DISSERTATION

Simultaneous Object Detection and Pose Estimation under Domain Shift

conducted in partial fulfillment of the requirements for the degree of a

Doktor der technischen Wissenschaften (Dr. techn.)

supervised by

Ao.Univ. Prof. Dipl.-Ing. Dr. techn. Markus Vincze
E376 Automation and Control Institute

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology

by

Stefan Thalhammer M.Sc.
DOB 08.08.1987
Matr. Nr.: 0904045

Vienna, August 2022 Stefan Thalhammer

Acknowledgment

This thesis would not have been accomplished without the help and guidance of the current and former members of the Vision for Robotics research group (V4R). I greatly acknowledge the guidance and supervision of Markus Vincze and I am thankful for the provided opportunity to have been a member of V4R. I am grateful for the help, guidance and countless hours of proofreading provided by Timothy Patten. Finally, I would like thank all the colleagues for providing helping hands and for lending ears during fruitful discussion.

This work has been supported by the Austrian Research Promotion Agency in the program *Production of the Future* funded project MMAssist_II (FFG No. 858623) and under grant agreement No. 879878, project K4R. Further support of this work has been provided by the Austrian Science Fund (FWF) under grant agreement No. I3969-N30, project InDex. We gratefully acknowledge the support of the EU-program EC Horizon 2020 for Research and Innovation under grant agreement No. 101017089, project TraceBot. We thank the NVIDIA Corporation for donations of GPUs used in this research.

Abstract

Robotic systems are meant to provide support to humans, be it in their homes or work places. Manipulating objects is one of the most fundamental tasks for such systems, since this enables robots to autonomously take over tasks from humans. Relevant applications are diverse, ranging from making predictions for mobile manipulation, bin picking and augmented reality. Using real-world data for training is not desired since estimators are biased towards the data characteristics, capturing and annotating data is cumbersome and data in the target domain is not always available. Thus, using synthetic data is preferable. However, this requires that algorithms for object localisation need to generalise to novel domains. Another challenge is handling Relevant applications require object pose estimators to handle multiple distinct objects. Efficiently handling involved objects keeps cycle times short and computational system load low. The challenge is to effectively encode feature space for object sets with different shape complexities and symmetries.

This thesis presents methods to adapt, respectively generalise to novel domains and formulations for object pose estimators that handle multiple objects with different scales well with respect to the number of object instances in the image. Generalising to novel domains requires unbiased estimators. Rendering training data allows to randomise the relevant data characteristics, creating unbiased data well suited for training models meant to generalise well. Solutions for adapting synthetic depth data to the real-world domain and for generalising in the RGB-domain are presented. We formulate object pose estimation as a multi-task problem, performing detection, classification and pose correspondence estimation simultaneously for multiple objects. This approach is extended with direct pose regression resulting in scalable object pose estimation with constant runtime with respect to the number of object instances in the image.

Evaluations are provided on five datasets and on two different grasping scenarios. Presented experiments indicate that synthetic training data is well suited for learning-based object localisation. Training the presented object pose estimators using the domain adaptation for the depth domain and our domain generalisation strategies for the RGB domain results in competitive performance compared to the state of the art. The direct pose regression extension for scalable object pose estimation improves over other single-staged approaches and results in negligible runtime increase for up to 90 object instances in an image. We present grasping experiments showing the suitability of the presented methods for real-world deployment.

Contents

Chapter 1

Introduction

Robots, autonomous systems and agents are there to support humans. These systems are confronted with diverse applications. Mobile robots in retirement homes are meant to help elderly people to master their day. Facing tasks such as picking up objects from the ground and delivering requested items to humans they aid. Industrial robots are designed to support or autonomously handle production processes. A typical task is to provide scene information to ontology-based reasoning systems to perform complex assemblies or the reverse processes. Agents for augmenting and enhancing user experience display environmental information or highlight scene context to heads-on devices. These applications are solved with pipelines sequentially solving the involved tasks, such as sensing the environment, processing the resulting sensor readings and navigating to and interacting with their environment and users. Processing sensor readings is one of the truly fundamental challenges for essentially all of these applications which take place in man-made environments. Human environments are inherently designed to be processed through human vision. As such, visual perception of the environment naturally provides crucial information to solve these challenges. Thus, it is natural to base the perception of these systems on vision systems.

Visually perceiving the environment requires estimators to differentiate between useful information and noise in the image, such as foreground-background separation. The feature space of the image is filtered for prior task-specific information. Those complex computer vision applications mentioned above hold challenges that consider objects to be the relevant foreground information. Objects are part of everyday life and come in diverse shapes, appearances and sizes, like containers, tools, toys and consumables. Manipulating objects enables robots to autonomously perform tasks, including mobile manipulation [1]–[3], bin picking [4]–[6], augmented reality [7], [8] and reconstruction [9], [10]. During deployment the most crucial first step to realise such tasks is object localisation. After extracting object hypotheses task specific problem solving is performed, e.g. object pose refinement [1], [11], object tracking [12], [13] and grasp pose estimation [2].

Currently, Convolutional Neural Networks (CNN) are the Swiss army knife of computer vision. Tremendous results are achieved for image recognition [14], object detection [15], image segmentation [16], pose estimation [17], and many more. CNNs are dominating computer vision since encoding a latent representation of training samples

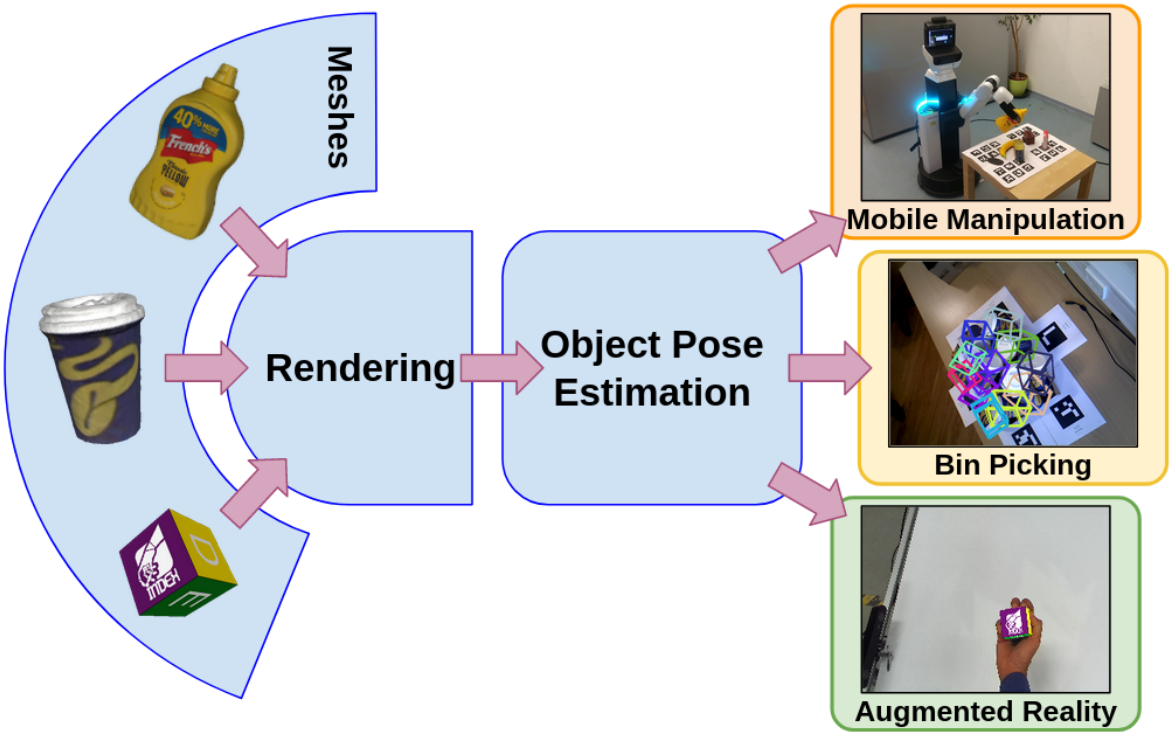

Figure 1.1: **From Meshes to 6D Object Locations:** It is of great applicability and feasibility to train object pose estimators exclusively from geometry and appearance priors. Handling the domain shift effectively enables deployment on diverse applications with robust performance.

enables formulating efficient problem solvers with excellent interpolation between data points [18], [19]. Yet, in order to provide robust problem solving large amounts of data are needed [14]. For one thing capturing an annotating training data in the real-world is very demanding in manual labor and thus the aim is to automate that process. Additionally, encoding a latent representation of the captured and annotated data induces a bias [20], [21]. This overfitting to the latent data distribution is disadvantageous for generalising to unseen domains. Important dimensions of this distribution are scene illumination, occlusion patterns, object configurations and object appearance [20], [22], [23]. Deep learning models generalise poorly under domain shift, i.e. different distribution of training and test data with respect to these dimensions. Due to these reasons it is desired to automatically synthesise data with sufficient variation along these dimensions and employ regularisation strategies to provide deep learning models that are robust across domains.

In this work we improve object localisation under domain shift. Figure 1.1 presents a visual representation. We solely assume the availability of textured or not-textured geometry priors of the objects of interest. Using rendering engines, training data is created. During training diverse means for regularising learning are employed to generalise the trained models to unseen domains.

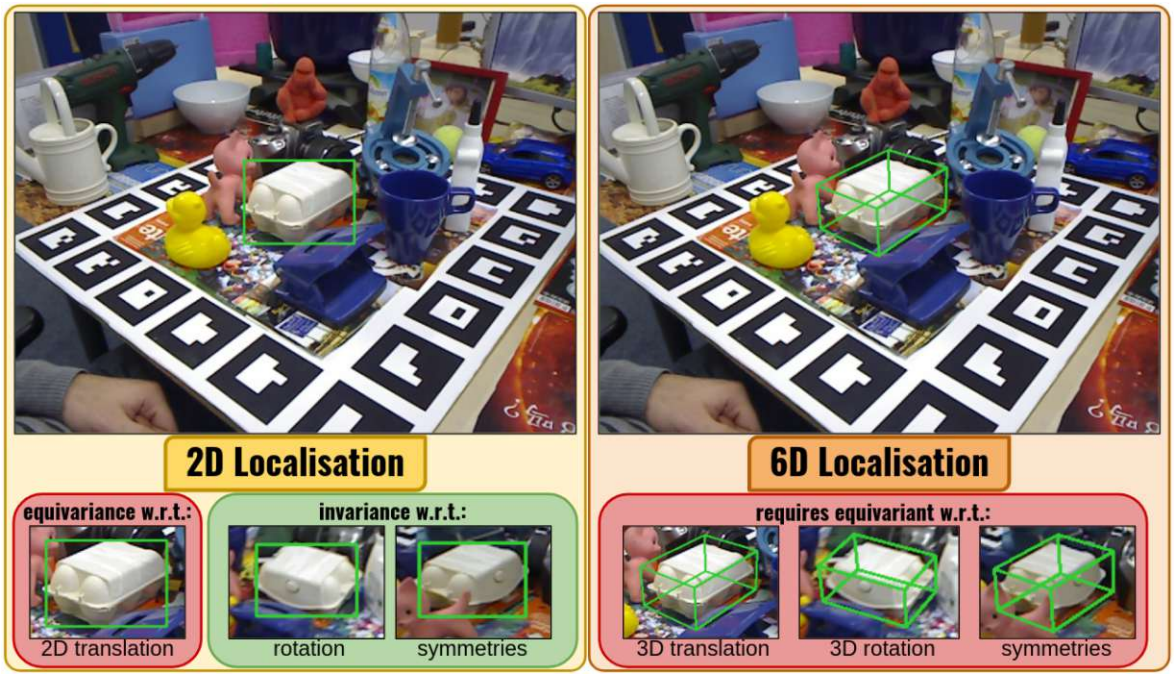

Figure 1.2: **2D versus 6D Object Localisation:** Object detection, respectively 2D localisation aims to recover the rough image location containing an object of interest. This requires the learned encoding to be equivariant with respect to image space translation and scale, allowing invariant mappings w.r.t object rotation and ambiguities resulting from object symmetries. Object pose estimation, respectively 6D localisation, requires learning an encoding equivariant w.r.t. 3D translation and rotation and object symmetries.

1.1 Object Localisation

The main variants of object localisation that this work focuses on are object detection and object pose estimation. Object detection refers to estimating the object location in the two-dimensional image space, and object pose estimation refers to estimating the object pose in the three-dimensional camera space. Additionally, we assume no availability of real world images, as is the case for classical pose estimation approaches [24]–[27]. We aim to solve the problem of localising one or more objects (multi-object) with multiple instances of each (multi-instance). Standard challenges for such localisation problems are cluttered environments, partial object visibility, object symmetries and changing appearance in different domains. Solutions to solve these challenges vary for object detection and pose estimation, see Figure 1.2. The following paragraphs provide a comparison of the problems object detection and object pose estimation.

Object Detection The task of object detection refers to localising the object of interest in an RGB image, thus in $2D$ space. As such the desired information to be estimated are the upper left and the lower right image coordinates forming the smallest bounding box encapsulating the respective object [15], [28]–[30]. Thus object detectors need to be equivariant with respect to image space translation and scale of the object.

Properties such as the object's relative rotation to the camera and its geometrical and textural symmetries are treated as invariant.

Due to the invariance with respect to the aforementioned properties, the amount of trainable parameters can be quite low. As such, many parameter-efficient object detectors scale well with respect to the number of object classes [29], [31]–[33]. A trait needed for object detection due to the large amount of object classes [34]. Additionally, object detectors scale well with respect to the amount of object instances due to the low output dimensionality.

Object Pose Estimation The task of object pose estimation requires estimating the relative transformation of one or multiple objects and instances with respect to the camera. The output space is the relative transformation from camera to object coordinate system, and thus parameterised with three dimensions for translation and three for rotation, as such a $6D$ space. Pose estimators require learning an encoding and output space that is equivariant to translation in 3D, rotation in 3D and visual object symmetries. This increased complexity leads to a fundamental difference in how objects are detected and how their pose is estimated.

An important aspect of this work is to demonstrate the similarity of 2D and 6D localisation problems in terms of formulation, and to present the substantial difference of how they are handled due to the difference in complexity. By focusing and building on the similarities of those two problems we aim to provide solutions for solving the 2D and the 6D object localisation simultaneously. As such, findings and contributions presented in this thesis are meant to introduce the advantages of 2D object localisation, such as end-to-end trainability, robustness to domain shift and constant runtime with respect to the number of object instances in an image to 6D object localisation.

1.2 Problems and Research Questions

The following sections outline the problems connected to training pose estimators from geometry priors, and bridging their applicability to that of object detectors. Solving the problem of object localisation for diverse real-world scenarios is challenging due to the many latent aspects of the data, such as camera intrinsics and extrinsics, imaging noise, scene context and photometric attributes. Using real-world data for training leads to a drop in performance in novel domains since deep learning approaches encode a representation of the training data, which induces a bias as to such latent data characteristics [20], [21], [35]. In order to generalise object detection and pose estimation performance to novel domains, deep learning models need to be invariant with respect to these aspects. Thus, training with synthetically created data is preferable [20]. Yet, this requires overcoming the domain shift between synthetically created renderings and the real-world test images.

The low dimensional output space of object detectors allows high-performance single-stage approaches. However, the state of the art for pose estimation are multi-staged, multi-model approaches in order to maximise performance as compared to single-model ones. Object pose estimation detects objects in the first stage, estimates pose correspondences in the second, and derives poses using Perspective-n-Points (PnP) [36]

in the third [17], [37]–[41]. Often poses are subsequently refined using additional stages [1], [11], [17], [38]. Simultaneously performing these individual steps provides scalable, parameter-efficient and easy-to-use formulations.

Besides solving object detection and pose estimation with separate models, state-of-the-art approaches split pose estimation into correspondence prediction and pose estimation with PnP. Correspondences are adopted as regression targets since directly regressing object poses from image features leads to severely reduced performance. Directly estimating the 6D object pose, however, enables scalable formulations with improved performance through the more meaningful guidance of solving the actual problem and not a proxy one. The following paragraphs present the problems outlined above in detail.

Domain shift Classical pose estimation approaches encode look-up tables that contain descriptors of template poses for matching during runtime [24]–[27]. These templates are created from reconstructed or designed object models and as such, no domain-specific real-world data is required for training. Due to the diverse hyperparameters of those methods, the method can be tuned specifically to the scene characteristics during deployment. Deep-learning-based approaches require task and domain specific data to encode a representation for problem solving. Using real-world data for training, however, has the major disadvantage of generalising badly to novel domains. Data characteristics of the test scene such as photometric attributes, scene context, imaging noise and the data distribution in $SE(3)$ are not always known or controllable [20].

Important characteristics of the data are photometric attributes of the scene, such as illumination and the interaction of the light rays with the physical world. These are directly reproducible through rendering when parameters are known. However, inverting this process to estimate parameters is ill-posed due to the infeasibility of tracing the ray of light and its interaction with different materials and their physical properties. Similarly unconstrained is the scene context in the real world. Consider the case where the real-world training examples are captured on the surface of a desk and the task is grasping objects in a shop floor. During test time, predictions have to be made from images that feature many more of different objects as compared to what is known from the training set. The imaging sensor noise also introduces uncertainties regarding the domain. It can be measured to a certain extent, but exhaustively determining all the relevant parameters and their interconnectedness is similarly intangible as estimating photometric attributes. Creating real-world training data with unbiased pose sampling is not as intangible as the aspects mentioned above, yet, it is difficult and costly to do. Exhaustive and uniform sampling of object poses in $SE(3)$ camera space and producing annotations leads to an extensive involvement of humans, and thus manual labor.

Considering these aspects it is desirable to automatically synthesise training data. Generalising to novel domains requires the latent synthetic data characteristics to be a super-set of those to expect during application. Since generalising to novel domains, respectively adapting to them, is a difficult task in RGB images, many works address this domain gap in depth images [6], [42]–[45]. However, capturing depth data comes with disadvantages and limited applicability due to its comparably costly sensors, missing textural cues and low image resolution. Though, overcoming the domain gap in RGB is more difficult due to the higher variations of the image noise and the variations of the

latent data characteristics, numerous works thus prefer using RGB and either addressing the domain gap by using annotated data with the expected characteristics [22], [39], [40], [46]–[49] or by using not annotated samples of the expected domain [49]–[52]. For one thing, using real-world samples for training induces a bias for estimation making. Then again, assuming the availability of real-world images with the expected characteristics is infeasible for real-world applications. The domain of deployment and as such also the camera characteristics are not always known beforehand and the physical objects are often unavailable for training. The authors of [20] show that a severe reduction in performance is to be expected when the available real-world data characteristics diverge from those during deployment.

As a consequence, researchers tend to train on mixed data, consisting of real-world samples, with the expected characteristics, and rendered samples [22], [40], [46], [47], [53], [54]. The real-world samples guide trained models to perform well on the expected test sets and the renderings increase the data variation in terms of poses, occlusion patterns, lighting conditions and background. Exclusively training on synthetic data reduces performance as compared to using real-world data for training [55].

Reduced Performance of Single-staged Approaches A general problem for object localisation is that single-stage approaches result in lower detection performance than their two-staged counterparts, 10% to 40% for object detectors as of 2017 [56]. Though exhibiting that performance gap, in the meantime single-staged detectors superseded their multi-staged counterparts as the State of the Art [33], [57], providing estimates in a single stage enables parameter-efficient, scalable and simpler solutions. As a consequence of their end-to-end trainable nature, object detectors scale well with respect to the number of object classes [28], [29], [31]–[33].

Learning-based object pose estimation is currently a multi-staged problem. Many top-performing approaches expect sparse 2D location priors from object detectors as image location priors [17], [23], [38]–[40], [46], [47], [54]. Some of these and others [22], [49], [53], [58], [59] adopt geometric correspondences as training targets to reduce the dimensionality of the output space, and thus to improve performance. In the final stage poses are estimated using Perspective-n-Points (PnP) algorithm [36] from these correspondences. Research for combining detection and geometric correspondence estimation is also conducted [22], [49], [58], [59], yet only [58] provides results indicating performance comparable to state-of-the-art approaches with more stages and end-to-end trainability. A downside of this method is that computationally expensive post-processing is required to cluster predicted geometric correspondences to distinct object instances [60].

Missing Scalability in Object Pose Estimation Early deep learning-based pose estimators directly regress the 6D pose [6], [48], [61], [62]. Later research adopts surrogate training targets, corresponding to the observed pose. Since predictions are made in a lower dimensional space than 6D, either 2D or 3D, these approaches provide a more natural representation for CNNs and thus improve in performance, as outline in Section 1.1 [17], [23], [38]–[40], [46], [47], [54]. Using these so-called surrogate training targets requires deriving the pose through algorithms like Perspective-n-Points (PnP) [36]. Nonetheless an accompanying problem is that adopting such means reduces the ability of the network to learn descriptive features for the actual tasks. Additionally,

multi-staged problem solvers are required, effectively leading to reduced scalability.

Ongoing research directly estimates 6D poses from intermediate geometric correspondences [37], [41], [63], [64]. In [41] and [37], detection is separated from the pose estimation stage, thus, these are not end-to-end trainable since an object detector is required for sparse location prior prediction. The work of [64] is end-to-end trainable but separate networks need to be trained and pooling geometric correspondences means multiple objects and instances cannot be handled simultaneously. While regressing object poses from intermediate geometric correspondences circumvents the drop in pose estimation performance, current state of the art does not acknowledge the resulting potential for scalability. This strategy can naturally implement the desired traits of object detectors to object pose estimators: scalability, simpler application (i.e. less post-processing) and end-to-end trainability for all objects.

The goal of this work is to provide a solution for single-stage object pose estimation assuming only the availability of geometric object priors. Considering the state of the art and the open problems outlined above, the following research questions emerge:

1. How to generalise object localisation to unknown domains?
2. How to enable simultaneous object detection and correspondence estimation?
3. How to formulate scalable, single-staged pose estimation?

1.3 Scientific Contribution

By adopting and improving recent advancements we provide solutions for the problems connected to aforementioned research questions. We improve domain generalisation when using RGB and domain adaptation when using depth, in order to train object pose estimators from synthetic data. By adopting feature pyramid-based network architectures, commonly used for object detection, we enable efficient simultaneous object detection and correspondence estimation. Empirical analysis that this formulation scales well with the number of object classes of interest is provided. Ultimately, by learning to directly regress 6D object poses from latent geometric correspondences, we provide an efficient formulation for processing all object instances in an image in parallel. Summarising the scientific contributions of this thesis:

1. We present means for improving domain adaptation in the depth domain and domain generalisation in RGB.
2. We present an efficient feature pyramid-based solution for simultaneous object detection and pose correspondence estimation.
3. We present end-to-end trainable direct pose estimation with constant runtime. Providing object poses with similar scalability as object detectors estimate bounding boxes.

The remainder of this section presents an overview of each of these contributions, followed by the list of publications in which these are presented.

1.3.1 Domain Generalisation for Object Localisation

For generalising to novel domains we address the problem at different stages of the deep learning training process. We improve training data distributions, the training process, and explore better suited formulations than the standard ones, see Figure 1.3. The principles we adapt and improve upon for effective domain generalisation are:

- **Domain Randomisation** For synthetic data rendering, we randomise different aspects of the physical properties used for rendering training images from virtual scenes. Depending on the expected use case and domain, aspects such as scene background, illumination, object and camera poses and material properties of the objects are randomised.

- **Data Augmentation** As a preprocessing step for training, images are augmented to create a superset of the expected real-world data variations. We present findings showing improvements in RGB and depth.

- **Restricted Training** We use a pre-trained backbone with frozen layers during fine-tuning. Initial stages of the backbone are not updated during training, to force the network to learn high-level features conditioned on low-level ones learned from rich real-world data. In order to maximise the effect of the frozen initial stages batch normalization parameters are also not updated during fine-tuning.

- **Network Architecture** We introduce formulations of common object detectors to object pose estimation. We show that these lead to efficient object pose estimation reaching state-of-the-art performance with comparably little network parameters.

The following paragraphs outline the means taken to transfer trained models to a novel domain in the RGB and depth domain, respectively.

Domain Adaptation in Depth We show that randomising the domain of the virtual scene used for training data rendering and augmenting the data with errors expected from the imaging process leads to improved object detection, compared to using available real-world data. Using the same process for creating object pose estimation training data results in state-of-the-art performance.

Domain Generalisation in RGB Many works provide analysis that the depth domain is well suited for problems with domain shift because of of its limited variations in comparison with RGB [45]. This limitedness however leads to less informative value to draw conclusions from, e.g. when handling transparent objects or similarly shaped objects where the discriminative part is the texture. For such cases it is advisable to switch to RGB. Yet, due to the richness of the domain it is difficult to generalise well. We improve over commonly used data augmentation techniques by introducing contrast noise. Furthermore, forcing the network to encode high-level features from low-level features learned during pre-training improves performance in the real domain tremendously. Lastly, our networks are built on feature pyramids, which are known to generalise well. The following section presents the feature-pyramid-based estimation making more in detail.

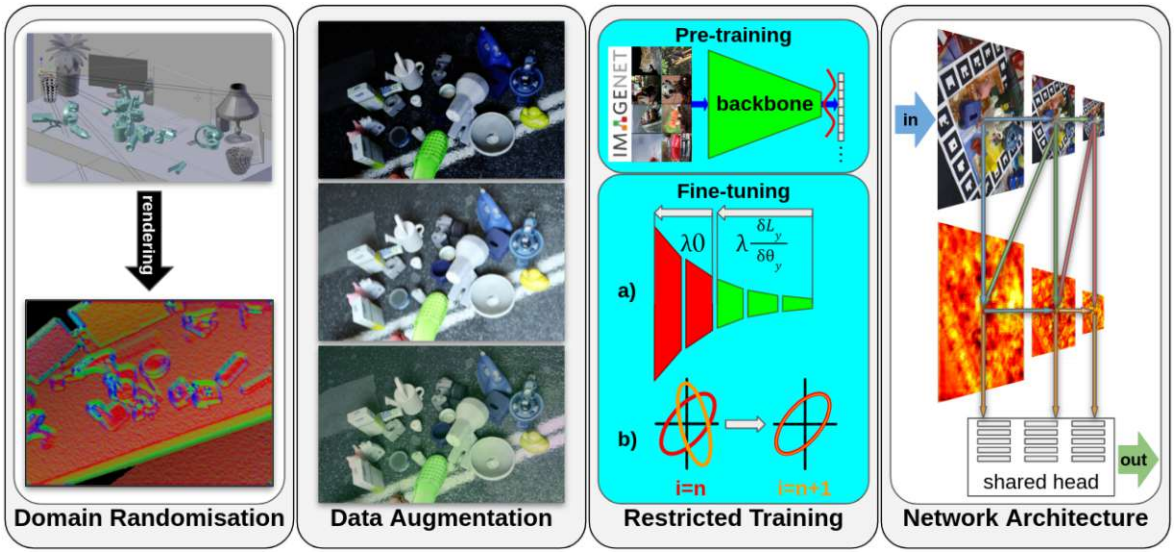

Figure 1.3: **Domain Generalisation:** We adopt and improve on multiple approaches for domain generalisation. By randomising synthetic data creation, varying online data augmentation, pre-training on real-world data and restricting the layers that are updated during training, and by using regularising network architectures, our network generalises well to novel domains.

