *ICRA*. 2022.

[216] Kentaro Wada, Edgar Sucar, Stephen James, Daniel Lenton, and Andrew J. Davison. "MoreFusion: Multi-object Reasoning for 6D Pose Estimation from Volumetric Fusion". In: *CVPR*. 2020.

[217] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion". In: *CVPR*. 2019.

[218] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. "GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation". In: *CVPR*. 2021.

[219] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. "Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation". In: *CVPR*. 2019.

[220] Jingwen Wang, Martin Rünz, and Lourdes Agapito. "DSP-SLAM: Object Oriented SLAM with Deep Shape Priors". In: *3DV*. 2021.

[221] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. "CaTGrasp: Learning Category-Level Task-Relevant Grasping in Clutter from Simulation". In: *ICRA*. 2022.

[222] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. "You Only Demonstrate Once: Category-Level Manipulation from Single Visual Demonstration". In: *RSS*. 2022.

[223] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E. Bekris. "Se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains". In: *IROS*. 2020.

[224] Felix Widmaier, Daniel Kappler, Stefan Schaal, and Jeannette Bohg. "Robot Arm Pose Estimation by Pixel-Wise Regression of Joint Angles". In: *ICRA*. 2016.

[225] Gordon Wilfong. "Motion Planning in the Presence of Movable Obstacles". In: *Annals of Mathematics and Artificial Intelligence* (1991).

[226] Paul Wohlhart and Vincent Lepetit. "Learning Descriptors for Object Recognition and 3D Pose Estimation". In: *CVPR*. 2015.

[227] Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. "Handheld Multi-Frame Super-Resolution". In: *SIGGRAPH*. 2019.

[228] Jimmy Wu, Bolei Zhou, Rebecca Russell, Vincent Kee, Syler Wagner, Mitchell Hebert, Antonio Torralba, and David M S Johnson. "Real-Time Object Pose Estimation with Pose Interpreter Networks". In: *IROS*. 2018.

[229] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. "3D ShapeNets: A Deep Representation for Volumetric Shapes". In: *CVPR*. 2015.

[230] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". In: *RSS*. 2018.

[231] Yang Xiao, Yuming Du, and Renaud Marlet. "PoseContrast: Class-Agnostic Object Viewpoint Estimation in the Wild with Pose-Aware Contrastive Learning". In: *3DV*. 2021.

[232] Yang Xiao, Vincent Lepetit, and Renaud Marlet. "Few-Shot Object Detection and Viewpoint Estimation for Objects in the Wild". In: *TPAMI* (2022).

[233] Yang Xiao and Renaud Marlet. “Few-Shot Object Detection and Viewpoint Estimation for Objects in the Wild”. In: *ECCV*. 2020.

[234] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry, and Renaud Marlet. “Pose from Shape: Deep Pose Estimation for Arbitrary 3D Objects”. In: *BMVC*. 2019.

[235] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. “Unseen Object Instance Segmentation for Robotic Environments”. In: *TRO* (2021).

[236] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. “MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM”. In: *ICRA*. 2019.

[237] Dekun Yang and John Illingworth. “Calibrating a Robot Camera”. In: *BMVC*. 1994.

[238] Shichao Yang and Sebastian Scherer. “CubeSLAM: Monocular 3-D Object SLAM”. In: *TRO* (2019).

[239] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. “Deep Part Induction from Articulated Object Pairs”. In: *SIGGRAPH Asia*. 2018.

[240] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. “DPOD: 6D Pose Object Detector and Refiner”. In: *ICCV*. 2019.

[241] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr., Alberto Rodriguez, and Jianxiong Xiao. “Multi-View Self-supervised Deep Learning for 6D Pose Estimation in the Amazon Picking Challenge”. In: *ICRA*. 2017.

[242] Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila. *VDO-SLAM: A Visual Dynamic Object-aware SLAM System*. 2021. arXiv: 2005.11052.

[243] Zhengyou Zhang. “Iterative Point Matching for Registration of Free-Form Curves and Surfaces”. In: *IJCV* (1994).

[244] Yu Zhong. “Intrinsic Shape Signatures: A Shape Descriptor for 3D Object Recognition”. In: *ECCV Workshops*. 2009.

[245] Tao Zhou and B E Shi. “Simultaneous Learning of the Structure and Kinematic Model of an Articulated Body from Point Clouds”. In: *IJCNN*. 2016.

[246] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. “On the Continuity of Rotation Representations in Neural Networks”. In: *CVPR*. 2019.

[247] Yiming Zuo, Weichao Qiu, Lingxi Xie, Fangwei Zhong, Yizhou Wang, and Alan L Yuille. “CRAVES: Controlling Robotic Arm with a Vision-based Economic System”. In: *CVPR*. 2019.

## RÉSUMÉ

Cette thèse présente des méthodes permettant de récupérer la configuration 3D de scènes contenant des objets rigides et des robots articulés en utilisant une ou plusieurs images RGB en entrées. Les scènes et conditions visuelles considérées incluent des objets sans texture et/ou symétriques, des robots à plusieurs degrés de liberté, des scènes imagées dans des conditions difficiles et des objets ou robots partiellement occlus. Les contributions comprennent des méthodes pour (i) la détection et l'estimation de la position 2D des objets dans l'espace de travail d'un robot, (ii) la résolution efficace du problème de réarrangement 2D, (iii) l'estimation de pose 6D d'objets basées sur l'apprentissage statistique, (iv) l'estimation de pose multi-vues multi-objects, (v) l'estimation de la pose de nouveaux objets (c'est-à-dire inconnus pendant l'entraînement) et (vi) l'estimation de la pose 6D et des angles articulaires d'un robot. Ces méthodes font progresser l'état de l'art sur les benchmarks existants pour l'estimation de pose d'objets et de robots.

## MOTS CLÉS

Vision par ordinateur, robotique, estimation de pose, apprentissage automatique.

## ABSTRACT

This thesis develops methods for recovering the 3D configuration of scenes containing rigid objects and articulated robots using one or multiple RGB images as inputs. The scenes and visual conditions considered include textureless and/or symmetric objects, robot arms with several degrees of freedom, scenes imaged under challenging conditions, and objects or robots partially occluded. The contributions include methods for (i) detection and 2D position estimation of objects in a robot's workspace, (ii) efficient planar rearrangement planning, (iii) learning-based 6D pose estimation of rigid objects with known 3D models, (iv) multi-view multi-object 6D pose estimation, (v) pose estimation of novel objects (i.e. unseen during training), and (vi) estimating the 6D pose and joint angles of an articulated robot. These methods advance the state-of-the-art on existing datasets and benchmarks for object and robot pose estimation.

## KEYWORDS

Computer vision, robotics, pose estimation, machine learning.