1.3.2 Simultaneous Object Detection and Pose Estimation

The previously presented methods provide synthetic data and means to generalise trained models to novel domains. This section outlines how object detection and pose estimation is performed simultaneously. In order to solve that challenge, formulations for multi-task learning are required that scale well with regards to the amount of the objects of interest. Object detection solves the multi-task problem by sliding sub-networks over query image locations, producing outputs for each required task. As a consequence, downstream network parts are trained with guidance from all involved tasks.

Many single- and multi-stage object detection approaches are based on feature pyramids [32] for producing latent multi-scale feature maps. Feature pyramids provide efficient solutions for object detection, that generalise well. Thus, they are ideally suited for object pose estimation under domain shift. Anchors [28] are used as location priors and for target standardisation in the feature pyramid. Shared sub-networks for prediction making slide over those feature maps and estimate bounding boxes while differentiating foreground from background.

These principles are adopted and improved upon for object pose estimation, see Figure 1.4. We present a network for simultaneous multi-object detection and pose estimation based on a modified feature pyramid. We sample image locations in the latent multi-scale feature maps of our feature pyramid, using an anchor-based approach to additionally estimate geometric correspondences for each object, and furthermore estimate the object pose using RANSAC-PnP [36], [65], [66]. The presented results show that such a formulation lead to improved performance and scalability as compared to state-of-the-art multi-stage approaches. Furthermore, a specialised loss for 3D bounding

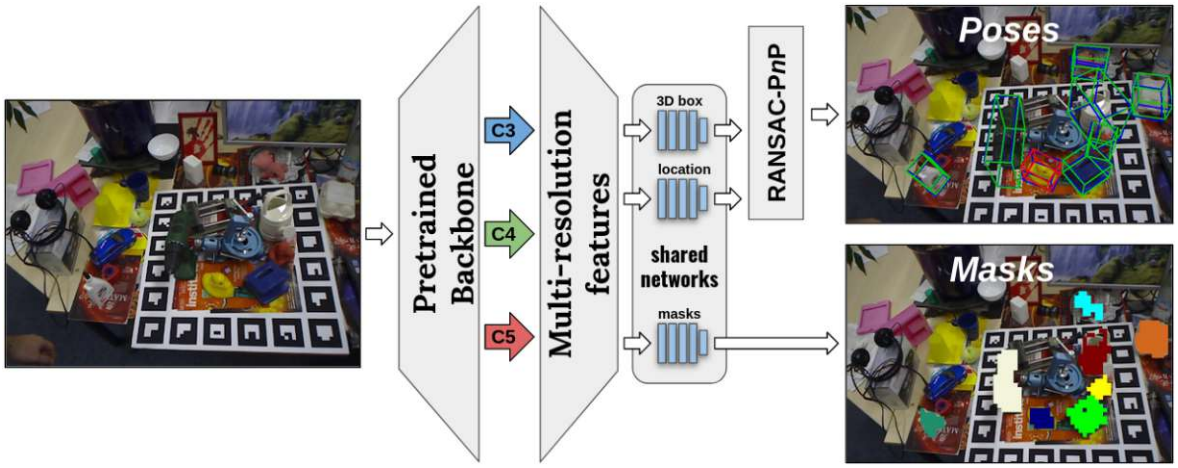

Figure 1.4: **Simultaneous Detection and Pose Estimation:** A pre-trained backbone extracts multi-scale image features to solve object localisation and classification in 2D, and to predict geometric correspondences of multiple objects simultaneously. Poses are ultimately derived using RANSAC-PnP.

box estimation that improves performance is proposed. The following sections present an extension for direct pose regression for multi-object and multi-instance scenarios that features similar scalability to object detection approaches, though solving a problem of higher dimensionality (6D for object pose estimation, as compared to 2D, for object detection).

1.3.3 Multi-Instance Direct Pose Regression with Constant Runtime

Having robust geometric correspondence estimation provides a good basis for directly learning to derive 6D poses. However, while object detectors provide 2D estimates that are directly used in sequential problem solvers of vision pipelines, providing 6D poses directly is not as straightforward. Current object pose estimators require PnP to compute the 6D pose from the predicted 2D correspondences.

In order to provide the same ease of application and scalability of object detectors, we equip our pose estimator presented above with a direct pose regression network, see Figure 1.5. The additional sub-network slides over the predicted geometric correspondences and learns to directly estimate the 6D pose from them. This additional guidance during training improves the downstream task of geometric correspondence estimation, and removes the additional requirement of applying RANSAC-PnP to each set of object pose hypotheses. Modifying the commonly used non-maximum suppression in object detection, we filter for the predicted 6D poses with the highest confidence per object instance. We provide experiments showing that direct pose regression, as upstream task, guides learning towards better geometric correspondence estimation, while also allowing scalable processing of multiple objects and their instances, as is the case for object detectors.

Despite their effectiveness, anchors [30] also come with downsides. In order to

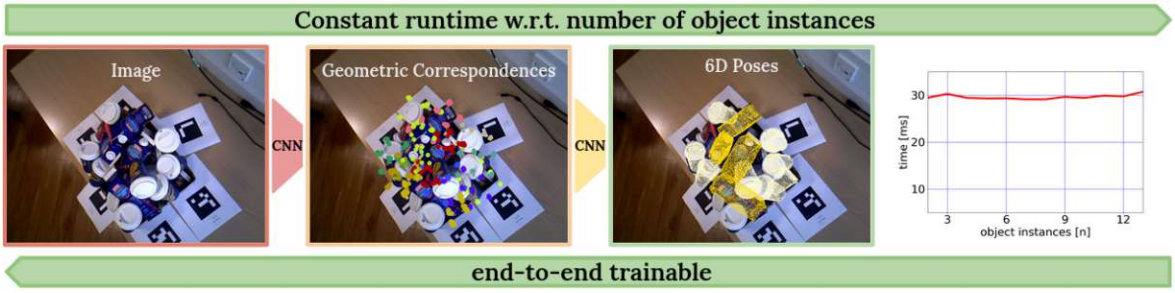

Figure 1.5: **Direct Pose Estimation with Constant Runtime:** We formulate an end-to-end trainable learning task, with the downstream task of geometric correspondence learning and the upstream task of direct pose estimation. Computationally efficient multi-instance clustering is applied to extract the per-instance poses in an image. This procedure results in constant network time and improved pose estimation due to the additional supervision.

detect objects from specific feature map resolutions 16 hyperparameters are required for sampling a dense bounding box prior space. Due to the large output space of sampling nine anchors per feature map location, convergence and training speed are low. We directly estimate object poses from locations of the multi-scale feature map, which overlap with the object mask in the image space. The respective feature map resolution is chosen using a scalar object size parameter projected to the image space. This parameter is calculated from the object dimensions. Geometric correspondences are also standardised using that parameter. This results in increased convergence and reduced training time. The following section presents a list of publications this thesis is derived from.

1.4 List of Publications

The content in this dissertations has been partially previously published in the following publications:

- ***Thalhammer, S.*** *and Vincze, M.* (2022). COPE: End-to-end Trainable Constant Runtime Object Pose Estimation. *Accepted for publication at the Winter Conference in Applications for Computer Vision (WACV).*
- *Gupta, H.,* ***Thalhammer, S.*** *and Vincze, M.* (2022). Grasping the Inconspicuous. *Accepted for publication at the Austrian Robotics Workshop.*
- *Gallauner, B.,* ***Thalhammer, S.*** *and Vincze, M.* (2021). Enabling Classification of Heavily-occluded Objects through Class-agnostic Image Manipulation. *OAGM Workshop*, 27-31. DOI: 10.3217/978-3-85125-869-1-05
- ***Thalhammer, S.****, Leitner, M., Patten, T. and Vincze, M.* (2021). PyraPose: Feature Pyramids for Fast and Accurate Object Pose Estimation under Domain Shift, *IEEE International Conference on Robotics and Automation (ICRA)*, 13909-13915. DOI: 10.1109/ICRA48506.2021.9562108

- ***Thalhammer, S.**, Patten, T. and Vincze, M.* (2020). Usability Study of Learning-Based Pose Estimation of Industrial Objects from Synthetic Depth Data. *International Precision Assembly Seminar (IPAS)*, 9, 285-296. DOI: 10.1007/978-3-030-72632-4_21

- *Flatz, L., **Thalhammer, S.**, Patten, T. and Vincze, M.* (2020). Improving Out-of-distribution Distractor Handling through Data Augmentation. *RO-MAN 2020 Workshop on Hand-OBject Interaction: From human demonstrations to robot manipulation (HOBI).*

- *Peer, M., **Thalhammer, S.** and Vincze, M.* (2020). Image Synthesis in $SO(3)$ by Learning Equivariant Feature Spaces. *Joint Austrian Computer Vision and Robotics Workshop*, 108-113. DOI: 10.3217/978-3-85125-752-6-25

- ***Thalhammer, S.**, Patten, T. and Vincze, M.* (2019). SyDPose: Object detection and pose estimation in cluttered real-world depth images trained using only synthetic data. *International Conference on 3D Vision (3DV)*, 106-115. DOI: 10.1109/3DV.2019.00021

- ***Thalhammer, S.**, Patten, T. and Vincze, M.* (2019). Towards object detection and pose estimation in clutter using only synthetic depth data for training. *Joint ARW and OAGM Workshop*, 203-208. DOI: 10.3217/978-3-85125-663-5-46

- ***Thalhammer, S.**, Patten, T. and Vincze, M.* (2019). SyDD: Synthetic depth data randomization for object detection using domain-relevant background. *Computer Vision Winter Workshop (CVWW)*, 14-22. DOI: 10.3217/978-3-85125-652-9-02

Chapter 2

Related Work

The first part of this section presents the state of the art for synthetic data creation for the RGB and the depth domain and means for generalising and adapting to unknown domains. As follow up object detection is summarised, the state of the art for object pose estimation is presented and the chapter concludes with a presentation of relevant metrics for quantitative evaluation.

2.1 Synthetic Data Creation

Due to the large amounts of requirement training data and the challenges accompanying capturing real-world, deep learning research for creating synthetic training data is multifaceted [23], [43], [62], [67]–[74]. Rozantsev *et al.* [73] project the object geometry, taken from CAD models, into RGB images. A texture filling algorithm varies the object appearance with respect to blur, noise and material properties. Su *et al.* [74] render multiple views of 3D objects to generate a single compact descriptor of that object using a CNN. This is a common strategy for pre-deep-learning object pose estimation approaches [25]. In [70] the authors create annotated synthetic indoor scenes using an automatic furniture arrangement mechanism. They use a simulated Kinect noise model to include noise in the synthetic depth scans. Carlucci *et al.* [67] use *Blender*[1] to create a synthetic depth image dataset for object recognition. They use 3D CAD models downloaded from different databases to create object categories. Views are rendered from a configuration space consisting of object distance, camera position, focal length and random object warping. The amount of identical rendered images is minimised. Planche *et al.* [44] present a pipeline to render realistic depth images for object recognition. They simulate the image appearance for a wide range of sensors. Their pipeline consists of a pattern projection mechanism, an intermediate step injecting sensor noise followed by stereo matching and post-processing to reproduce the spatial sensitivity of the sensors, and to simulate the impact of surface materials. Backgrounds such as primitive shapes and captured real-world scans can be added. The authors of [62] project rendered views of object models on top of Microsoft COCO [34] to provide training data for their object pose estimation approach. This approach has

[1] https://www.blender.org/

been successfully adopted by the object pose estimation community [22], [40]. The authors of [69] employ a similar strategy to increase the amount of real-world training images, by extracting the object of interest based on its mask from training images and pasting them on random background. Sundermeyer *et al.* [23] use synthetically created RGB-crops of objects for object pose estimation. Park *et al.* [43] render RGB-D images from physically sampled virtual scenes. Hodan *et al.* [71] show the suitability of physically-based-rendering (pbr) for synthetic-to-real object detection. Denninger *et al.* [68] provide a framework for creating synthetic training data for diverse computer vision problems using PBR, which in turns is used to provide the training data base for the BOP-challenge [75].

2.2 Transfer Learning

The generality of trained deep learning models, as a family of algorithms that learns a representation of the provided training data, is highly dependent on the amount and the domain of that data. Thus, a bias is induced during training. Transfer learning subsumes those cases for which labelled data is not available over all domains [76]. Depending on the availability of data and corresponding annotations, diverse solving strategies exist, ranging from domain randomisation over domain alignment to unsupervised domain adaptation [42], [67], [77]–[79]. This work expects neither availability of target domain annotations, nor target domain samples, since it is infeasible to always expect the availability of data in the test domain. Therefore, this work aims to generalise to unseen domains.

Diverse techniques to improve the trained model's generality exist. Regularising the kernels' weight updates improves generality of learned representations [80]. Randomly dropping network weights from updating during training can be seen as a form of ensemble learning which leads to improved generalisation [81]. Aligning the statistics of learned feature spaces also helps to improve transfer when facing data with a shift in domain [82]. The parameter space of the model itself specifies the model's capacity. As such it has an influence on the learned representation and thus controls the generalisation ability of the trained model. Additional training hyperparameters such as batch size and learning rate also influence the performance under domain shift, and in practice need to be adjusted based on the specific problem [83]. The following paragraphs present specific techniques to improve domain transfer when using depth images, followed by those for the RGB domain.

Transfer Learning in the Depth Domain: Transfer learning subsumes the cases where labelled data is not available over all domains for the task at hand. In this work we consider the case that we have the same task for source and target domain, but labelled data is only available in the source domain. This requires the application of domain adaptation techniques for the transductive transfer learning task [76]. Solutions to this problem include domain randomisation, domain alignment and unsupervised domain adaptation [42], [67], [77]–[79], [84]. We use an off-the-shelf architecture as an detector, which requires strategies to transfer the image domain.

In [85] collision avoidance for autonomous flight is learned from simulation. They

render images from synthetic 3D hallways. Parameters such as wall textures, furniture position, illumination and camera pose are randomised. Tobin *et al.* [72] use domain randomisation to produce sufficient variability at training time to enable robot grasping. Their approach randomises shape, position, orientation, texture of the objects involved, illumination and camera extrinsics. In [45], Zakharov *et al.* use domain randomisation to augment depth images. Perlin noise [86], Voronoi texturing and white noise is used as background for rendered 3D object models. Perlin noise is an inexpensive way to simulate sensor noise. Randomised patterns are used to simulate occlusion.

A remaining challenge is to adapt to the target domain when no data is available. We address this by augmenting synthetic depth scans with Perlin noise and a randomised sensor noise model.

Transfer Learning in the RGB Domain: The two most common methods for synthetic-to-real transfer for learning-based pose estimation are unsupervised domain adaptation and domain randomisation [72]. For unsupervised domain adaptation, Generative Adversarial Networks (GAN) are employed to map images from a source to a target domain [42], [45]. Bousmalis *et al.* [42] train one model to map images from the synthetic source domain to the real target domain, followed by a second model to classify and estimate 3D poses. The authors of [87] learn a mapping between the synthetic and real-world features spaces for domain adaptation. A self-supervised approach using differential rendering is proposed to overcome the requirement of annotated real-world images [52]. However, while neither [87] nor [52] need annotated real-world images, both require a considerable amount of images in the target domain. Our approach works purely with synthetic data without the requirement of any real-world images during training. We do not aim to adapt to the target domain, we generalise to unseen domains via randomising the noise applied to the synthetic domain and by regularising training.

When randomising domains, the image space is augmented by inducing noise [23], [62] to create a synthetic domain that is a superset of the real-world domain. In [62], rendered images are pasted onto images from MS COCO [34] and trained for end-to-end object detection, classification and pose prediction using a network based on [28]. In [49], the authors follow the same strategy to estimate poses using an encoder-decoder architecture. Sundermeyer *et al.* [23] train separate networks for detection and pose estimation. For detection, the models from [28] and [56] are used, then an encoder-decoder network is trained to reconstruct rendered images from augmented renderings. In this work, we leverage and improve over recent RGB augmentation strategies to improve generalisation.

2.3 2D Localisation

Single-stage object detectors present efficient solutions for multi-object, multi-instance object localisation in 2D [28], [56], [57], [88]–[90]. Object detection benefits from making predictions from the multi-scale feature maps of a feature pyramid network [28], [32], [33], [91], [92]. In the next paragraph we present the state of the art for feature pyramids in deep networks in detail. It was shown that using location priors such as anchors [28],

[30], [56], [92] to sample training locations in specific pyramid levels improves object detection performance. The last paragraph of this section discusses the use of anchors.

Feature Pyramid Networks: Feature pyramid networks aggregate features over various spatially resolved feature maps. This provides local to global object appearance cues and thus encodes scale information into feature maps. By aggregating features using a top-down pathway then upsampling lower-resolved feature maps for feature fusion, Feature Pyramid Network (FPN) [32] improves performance over using the feature maps from a single image resolution. Building on FPN, the Path Aggregating Network [93] adds a bottom-up pathway, improving over FPN. Recently, [91] used Neural Architecture Search [94] to design an effective feature pyramid. In [33], an efficient modification of FPN is presented in which the introduction of skip connections leads to a more lightweight and better performing variant. We translate the concept to pose estimation. By leveraging recent findings for feature pyramids and with the specifics of 6D localisation scenarios in mind we design a Pose Feature Pyramid Network (PFPN).

Anchors for Object Detection: The standard approach for choosing training locations in the multi-scale feature maps is to threshold the overlaps of the ground truth bounding boxes and assumed box priors, so-called anchors [28], [30], [56], [92]. Bounding box priors with different sizes and aspect ratios are sampled depending on the individual feature map resolutions in the feature pyramid. For training, foreground image locations are chosen based on the Intersection-over-Union (IoU) between ground truth bounding boxes and the respective anchor priors. As such, training locations are correlated with the projected object shape in the image space. This leads to effective encoding and handling of objects with different scales. However, using anchors has two downsides, firstly, it requires manual specification of 16 hyperparameters that reflect the expected training and test dataset statistics; secondly, the size of the output space depends on the number of anchors sampled for each image location. Recent approaches propose alternative formulations to circumvent these shortcomings, while retaining the advantages of anchors [88]–[90], [95]. The authors of [90] choose the respective feature map resolution for training explicitly by using the bounding box size to overcome the necessity for sampling anchor boxes. True image locations of the feature maps are assigned based on the respective pixel's center-ness with respect to the ground truth bounding box. Alternatively, [89] models objects as paired-keypoints: the top-left and the bottom-right corner of the bounding box. Similarly, [88] addresses the inefficiency of anchor-based object detection by modelling objects as their center points and estimating the bounding box relative to it. The authors of [95] overcome the requirement of hyperparameters for assigning objects to anchors by designing a flexible maximum likelihood estimation assignment for network training. We propose to sample training locations based on the visible object mask and the 3D object dimensions. We also use the 3D object dimension to effectively replace the anchor-based target annotation standardisation. As such we encode objects of different sizes and eccentricities more effectively, while also reducing the size of the output space and the number of required hyperparameters.

2.4 6D Localisation

After an outline of recent developments in monocular 6D object pose estimation, we dive into deriving direct pose estimates from intermediate geometric object representations. This is followed by reviewing location sampling and target standardisation for object detection.

2.4.1 Classical Approaches:

Classical approaches use different types of hand-crafted features to encode view-dependent descriptors for object pose estimation. Early works encode templates from depth or RGB-D data [24]–[26], [96]. Drost *et al.* [26] match descriptors comprised of point-pair features to a point cloud of the test scene. Pose votes are generated based on the query's similarity to entries of a look-up hash table. Votes are accumulated and hypotheses are refined using ICP. Hypotheses are favoured when the detected 3D edges match the model contours. Hinterstoisser *et al.* [25] match queries to RGB-D template descriptors encoding local image gradients and surface normals. Aldoma *et al.* [97] encode SHOT descriptors [98] for object recognition and 6D pose estimation. A global hypotheses verification framework improves performance in clutter and under occlusion [96]. Hodan *et al.* [24] use a sliding window with cascading evaluation. Pre-filtering differentiates between object and background. Hypotheses are generated for every window by hashing. Hypotheses verification consists of verifying size, normals, gradients, depth template and color. Object pose refinement is initialised from the verified hypotheses using particle swarm optimization. Vidal *et al.* [27] and Alexandrov *et al.* [99] improve the performance of [26] and thus show that point-pair feature templates still provide strong estimators today. For hypotheses generation, a local clustering step and an additional filtering step to remove non-discriminative pairs between neighbouring clusters[27]. After hypotheses generation, a re-scoring is used on the top 500 based on how well they fit to the scene data. The top 200 are then refined using projective ICP. During post-processing, hypotheses that only partially fit the scene are removed. Additionally object silhouettes that do not overlap with scene edges are filtered. Apart from point-pair feature-based methods, template-matching methods are also relevant for learning-based pose estimation [100].

2.4.2 Learning-based Approaches:

It has been shown that decision forests are a learning-based alternative to hand-crafted matching frameworks [101], [102]. Tejani adopts the multi-modal templates of [25] to predict the probability distribution over templates for detection and object pose estimation. Brachmann *et al.* [102] also employ a random forest approach to predict object class and pose per pixel. A RANSAC-based optimisation scheme is used to sample poses. The pose is subsequently iteratively refined by maximising the likelihood over the distributions estimated by their random forest. Kehl *et al.* [62] present a deep-learning solution modifying SSD [28] as backbone network to regress bounding boxes and to predict class, viewpoints and in-plane rotation. The 3D translation is

derived from the detected bounding box. They argue that classifying poses is more accurate than regressing and allows parsing multiple detections as well as handling symmetries in a straightforward way. They train SSD on real images with objects of interest rendered into the scene. Xiang *et al.* [48] predict instance segmentations and pose from RGB images. They regress the center direction per pixel to estimate the object's center using a Hough voting layer. The authors introduce the ShapeMatch-Loss that enables handling symmetrical objects. During training the offset between the estimated model points and the ground truth model points is minimised. ICP is used as refinement. Rad *et al.* [47] employ VGG [103] to obtain coarse object segmentation and passes the object crop through two other networks. One network estimates the pose based on control points and another network subsequently refines the initial estimate. Sundermeyer *et al.* [23] use RetinaNet [56] for object detection and 3D translation derivation. An autoencoder is trained to extract a latent feature vector to derive the object rotation. Sock *et al.* [6] employ Faster-RCNN [104] for feature extraction in depth images and add a pose branch, separately estimating the object center, depth and rotation. These initial estimates are coarse. Since they estimate object poses in crowd scenarios, they apply a joint registration module taking neighbouring object hypotheses into account separates true from false positives.

2.4.3 Pose representation

Xiang *et al.* [48] directly regress the quaternion representing the object rotation and the object's translation. Kehl *et al.* [62] argue that rotation estimation improves when formulating it as a classification problem. They classify the viewpoint and derive the 3D translation from a precomputed bounding box depending on the rotation. Do *et al.* [61] regress values of the Rodrigues rotation formulation. Sock *et al.* [6] separately classify the three Euler angles and the object's depth. Crivellaro *et al.* [105] propose to estimate geometric correspondences and to regress their image locations. This keypoint prediction can be formulated similarly efficient as bounding box estimation. The pose is derived by solving PnP with these 2D estimates of corresponding 3D points in the object coordinate frame. This approach is widely adopted by the community [22], [47], [53], [59], [106], [107]. An alternative strategy is to use encoder-decoder architectures for dense hypotheses generation and subsequent pose estimation using the PnP algorithm [22], [39], [40], [46], [49], [54], [58], [107]. The majority of these approaches colour the mesh model based on the *uv*-coordinates. The networks are trained to regress the vertex location in the object coordinate system, i.e., 2D-3D correspondences, that are the input to PnP. While many state-of-the-art methods adopt these trends it is computationally expensive, since dense correspondence prediction requires an output space of the input space's size, while keypoint estimation enables a scalable pose estimation formulation.

2.4.4 Multi-model Pose Estimation

Since single-model approaches result in inferior performance as compared multi-model approaches [56], researchers tend to train separate models for each object of interest to improve performance [22], [43], [47], [49], [59]. Even more performance gain is achieved

when first detecting objects in images, secondly predicting geometric correspondences as surrogate targets and lastly estimating the pose using the PnP algorithm [22], [39], [40], [46], [49], [54], [107]. The performance improvement occurs from sequentially estimating multiple representations and as such using all of the network's capacity for the respective task.

We argue that the state of the art for object detection, feature pyramid networks, provides a solution to alleviate that problem. Making predictions from multi-scale features better encodes object scale information, since differently resolved feature maps are used for making predictions of objects with varying scales. Which is beneficial for RGB, as no direct depth information is available. Additionally, solving multi-task problems encodes more general latent feature spaces that are well suited for providing estimates under domain shift.

2.4.5 Direct Pose Regression

Considering that direct pose regression from the feature space leads to inferior performance, the best-performing monocular object pose estimation approaches leverage geometric correspondences as regression targets [39], [40], [49], [53], [54], [58], [108]. Poses are derived for each estimated set of object correspondences using variants of PnP [36], [66], [109]. Recent trends replace the classical solver with trainable versions [37], [41], [63], [64], [110] to infer the 6D pose directly from the intermediate geometric correspondences. This enables end-to-end trainable object pose estimation, as it provides the additional supervision for the down-stream network parts with the 6D pose. Their findings indicate that direct 6D pose estimation also results in state-of-the-art performance by sharing the pose regressor over the objects classes [37], [41]. However, efficient and simultaneous single-stage multi-object instances handling is a problem that still remains [58].

The BOP challenge [75] is a benchmark that aims to provide a standardised protocol for unbiased comparison of object pose estimators. Considering the top performing approaches, a frequently used technique to handle multiple object instances in an image is to separate object detection from pose estimation [17], [38]–[40], [58], [111]. In the first stage, 2D location hypotheses are provided using common object detectors such as FasterRCNN [30], RetinaNet [56] or FCOS [90]. In the second stage, object crops are passed to the pose estimator but this leads to considerable temporal and computational cost, and in the third stage poses are derived. An exception is EPOS [58] where multi-instance handling is facilitated by using Graph-Cut RANSAC [60] to cluster the predicted geometric correspondences to individual instances. Despite providing a sophisticated approach for addressing object symmetries and multiple object instances with one forward pass through a network, their multi-instance fitting of poses using [60] is computationally very demanding. In our work, we alleviate this issue by adopting ideas from object detectors and incorporate direct pose regression into the detection stage.

2.5 Metrics

This section provides the reader with an overview of the metrics used throughout the thesis. The next paragraph introduces metrics for object detection. Afterwards common metrics for object pose estimation are presented.

2D Localisation The most common heuristic that defines if a bounding box detection counts as correct is the Intersection-over-union, abbreviated to IoU:

$$IoU = \frac{\hat{box} \cap box}{\hat{box} \cup box}, \tag{2.1}$$

where $\hat{box}$ and box are the estimated and the ground truth 2D bounding box, respectively. Based on their IoU-values true positive detections are distinguished from false positive detections. Using the IoU for thresholding one of the most basic metrics for object detection is the recall:

$$recall = \frac{tp}{tp + fn}, \tag{2.2}$$

Where tp, fn are the number of true positive and false negative detections, respectively. The precision:

$$precision = \frac{tp}{tp + fp}, \tag{2.3}$$

with fp being the number of false positive detections. Using these metrics the harmonic mean of recall and precision is defined as:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{2.4}$$

Results for object detection according to the Microsoft COCO object detection challenge [34] are reported using the the mean Average Precision (mAP):

$$mAP = \frac{1}{a} \sum_{a=0}^{a} \sum_{t=0}^{t-1} (recall_t \cdot recall_{t+1}) \cdot precision_t, \tag{2.5}$$

for different values of IoU thresholds t and the mean over the number of object classes a. When reported we use the discrete integral of t from 0.5 to 0.95 in 0.05 steps.

6D Localisation A commonly used metric to report object pose estimation quality is the ADD(-S) error [112]:

$$e_{ADD} = \underset{m \in M_i}{avg} ||\hat{P}m - Pm||, \tag{2.6}$$

$$e_{ADDS} = \underset{m_1 \in M_i}{avg} \underset{m_2 \in M_i}{min} ||\hat{P}m_1 - Pm_2||. \tag{2.7}$$

ADD measures the average deviation of model points using the corresponding point distance. For objects exhibiting coordinate frame transformations that result in ambiguous views, i.e. visual object symmetries, the ADDS error, using the closest point

distance, is calculated. We report the fraction of poses below the commonly used error threshold of 10% of the object diameter, the ADD(-S) recall.

When comparing against the BOP challenge [75], we use the respective metric. The deviation of the estimated pose $\hat{P}$ to the ground truth P is projected to a scalar value using the average recall of three error metrics. These are the Visual Surface Discrepancy, the Maximum Symmetry-Aware Surface Distance and the Maximum Symmetry-Aware Projection Distance:

$$e_{VSD} = \underset{p \in \hat{V} \cup V}{avg} \begin{cases} 0 & \text{if } p \in \hat{V} \cap V \wedge |\hat{D}(p) - D(p)| < \tau \ , \\ 1 & \text{otherwise} \end{cases} \tag{2.8}$$

$$e_{MSSD} = \min_{s \in S_i} \max_{m \in M_i} ||\hat{P}m - Ps||_2, \tag{2.9}$$

$$e_{MSPD} = \min_{s \in S_i} \max_{m \in M_i} ||proj_{3D \to 2D}(\hat{P}m) - proj_{3D \to 2D}(Psm)||_2, \tag{2.10}$$

where $\hat{V}$ and V are sets of image pixels; $\hat{D}$ and D are distance maps and τ is a misalignment tolerance. Distance maps are rendered and compared to the distance map of the test image to derive $\hat{V}$ and V. S_i is a set of symmetry transformations that depend on the visual ambiguities of the object mesh. M_i is a subset of the mesh vertices and $proj_{3D \to 2D}(.)$ denotes the projection to the image space. For each of these metrics the average recall (AR) is measured when comparing errors to multiple error thresholds (and τ in the case of e_{VSD}). Results are then reported as the Average Recall: $AR = (AR_{VSD} + AR_{MSSD} + AR_{MSPD})/3$.

Chapter 3

Object Localisation under Domain Shift

This section presents detailed descriptions of the methods proposed in this thesis. After introducing the relevant concepts for creating rich background information and physical sampling of scenes for synthetic data creation, we present techniques to adapt to the depth, respectively to generalise within the RGB domain. Thereafter, we demonstrate the basic approach to solve the problem of efficient simultaneous detection and 6D pose estimation of multiple objects of interest. Subsequently, this approach is extended to RGB-based pose estimation, specifically addressing the challenges accompanying domain shift in this richer domain. Ultimately, our method is extended by proposing a more efficient true location sampling scheme for training, and by providing end-to-end trainability for constant runtime object pose estimation. For training object meshes either generated using Computer Aided Design (CAD) or reconstructed from physical object models are used.

3.1 Training Data Creation and Transfer Learning

The following paragraphs present physically-based scene sampling for training data rendering. We render synthetic depth data of a virtual scene resembling the area of deployment of our model, using Blender[1]. These data are subsequently augmented and annotated using a randomised noise model and are used for supervised training. Domain randomisation, i.e. diverse scene setups and various background information, produces training data with high variation regarding views and occlusion patterns. Additionally, noise heuristics are applied. We create an augmented domain X^{a} that creates a superset $X^{\mathrm{a}} \supseteq X^{\mathrm{r}}$ of the variations in real-world RGB-D images. We take care that X^{a} does not diverge far from X^{r} by choosing variations to not violate the sampling theory. We show that this generates high-quality data to train deep architectures for object detection and pose estimation in depth images. An example of a virtual scene is presented in Figure 3.1.

[1] www.blender.org

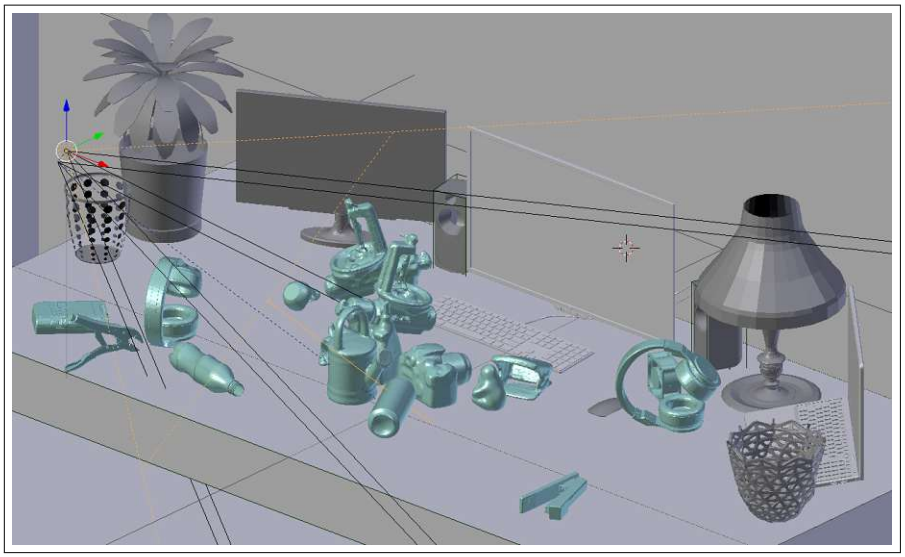

Figure 3.1: **Virtual Scene Setup** An example of a randomly sampled virtual scene for synthetic depth data rendering.

3.1.1 Training Data Rendering for the Depth Domain

We create synthetic data with diverse scene setups and background information in order to produce data with high variation to train object detectors and pose estimators. Three different approaches to create synthetic scenes are chosen in order to evaluate the importance of the background information:

- *simple*: Objects are arranged on a table, without further background information.
- *limited*: Objects are arranged on a table with static domain-relevant background objects.
- *realistic*: Objects are arranged on a table with static domain-relevant background objects and randomly placed domain-relevant objects.

Table 3.1 presents a list of background objects used for rendering. The additional objects are downloaded from GrabCAD[2].

In order to render training images from virtual scenes exhibiting the expected variations regarding views, object poses and occlusion patterns, for each image we:

1. Set a plane that acts as a ground plane for object sampling. Domain-related objects are set to fixed positions, depending on the experiment in Table 3.1. These are either none for *simple* or those listed under *limited* and *realistic*. The center of the tabletop is colocated with the origin of the scene's coordinate system. The x-y-plane of the coordinate system is parallel to the tabletop.

[2]https://grabcad.com

Table 3.1: **Virtual Scene Background** Objects sampled for different virtual scenes' background clutter.

simple	no additional background information
limited	Apple IMac, bin, keyboard, lamp, laptop, two types of screens, mouse, pot plant, speakers
realistic	All from the *limited* objects, Apple Iphone, ball, BeatsAudio, two types of cans, bottle, Buick model, bulb, DualShock 4 controller, pc fan, knife, Nintendo Gameboy, Nvidia GeForce GTX 1080, plier, spacer, stapler, tablet

2. Draw nine objects of interest and drop them onto the tabletop. The x and y offset is sampled uniformly and independently up to 30 centimeters away from the scene coordinate system's center.

3. Draw either no domain related object for the case of testing *simple* background (Table 3.1), or between six to ten domain related objects for testing the *realistic* case and drop them anywhere on the tabletop.

4. Sample the camera pose similar to the expected poses uniformly in the upper hemisphere of the table. The viewing direction is fixed to the origin of the scene's coordinate system. The virtual camera system consists of one camera and a light source mimicking an infrared light projector.

5. Render a depth image, a binary mask indicating visible image regions, object masks indicating pixel-level class correspondences of the visible pixels and individual object masks without occlusion.

6. Compute the ground truth for bounding boxes, class identities, object poses in the camera coordinate frame and per object visibility ratios using the previously rendered object masks.

The output of the rendering process is a quadruplet of a depth image, binary visibility mask, object masks and the ground truth annotations. The binary mask provides information about image regions with invalid depth values depending on the imaging geometry of the virtual infrared depth sensor setup. Objects are annotated with a bounding box, 6-DoF pose and visibility ratios in the image plane. Figure 3.2 shows an example of the synthetically rendered depth images and visibility masks.

In theory this procedure can easily be extended to RGB rendering by assigning material properties to the scene objects and using a ray tracing-based rendering engine. In practice we use the renderer of [68], which was released during the creation of this thesis. Both versions are interchangeable, yet [68] provide a stand-alone Python [3] API, which makes their approach more modular.

[3] https://www.python.org/

Figure 3.2: **Renderer Output** Synthetic depth image (left), visibility mask (middle) and pixel level class correspondence (right).

3.1.2 Adaptation in the Depth Domain

Different works address the problem of creating synthetic depth data with realistic noise characteristics [44], [73], [113]–[115]. We combine their findings with domain randomisation [72] to apply an advanced augmentation strategy. We pair a sensor model with Perlin [86] noise-based pixel warping to augment rendered depth images with realistic noise. In order to create a superset of the variations of real-world depth images, and to force the trained networks to handle diverse levels of noise, we randomly choose the parameters of our augmentation for each image. Various works evaluate and quantify the errors of the depth scans from infrared-based structured light cameras such as the *Microsoft Kinect V1* [116]–[118]. The most common sources of error are the depth sensor itself, the measurement setup and the properties of the object surface. Missing depth values are typically caused by infrared occlusion, specular surface reflection and gaps in the depth images due to strong light [118]. Our approach is designed for objects of interest with surface materials that diffusely scatter incoming light, hence omitting the simulation of specular reflections. We propose to randomise the parameters of our augmentation to account for the intractable number of variations and combinations of the influences in the depth image capturing process.

Randomised Sensor Model Based on the imaging geometry, parts of the scene are occluded, these occlusions are affected by strong light illuminating the scene. In order to simulate that influence, morphological opening and subsequent median filtering is applied to the mask image, which is created by the rendering script. The binary mask is applied to the synthetic depth images to remove the occluded image regions. The kernel sizes are sampled from $\{3, 5, 7\}$. These kernel sizes are also used for blurring. For further augmentation, depth images are resized to 320 by 240 pixels, since that is the resolution of the infra-red based structured light camera, the *Microsoft Kinect V1*. The images are down-sampled using area interpolation to avoid aliasing. Blur is added to minimise the discrepancy between depth gradients in the real-world and synthetic images. The standard deviation of the blurring operation is chosen uniformly in a range from 0.25 to 3.5. The synthetic depth values are rounded to the nearest quantisation value, based on the hypothesised sensor's depth resolution [118] to obtain synthetic depth values in an eleven bit range. Additional noise is added to the quantized depth values using an offset chosen randomly from a Gaussian distribution. The depth noise of the sensor increases non-linearly with depth, though since the expected object

placement lies in the range between 30 centimeters and 150 centimeters we approximate it linearly, similar to [119]. The offset is calculated per pixel using its nearest quantised value, scaled by the parameter n_{sd}. The randomized parameter n_{sd} is drawn uniformly between 0.002 and 0.004. This range is based on the actual depth noise of the *Microsoft Kinect V1*.

Perlin Noise Further randomness of the appearance of occluded scene parts, depth and lateral noise is added by warping the depth images through the application of pixel offsets, using the Perlin noise technique [86]. This approach is similar to Zakharov *et al.* [45]. The basic concept is that a 3D vector field is generated to randomly distort synthetic depth images. Pixel locations are warped by applying the sampled vector field to the already augmented depth images. We use their proposed parameter ranges. For training, images are colour coded using the approach of [120]. A comparison of a real world depth image (left) and a synthetic depth image (right) is presented in Figure 3.3. Figure 3.4 compares different levels of augmentation.

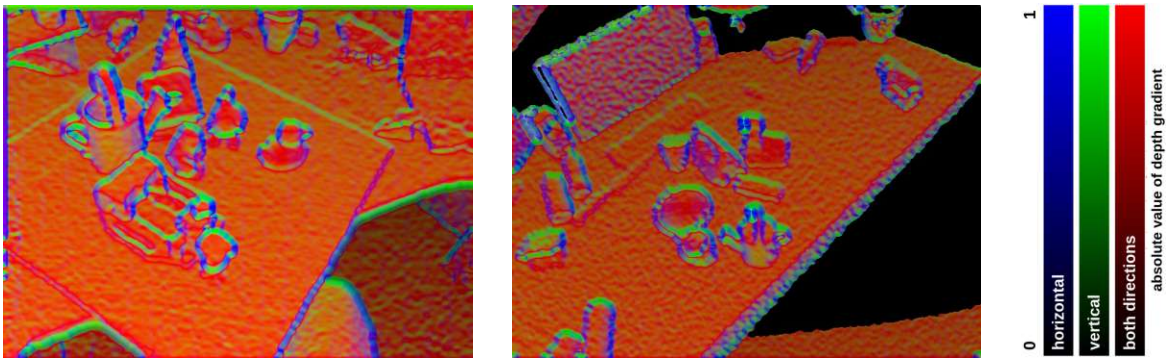

Figure 3.3: **Depth Domain Comparison** Comparison of a real-world (left) and a synthetic augmented (right) depth image, colour coded to using [120].

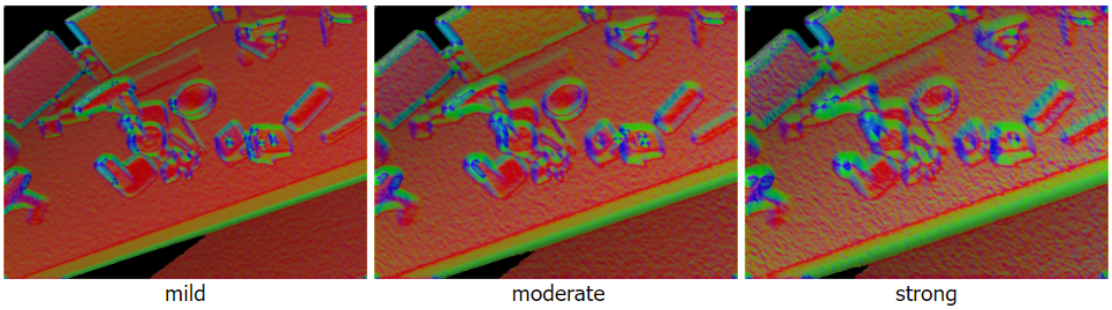

Figure 3.4: **Depth Augmentation** Synthetic depth images with different intensity levels of our camera model, colour coded using [120].

3.1.3 Training Data Rendering for the RGB Domain

Early deep learning-based object pose estimation approaches train on synthetic data generated using rasterisation [23], [48], [62]. Hodan *et al.* [71] show that physically-

based pose, camera and appearance sampling improve object detection rates. The authors of [68] present a publicly available toolkit for creating training data with rich annotations suited for diverse computer vision problems. Rendered using physically-based rendering (pbr). For standard datasets we use the data generated by [68] provided with the BOP-challenge [75].

For high-performance custom dataset creation we generate data using OpenGL[4], rendering approximately 50,000 training images with a resolution of 640×480 in ~ 5 hours. Data sampling per image is performed with the following steps:

1. Randomly sample background image from [34].
2. Randomly choose objects from the set of object meshes and uniformly sample the 6D object pose in the camera frustum. Randomise rendering parameters based on ranges centered at physically plausible values, such as index of refraction (ior). In practice the ior can be derived from look-up tables depending on the material. Approximations based on parameters such as the ior provide practical mean values of the distributions to sample the respective parameter such as diffuse, specular and metallic reflection from.
3. Randomise illumination and light color.
4. Render objects starting with the farthest from the camera. Iteratively update the image and the mask image.
5. Render all sampled objects separately transformed to the image center using the lookAt-function[5] to create per instance mask images and compute visibility fractions.
6. Derive annotations such as amodal and visible bounding box from the per instance mask or the mask image, respectively.

For custom PBR data we use the BlenderProc toolbox of [68], which takes roughly 50 hours for rendering 50,000 images on an NVIDIA Geforce RTX 2080 Ti. Figure 3.5 presents training images rendered using OpenGL with the left image pair and BlenderProc with the right image pair. The highlighted crops, the second and the fourth image from the left present appearance changes based on the respective rendering type.

3.1.4 Generalisation in the RGB Domain

This section provides descriptions about the main concepts used for generalisation within the RGB domain. We apply randomised image augmentations during training, training is initialised using pre-trained weights and during fine-tuning with the presented synthetic data, parts of the network are not updated to improve domain generalisation.

Data Augmentation Domain generalisation is encouraged by applying randomised image augmentations during training. We apply augmentations to the synthetic training

[4] https://www.opengl.org/

[5] https://www.euclideanspace.com/maths/algebra/vectors/lookat/index.htm

Figure 3.5: **Training Data Rendering** Comparison of rendering object views on top of Microsoft COCO images [34] using rasterisation (left image pair) and rendering from cluttered virtual scenes using raytracing with BlenderProc [68] (right image pair).

Table 3.2: **Data Augmentations** RGB image augmentations are sampled from the listed chances and parameter distributions.

Augmentation	Chance (per channel)	Range
gaussian blur	0.2	$\sigma \sim \mathcal{U}(0.0, 2.0)$
average/median/motion blur	0.2	$\sigma \sim \mathcal{U}(3, 7)$
bilateral blur	0.2	$\sigma \sim \mathcal{U}(1, 7)$
hue/saturation	0.5	$\mathcal{U}(-15, 15)$
grayscale	0.5	$\mathcal{U}(0.0, 0.2)$
add	0.5 (0.5)	$\mathcal{U}(-0.04, 0.04)$
multiply	0.5 (0.5)	$\mathcal{U}(0.75, 1.25)$
gamma contrast	0.5 (0.5)	$\mathcal{U}(0.75, 1.25)$
sigmoid contrast	0.5 (0.5)	$\mathcal{U}(0, 10)$
logarithmic contrast	0.5 (0.5)	$\mathcal{U}(0.75, 1.0)$
linear contrast	0.5 (0.5)	$\mathcal{U}(0.7, 1.3)$

images similar to those of [23], [39], [40]. All parameters and ranges can be viewed in Table 3.2. However, two aspects are not sufficiently addressed by the state of the art. Rendering synthetic data produces smooth object contours, especially when rendering object views and pasting these on random backgrounds. In order to induce increased noise on object contours, linear, gamma, sigmoid and logarithmic contrast functions are applied as augmenters. Since illumination changes largely contribute to the image appearance, we choose to increase local image variations by sampling blob-like patterns using frequency noise and changing contrast and brightness locally. The scale for the noise function is sampled uniformly between 0 and 4. Masked image regions are additionally augmented sampling *multiply* and *linear contrast* of Table 3.2. Figure 3.6 compares a raw rendered image to three random augmentation samples.

Pretrained CNN-layers Initialising backbone networks [104], [121], [122] of CNNs with weights that are already trained for image recognition has widely been shown to improve learned representations [40], [49], [50], [58] and enables fast network fine-tuning for problems such as object detection [123]. As such, our networks used for generating

Figure 3.6: **Exemplary Augmented Images** Comparison of raw and randomly augmented RGB images.

the results from RGB data are pre-trained on ImageNet [14]. This initialisation also allows to freeze the weights of certain layers to improve generalisation.

Restricted Training The performance reduction occurring when estimating poses under domain shift is partially alleviated by setting low-level stages of the backbone to non-trainable [49], [58], [124]. EPOS [58] freezes the majority of the backbone, i.e. early and middle flow of Xception-65 [121] when training exclusively on synthetic data. We experienced this strategy to be infeasible for feature pyramid-based approaches, since predictions are made from different feature map resolutions taken also from early and intermediate feature maps of the backbone. Thus, freezing the weights of all layers up to the output layers reduces the pose estimation performance due to the fact feature learning for the present task is limited. We only freeze 14 convolution layers, similar to [49], [106]. As such, the first 2 stages of Resnet [104] are effectively frozen. We additionally experience performance drops when updating any of the batch normalization layer's parameters. Best results are achieved by not updating these layers at all during training. Additional performance improvements are observed when exploiting the self-regularising effect of Mish [125] in task specific network heads.

3.2 Supervised Simultaneous Detection and Pose Estimation

This section starts by presenting the basic principles adopted for simultaneous object detection and pose estimation. Subsequently, these principles are adopted and modified for object detection and pose estimation from depth images. The presented formulation is extended for providing estimates from RGB data. Ultimately, the approach is improved to provide scalable object detection and pose estimation. The approach results in negligible runtime increase with respect to the number of processed object instances.

3.2.1 Basic Approach

Our aim is to classify and estimate the poses of all object instances in a single RGB input image. The 6D pose is defined as $\hat{P} \in SE(3)$, which represents the object's rotation $R \in \mathbb{R}^3$ and translation $t \in \mathbb{R}^3$ with respect to the camera's coordinate frame.

Single-stage feature pyramid-based object detectors provide efficient multi-object handling and generalise well under domain shift [28], [29], [33], [56]. Their main building blocks are a backbone network for feature extraction, a feature pyramid network for creating hierarchical multi-scale semantic feature maps [32], [33], [91], [93] and task specific network heads. Our approach is based on Retinanet[6] [56] with Resnet backbone [104], pre-trained on ImageNet[126], with the Feature Pyramid Network [32] (FPN) for multi-scale feature learning and two head networks. The first predicts a one-hot encoding of the object class per feature map location, learning is supervised using focal loss [56]. The second head estimates a bounding box per location and is supervised using smooth-l_1. Figure 3.7 presents the basic network for multi-object pose estimation. An additional network head is added to Retinanet. Like the two generic heads, the additional head takes the hierarchical features from the feature pyramid network as input. The output per location is an estimate of the standardised encoding of the sampled training targets for pose estimation, i.e. geometric object correspondences.

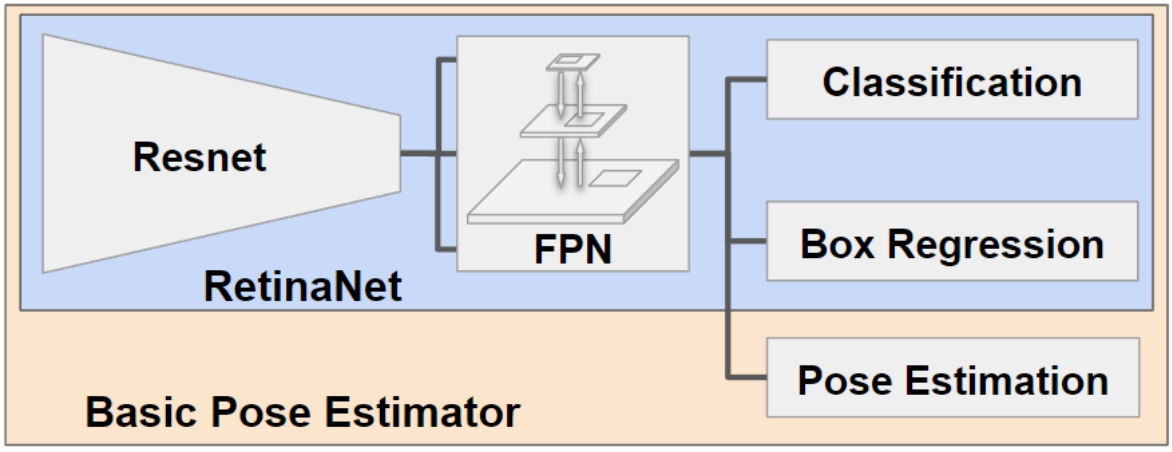

Figure 3.7: **Basic Pose Estimator** We adopt Retinanet [56]. Resnet [104] is used as feature extractor, FPN [32] for hierarchical feature learning and we add a head-network for pose estimation.

3.2.2 Anchors

Effective assignment of true image locations for updating network weights during training is an ongoing research problem [56], [88]–[90], [95], [127]. For feature pyramid-based networks these locations are sampled in the hierarchical feature maps of feature pyramids [32]. Anchors [30] are a widely-used approach for providing bounding box priors to choose these locations [28], [30], [56], [92]. We adopt this sampling strategy for our basic approach. For each image location in the multi-scale feature map, 9 differently shaped and sized bounding box priors are sampled. This requires 16 hyperparameters: 5 each for *base sizes* and *strides* and 3 each for *ratios* and *scales* [56]. The resulting priors that overlap the 2D ground truth bounding boxes with an IoU of more than 0.5

[6] https://github.com/fizyr/keras-retinanet

are considered as true training locations and are used to update the weights. Training targets for regression are standardised using the offset of the respective location from the image space origin as mean, and the prior's width and height as standard deviation. In practice, ratios, scales and sizes are adjusted based on the expected bounding box sizes. This results in two convenient traits:

- Sampling anchors leads to a uniform scale space for the expected bounding boxes. As such, a similar amount of training locations are sampled per object, independent of the object's size in the image space.
- Regression targets are standardized using the respective anchor's center, width and height. This means that the regression target space has similar statistics for differently sized objects.

3.2.3 Pose Representation

Providing appropriate regression targets is very important in order to enable effective task learning, and for deriving poses from the predicted representation. Early learning-based approaches tried to directly estimate or classify rotation in terms of Euler angles, Quaternion or using the Rodrigues' rotation representation [6], [23], [48], [61], [62]. Object translation is inferred from the 2D bounding box size and the estimated object rotation, or is directly regressed or classified. An arguably better suited representation for learning-based pose estimation are geometric correspondences [22], [39], [40], [49], [53], [54], [58], [59]. The two standard ways to use geometric correspondences as regression targets are either to predict uv-coordinates [39], [40], [49], [58] or keypoints [22], [53], [54], [59]. Using the 3D geometric correspondences in the object frame and their corresponding predictions in the camera frame, solving PnP estimates the 6D transformation that solves the perspective projection equation. This is equal to the object pose. While no clear study exists comparing uv-coordinates and keypoints, the two obvious main differences are output size and number of hypotheses. Predicting uv-coordinates in practice requires encoder-decoder networks since the output space has to be dense. Most of uv-coordinate-based approaches make predictions using separate networks per object, with object crops derived from the estimated bounding box as the input. Keypoints are sparse geometric correspondences, thus can be predicted from latent feature spaces like those of hierarchical feature maps. As an example, the required output parameters for Pix2Pose [40] to produce geometric correspondences for one object are $256 \times 256 \times 3$, thus 196,608. EPOS [58] presents a single-model estimator independent of the number of objects and instances involved, their output space for the standard pose estimation image space of 640×480 [75], is $640 \times 480 \times 3$, thus 921,600. In contrast to these, estimating a sparse set of correspondences using the approach presented in Section 3.5 only requires 6300×16, thus 108,000 output parameters for processing multi-object multi-instance scenarios. The state of the art reports a higher performance for dense correspondences due to their larger hypotheses space, compared to keypoint-based approaches [47], [54], [59]. Notably, the authors of [22] show that providing a large set of keypoint hypotheses also leads to good pose estimation results. Since we use all anchors that sufficiently overlap with an object

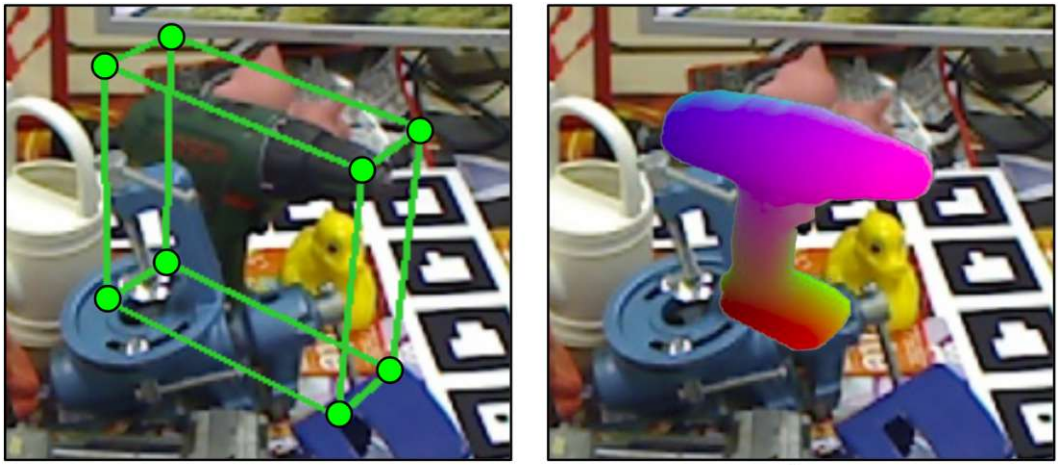

Figure 3.8: **Geometric Correspondences** a) Keypoints, green corner points of smallest cuboid enclosing the object and b) color values based on the vertex location in the object coordinate frame.

bounding box for location sampling, our formulation also results in many hypotheses per object instance. Therefore, a diverse set of hypotheses from multiple latent feature vectors is created. We argue that this procedure leads to similarly rich hypotheses space as those of approaches using dense correspondences, while being computationally more efficient.

Object meshes are considered to be known in advance, but no additional information regarding the test scene is required. We define the corner points of the smallest cuboid enclosing the respective object mesh in its coordinate frame as the geometric correspondences (G_{3D}). Figure 3.8 provides an example of the smallest enclosing cuboid of LM's Drill [112] in *a*. Keypoint locations are indicated as green dots with black borders. The image in *b* presents a dense set of correspondences, coloured based on their uv-coordinates.

3.2.4 Supervised Learning

The basic network learns to generate the outputs $\hat{O}$, $\hat{B}$ and $\hat{G}$. The first predictions are the set of object class probabilities $\hat{O} \coloneqq \{\, \hat{o}_0, ..., \hat{o}_k\}$, where k is the number of image locations in the multi-scale feature map and $\hat{o}_k \in \mathbb{R}^a$ is the Bernoulli-distributed object class prediction. We denote the number of object classes in the dataset with a. The second module predicts the amodal bounding boxes $\hat{B} \coloneqq \{\, \hat{b}_0, ..., \hat{b}_k\}$, where $\hat{b}_k \in \mathbb{R}^4$. The third module predicts the projection of G_{3D} to the image space $\hat{G} \coloneqq \{\, \hat{g}_0, ..., \hat{g}_k\}$, where $\hat{g}_k \in \mathbb{R}^{16}$. The overall loss function is defined as:

$$L = \alpha \cdot L_O + \beta \cdot L_B + \gamma \cdot L_G, \tag{3.1}$$

where L_O, L_B and L_G are the losses for object classification, bounding box and correspondence regression and where α, β and γ are loss weights. Learning to predict

the Bernoulli distribution for the object class is supervised using the Focal loss [56] and the loss for learning bounding box and keypoint prediction is supervised using the Smooth-l_1 [56].

We weigh the contribution of each component of the overall loss differently. In practice the weights are set in a way that the starting loss for L_B is twice and that for L_g four times as much as L_O. Our networks are trained using Adam [128] optimiser with a base learning rate of 10^{-5}, reducing the learning rate to one tenth every time the overall loss didn't reduce for two consecutive epochs.

3.2.5 Symmetry Handling

Symmetry handling in the context of deep learning is necessary to not hinder the learning process while training networks [23], [40], [47], [48], [58], [129]. Objects with continuous symmetries can be rotated arbitrarily around the axis of symmetry, resulting in an infinite number of similar, thus ambiguous views. When training these cases, without employing special measures to keep the angle error between the predicted and the ground truth pose near zero, the training process is hindered. The left plot of Figure 3.9 compares the calculated angle error and the desired angle error for the loss calculation of objects with continuous symmetries. When handling objects with discrete symmetry transformations, transforming the objects with said transformation leads to views resulting in the same object appearance. This can lead to ambiguities during loss computation. The right plot of Figure 3.9 compares the calculated and the desired angle error for the loss calculation of an object with discrete symmetry. The desired absolute angle error is zero for similar views.

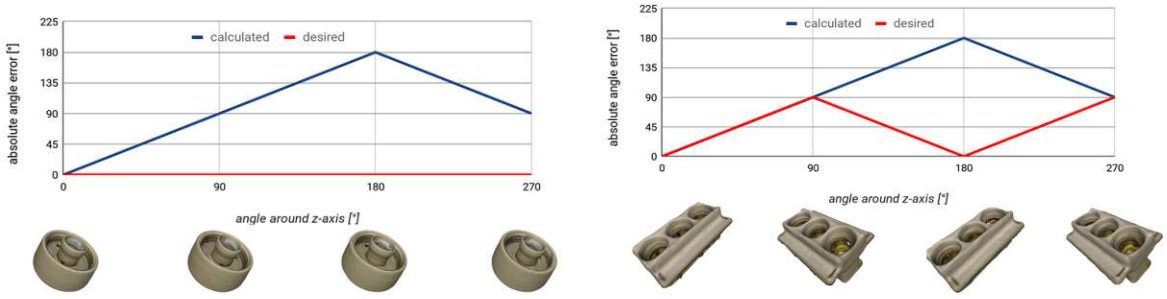

Figure 3.9: **The Problem of Object Symmetries** Calculated versus desired angle error for loss calculation of object 17 of T-LESS, revolved around its axis of symmetry (left) and calculated versus desired angle error for loss calculation of object 9 of T-LESS, revolved around z-axis, resulting in ambiguous views (right).

The standard case is that object models and annotations of object transformations that result in ambiguous views are available, or can easily be manually defined if not available. As such, common strategies to take these symmetries into consideration are restricted training data creation [23], [47], [130] and formulating learning in a way to not hinder convergence [38], [58], [129], [131], [132]. However, there are works considering learning to recognise such symmetries [133].

In this work, in order to disambiguate continuous symmetries, the annotated object pose is transformed by rotating the object along its axis of symmetry. The object is transformed so that the plane spanned by the axis of symmetry and an arbitrary orthogonal axis of the object frame pass through the camera center. The left portion of Figure 3.10 presents the annotation before and after transformation. Hence, we overcome the problem of mapping identical views to a continuous set of possible poses. In order to disambiguate discrete cases, object pose annotations are transformed such

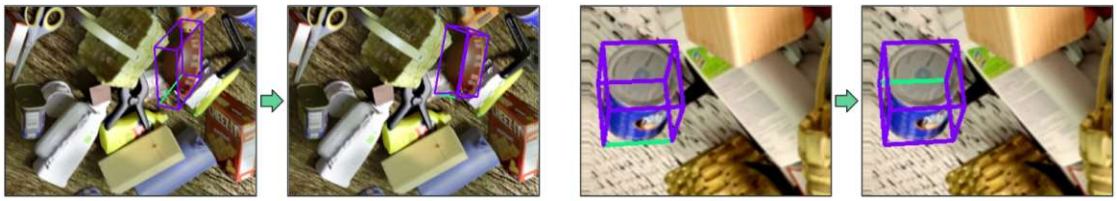

Figure 3.10: **Transformed Annotations** Left presents a solution to map continuous symmetries to unambiguous pose annotations, right presents mapping discrete symmetries to unambiguous pose annotations, on objects of YCB-video [48].

that the normal of the plane of symmetry always points to the same side of the image, i.e., to the positive or negative x-direction of the image frame. Consequently, the same control points are consistently either on the right or on the left side of the object during training. As such ambiguous views are consistently annotated with only one ground truth pose, the right side of Figure 3.10 presents an example. These simple transformations disambiguate object symmetries and thus do not hinder network convergence during training. The proposed transformations can already be applied during rendering. Yet are also applicable online, during training data annotation or training target sampling.

3.2.6 Deriving Object Poses

Each location sampled in the hierarchical feature maps outputs the image locations of the corner points of the smallest cuboid enclosing the respective object. During runtime anchors with a sigmoid classification score above the detection threshold are considered as true anchor locations, and are used to generate the hypotheses for geometric correspondences in the test image. Having the 3D locations of the defined keypoints in the object frame g_{3D}, their estimated 2D projections in image frame g_{2D} and the intrinsics K of the calibrated camera, solves the perspective projection model:

$$g_{2D} = K \cdot [R|t] \cdot g_{3D} \tag{3.2}$$

for R, the 3D rotation, and t, the 3D translation retrieves the object pose. Figure 3.11 shows a visual representation of the relationship between the geometric correspondences in object and in the image frame. The Perspective-n-Points (PnP) algorithm provides a solver for deriving R and t [36], [66], [109]. Since the set of estimated correspondences is

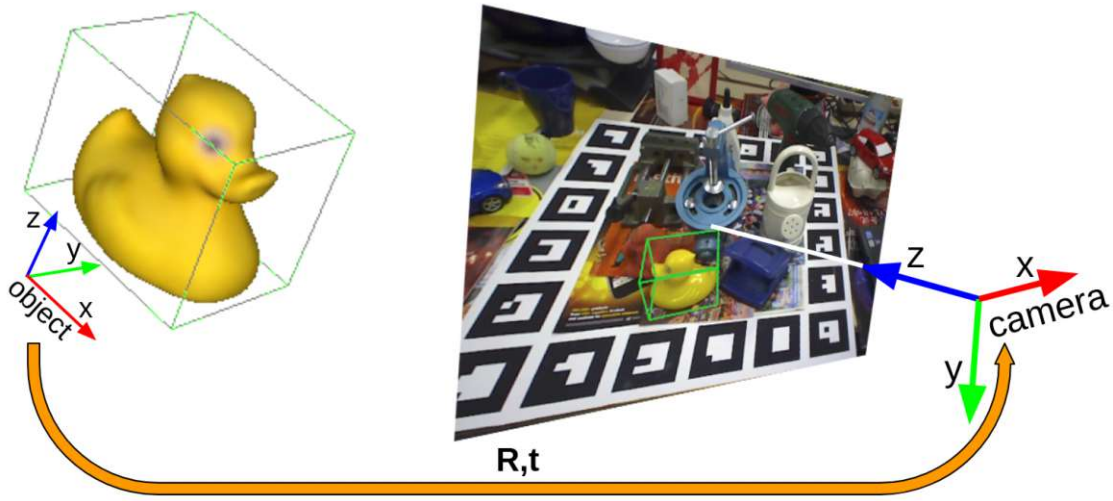

Figure 3.11: **Perspective Projection** Knowing the corresponding 3D locations in the object frame of the estimated keypoints in the 2D image frame allows estimating the relative transformation using PnP.

noisy, the RANSAC version of EPnP is used [36], [65]. As break conditions we use 300 iterations or an accumulated reprojection error of 5 millimeters with an inlier threshold of 0.99. If any of these conditions is met, the estimated pose leading to the lowest reprojection error is returned and used as object pose.

3.3 SyDPose: Object Detection and Pose Estimation in Cluttered Real-World Depth Images Trained using only Synthetic Data

Recently deep learning advanced the state of the art for computer vision tasks. Nevertheless, the advent of deep networks for 3D pose estimation has yet to be fully realised, given that classical approaches predict poses exclusively from geometry priors with higher performance compared to deep-leaning approaches [55]. While deep networks achieve superior performance, they require a huge amount of training data [14]. Capturing and annotating these data is time-consuming and labour-intensive, requiring physical object instances, which is problematic in fast paced manufacturing environments. Thus, for industrial applications it is desirable to train pose estimation pipelines from CAD data only, since it is usually readily available.

Classical feature- and template-based approaches for pose estimation employ meshes or point clouds, to create templates or hash tables in order to detect objects and estimate their pose at runtime [24]–[26], [134]. Consequently, these methods require no real-world data for training. Current deep learning approaches do not close the domain gap, i.e. traversing from synthetic to real-world data without a decrease in performance, therefore they need real-world data during training [47], [48], [54], [59]. Recent approaches showed that domain adaptation for 6D pose estimation is easier in

the depth domain [6], [45], [87]. Therefore, we address the task of training deep pose estimators only from synthetic depth data by rendering and augmenting these data in terms of background information and sensor noise through random shape perturbations, presented in Section 3.1.2 and Section 3.1.2.

Pose estimation is a non-trivial task for learning-based approaches, consequently researchers tend to separate detection from pose estimation or even train separately for distinct classes [23], [45], [47], [48], [59]. However, end-to-end learning, i.e. training and deploying multiple stages of a vision pipeline at once, is desirable to reach a high frame rate and to allow practical application. Additionally, when deep architectures are trained on multiple objectives, i.e. in a multi-task fashion, the learned features are stronger, which has been shown to be beneficial for each individual task [92]. Especially when employing pre-trained models in a domain different from RGB, e.g. on depth data, retraining the backbone with additional guidance is required to adopt the learned feature extraction to the domain. We experiment with the output space for multi-object 6D pose estimation. By providing specific output parameters per object class we improve pose estimation performance. Our multi-task, end-to-end models for pose estimation are thus trained with the capacity to simultaneously localise the distinct desired objects in image space, and to estimate their poses in 6D.

A standard approach to predict 6D object poses is the prediction of the 3D bounding box encapsulating the object of interest. The bounding box's control points are projected into 3D using PnP to estimate the pose [47], [54], [59], [105]. These approaches require a lot of training data and time to converge to a usable state. We provide a simple extension that improves convergence. Comparing edge length differences in image space enforces orthogonality on the estimated 3D bounding box. Since this is done without having to re-project the control points to the 3D space while training, the computational overhead is kept low.

In summary, we propose a method for texture-less object pose estimation in real-world depth images using only synthetic data for training. Our approach runs at almost 20 frames per second (fps) using an Nvidia Geforce GTX 1080.

- To the best of our knowledge we present the first end-to-end learning-based approach for simultaneous object detection, classification and pose estimation of an arbitrary number of texture-less objects in real-world depth images trained using only meshes.
- We present a general extension to the loss function used for 3D bounding box estimation. By comparing all parallel 3D bounding box edge length differences, we favour 3D orthogonality and thus increase performance.
- State-of-the-art performance on the LM [112] and LM-O [135] dataset.

3.3.1 Approach Description

Figure 3.12 presents an overview of the 6D pose estimation approach. One end-to-end trainable network produces outputs for all objects of interest, described in Section 3.2.1.

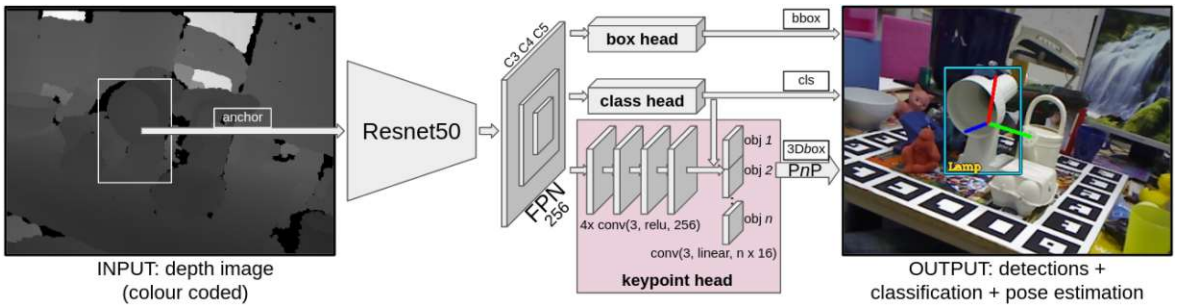

Figure 3.12: **SyDPose** Multi-task, end-to-end network for object detection, classification and pose estimation. Trained on augmented synthetic data, poses are represented as geometric correspondence locations in the image space of real-world depth images. 6D poses are estimated using RANSAC-PnP.

Estimated outputs are multiple one-hot encoding vectors presenting the object class, 2D bounding boxes and geometric correspondences, for all object instances in an image. Training data is created as described in Section 3.1.1. Since only synthetic data is used for training, the domain adaptation presented in Section 3.1.2 is applied to the data. The network is trained on this synthetic data and estimates are produced on real-world depth images. During training, compared to the vanilla anchors used in RetinaNet, we do not use different scales for anchors creation, described in Section 3.2.2. This is done since no performance reduction is observed, but training time is decreased considerably due to the reduction of the output space to one third. We additionally regularise the layers of the keypoint head using *l2* regularisation with the hyperparameter set to 0.001. During inference, foreground-background separation is performed using a detection threshold of 0.5 on the one-hot class encoding vector. Non-maximum suppression is applied to the foreground samples per class. Using an IoU of 0.5 the estimated bounding boxes are filtered for that with the highest detection score. The final output is one set bounding box points per object instance. Using these, poses are estimated using RANSAC-PnP, see Section 3.2.6.

3.3.2 Multi-Object Handling

The basic pose estimation network described in 3.2.1 employs the same network head architecture for bounding box estimation and keypoint prediction. These two problems are similar in terms of formulation, yet geometric correspondence prediction is inherently more demanding with respect to the learned encoding, see Section 1.1. Thus, in order to improve multi-object pose estimation performance, we split the output space of the keypoint network head.

The output space for bounding box prediction is $k \times 4$, with k again referring to the amount of latent image locations and 4 values for the x and y coordinates of the upper left and lower right corner of the bounding box. We observe that directly using the same head network for keypoint prediction, with $k \times 16$, the x and y coordinates of the 8 cuboid corners per image location, leads to a low convergence of L_G, see Equation 3.1. In order to improve convergence and the localisation accuracy of the keypoints for

pose estimation we generate an output space of $k \times a \times 16$. In practice we assign 16 specific output channels for each object class in a. During training we only update the corresponding parameters.

3.3.3 Orthogonality Favouring Loss

We observe that the edges of the predicted 3D bounding boxes are not orthogonal to each other when projected to 3D. The state of the art does not take the geometry of the bounding box in 3D into consideration during training [47], [54], [59], [105]. We hypothesise that guiding learning with that additional information improves the learned representation and improves physical plausibility of the estimates. This in turns leads to improved performance using PnP since the set of correspondences to estimate the pose from has less error.

As such, we propose to additionally supervise the keypoint loss L_G of Equation 3.1 with an orthogonality constraint. Thus, L_G becomes:

$$L_G = \delta \cdot L_{pts} + (1 - \delta) \cdot L_{orth} \tag{3.3}$$

where L_{pts} is the augmented l_1 loss used in [56], [136] and L_{orth} is an orthogonality favouring loss with weighting parameter δ. The orthogonality constraint is formally defined as

$$L_{orth} = \sum_{k=0}^{k} \frac{1}{12} \sum_{\{\forall c_i c_j \| i \neq j, e_i \| e_j\}} \Big| (\|e_i\| - \|e_j\|) - (\|\hat{e}_i\| - \|\hat{e}_j\|) \Big| \tag{3.4}$$

per image, where e are the edge lengths, respectively estimated edge lengths $\hat{e}$ of the 3D smallest cuboid enclosing the corresponding object. The factor for normalisation, 12, results from the amount of non-redundant parallel edges of the cuboid.

This loss penalises predicted length differences of two parallel edges on the 3D bounding box in proportion to the deviation from the ground truth length difference. Comparing these over all parallel edges favors orthogonality of the estimated cuboid in 3D space. Since this constraint is directly computed from the estimated keypoints without the need to project points to 3D, the computational overhead is negligible.

3.4 PyraPose: Feature Pyramids for Fast and Accurate Object Pose Estimation under Domain Shift

Exploiting synthetic data is a popular direction in order to generalise to novel domains and to fully automate the training procedure. The best performing deep learning-approaches for monocular 6D object pose estimation employ encoder-decoder architectures [22], [39], [40], [49], [53]. These methods only focus on local changes in image space,

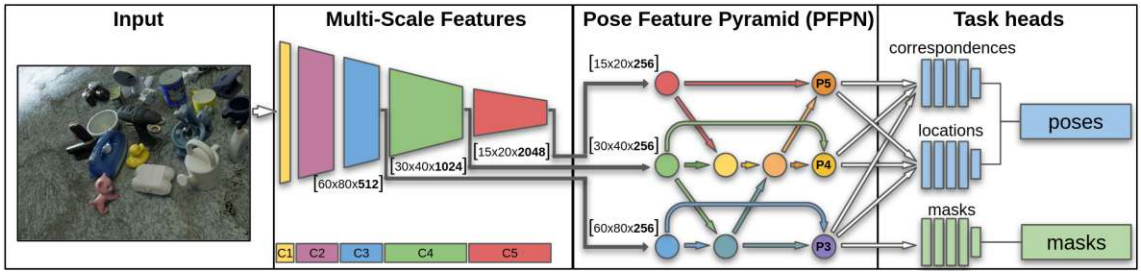

Figure 3.13: *PyraPose*: Monocular 6D object pose estimation under domain shift. Multi-resolution feature aggregation from different backbone stages using a specialized Pose Feature Pyramid Network (PFPN) enables domain generalisation and effective encoding of object scales to the feature space.

which is detrimental to pose estimation under domain shift, i.e., synthetic and real. Alternatively, predicting from coalesced features at multiple spatial resolutions enables processing of local to global information in parallel, and has shown remarkable success for object detection [28], [32], [33], [56], [91], [93]. We hypothesise that this is highly suitable for generalising to novel domains and therefore for object pose estimation under domain shift. RGB-based approaches lack in performance compared to depth-based ones, since the object depth is not directly observable [39]. Given that feature pyramids explicitly incorporate object scales in the learning process, we further hypothesise that they are well suited for RGB-based object pose estimation.

This work proposes to use feature pyramids as the main building block for extracting meaningful hierarchical features for pose estimation, in contrast to those generated by encoder-decoder networks. We show that making predictions from multi-resolution feature maps guides the network to learn robust pose estimation under occlusion and domain shift. Our approach PyraPose, shown in Figure 3.13, is based on a novel Pose Feature Pyramid Network (PFPN), specialised for pose estimation. Our design needs only one network (less than 43M parameters) per set of objects, which leads to fast inference time of ∼26 fps. In comparison to state-of-the-art methods also trained on synthetic data, we achieve up to ∼35% higher accuracy. We show that we effectively transition to novel domains by testing on data that has a severe shift in data characteristics, such as scene setup, illumination and camera intrinsics. Furthermore, real-world grasping experiments with a mobile manipulator demonstrate the full capability of our approach.

In summary, our contributions are the following:

- A novel RGB-based approach for pose estimation based on a new feature pyramid network, PFPN, that creates meaningful multi-scale features for pose estimation.
- The demonstration that multi-scale features are well suited for pose estimation under domain shift, allowing generalisation to different domains of deployment by only training on synthetic data.
- State-of-the-art performance on multiple standard datasets and demonstration of the usability for real-world robot grasping.

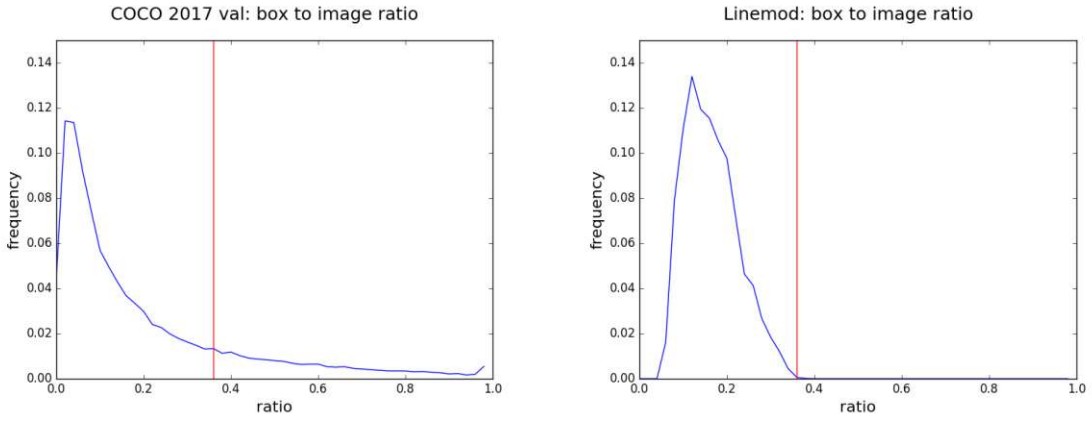

Figure 3.14: *Object Scale Ratios* Frequency of relative bounding box ratios for Microsoft COCO's 2017 validation set [34] and the original Linemod test set [112].

3.4.1 Approach Overview

Our proposed approach for single-shot 6D object pose estimation is trained with synthetic RGB data, introduced in Section 3.1.4. Physically-based rendering [68], [71] provides data with realistic interactions of the object materials with the scene illumination. The first stage is an end-to-end trainable multi-task CNN. This network detects objects in the image and estimates the image locations of the projected per-object geometric correspondences. The centerpiece of the proposed approach is a specialised feature pyramid network, shown in Figure 3.13. In the second stage, RANSAC-PnP computes the 6D pose from the sampled set of correspondence hypotheses per object.

Only one shared network is used, independent of the number of objects. Therefore, only one model is trained per dataset in contrast to the majority of state-of-the-art approaches [22], [23], [40], [46], [47], [49], [129]. This is advantageous because the memory footprint is low, it is easy to apply and training time is short. Furthermore, only one forward pass at runtime is required, independent of the application. As a result, pose estimation is achieved at $\sim$ 26 fps on an Nvidia Titan V.

3.4.2 Pose Feature Pyramid Network

This section presents a specialised Pose Feature Pyramid Network (PFPN) that is designed to learn hierarchical feature maps for object detection and pose estimation. We hypothesise that pose estimation benefits from focusing on low-level features during feature aggregation because:

1. Using Neural Architecture Search [94] to learn better hierarchical feature aggregation architectures [32] in object detection [91] results in an architecture that assigns more layers and feature map merging to low-level features maps than to the high-level ones. This can be explained by the findings of [137], which demonstrate that it is beneficial to encode more abstraction into low-level, respectively highly-resolved feature maps of the feature pyramid.

2. Object pose estimation tendentiously has to deal with objects in a clutter. Consequently, many standard pose estimation tasks are characterised by small object to image scale ratios, see Figure 3.14. Revisiting the anchor box computation of [30], anchors are sampled in the image locations, i.e. pixels, of the hierarchical feature map of feature pyramids. With an input of 640×480, P3 has a spatial resolution of 80×60 pixels, P4 has 40×30 pixels and P5 20×15. For each of these feature maps and for each pixel, a specific set of anchors is sampled using a base size, scales and aspect ratios. Scales and aspect ratios are the same over all resolution, but base sizes correlate with the ratio of image downsampling. The standard value for the base size for P3 is 32 pixels, for P4 it is 64 pixels and for P5 it is 128. The biggest sampled scale is 1.59. Thus, the biggest anchor to image ratio when using these 3 levels is 36%. While there are no objects with a bounding box to image diagonal ratio of more than 36% for Linemod, in Microsoft COCO's 2017 validation set [34] 16.9% of the featured objects are above that threshold, indicated with a red vertical line. Thus, object detection also requires a broader anchor box distribution to properly sample bounding box priors for all expected object scales. For pose estimation, these larger locations produce a considerable computational overhead for little performance improvement and therefore are removed.

3. Features from lower-levels of the backbone provide a larger hypotheses space due to the higher feature map resolution, and they condition the prediction making on local views. Using an IoU-threshold of 0.5 between bounding box and anchor prior is intuitive and in practice showed the best performance. However, in high-level feature maps more anchor locations with a low object to anchor ratio are sampled. This is not desired since learning is biased for holistic object appearances and not local ones, which has been shown to be beneficial [53], [54].

Thus, our network design omits higher-levels of the feature pyramid, which are usually used for object detection (i.e., C6 and C7), and we only rely on the backbone levels C3, C4, and C5 of Resnet [104]. In Figure 3.13, each node represents an add operation followed by a 3×3 convolution with stride 1 and ReLU activation. In order to create pyramid level P3, the low-level features from C3 and C4 are combined. Following [91], all spatial resolutions from the backbone are used to create pyramid level P4. This aggregates features from each available pyramid level and applies a higher amount of convolutions than for P5. The smallest resolved backbone level C5 is minimally processed to keep the computational overhead low. Cross connections are designed to provide information from all available backbone levels to create P5. Skip connections are used on P3 and P4, as is known to be beneficial [33], [104]. We observe that combining only two inputs per add operation and applying a convolution to the output achieves best performance. The aggregated features are then forwarded to the task heads.

3.4.3 Task Heads

Our approach is supervised with 2D and 3D bounding boxes as well as segmentation masks. The 3D bounding box and segmentation mask predictions update the trainable

Figure 3.15: **Sampled Locations and Predicted Masks:** Left shows all the sampled priors (white boxes) used to predict the pose of YCB-video's [48] mustard bottle. Images on the right and top present some partial views in more detail. Right shows coarse mask prediction on an image of Occlusion dataset [135]. Predictions made with resolution 80×60 pixels.

weights while the 2D bounding boxes are only used to compute image locations for pose hypotheses prediction. The location and correspondence heads are used for pose estimation and the mask head predicts the coarse instance segmentation. During training, 3D bounding boxes are projected into image space using the object pose in the camera frame and the camera intrinsics. During inference, 3D bounding boxes are estimated in image space and subsequently re-projected to 3D using PnP and the camera intrinsics to estimate the object's pose.

Pose Hypotheses Prediction The location and correspondence heads classify image regions that contain an object and create 2D-3D correspondence hypotheses, respectively. The left frame of Figure 3.15 shows examples of such image region priors.

Location sampling and correspondence standardisation is done using anchors, see Section 3.2.2. The predicted geometric correspondences representing the object pose are described in Section 3.2.3. Using anchor boxes as priors conditions the network on producing hypotheses from different scales and partial views. This includes a small object to background ratio, which is shown to be beneficial in [53], [54]. The location and correspondence heads use all three feature map resolutions coming from PFPN for prediction making (i.e., P3, P4, P5).

Mask Prediction Since depth data is widely available on robotic platforms, the Iterative Closest Point (ICP) algorithm can be used to refine initial pose estimates. The higher the ratio of object to background points the more accurate is the estimated transformation. Mask prediction is thus vital to achieve high quality performance with ICP. We apply a similar approach to [53] by predicting the mask at one eighth of the image resolution, see the right frame of Figure 3.15 for an example of predicted masks. This leads to reasonable masks while keeping the computational overhead low, as transposed convolutions are not used to upsample. Conversely to the location and the correspondence heads, the mask head takes only the features from P3 as input.

Head Architectures All three task heads share the same architecture. Each consists of four convolution layers with a kernel size of 3×3, a stride of 1 and ReLU [138] activation, followed by another 3×3 convolution to regress the image locations of the eight 3D bounding box corners with linear activation, or to classify with sigmoid

activation. The first four convolutions in the location and mask heads extract 256 features per location. Conversely to the basic approach presented in Section 3.2.1 and the keypoint regression head of Section 3.3 we use 512 features channels. This results in similar performance as compared to using specific output channels per object class, yet improves convergence and simplifies loss weight determination.

Loss Functions For the correspondence head, the orthogonality favouring loss, introduced in Section 3.3.3 is used with δ set to 0.8. By punishing edge length differences of the ground truth and estimated 3D bounding boxes, the regression incorporates 3D information. Both the location and mask heads are optimised using the focal loss [56], which is commonly used to improve the cross entropy loss by overcoming imbalances of per-class samples and in the amount of foreground and background samples. This is achieved by dynamically re-weighting difficult examples during loss computations. During training, the losses are normalised over batches, anchors (location head) and anchor locations (mask head). Weights for the loss of the correspondence, location and mask heads are 0.125, 1.0 and 0.1, respectively.

3.4.4 Domain Generalisation

We use the pbr-data introduced in Section 3.1.3 as training data for the proposed approach. This data contributes to the generalisation of our trained models. Refer to the ablations in Section 4.2.2 for more details on that matter. The generalisation of our trained models to novel domains is encouraged by applying different means for regularisation during training, as described in Section 3.1.4. Random image augmentations are sampled online, individually for each image in a batch. The weights of the backbone are pretrained on ImageNet [14]. The first two stages of Resnet [104] are not updated during fine-tuning, similar to [49], [58], [124]. While [58], [124] freeze the majority of the convolution layers in the backbones, we only freeze the first 28% since detrimental otherwise. This might be connected to the usage of a feature pyramid based approach, where the earliest features used in the feature pyramid are outputs after the 24th convolution, as such after $\sim$ 48% of the backbone. This indicates the need to fine-tune some convolutions on task. Convolution layers of PFPN and the network heads are regularised using the l_2 penalty term.

3.5 COPE: End-to-end trainable Constant Runtime Object Pose Estimation

Learning-based object pose estimation research focuses on maximising the performance under challenging conditions like domain shift, object occlusion and object symmetries by tendentiously separating the detection from the pose correspondence estimation stage [17], [23], [38]–[40], [47], [54] then deriving the 6D pose with the Perspective-n-Points (PnP) algorithm [36], [66], [109] using the estimated geometric correspondences. This leads to shortcomings because a) adopting surrogate training targets decouples pose estimation from the training process and thus limits learning [37] and b) running inference for multi-instance scenarios leads to a computational complexity of at

least $\mathcal{O}(n)$ with respect to the number of objects (n) for the pose estimation stage. Consequently, this type of approach has severely diminishing applicability for realistic scenarios.

Recent object pose estimation research trends recognise those shortcomings and partially alleviate them by directly regressing the 6D pose from the intermediate pose correspondences, and achieve tremendous results [37], [41], [63], [64]. In [41] and [37], detection is separated from the pose estimation stage, which makes them not end-to-end trainable because they require an object detector. The work in [64] is end-to-end trainable but separate networks need to be trained and pooling geometric correspondences means multiple objects and instances cannot be handled simultaneously. We improve upon these findings by proposing a natural extension to efficiently handle multi-object multi-instance scenarios.

This work solves the aforementioned shortcomings by sharing the latent representation, as well as the direct pose regressor over objects and their instances. We classify image locations in the feature maps, regress bounding box and view-dependent object geometric correspondences and regress the direct 6D pose. While the geometric correspondences are intermediate representations, the direct 6D pose head takes this intermediate output as input. Consequently, the loss related to the 6D pose is also backpropagated to the down-stream task of geometric correspondence estimation. This design also allows further guidance of the learning process by enforcing consistency between these consecutive tasks, which additionally improves each of them. Additionally, we propose a concurrent solution to anchors [30] for true location sampling during training that does not require manually choosing hyperparameters based on the expected test data distribution and reduces the output space by a factor of 9. True locations are sampled and regression targets are standardised from a scalar shape prior derived from the respective object mesh and the backpropagated loss is normalised for each object class. Thus, training is not biased towards larger objects and no prior assumptions need to be made in contrast to the case for anchors.

In summary, our contributions are:

- A simple and efficient solution for multi-object multi-instance object pose estimation that improves over the state of the art.
- A training target sampling scheme that requires no assumptions about the test data distribution.
- Experiments quantitatively showing the advantage of directly regressing the pose from estimated surrogate targets.

Efficiently sharing internal representations over objects and instances enables end-to-end trainability that requires only one forward pass through the network to process all object instances in a single input image. We show that, processing more than 90 object instances in a single image with more than $24fps$ on an Nvidia Geforce RTX 3090, our method's performance is competitive to similar state-of-the-art approaches but up to 35 times faster.

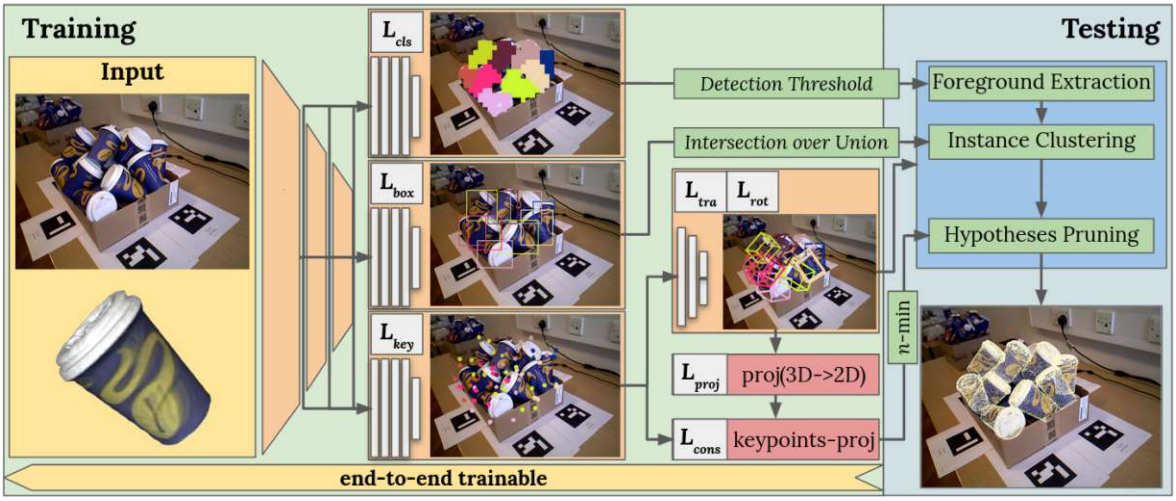

Figure 3.16: **Constant Runtime Object Pose Estimation.** Given an input image and a 3D model, image locations are classified while bounding boxes and geometric correspondences are regressed. A direct pose regression module slides over the image locations and regresses the 6D pose from the geometric correspondences. Training is supervised with losses for each module (L_{cls}, L_{box}, L_{key}, L_{tra} and L_{rot}) as well as auxiliary losses (L_{proj} and L_{cons}) to enforce consistency between estimated correspondences and direct poses. During testing, instances are efficiently clustered using their 2D IoU then the n hypotheses with the highest consistency generate the 6D output.

3.5.1 Constant Runtime via Direct-pose regression

Our aim is to classify and estimate the poses of all object instances in a single RGB input image. The 6D pose is defined as $\hat{P} \in SE(3)$, which represents the object's rotation $R \in \mathbb{R}^3$ and translation $t \in \mathbb{R}^3$ with respect to the camera's coordinate frame. Object meshes are considered to be known in advance but no additional information regarding the test scene is required. We define the corner points of the smallest cuboid enclosing the respective object mesh in its coordinate frame as the geometric correspondences (G_{3D}). COPE, outlined in Figure 3.16, outputs the set of object instances visible in the image, parameterised with object type and 6D pose.

COPE builds upon the success of recent efficient object detection approaches [56], [88]–[90]. The RGB input image is first processed with a CNN backbone and then multi-scale features are computed using a feature pyramid to estimate the intermediate object representation. Three modules shared over feature maps of sizes [$s/8$, $s/16$, $s/32$], with s being the input image resolution, generate the intermediate outputs $\hat{O}$, $\hat{B}$ and $\hat{G}$. The first predicts the set of object class probabilities $\hat{O} \coloneqq \{\ \hat{o}_0, ..., \hat{o}_k\}$, where k is the number of image locations in the multi-scale feature map and $\hat{o}_k \in \mathbb{R}^a$ is the Bernoulli-distributed object class prediction. We denote the number of object classes in the dataset with a. The second module predicts the amodal bounding boxes $\hat{B} \coloneqq \{\ \hat{b}_0, ..., \hat{b}_k\}$, where $\hat{b}_k \in \mathbb{R}^4$. The third module predicts the projection of G_{3D} to the image space $\hat{G} \coloneqq \{\ \hat{g}_0, ..., \hat{g}_k\}$, where $\hat{g}_k \in \mathbb{R}^{16}$. A shared direct pose module slides

over the set $\hat{G}$ estimating direct pose hypotheses $\hat{P} \coloneqq \{\ \hat{p}_0, ..., \hat{p}_k\}$, where $\hat{p}_k \in \mathbb{R}^9$. The pose output is parameterized by 3 values for translation in $\mathbb{R}^3$ and the first two basis vectors of the rotation matrix in $\mathbb{R}^3$, thus 6 values [139]. A set $\hat{C} \coloneqq \{\ \hat{c}_0, ..., \hat{c}_k\}$ is computed that quantifies the consistency $\hat{c}_k \in \mathbb{R}^1$ between $\hat{g}_k$ and $proj_{3D \to 2D}(G_{3D} \cdot \hat{p}_k)$ for each image location separately.

During inference, given a query image the network predicts $H = \{\hat{O},\ \hat{B},\ \hat{G},\ \hat{P},\ \hat{C}\}$ with constant runtime. Corresponding elements of H with an image location k of maximum class probability $\hat{o}_k$ below the detection threshold are discarded. The resulting subsets are clustered into object instances using the IoU between elements of $\hat{B}$. We define a hyperparameter n for the highest number of consistencies in $\hat{C}$. This parameter is set to 10 for the presented experiments; see the ablation in Table 4.16. Finally, the detected object classes and the mean of the n poses with the highest consistencies per instance are returned with negligible increase in runtime with respect to the number of object instances. Through this procedure, our method can estimate the poses of a large number of object instances in a single test image in real-time ($> 24fps$) on an Nvidia Geforce RTX 3090.

3.5.2 Training Target Sampling

Effective assignment of true image locations for updating network weights during training is an ongoing research problem [56], [88]–[90], [95], [127]. These true image locations are often sampled in the output feature maps of feature pyramids [32], which are a great tool to efficiently encode scale information to the feature space. Anchors [30] are the standard representation for providing bounding box priors to sample true image location based on the IoU with the ground truth during training [28], [30], [56]. For each image location in the multi-scale feature map, 9 differently shaped and sized bounding box priors are sampled. This requires 16 hyperparameters: 5 each for *base sizes* and *strides* and 3 each for *ratios* and *scales* [56]. This results in two convenient traits since anchor locations used for updating the network's weights are chosen based on a threshold-parameter for the IoU with the ground truth:

- Sampling anchors leads to a uniform scale space for the expected bounding boxes. As such, a similar amount of training locations are sampled per object, independent of the object's size in the image space.

- Regression targets are standardised using the respective anchor's center, width and height. This way regression targets have similar statistics for differently sized objects.

Despite these convenient traits, training target sampling can still be improved since anchors require a) choosing 16 hyperparameters depending on the expected object scales in image space and b) generating 9 anchors per feature map location, which results in a large output space that slows down convergence. We overcome these shortcomings by using a regression target standardisation scheme that reflects the object's geometry and scale.

3.5.2.1 True Location Sampling

Object masks are used for true training location sampling as in [53]. However, instead of predicting object masks and correspondences from a single feature map resolution, our work adopts the divide-and-conquer strategy of feature pyramids to make predictions from multiple feature map resolutions. To overcome the necessity of requiring hyperparameters [30], [90] for choosing the best suited feature map resolution for locating an object, we propose a geometry-based approach to assign true training locations. We supplement true location sampling with a scalar shape prior:

$$\delta_o = max||(m_i - m_j)||_2 \ \forall \ m_i, m_j \in M, i \neq j \tag{3.5}$$

where M is the set of object model vertices. Since the spatial downsampling of the input image through the backbone follows an exponential function, it is intuitive to explicitly choose pyramid levels using a logarithmic function. As such, we choose the respective feature pyramid level with:

$$level = f + log_d(\delta_o / t_z), \tag{3.6}$$

where f depends on the number of pyramid levels used, t_z corresponds to the object's distance from the camera and d is the only remaining hyperparameter. Since we use PFPN presented in Section 3.4.2 with three pyramid levels for prediction making, this requires choosing 6 hyperparameters when using the FCOS sampling scheme and 12 when using anchors. An additional advantage is that δ_o better reflects the object shape in all three spatial dimensions and thus also the visible object surface in the image space compared to using the bounding boxes for the assignment of true training locations. As a consequence, elongated objects are tendentiously sampled in higher resolved feature pyramid levels than boxy shaped objects. Despite needing fewer hyperparameters, we retain a similar amount of true locations used for training. Classifying true image locations (L_{cls}) is supervised using the focal loss [56].

3.5.2.2 Geometric Correspondence Standardisation

Instead of standardising the projected object correspondences G using anchor priors or with a scalar value agnostic to object shape [90], we directly incorporate δ_o to scale regression targets of different objects to a similar magnitude:

$$y_G = (c - G)/\delta_o, \tag{3.7}$$

where c is the center of the respective feature map location, G are the image locations of the geometric correspondences and y_G are the standardised regression targets. As such, regression targets are encoded similarly as with anchors (with similar σ for G of all objects independent of their scale or shape eccentricity). Thus, the computed error is independent of the object's scale in the image space and the training process is not biased for larger objects. Our approach needs no hyperparameters for standardisation and since 9 times fewer network output parameters per feature map location are required as compared to anchors, convergence is improved.

3.5.2.3 Imbalance Problem of Target Locations

Choosing training target locations based on the object mask leads to a training process that is biased towards objects with a larger projected image surface. For classification this is commonly circumvented using the focal loss [30]. Using anchors as location priors alleviates the issue since anchors are sampled uniformly over the expected object scale space. We define a concurrent solution by normalising over the number of true training locations l and accumulating the gradient afterwards. The regression loss is:

$$L_{reg}(\hat{y},y) = \frac{1}{a} \cdot \sum_{i=0}^{a} \frac{1}{l_i} \cdot \sum_{j=0}^{l_i} huber(\hat{y}_j, y_j), \tag{3.8}$$

where *huber* is the loss used in RetinaNet [56], [136] and y and $\hat{y}$ are ground truth and the estimate, respectively. This procedure requires no additional trainable parameters and leads to minor computational overhead during training time, and to none during test time, despite improving multi-object handling.

3.5.2.4 Direct Pose Regression

The direct pose is regressed using the output $\hat{y}$ of the module, estimating intermediate geometric correspondences as done in [37], [41], [64], [110]. The 6D pose is parameterised as $P \in SE(3)$, with $t \in \mathbb{R}^3$ being the 3D translation vector and $R \in \mathbb{R}^6$ the first two base vectors of the $SO(3)$ rotation matrix as done in [37], [41], [140].

Prior methods perform pose estimation on zoomed crops of the detected objects of interest. Using the object rotation in the camera coordinate system, i.e. the allocentric rotation [140], leads to mapping different object views to the same rotation. To alleviate that problem, these approaches use the rotation of the camera in the object coordinate frame, i.e. the egocentric rotation.

In contrast, we learn to predict geometric correspondences directly in the image space. These correspondences are destandardised with the inversion of Equation (3.7) and fed to the direct pose estimation module. As such, our approach correlates object rotation with its image location. This means that we are able to directly regress the allocentric rotation since we require no zooming or cropping. Additionally, we can directly regress the 3D translation without requiring a scale-invariant translation representation as used in [37], [39], [41]. The network training is supervised using the image locations sampled with Equation (3.6).

3.5.3 Symmetry-aware Loss

Objects exhibiting discrete or continuous symmetries, i.e. similar views that correspond to different ground truth poses P, are detrimental to the convergence of the network training [40], [129], [132]. We adopt the transformer loss of [40] since symmetries are efficiently handled during loss computation and require no additional trainable weights. We define our keypoint estimation loss for supervising the training of the geometric

correspondence learning with:

$$L_{key} = \min_{s \in S_i} L_{reg}(\hat{y}, hy), \tag{3.9}$$

where S_i is a set of symmetry transformations that depend on the visual ambiguities of the object. We observed that separately choosing hypotheses with L_{key}, and substituting the direct pose losses for L_{rot} and L_{tra} with Equations (3.9) introduces ambiguities since the 6D pose is directly derived from the estimated intermediate geometric correspondences. To alleviate this issue we define an indicator function $\mathbb{I}$, indicating the symmetry that minimizes L_{key}. As such, we supervise the direct pose regression with:

$$L_{rot/tra} = L_{reg}(\hat{y}, \mathbb{I}(S)y). \tag{3.10}$$

Since only one set of $\hat{G}$ is predicted per image location and Equations (3.9) and (3.10) sufficiently account for object symmetries, L_{proj} and L_{cons} can directly be computed with L_{key}. The projection and the consistency loss are thus defined as:

$$L_{proj} = L_{reg}(G_{3D}\hat{P}, G), \tag{3.11}$$

$$L_{cons} = L_{reg}(G_{3D}\hat{P}, \hat{G}). \tag{3.12}$$

The overall loss is presented in Equation 3.1 is thus extended to:

$$L = \alpha \cdot L_O + \beta \cdot L_{box} + \gamma \cdot L_{key} + \delta \cdot L_{rot} + \epsilon \cdot L_{tra} + \zeta \cdot L_{proj} + \eta \cdot L_{cons}, \tag{3.13}$$

where α, β, γ, δ, ϵ, ζ and η are loss weights. The bounding box estimation, L_{box}, is supervised using Equation (3.8).

3.5.4 Multi-instance Handling

Commonly, multiple instances of the same object in a single image are handled before correspondence estimation, by non-maximum suppression of the detection stage [17], [39], [40], [52] or by clustering correspondences afterwards [58]. The first family of methods individually processes each instance's image crop to estimate the 6D pose. The second family of methods is more advantageous, however, because the network is shared over all objects of interest. Unfortunately, since [58] predicts dense geometric correspondences, the method has a high runtime. This is due to the clustering of correspondences to object instances using [60], which is computationally demanding.

In our work, a first filtering stage of H is performed by discarding the non-maximally scoring object classes for each image location k. Subsequently, image locations with a detection score below the detection threshold are pruned. The remaining hypotheses correspond to detected objects. The 2D bounding boxes, $\hat{B}$, are used to cluster object instances based on the respective IoU between the outputs of different image locations. Ultimately, using the computed consistency $\hat{C}$, the pose is averaged over the n hypotheses of $\hat{P}$ with the highest consistency for each object instance.

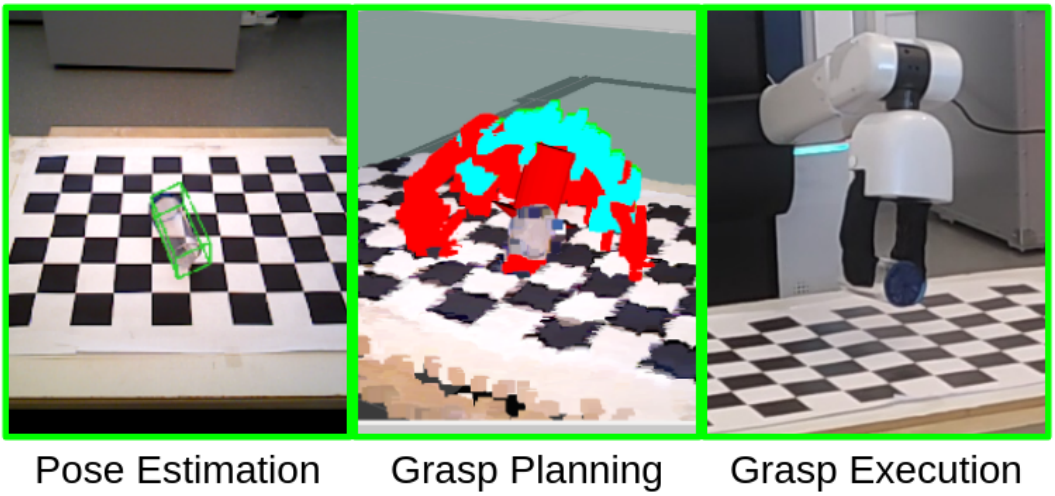

Figure 3.17: **Grasp Pipeline Overview** Poses are estimated using our presented methods. Grasps are planned using pre-defined grasp points. The closest collision-free trajectory is executed to grasp the object of interest.

3.6 Object Grasping

Performance of pose estimation methods often deteriorates when deployed on real-world robots [1], [141], [142]. This section presents the setup used for evaluating the suitability of the poses, estimated by the presented methods for real-world object grasping. An overview of the grasping pipeline is given in Figure 3.17. Poses are estimated, grasp points are sampled and filtered for their grasping success, and the grasp is executed.

Grasping experiments are conducted with the Toyota Human Support Robot (HSR) [143], [144] to demonstrate the transition to real-world applications using only synthetic data for training. The proposed approaches are trained using the meshes of the objects of interest. All known objects are detected and poses are estimated in the camera frame of the RGB-D camera in its head, the Xtion PRO LIVE [7]. The poses of the detected object instances of interest are used to transform manually pre-defined grasp poses to the robot coordinate frame. Pre-defined grasp poses are annotated per object using a model of the gripper, as shown in the left image of Figure 3.18. These annotated grasp poses are then transformed to the robot base frame, using the estimated pose. For each potential grasp pose trajectories for execution are computed, and grasp success is evaluated. The right image of Figure 3.18 displays computed gripper positions. Point clouds coloured in red indicate infeasible gripper configurations due to collisions, blue indicates valid gripper configurations and green indicates the chosen grasp. The grasp with the shortest gripper trajectory is executed.

[7] http://xtionprolive.com/asus-3d-depth-camera

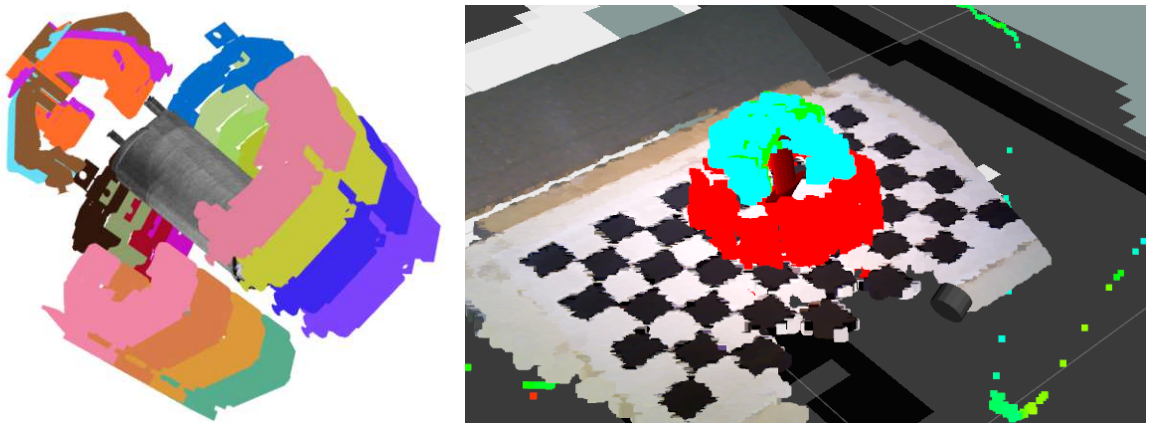

Figure 3.18: **Grasp Annotation and Success Checking** Left image: 20 possible grasp configurations are annotated. The randomly coloured grippers are scaled to 50% for visibility. Right image: Sampled grasp positions based on the object pose, invalid grasps in red, possible grasps in blue and green, where green also presents the executed grasp.

Chapter 4

Experiments

This chapter presents experimental setups, quantitative and qualitative evaluations, and robotic grasping experiments of the methods and strategies presented in Chapter 3. At first results for object localisation when using synthetic augmented depth data for training are presented. Subsequently, object localisation results when using our proposed formulations with RGB are demonstrated. After providing robotic grasping experiments the section concludes with a direct comparison of our presented pose estimation methods.

4.1 Object Localisation from Depth Data under Domain Shift

This section analyses the creation process of synthetic depth data presented in Section 3.1.1. The influence of the augmentations for domain adaptation presented in Section 3.1.2 is evaluated. Lastly, pose estimation from this data is compared to the state of the art.

4.1.1 Object Detection from Synthetic Depth Data

Synthetic training data created using the method presented in Section 3.1.1 and the domain adaptation of Section 3.1.2 is denoted as Synthetic Depth Data (SyDD). Results for object detection are presented on the Linemod dataset [112]. A standard and well-known baseline for object recognition and pose estimation in RGB-D. The test set of the LineMOD dataset consists of 15 test sets, one for each dataset object, with approximately 1200 captured images per scene. Every set has different object instances visible, although only the object in the center of the image is annotated with a bounding box, class and pose. Since different object instances without annotation are visible in the test images, only the annotated object is considered for calculating the detection recall. In all experiments we report the percentage of the amount of annotated objects per test set that is correctly detected and classified. An estimate counts as correct if the estimated bounding box has at least an IoU of 0.5 with the ground truth.

Experimental Setup All tests are conducted with the following preprocessing and

Table 4.1: **Detection Recall on Linemod** Recall of FasterRCNN [30] trained on real-world and on synthetic data. Numbers are the percent of correctly estimated bounding boxes.

Classes	real	*SyDD*
ape	53.56	**76.86**
can	**97.24**	94.15
cat	41.31	**82.70**
driller	**96.21**	92.76
duck	89.39	**93.70**
eggbox	64.8	**81.01**
glue	**81.72**	70.08
holepuncher	**89.89**	77.69
overall	76.77	**83.62**

network configuration. All real-world and synthetic images are converted to three channel RGB images and coloured based on the normal direction using the approach of Nakagawa *et al.* [120]. Image regions with missing depth values are inpainted using [145] and depth cuts are applied to image regions up to 0.2 meters and regions further away than 1.8 meters.

We use FasterRCNN [30] with Resnet101 [104] backbone, pretrained on ImageNet [126], with the standard optimizer and loss functions. The learning rate starts at $1 \cdot 10^{-2}$ and decays to $1 \cdot 10^{-4}$. We train for 180,000 iterations using a batch size of one and a weight decay of $1 \cdot 10^{-4}$.

Performance on Real-world Data We compare training the object detector on real-world images and on our augmented synthetic depth data. Results on Linemod are provided using the re-annotated test set of benchvise by [135] for training and using the same amount of images created using SyDD. Table 4.1 compares the detection recall for the respective per-instance annotated sets of Linemod.

The average recall of the detector trained on our synthetic dataset outperforms the recogniser trained on real-world data. The performance margin results from the higher variability in the synthetic dataset. The biggest differences in detection recall are visible for the objects ape, cat and eggbox. This is caused by the scene setup of the real-world depth scans used for training. The ape is placed in different poses in the scene and is either not occluded or completely occluded in most of the images. In comparison, using randomly sampled scene setups as done in our rendering creates samples with diverse occlusion ratios. The cat is viewed with similar rotation respective to the camera in many of the scans used for training on real-world data, which again results low variability of the pose space, leading to an biased estimator. The training samples of eggbox also exhibit a low variability of viewpoints. Furthermore, occlusion is mostly caused by the same object. The randomised augmentation covers a wider range of variations influencing the image creation process, as well as the placement of objects in the virtual scene. This increases the variation of occlusions and views in comparison to the real-world images of [135]. Figure 4.1 shows that while using randomised

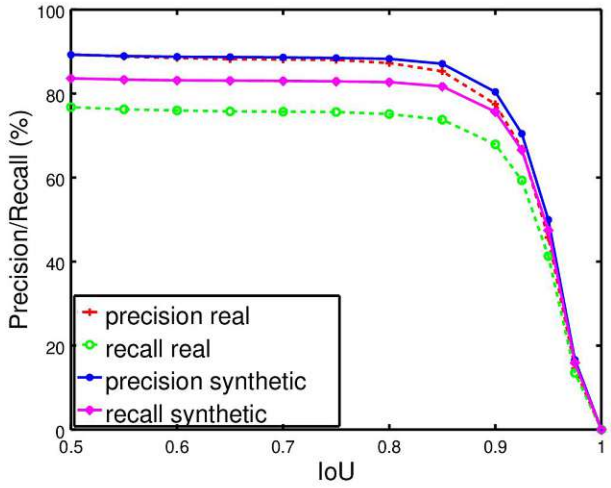

Figure 4.1: **Detection Rates with Respect to IoU** Recall and precision curve comparison of real-world and synthetic data using different IoU scores on Linemod.

synthetic data improves detection recall, compared to using available real-world data for training, the precision does not decrease. Thus our presented synthetic data generation and augmentation method even leads to slightly improved robustness with respect to false-positive detections and as such presents effective domain adaptation.

Influence of the Background Information The importance of the background information provided in the training data is evaluated by training FasterRCNN, with data generated using the different background sets presented in Section 3.1.1. Each training set consists of 10,000 synthetic images. Table 4.2 shows a comparison of the detection recall. The results indicate that providing richer background information during synthetic data generation improves the detection recall. Results also indicate that it is not necessary to use the same background objects for training as observable during deployment. Our findings show that domain specific background objects are sufficient for detectors to yield similar performance to detectors trained on real-world, hand-annotated images. When providing richer background information object detection rates improve for all objects but bowl. The recall for this object decreases when using realistic background sampling.

Evaluation of the Augmentation Method The performance influence of the separate parts of the augmentation for domain adaptation presented in Section 3.1.2 is evaluated using 10,000 generated images. The augmentation methods are:

- *synth*: non augmented synthetic depth data.
- *perlin*: augmenting the synthetic images using only Perlin noise [86] with the parameters from [45], after removing occluded image regions based on the imaging geometry of the Microsoft Kinect V1, using the randomised visibility masks.
- *auth*: randomised realistic sensor model, where the difference to our proposed

Table 4.2: **Background Information Comparison** Detection recall of FasterRCNN trained on SyDD, with different backgrounds in the virtual scenes. Numbers are the percent of correctly estimated bounding boxes.

Classes	simple	limited	realistic
ape	57.79	59.79	**79.69**
benchvise	65.16	64.09	**96.05**
bowl	91.24	**93.03**	85.81
camera	66.61	74.94	**94.17**
can	57.02	79.10	**91.97**
cat	58.95	80.75	**97.71**
cup	76.13	83.06	**88.06**
driller	65.99	84.76	**96.72**
duck	82.30	81.58	**95.37**
eggbox	83.32	92.42	**93.77**
glue	65.16	79.98	**82.79**
holepuncher	74.54	85.53	**92.97**
iron	42.19	64.58	**89.84**
lamp	50.77	65.69	**96.09**
phone	71.36	68.22	**93.00**
overall	63.03	72.29	**85.88**

method SyDD is that the depth noise is added before quantising to eleven bit range.

- *SyDD*: Full augmentation presented in Section 3.1.2.

The results presented in Table 4.3 indicate that strong average detection recall is already achieved by adding Perlin noise. However, even better performance is achieved

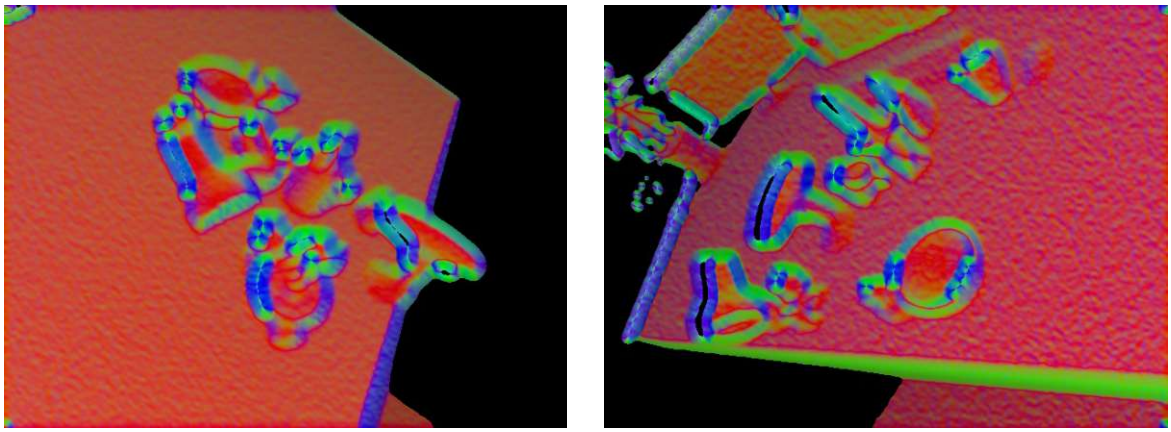

Figure 4.2: **Example Images for Background Complexity** Images displaying the synthetic training dataset, with simple background information (left) and limited background information (right).

Table 4.3: **Augmentation Method Comparison** Detection recall of Faster-RCNN trained using different augmentation methods. Numbers in percent.

Classes	*synth.*	*perlin*	*auth.*	*SyDD*
ape	59.06	71.76	67.96	**79.69**
benchvise	71.99	93.90	91.85	**96.05**
bowl	**91.64**	91.48	91.08	85.81
camera	56.54	84.60	89.84	**94.17**
can	53.51	94.15	**95.32**	91.97
cat	89.91	97.20	91.18	**97.71**
cup	73.23	84.19	81.21	**88.06**
driller	89.31	95.62	95.71	**96.72**
duck	62.04	93.78	89.63	**95.37**
eggbox	45.49	81.80	90.90	**93.77**
glue	44.02	**85.08**	83.44	82.79
holepuncher	59.18	**93.37**	81.33	92.97
iron	39.15	89.41	78.83	**89.84**
lamp	75.31	93.48	**97.96**	96.09
phone	45.78	91.31	90.27	**93.00**
overall	59.76	83.82	82.28	**85.88**

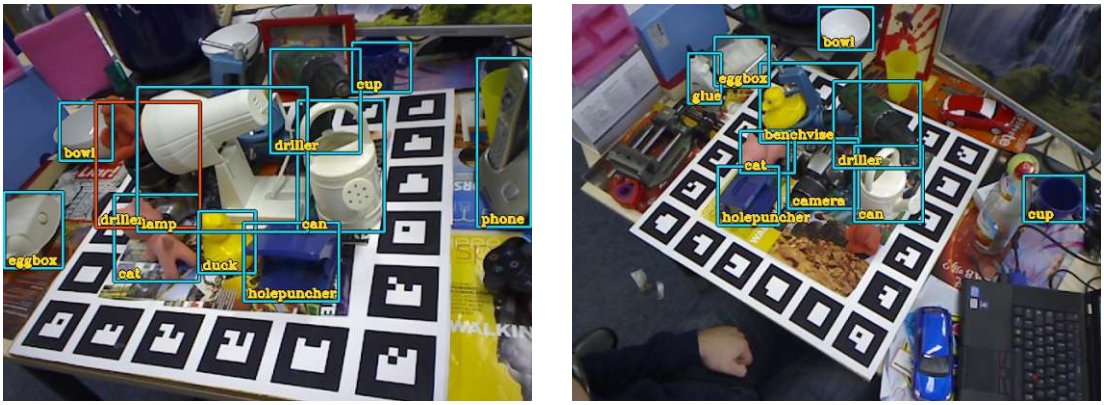

Figure 4.3: **False Negative Detections** Incorrectly classified driller, indicated with a red bounding box, left image. Missed detection of duck, right image.

by combining Perlin noise with the proposed randomised camera model. We conclude that augmenting images with Perlin noise can effectively close the domain gap, however, it lacks the artefacts of the physical depth data capturing process. Re-sampling the augmented images to the Microsoft Kinect V1 depth resolution decreases detection recall as compared to Perlin and SyDD.

Open Problems Qualitative results of object detection using the training images from SyDD and test images from Linemod are presented in Figure 4.3 and Figure 4.4.

The RGB-images are only used for visualisation. The top of Figure 4.3 shows ape placed on top of the camera, which is incorrectly classified as driller. A similar error is visible in the bottom of Figure 4.3. The benchvise is correctly classified but the duck is

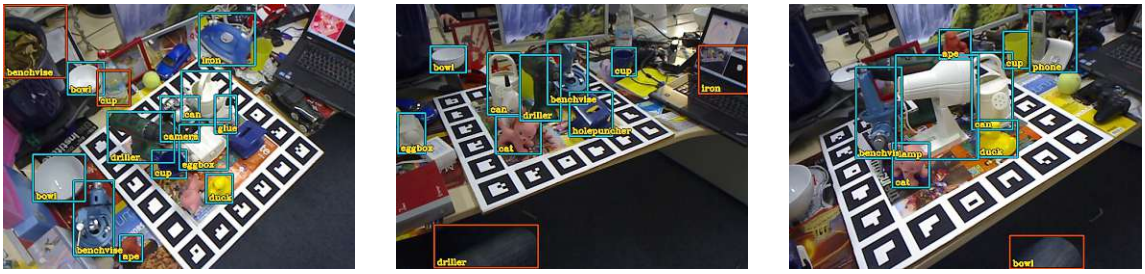

Figure 4.4: **False Positive Detections** Detection result with incorrect detections on boundary regions of the image or fabric.

not detected. This error occurs because objects in the virtual scene are enclosed by a convex hull. Consequently, stacked objects are not sampled for data rendering. A convex hull is used to represent the collision shape of objects to minimise errors when performing the physics simulation.

Figure 4.4 shows detection results with objects incorrectly detected on fabric, near the image boundary. The left image shows an incorrect detection of benchvise in the upper left corner of the image and one of cup nearby. The middle and the right image show incorrect detections of driller and bowl on fabric featuring similar curvatures. Another incorrectly detected instance of iron is visible on the right edge of the middle image. These detections result from training on partially visible objects that are cut off by the image boundary. Another common error is the detection of objects on smoothly curved fabric surfaces as can be seen in the bottom parts of the middle and the right image in Figure 4.4. This error is a combination of training on boundary regions and missing background information during the rendering process.

4.1.2 Domain Adaptation for Object Pose Estimation using SyDPose

The approach presented in Section 3.3 is evaluated for object pose estimation on the Linemod [112] and the Occlusion dataset [135]. We provide results comparing to the deep learning-based state of the art and against the point-pair-feature approach of [26]. Additionally, ablations to evaluate the influence of our proposed orthogonality constraint and our multi-object formulation are presented.

Experimental Setup For testing we use only the depth images of the datasets that are captured using a Microsoft Kinect V1. Images are converted to three channel RGB images, coloured based on the depth gradient using the approach of Nakagawa *et al.* [120]. Image regions with missing depth values are inpainted using [145] and depth cuts are applied to image regions farther than two meters. Our networks are trained using the Adam optimiser [128] with adaptive learning rate initialised at $1 \cdot 10^{-5}$. In order to prevent the network from overfitting to the limited amount of training data, we apply extensive geometric data augmentation of the training images. Every input image is randomly augmented online using a superposition of translation and scaling up to 20 percent. All experiments are conducted on a Nvidia Geforce GTX 1080.

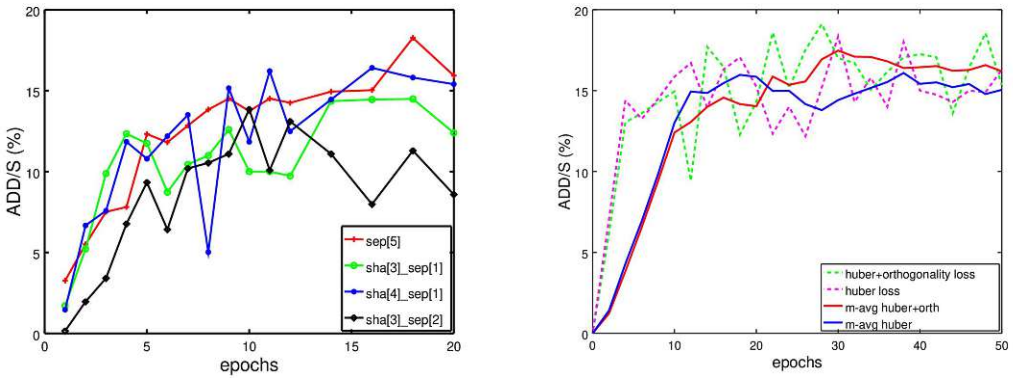

Figure 4.5: **Multi-object Keypoint Branch** Right: Comparison of different convolution layer setups of our keypoint regression branch. Left: Comparison of our architecture with and without our orthogonality favouring loss for 3D bounding box estimation.

Evaluation Metrics Evaluation is performed using the average distance between points, comparing the estimated and the ground truth pose (*ADD/S*) [112], defined in Equation 2.6 and the *F1*-score based on the *ADD/S* metric comparing against the state of the art [101]. Using the *ADD/S* metric, the pose is considered correct if the average distance difference of corresponding transformed points, using the estimated and the ground truth pose, is below a certain threshold multiplied with the object model diameter. We use the commonly used threshold of 10% of the object diameter. For symmetric objects, eggbox and glue, the closest point distance is used. The *F1*-score represents the harmonic mean between recall and precision. Thus, it considers the false positive rate when calculating the amount of correctly predicted poses. Using the *F1*-score, predicted poses are considered correct if the *ADD/S* score is below the threshold set to 0.15, as in [101].

Pose Estimation Branch Architecture Training one network for simultaneous object detection, classification and pose estimation on multiple objects leads to a decrease in overall pose estimation performance, compared to training one model per object of interest. We provide studies regarding our architectural choices to show that we are able to circumvent losing the major part of performance, by simply training the last convolution layer separately for each object of interest.

The left plot of Figure 4.5 presents a self-comparison for different network configurations using the Linemod dataset. Using four shared convolutional layers and one separate layer per class (*sha[4]+sep[1]*) leads to similar pose estimation performance as using separately trained networks and higher pose estimation performance than using three shared and two separate layers (*sha[3]+sep[2]*). We also provide results for using three shared and one separate layer (*sha[3]+sep[1]*), showing that reducing the amount of convolution layers reduces performance. Using three shared and two separate layers (*sha[3]+sep[2]*) also reduces performance due to the different convergence speeds of the network branches.

A clear advantage of using shared feature extraction over all objects of interest for

Table 4.4: **Parameters and Runtimes** Amount of trainable parameters and training and inference time of different keypoint regression branch architectures.

Design	*Parameters*	*Train [ms]*	*Test [ms]*
sep[5]	40,333,727	202	56
sha[3]+sep[1]	39,840,457	199	55
sha[4]+sep[1]	40,430,537	208	56
sha[3]+sep[2]	48,691,657	282	79

pose estimation is that less than one sixth of the total training iterations is needed to achieve similar performance compared to training separate networks. Table 4.4 shows parameters, training (*train*) and inference (*test*) time of our end-to-end models using different pose estimation head designs. Architectures that are combinations of separate and shared layers lead only to marginal differences regarding time consumption per training iteration and during inference. While replacing the last shared convolutional layer in the control point regression branch with separate layers leads to increased results, comparable to training separate networks per object, the computational overhead for this configuration is still reasonable.

Performance Improvement via Orthogonality Using the proposed orthogonality loss, presented in Equation 3.3, improves the trained model's performance when nearing convergence. We present a comparison between regressing the keypoints of the 3D bounding box using only Huber loss, see Equation 3.1, and regressing the keypoints incorporating our orthogonality favouring loss with the parameter δ of Equation 3.4 set to 0.8. We also calculate the moving averages for easier comparison, using a kernel size of five. The right plot of Figure 4.5 presents the percentage of correct pose estimates using the *ADD/S* metric plotted against epochs.

Comparison to the State of the Art Deep learning architectures for 6-DoF pose estimation from synthetic data using only depth images exist, however, it is difficult to evaluate against these since the code is not publicly available [6], [146]. Consequently, we present comparison against fully-synthetic learning-based pipelines for pose estimation using RGB and against the point-pair-feature (PPF) method of [26]. Figure 4.6 shows regressed 3D bounding boxes for scenes of the Occlusion dataset.

Recent deep learning approaches achieve strong performance on the Linemod dataset which contains approximately 1,200 test images for each of the 13 dataset objects. Each object is placed in a heavily cluttered scene and annotated with a bounding box, class and 6D pose. However, many approaches reach the reported performance using real-world RGB data for training to overcome the domain shift [47], [53], [54], [59]. Consequently, these are not well suited for practical scenarios where object meshes are often available or can be generated at low cost, while capturing and annotating real-world images requires a significant effort. Table 4.5 compares our method to the fully-synthetic approaches of Kehl *et al.* [62] and Sundermeyer *et al.* [23] using the *ADD/S* metric. The numbers for [62] are taken from [59]. Both approaches train and test on RGB data only, while ours uses depth data only. As such, we only require

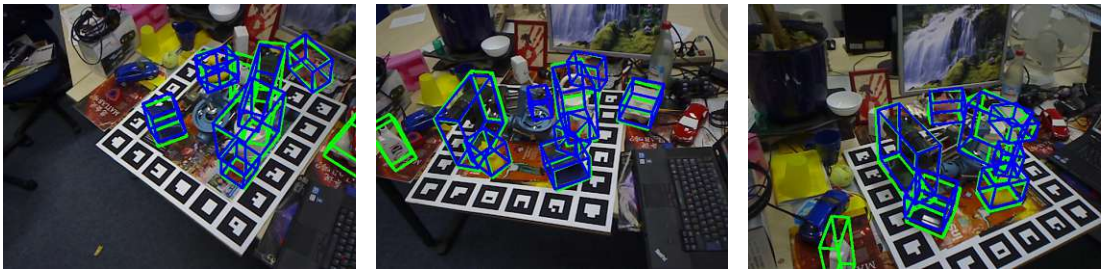

Figure 4.6: **Multi-object Pose Estimation** Visualisation of the 3D box estimation on images of the Occlusion dataset [135], yellow represents the ground truth and blue our estimated boxes. RGB is only used for visualization.

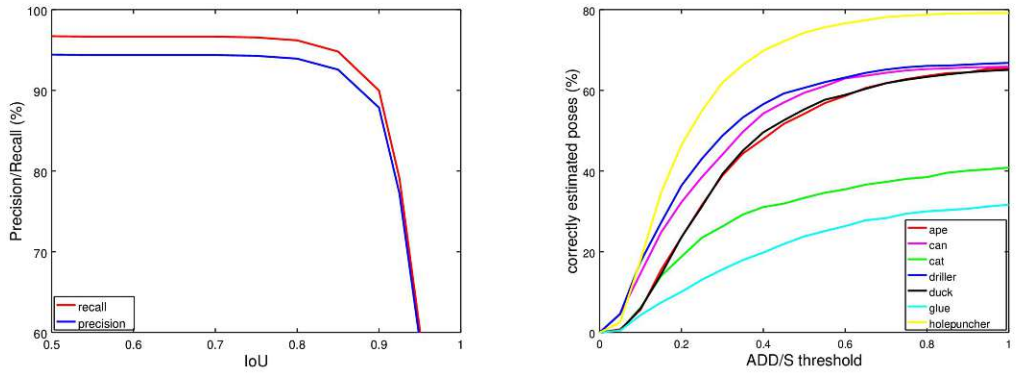

Figure 4.7: **Object Detection and Multi-Object Pose Estimation** Left: Detection recall and precision on Linemod. Right: Percentage of correct poses on the Occlusion dataset [135] for different *ADD/S* thresholds.

geometry priors, although the methods we compare against require texture information. We also present the results of the PPF method of Drost *et al.* [26], showing their superiority to CNN approaches when disregarding false positives. PPF methods [26], [27] are the top-performing pose estimation methods [55]. However, when incorporating the false positive ratio in the evaluation as done by [101], we achieve results that outperform the PPF method of [26]. The left plot of Figure 4.7 shows the recall and precision for detection with respect to different IoU thresholds on the Linemod dataset. Given that we only employ one model we do this at approximately 20 fps, which is faster than the PPF methods that report runtimes of multiple seconds. Nevertheless, we have to mention that our method is not on par with the most recent advancement of PPF [27], since this method has strong refinement and verification steps.

The Occlusion dataset [135] is a subset of Linemod, re-annotated to provide annotations for all Linemod objects present in the image sequence of the benchvise. We follow the standard procedure excluding the benchvise from evaluation.

Table 4.6 reports the mAP for object detection, presented in Equation 2.5, in comparison to state-of-the-art learning-based pose estimation methods. We achieve a mAP of 0.517 when using an IoU threshold of 0.5 over all classes. The right plot

Table 4.5: **Comparison to the State of the Art with SydPose** Comparison to synthetically trained pose estimation pipelines, on Linemod [112]. Reported is the percentage of correctly estimated poses using the respective metric.

Metric	ADD/S				F1	
Type	CNN			PPF	PPF	CNN
Method	SSD-6D[62]	AAE[23]	Ours	Drost[26]	Drost[26]	Ours
Ape	0.00	3.96	**16.35**	**86.5**	**62.8**	43.8
Benchvise	0.18	20.92	**35.17**	**70.7**	23.7	**63.4**
Cam	0.41	**30.47**	16.67	**78.6**	**51.3**	44.0
Can	1.35	**35.87**	27.27	**80.2**	51.0	**59.1**
Cat	0.51	17.9	**34.19**	**85.4**	56.6	**65.8**
Driller	2.58	23.99	**30.71**	**87.3**	59.7	**65.4**
Duck	0.00	4.86	**9.32**	**46.0**	31.3	**35.5**
Eggbox	8.90	**81.01**	52.76	**97.0**	**82.6**	74.8
Glue	0.00	45.49	**51.66**	**57.2**	38.2	**77.7**
Holepuncher	0.30	17.6	**29.21**	**77.4**	50.0	**58.6**
Iron	8.86	32.03	**34.25**	**84.9**	40.5	**69.0**
Lamp	8.20	**60.47**	37.50	**93.3**	**77.6**	68.4
Phone	0.18	**33.79**	17.24	**80.7**	**47.1**	42.9
Average	2.42	28.65	**30.21**	**78.9**	51.7	**59.1**

of Figure 4.7 shows pose estimation results for heavily occluded and cluttered scenes. We produce good pose estimates on single-shot multi-object scenarios with high object occlusion.

4.2 Generalising Object Pose Estimation within the RGB Domain using PyraPose

This section presents experiments to evaluate PyraPose, presented in Section 3.4. We use three different datasets to compare to state of the art, show the advantage of using synthetic data to generalise to new domains, perform ablations of our architectural design and conduct grasping experiments to demonstrate the suitability for real-world robotic applications.

Datasets Single-object pose estimation of objects with little texture is evaluated on the Linemod [112] dataset, which consists of approximately 1200 images of cluttered scenes per object. Linemod comes with 15 textured meshes of the objects of interest and corresponding pose annotations on the frame level. We use the same subset of objects as [23], [49], [52], [87]. Multi-object experiments for objects with little texture are performed on Occlusion [135], which is the test set of Linemod's benchvise object featuring annotations for all dataset objects that appear in the images.

Table 4.6: **Detection on Occlusion using SydPose** Detection performance on the Occlusion dataset [135].

method	*mAP*
Brachmann *et al.* [102]	0.51
Kehl *et al.* [62]	0.38
Tekin *et al.* [59]	0.48
ours	**0.52**

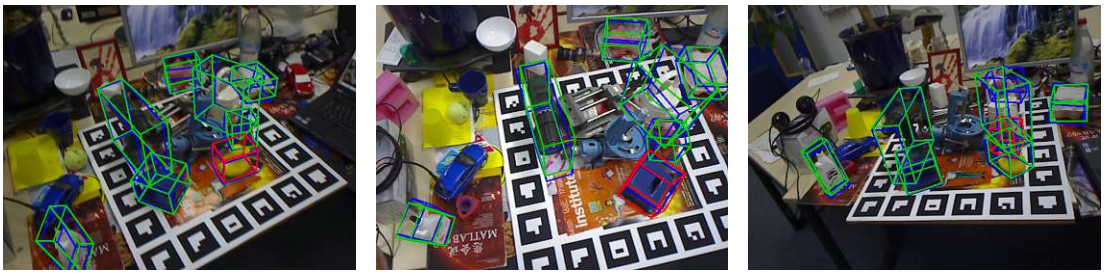

Figure 4.8: **Visualisation of PyraPose on Occlusion** Examples of correctly estimated (green), incorrectly estimated (red) and ground truth (blue) poses on images from the Occlusion dataset [135].

In addition to single- and multi-object experiments, we also present experiments on Homebrewed's [20] second sequence to show that using synthetic data for training is better for generalising to new settings. The specific sequence of Homebrewed features three of Linemod's objects in a different setting, which includes scene setup, illumination and camera intrinsics. We show that PyraPose trained on synthetic images for the Linemod setting generalises well to Homebrewed. Lastly, in order to bridge from training on synthetic data to performing tasks in the physical world, we present grasping experiments on YCB-video objects [48] following the layout of the GRASPA 1.0 benchmark [147].

Training Data Training images are rendered from complete scenes of textured 3D object models with realistic material and lighting interactions. The geometric configuration of objects and cameras is generated using physics simulation. Synthesising images is performed by physically-based rendering [68], [71]. For Linemod and Occlusion, we use the same training images. The synthetic Linemod, Homebrewed and YCB-video datasets contain $50k$ samples taken from the BOP challenge [55].

Implementation Details Our networks are trained using the Adam optimiser with a learning rate initialized with $1 \cdot 10^{-5}$ and a batch size of 8. The learning rate is decreased by one magnitude if the loss does not decrease for two epochs. Results are provided after 200 epochs of training on $50k$ synthetic RGB images on an Nvidia Geforce Titan V.

The domain generalisation of our models is encouraged by applying different strategies described in Section 3.1.4. Synthetic training images are randomly augmented in random

order to minimise the domain gap. We apply randomised image augmentations similar to [23]. However, we found slightly different ranges of the parameters to be beneficial, presented in Section 3.1.4. We also additionally vary the contrast, which is not performed in [23]. The weights of the first 14 convolutions are frozen similar to [49], [106]. Batch normalization layers are set to non-trainable and parameters are frozen.

Evaluation Setting For object detection, we consider the existence of an object as true if the detection score of at least one anchor is above 0.5. For these true anchors, the estimated 2D-3D correspondences are considered as hypotheses. For pose estimation, the ADD-(S) recall is used, as presented in Section 2.5. This measures the average distance difference between the corresponding transformed object points from the estimated and the ground truth pose. For the objects eggbox and glue in Linemod the symmetric version (ADDS) is used, in which the closest point distance is used for calculation. For all experiments, we report the percent of correctly estimated poses when the ADD(-S)-score is less than the standard threshold of 10% of the model diameter [112].

4.2.1 Multi-object Pose Estimation trained on Synthetic RGB Data

We provide a quantitative comparison to recent learning-based object pose estimation methods of [23], [129] and [49] that use only synthetic data for training, and therefore, are the most similar to our approach, as well as against two winning methods of the 2019 BOP challenge [148], i.e., [39] and [40].

Single-object scenario Table 4.7 presents results on Linemod [112]. Our method significantly outperforms [23], [49], [129][1], which are methods training on synthetic data with domain randomisation applied for domain adaptation. We achieve the best results on average, and for all individual objects except from the ape. The poor results for the ape might be caused by the suboptimal anchor parameters for the small object scale. The last two columns show results of [87] and [52]. Both methods use the test set of Linemod without pose annotation for unsupervised and self-supervised domain adaptation for training, respectively. We achieve better performance in comparison to both, even though no real images are used to train our network.

Our method provides strong estimates over all object scales due to the multi-scale nature when using feature pyramids. Especially for robotic scenarios, our approach is beneficial because we train only one network for all objects, while all competing methods [23], [49], [52], [87], [129] train one network per object to gain better performance. Our approach keeps the memory load low, provides easier usage and still outperforms the state of the art.

Multi-object scenario Table 4.8 reports the ADD(-S)-recall for multi-object detection and pose estimation in comparison to the state-of-the-art learning-based pose estimation approaches of [39], [40], [49]. These three methods employ encoder-decoder networks for pose hypotheses creation and we achieve significantly better performance

[1]Results of [49] provided by the authors; these differ from their presented results because ADD-precision is presented while we present ADD-recall

Table 4.7: **PyraPose on Linemod** ADD/S-recall on Linemod [112] in comparison to synthetically trained methods. Last two columns show results of methods using test images without pose annotation for training. Objects with (*) are evaluated using ADDS.

Training	Synthetic				Real w/o pose	
Method	AAE [23]	MHP [129]	DPOD [49]	PyraPose (ours)	DTPE [87]	Self6D [52]
Ape	4.2	11.9	**35.1**	22.8	19.8	38.9
Benchv.	22.9	66.2	59.4	**78.6**	69.0	75.2
Cam	32.9	22.4	15.5	**56.5**	37.6	36.9
Can	37.0	59.8	48.8	**81.9**	42.3	65.6
Cat	18.7	26.9	28.1	**56.2**	35.4	57.9
Drill	24.8	44.6	59.3	**70.2**	54.7	67.0
Duck	5.9	8.3	25.6	**40.4**	29.4	19.6
Eggbox*	81.0	55.7	51.2	**84.4**	85.2	99.0
Glue*	46.2	54.6	34.6	**82.4**	77.8	94.1
Holep.	18.2	15.5	17.7	**42.6**	36.0	16.2
Iron	35.1	60.8	84.7	**86.4**	63.1	77.9
Lamp	61.2	-	45.0	**62.0**	75.1	68.2
Phone	36.3	34.4	20.9	**59.5**	44.8	50.1
Avg.	32.6	38.8	40.5	**63.4**	51.6	58.9

using only synthetic data for training. Our multi-hypotheses creation scheme in combination with PFPN, presented in Section 3.4.2 leads to robust prediction making under occlusion and effective multi-object pose estimation, as such achieving a relative improvement of ∼35 % over the state of the art.

Runtime A comparison of runtime is provided against other single-shot object pose estimators that are designed for fast inference. A forward pass of our method takes on average 39 ms on an NVIDIA Titan V (∼26 fps), excluding PnP. RANSAC-PnP produces an overhead of approx 1 ms per detected object. In comparison, AAE [23] computes estimates at a rate of 13 fps when using RetinaNet [56] as backbone (used for the results in Table 4.7) and DPOD at 33 fps. The best performing methods [40] and [39] are significantly slower, computing estimates at a rate of ∼0.8 and ∼1 fps, respectively, without refinement[2].

Feature aggregation with PFPN Results provided in Table 4.9 compare PFPN, presented in Section 3.4.2 to FPN [56]. Using 3 pyramid levels for prediction making for PFPN shows a relative improvement of 5% and 3% over FPN on Linemod and Occlusion, respectively. Using 5 levels of features instead of 3 improves the performance of FPN but is detrimental to PFPN, since PFPN is conditioned on low-level features and more levels require prediction making from coarse pyramid levels with high-level

[2]Inference times taken from [148]

Table 4.8: **PyraPose on Occlusion** ADD/S-recall on Occlusion [135] in comparison to methods only using synthetic data for training. Objects with (*) are evaluated using ADDS. Methods indicated with (†) use the training data of [75].

Method	DPOD [49]	CDPN [39]	Pix2Pose† [40]	PyraPose† (ours)
Ape	2.3	**20.0**	11.3	18.5
Can	4.0	15.1	18.5	**46.4**
Cat	1.2	16.4	**17.1**	11.7
Drill	10.5	5.0	34.5	**48.2**
Duck	7.2	22.2	**25.3**	19.4
Eggbox*	4.4	**36.1**	12.0	16.7
Glue*	12.9	27.9	**30.8**	30.7
Holep.	7.5	24.0	12.2	**33.0**
Avg.	6.3	20.8	20.2	**28.1**

Table 4.9: **Evaluation of PFPN** Multi-scale feature aggregation comparison of PFPN, Section 3.4.2, and FPN [56] on Linemod [112] and Occlusion [135] using the ADD-recall.

Feature Aggregation	Linemod[112]	Occlusion[135]
none	46.5	21.3
FPN[32] (3 lvl.)	60.2	27.3
PFPN(ours) (3 lvl.)	**63.4**	**28.1**
FPN[32] (5 lvl.)	**64.1**	**29.6**
PFPN(ours) (5 lvl.)	61.7	28.2

features. Thus, the advantage of PFPN with 3 feature levels is that it effectively bridges the performance gap of 3 to 5-level FPN [32] and also reduces runtime by $\sim 10\%$, compared to 5-level FPN. Using no feature aggregation results in a considerable decrease in performance and supports the hypothesis that feature aggregation is a useful tool for pose estimation.

4.2.2 Domain Generalisation of PyraPose

This section presents an experiment evaluating PyraPose with respect to pose estimation performance in a novel domain and ablations for the augmentation parameter choices and ranges.

Novel domain Pose estimators trained on real-world data tend to overfit to certain characteristics of the data that they are trained on [20]. Training on synthetic data

Table 4.10: **PyraPose in a Novel Domain** ADD-recall on second test sequence of Homebrewed [20] in comparison to the baseline method of [49].

Method	Mesh origin	benchvise	drill	phone	Avg.
DPOD [49]	Homebrewed[20]	52.9	37.8	7.3	32.7
PyraPose (ours)	Homebrewed[20]	**62.9**	22.6	38.5	**41.3**
PyraPose (ours)	Linemod [112]	10.9	**60.0**	**44.4**	38.4

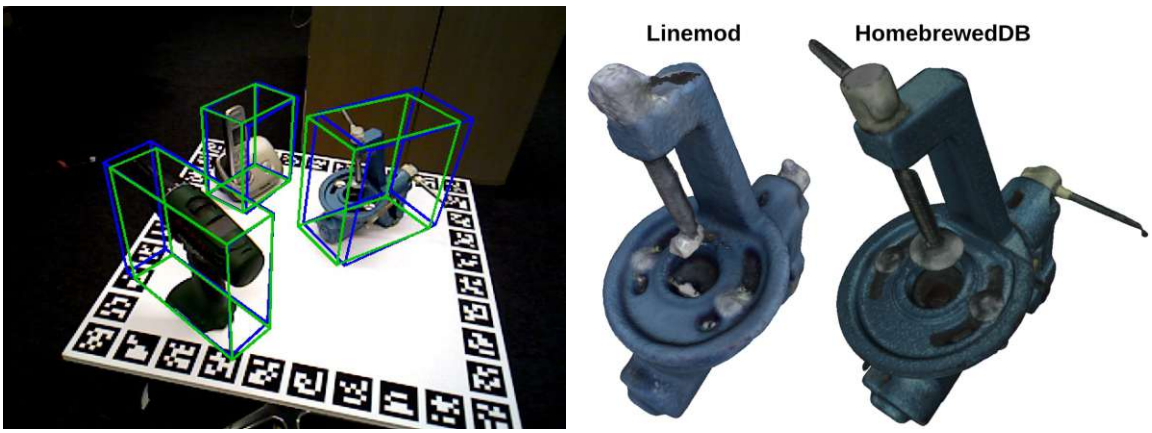

Figure 4.9: **Visualisation of PyraPose on Homebrewed** Left image shows pose estimates from HomebrewedDB [20]. Ground truth in blue and correctly estimates poses in green. Right image shows the mesh of the *Benchvise* from [112] and [20], respectively.

provides the benefit of translating better to novel domains, i.e., new places of deployment. As such, we show results on the second sequence of Homebrewed [20], which has images with pose annotations of Linemod's benchvise, drill and phone. DPOD [49][3] is used by the authors of Homebrewed [20] as a baseline for the dataset. This method is trained on synthetic images using the provided meshes.

The results provided in Table 4.10, using the ADD-recall on the validation set of [75], show that PyraPose outperforms DPOD when trained with synthetic data rendered for the Homebrewed setting (using meshes of all dataset objects for training). Figure 4.9 (left) shows an example of pose estimates. The last row in Table 4.10 gives results of our method when reusing our network trained for Linemod [112]. In order to use our model trained on Linemod, the corresponding object models are manually aligned to compute the ADD-score, since the coordinate frame origin of all three models differ in [112] and [20]. The resulting relative transformations between models are added as a constant offset to Homebrewed's test annotations. For deriving the pose with RANSAC-PnP, see Section 3.2.6, we simply use the intrinsics of Homebrewed, though PyraPose is trained with Linemod intrinsics. Our method effectively transitions to the novel place of deployment, i.e., overcomes the domain shift by training on synthetic

[3]Results are taken from [52]

Table 4.11: **RGB Image Data Augmentation** Augmentation strategy, multi-scale feature aggregation, network design and comparison on Linemod [112] and Occlusion [135] using the ADD-recall.

Strategy	Linemod[112]	Occlusion[135]
none	34.6	14.3
Pix2Pose[40]	53.3	23.1
CDPN[39]	56.7	24.4
PyraPose(ours)	**63.4**	**28.1**

data, which is highly useful in robotics. The results for the benchvise, using Linemod, are low. However, we conjecture that is due to the considerably different reconstruction used to render training data for Linemod, as shown in Figure 4.9 (right).

Image Augmentation in RGB Our proposed augmentation scheme yields the best performance compared to the augmentations applied by [39] and [40], see Table 4.11. We remark that our model trained without colour space augmentations, already results in better performance than the methods of [23] and [49], see Table 4.7 and Table 4.8, respectively. This shows the effectiveness of feature pyramids for generalising to new domains. This however, is at least partially to be attributed to the better training data, see Section 3.1.3. In comparison to the augmentations of [40] and [39], our augmentations lead to 21% and 15% relative improvement, respectively.

Table 4.12 compares augmentation strategies of [23], [40] and [39]. The main difference in comparison to the augmentations used by PyraPose, presented in Section 3.1.4, is that we significantly vary the contrast. We randomly sample gamma, sigmoid, logarithmic and linear contrast with a probability of 0.5 for each, per image. As such, the results in Table 4.11 indicate that varying contrast is important for synthetic to real object pose estimation.

4.2.3 Object Grasping in the Real-world using PyraPose

Grasping experiments are conducted with the Toyota HSR [143], [144] to demonstrate that PyraPose is suited for real-world applications using only synthetic data for training. The printable GRASPA benchmark layout 0 [147] is used, which consists of five objects from YCB-video [48]: mustard bottle (mustard), gelatine box (jell-o), potted meat can (spam), banana and foam brick (foam). The grasping pipeline is described in Section 3.6: Grasp poses are annotated for each object and transformed to the robot base frame using the estimated object pose; based on theses grasp poses, multiple trajectories are calculated and the grasp that is expected to result in the least positioning error is chosen; a grasp is successful if the object is lifted and remains stable in the gripper.

Table 4.13 compares grasping performance using only the initial pose estimate and refining that estimate using the predicted instance segmentation mask and ICP. Good performance is achieved for many objects, in particular, mustard, spam and foam (after refinement). The poor performance for jell-o is explained by the significantly different

Table 4.12: **Augmentation strategies in comparison.**

Augmentation	AAE [23]	Pix2Pose [40]	CDPN [39]
Add	$p = 0.5(0.3)$ $\mathcal{U}(-0.01, 0.01)$	$p = 1.0(1.0)$ $\mathcal{U}(-0.06, 0.06)$	$p = 0.5(0.3)$ $\mathcal{U}(-0.01, 0.01)$
Contrast Normalization	$p = 0.5(0.3)$ $\mathcal{U}(0.4, 2.3)$	$p = 1.0(0.0)$ $\mathcal{U}(0.8, 1.3)$	$p = 0.5(0.3)$ $\mathcal{U}(0.5, 2.2)$
Contrast Normalization	-	$p = 0.5(0.3)$ $\mathcal{U}(0.5, 2.2)$	-
Multiply	$p = 0.5(0.3)$ $\mathcal{U}(0.6, 1.4)$	$p = 1.0(0.5)$ $\mathcal{U}(0.8, 1.3)$	$p = 0.5(0.5)$ $\mathcal{U}(0.6, 1.4)$
Gaussian blur	$p = 1.0$ $\sigma \sim \mathcal{U}(0.0, 1.2)$	$p = 1.0$ $\sigma \sim \mathcal{U}(0.0, 0.5)$	$p = 1.0$ $\sigma \sim \mathcal{U}(0.0, 1.2 * \mathcal{N}(0.0, 1.0)$
Additive Gaussian noise	-	$p = 0.1(1.0)$ $\mathcal{U}(0.0, 0.1)$	-
Coarse dropout	-	-	$p = 0.5(0.0)$ $\mathcal{U}(0.6, 1.4)$ $size_{\%} \sim \mathcal{U}(0.1, 0.25)$
Invert	yes, n.a.	-	$p = 0.5(1.0)$ $\mathcal{U}(0.0, 0.2)$

Table 4.13: **Grasping Experiment with PyraPose**. Grasping YCB-video [48] objects, 10 trials are performed per object. Comparison is given for grasping without refinement (w/o ICP) and when refining the initial pose with instance segmentation and ICP (/w ICP).

Object	Mustard	Jell-o	Spam	Banana	Foam	Success
w/o ICP	10	1	9	7	3	60%
/w ICP	10	5	10	3	7	70%

texture of the real object compared to the model used to render the training data. Interestingly, the grasp success for the banana drops when using refinement. This is explained by the observation that the locally optimal output of ICP often results in a pose that is rotated towards the table, which leads to many grasp trajectories protruding the table plane. For the other objects, however, using instance segmentation to crop the point clouds for ICP-refinement improves performance. This highlight the advantage of computing the masks. Figure 4.10 shows an example of a successful grasp of mustard.

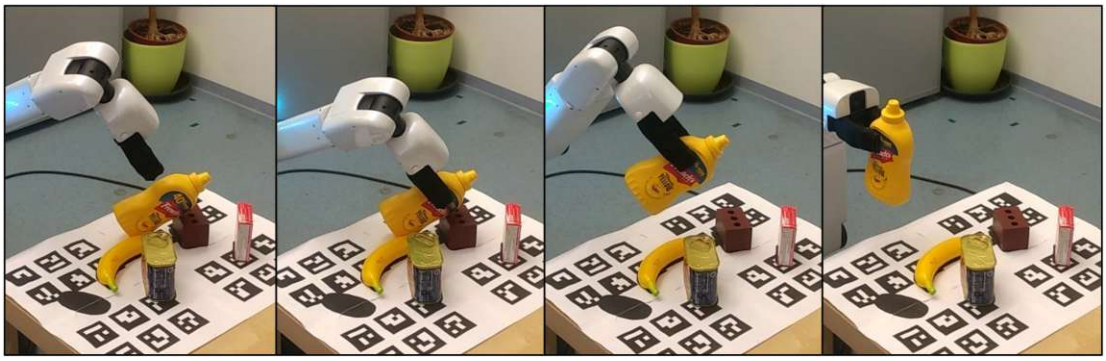

Figure 4.10: **Visualisation of a Mustard bottle Grasp** YCB-video's mustard bottle is successfully grasped from a pose supported by foam and remains stable in the gripper.

4.3 Scalable End-to-end Trainable Object Pose Estimation using COPE

This section provides quantitative and qualitative evaluation of COPE on several datasets. After introducing the experimental setup, we proceed with comparisons to the state of the art on two challenging datasets using the BOP protocol [75]. In addition, ablation studies are presented to quantify the influence of direct pose supervision on an additional dataset. To further validate and thoroughly test the capabilities of our method, we present results on a synthetic dataset with up to 100 object instances per image.

Datasets Evaluation is provided on three standard datasets: Linemod [112], Occlusion [135] and IC-bin [149]. For evaluation, we use the subsets provided with the BOP challenge. Linemod provides 200 test images for each of the 13 objects that come with watertight object models. Linemod constitutes a common benchmark for object pose estimation in cluttered environments. Occlusion consists of 200 test images of Linemod's second test sequence with all eight objects annotated in each. Linemod provides test images with challenging object occlusions. IC-bin presents 150 test images of up to 21 instances of two objects with heavy occlusion.

For training we use the 50k physically-based renderings for each dataset available through the BOP challenge [68]. These are generated using physically based rendering (*pbr*) [71], as described in Section 3.1.3. Results on Linemod and Linemod are provided with the same models trained on all 13 objects of Linemod. No annotated real images are used for training.

Evaluation Metrics Comparison to the state of the art is provided using the performance score of the BOP challenge [75]. Results for pose estimation are reported using the Average Recall (AR), the average over the metrics presented in Equation 2.8. Ablations are evaluated using the ADD recall, or ADDS recall for objects exhibiting symmetries [112], presented in Equation 2.6. We report the fraction of poses below the commonly used error threshold of 10% of the object diameter. Results for object detection are reported using the the mean Average Precision (mAP) of the Microsoft

COCO object detection challenge [34], presented in Equation 2.5. The results are those for the IoU values from 0.5 to 0.95 in 0.05 steps.

Implementation Details The weights of the backbone are pre-trained on ImageNet [126] and fine-tuned for 120 epochs using the Adam [128] optimiser with a learning rate of $1 \cdot 10^{-5}$ and a batch size of 8. Previous work suggests overcoming the domain gap between training on synthetic and testing on real images by not updating certain network weights during optimisation [49], [124]. Similarly, we do not update parameters of batch normalizations and the convolutions of the first two stages of the backbone during fine-tuning. We apply image augmentations as described in Section 3.1.4. The parameter d in Equation (3.6) is set to 3 for all experiments. Additionally, we exploit the self-regularising effect of Mish [125] in all layers up to the last layers of the task heads, which are linearly activated.

4.3.1 Scalable Multi-object Multi-instance Pose Estimation

This section compares the performance of COPE to the state of the art on IC-bin [149] and Occlusion [135]. Results using the BOP setting reporting the AR are provided in Table 4.14. The bottom section compares single-model methods, i.e approaches that produce estimates for all object classes and their instances in a single forward pass. Both DPOD [49] and EPOS [58] require PnP for deriving the 6D pose from the predicted geometric correspondences, while COPE directly outputs the 6D pose. COPE improves over both methods in AR on average. Compared to the previous single-model state of the art, EPOS, COPE achieves similar AR on Linemod 0.543 as compared to 0.547 but improves to 0.440 in comparison to 0.363 on IC-bin. More remarkable, however, is that the runtime of COPE is 37 times faster using the inference speed calculated by the BOP toolkit[4].

The top section of Table 4.14 presents the results of multi-model methods. These multi-model methods provide results using an object detector to sample sparse locations priors in the first stage and separately trained networks per object class for correspondence prediction, respectively pose estimation, in the second. For the methods [37], [150]–[152] no results on IC-bin are available. Compared to the best individually performing methods on both datasets, CosyPose on IC-bin and ZebraPose on Linemod, COPE results in $\sim 24\%$ relative performance decrease. This is in the expected range due to the known performance decrease for single-staged approaches [56].

Runtime Figure 4.11 presents the average runtime and standard deviation on IC-bin for five test runs of COPE, CDPNv2 and CosyPose on an Intel CPU with 3.6GHz and an Nvidia Geforce 3090 GPU. The times reported for CDPNv2 exclude the time required for detecting objects. Despite omitting the runtime of CDPNv2's first stage, our method is more than 12 times faster and 7 times faster than CosyPose when processing 15 object instances. Most notably, in contrast to multi-model approaches, COPE is capable of directly providing 6D poses for multi-object multi-instance cases at almost constant runtime, which makes it highly suitable for real-time scenarios.

Object Detection Table 4.15 compares the object detection accuracy, using mAP,

[4] https://github.com/thodan/bop_toolkit

Table 4.14: **Comparison to the State of the Art for Pose Estimation.** Presented are the *Average Recall* on IC-bin and Occlusion, the average over both and the inference speed using the BOP toolkit.

Method	IC-bin	Occlusion	Avg.	Time
Multi-model				
AAE [23]	0.217	0.146	0.182	**0.199**
Pix2Pose [40]	0.226	0.363	0.295	1.230
2Dto3D [111]	0.342	0.525	0.434	0.546
CDPNv2 [39]	0.473	0.624	0.549	1.010
SurfEmb [38]	0.550	0.623	0.587	6.296
CosyPose [17]	**0.574**	0.618	0.**596**	0.227
SO-Pose [37]	-	0.613	-	-
CIR [150]	-	0.655	-	-
PFA [151]	-	0.683	-	-
ZebraPose [152]	-	**0.718**	-	0.250
Single-model				
DPOD [49]	0.169	0.130	0.150	0.211
EPOS [58]	0.363	**0.547**	0.455	2.804
COPE(ours)	**0.440**	0.543	**0.492**	**0.075**

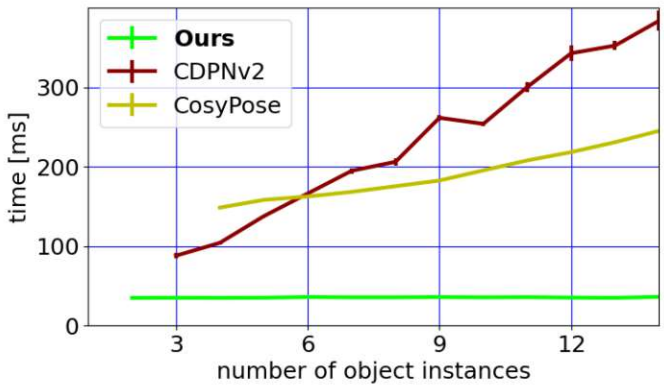

Figure 4.11: **Runtime Comparison to the State of the Art on IC-bin.** Provided are the times it takes to estimate poses for all object instances in a single image.

of COPE to the state of the art on IC-bin and Linemod using the same training data. The *mAP* metric quantifies the quality of bounding box estimation. On average, COPE outperforms both MaskRCNN [92] and FCOS [90]. COPE achieves the highest average *mAP* over both datasets. On Linemod, COPE is superior to MaskRCNN, achieving 0.532 as compared to 0.375 but slightly inferior to FCOS that reaches 0.622. However, we draw this conclusion with caution since FCOS only performs 2D Detection. The network size of FCOS is $\sim$ 50 million parameters just for object detection while COPE additionally predict geometric correspondences and direct 6D poses with only $\sim$ 17

Table 4.15: **Comparison to the State of the Art for Object Detection.** Presented are the *mAP* on IC-bin and Linemod and the average using the BOP toolkit.

Method	IC-bin	Linemod	Avg.
MaskRCNN [17], [38], [92]	0.316	0.375	0.346
FCOS [39], [90], [152]	0.323	**0.622**	0.473
COPE(ours)	**0.431**	0.532	**0.482**

Table 4.16: **Ablation Study for Pose Supervision.** Provided is the average *ADD/(-S)* recall. The objects eggbox and glue are considered as symmetric objects.

Supervision		Voting	Linemod	Occlusion
IM	2D	PnP	0.654	0.280
DR	2D	PnP	0.712	0.330
	6D	all	0.715	0.342
DR-P	2D	PnP	0.672	0.341
	6D	all	0.672	0.345
DR-PC	2D	PnP	0.712	0.348
	6D	n=1	0.722	0.338
	6D	n=5	0.724	0.346
	6D	n=10	0.732	**0.350**
	6D	all	**0.738**	0.349

million parameters more. Additionally, FCOS uses an input image resolution with up to 1333 pixels for the larger image side while COPE uses 640×480 input images. Thus, COPE solves twice as many tasks with higher complexity from images with half of the input resolution. COPE's detected bounding boxes are more precise than both standard detectors used by many multi-model methods on IC-bin, achieving 0.431 as compared to 0.323 and 0.316. As such our method provides excellent location priors for pose refinement.

Direct-pose Regression Table 4.16 displays the influence of direct pose regression on the end-to-end architecture on the Linemod [112] and Occlusion [135] datasets using the *ADD/(-S)* recall. The column *Voting* indicates the pose voting procedure using RANSAC-EPnP, an average of all direct pose hypotheses, or an average of the direct pose estimates with the best n hypotheses in terms of $\hat{C}$.

The results show that supervising the training process with direct pose regression (**DR**) improves the quality of the intermediate representation (**IM**) tremendously. The improvement is from 0.654 to 0.715 on Occlusion and from 0.280 to 0.342 on Linemod. Using **DR** direct pose estimates is superior to using those estimated by RANSAC-EPnP. Providing additional guidance with L_{proj} (**DR-P**) improves for the occluded scenario of Occlusion, but is detrimental for Linemod. Ultimately, enforcing consistency between the internal representation and the correspondences projected to 2D using the regressed 6D pose with L_{cons} (**DR-PC**) leads to good results for direct regression and when using the intermediate representation on both datasets. Figure 5 shows an example of Linemod's cat under occlusion. The re-projected model using the ground truth is colored

Figure 4.12: **Pose Supervision Comparison on Linemod's cat.** From left to right: raw image, pose obtained from geometric correspondences and RANSAC-EPnP, and direct pose regression. Blue, red and green meshes indicate ground truth, false positive and true positive pose (as measured by *ADD*).

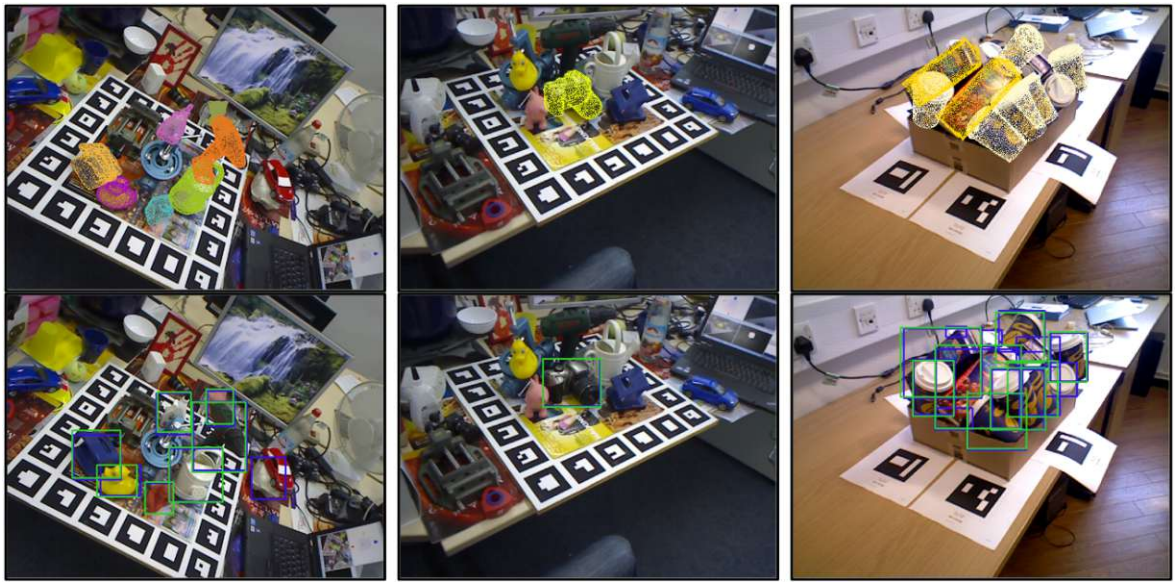

Figure 4.13: **Qualitative results on Occlusion, Linemod and IC-bin.** Top row shows reprojected object meshes based on the estimated poses, bottom row shows bounding box estimates. Blue boxes indicate the ground truth while green boxes indicate estimates.

blue and the wrong estimate based on *ADD*, using the intermediate representation and RANSAC-EPnP, is coloured red (middle image). Direct pose regression recovers from the incorrect intermediate representation, which is displayed in green (right image).

Error Cases Figure 4.13 shows results on Occlusion, Linemod and IC-bin. Projected object meshes based on the estimated pose are displayed in the top row, while estimated bounding boxes in comparison to the respective ground truth are illustrated in the bottom row. Green and red bounding boxes portray estimates and ground truth, respectively. The left image pair indicates a common error for Occlusion: a false negative detection of the eggbox. The right image pair shows that some of IC-bin's instances of juice are difficult to detect while detecting coffeecup works well even under heavy occlusion if more than the lid is visible.

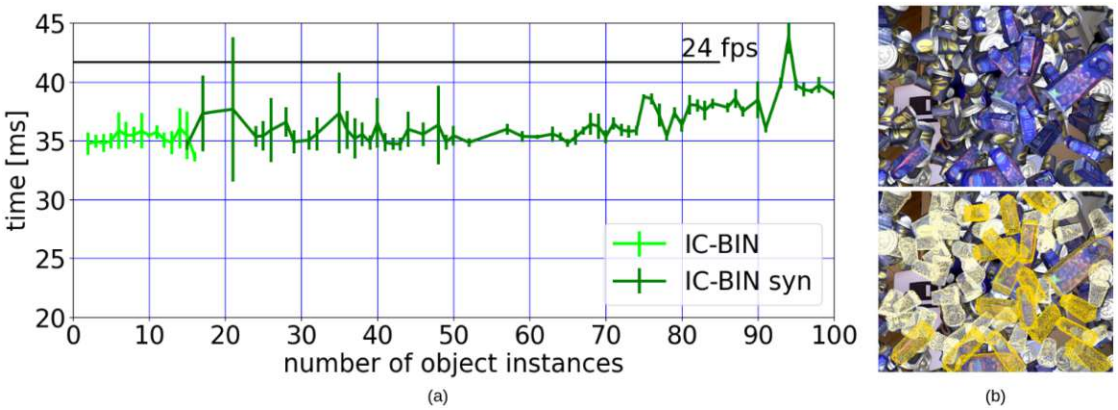

Figure 4.14: **Runtime Evaluation of COPE on IC-bin and IC-bin syn.** (a) COPE provides negligible runtime increase up to 70 object instances per image. The black line indicates the threshold for real-time processing. (b) An example of a test image of *IC-bin syn* and a visualization of the estimated poses.

4.3.2 Constant Runtime with Respect to the Number of Object Instances

In order to exhaustively test the runtime and scalability of COPE, we create a synthetic test dataset using the IC-bin objects and OpenGL[5] rendering, named IC-bin syn. The number of object instances to render per image is sampled from a uniform distribution with a lower bound of 10 and upper bound of 100. We render the sampled number of object instances randomly from the IC-bin objects onto the test images of IC-bin. Results are again provided for processing one test image on an Intel CPU with 3.6GHz and an Nvidia Geforce 3090 GPU.

Figure 4.14 (a) presents the runtime of our method for detecting and estimating the poses of up to 100 object instances in a single image. We report the average runtime and standard deviation for five test runs. The runtime increases negligibly up to 70 detected object instances. Our method exceeds real-time processing for more than 90 instances per image. Therefore, it provides quantitative proof of the tremendous scalability of the presented approach, and the constancy of the runtime with respect to the number of object instances in a single test image. Figure 4 (b) provides a rendered synthetic test image (top) and projected object models based on the estimated poses (bottom).

4.3.3 Grasping Transparent Objects with COPE

This section presents transparent object grasping using COPE. Grasping is performed with the Toyota HSR robot [143], [144]. Object poses are estimated in the camera frame of the RGB-D camera in its head, the Xtion PRO LIVE. A transparent canister is placed on a wooden table, where the robot is positioned to look at the table at

[5]https://github.com/thodan/bop_renderer

approximately 45° angle elevation. We place the canister on the table in three different ways: upright position, recumbent position, and attached to the base plate. We also use two different backgrounds, in-particular we use the checkerboard and the original wooden background of our table. The first is part of our training dataset and the other is for evaluating the generalisation to unseen backgrounds.

Real-world Training Data Real-world data is captured using the Realsense D435[6] and the ZED from stereolabs[7]. The camera is attached to the end-effector of the KUKA arm robot[8] and moved around the object in a sequence, taking 104 images from three different angles of elevation. In total, we record 15 sequences, 6 sequences with one object instance using the ZED, and 9 with two instances using the D435. In total 1352 training images are captured and annotated. Object poses, lighting and background patterns vary while data capturing. Four different backgrounds (a metallic surface, green fabric and 2 black-white checkerboard patterns) and three levels of illumination are combined.

Quantitative Grasping Results using COPE This section quantitatively evaluates grasping with COPE. In total, 5 grasps trails for the each of the four scenarios upright and recumbent with seen and unseen tabletop are performed. Three grasp results defined:

- *Full Grasp*: The object is grasped and remains stable in the gripper
- *Reached Grasp*: A suitable grasp position is reached, but the grasp is unsuccessful due to the gripper moving the object previous to grasping.
- *Failed Grasp*: Neither the object is grasped nor a suitable grasping position is reached.

We assign a score of 1, 0.5 and 0 for *Full Grasp*, *Reached Grasp* and *Failed Grasp*, respectively. Reported scores in Table 4.17 are normalised by the number of grasp attempts.

Table 4.17: **Grasping Experiments with COPE** Canister grasping from a tabletop with seen and unseen surface.

Tabletop	Seen		Unseen	
	upright	recumbent	upright	recumbent
Full Grasp	0.6	0.2	0.4	0.2
Reached Grasp	0.1	0.1	0.1	0.0
overall	0.7	0.3	0.5	0.2

Table 4.17 shows that COPE also provides pose estimates suited for grasping a transparent object. The grasping success rate is higher for the case of the seen background as

[6]https://www.intelrealsense.com/depth-camera-d435/
[7]https://www.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf
[8]https://www.kuka.com/de-at/produkte-leistungen/robotersysteme/industrieroboter/lbr-iiwa

compared to the unseen background. Yet, the trained model generalises well to unseen object appearances. Grasping success rate significantly drops when the canister is lying on the table (recumbant). This stems from the fact that estimating the depth of the object with respect to the camera is very challenging. Hence, estimating the object pose only slightly too far away as compared to the ground truth already leads to finding no executable gripper trajectory.

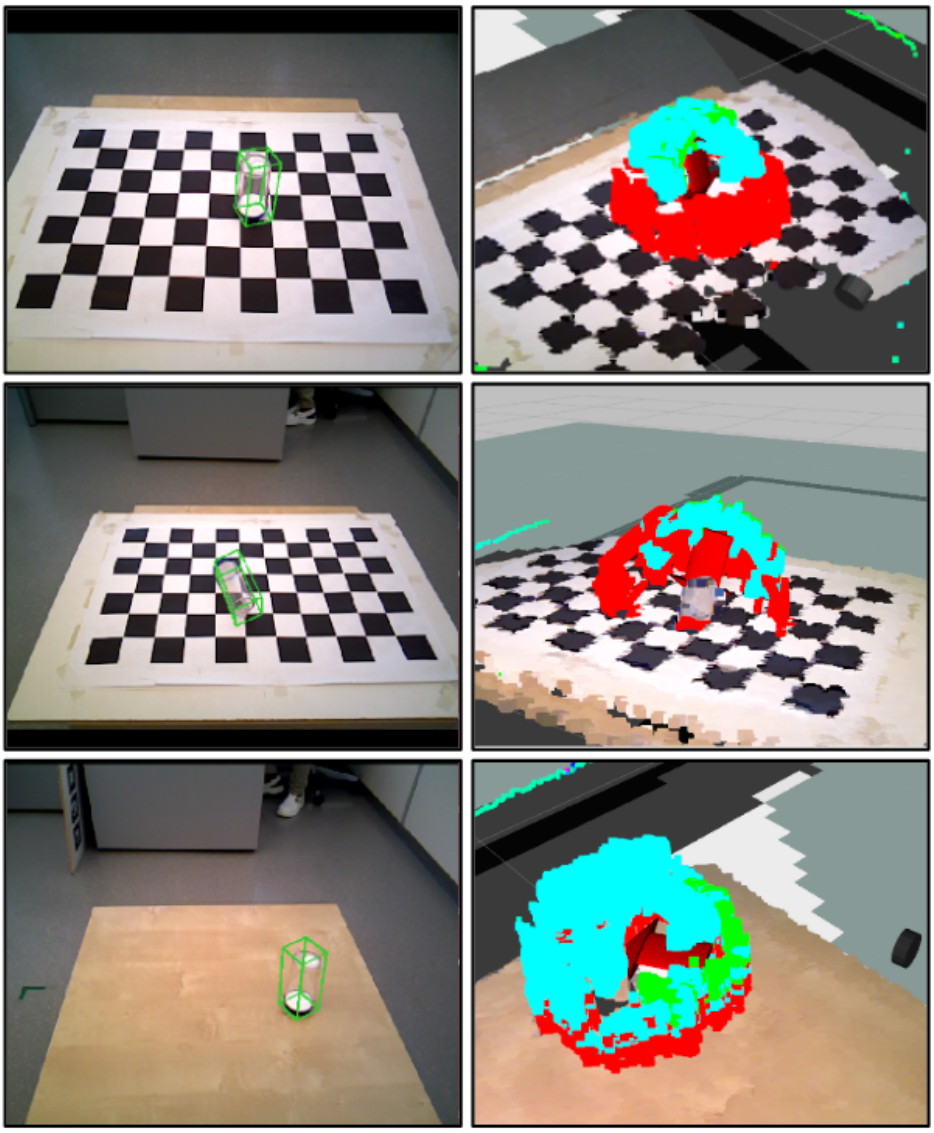

Figure 4.15: **Pose Estimation and Grasp Point Sampling** The left column of images indicates estimated poses with a green bounding box. Right shows all grasps, grasps protruding the table (red), grasps not protruding the table plane (blue and green), and chosen grasp (green).

Qualitative Grasping Results using COPE Figure 4.15 shows the estimated 6D poses for the transparent object and transformed grasp points. We observe a small offset of the estimated rotations in the left images. The rotation is difficult to estimate

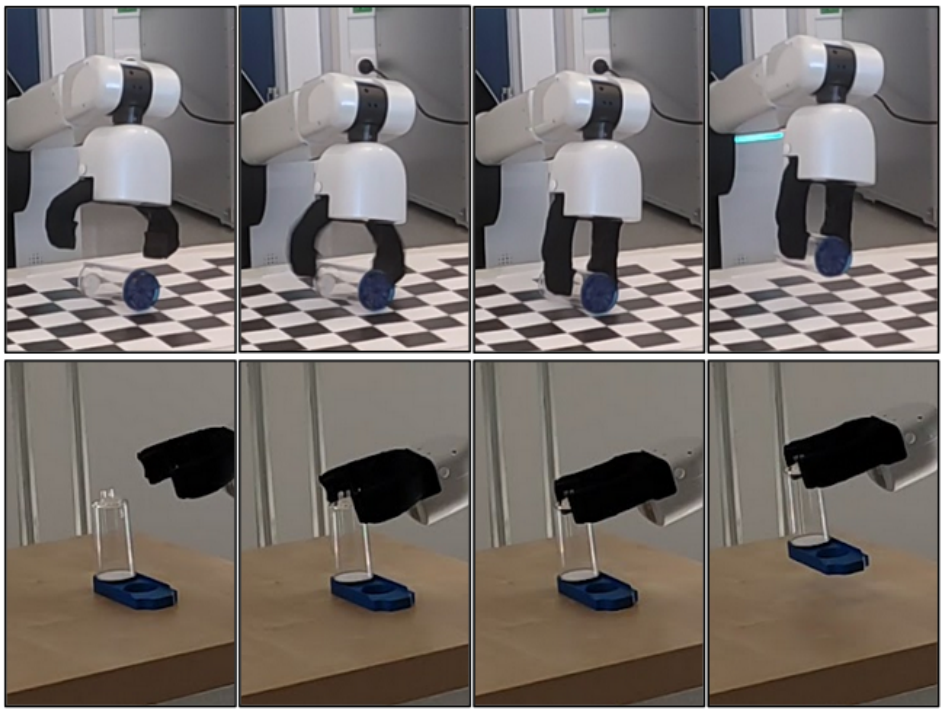

Figure 4.16: **Grasping Sequences with COPE** The Canister placed in its base plate, as such unseen during training, is picked from the table.

since only the top part of the canister provides cues to disambiguate the rotational symmetry along the longitudinal axis of the object. The right images shows the possible grasp points around the pose estimated canister. Red point clouds denote to grasp points that are protruding the table plane, blue and green point clouds are the valid configurations not protruding the table plane. The top and bottom row show that for the upright position of the canister it is easy to plan executable grasp trajectories, while the middle row shows few valid grasp points for the canister in the recumbent position.

Grasping the Canister The top two rows of Figure 4.16 show examples of successful grasp. The top row shows the canister being grasped from a lying position. The bottom row presents a successful grasp with a distractor, which is not visible during training. As such, we show that COPE is robust with respect to distractors.

4.4 Self-Comparison

This section provides a comparison of the methods presented in this thesis, SyDPose (Section 3.3), PyraPose (Section 3.4) and COPE (Section 3.5). Presented is the *ADD/(-S)* recall, Equation 2.6, on the Linemod [112] and the Occlusion [135] dataset.

SyDPose estimates 30.2% of Linemod's and 9.1% of Occlusion's poses correctly. PyraPose estimates 63.4% of Linemod's and 28.1% of Occlusion's poses correctly. Directly comparing these two methods, the majority of the performance improvement of PyraPose results from using RGB as input modality. Both methods use Resnet50 as backbone, pretrained on ImageNet. PyraPose drawns large benefit from this [153].

Table 4.18: **Self-Comparison of the presented Methods.** Provided is the average *ADD/(-S)* recall. The objects eggbox and glue are considered as symmetric objects.

Method	Linemod [112]	Occlusion [135]
SyDPose (Section 3.3)	0.302	0.091
PyraPose (Section 3.4)	0.634	0.281
COPE (Section 3.5)	**0.732**	**0.350**

Using pretrained network weights allows to freeze the first two stages of Resnet, thus improves generality of learned representations. Additionally, PyraPose is not restricted to a certain camera principle as SyDPose, thus more widely applicable. However, SyDPose has the advantage that no texture priors of the objects are required. This is also the case for Graph convolutional networks, which are better suited for depth data than CNN'S.

PyraPose estimates 63.4% of Linemod's and 28.1% of Occlusion's poses correctly. COPE estimates 73.2% of Linemod's and 35.0% of Occlusion poses correctly. The performance difference is due to having additional direct pose supervision with Cope and due to using Resnet101 as backbone for COPE and Resnet50 as backbone for PyraPose. Yet, Cope has multiple advantages over PyraPose. Using the true location sampling scheme of COPE only requires choosing 1 instead of 12 hyperparameters. We argue that our choice of setting COPE's hyperparameter d, in Equation 3.6, to 3 is quite general and as such valid for a broad variety of datasets. Choosing anchor parameters [30] as used by PyraPose in practice requires at least approximate statistics of the test set in terms of object poses and object sizes to maximise the performance. Additionally, COPE has a nine times smaller output space, thus convergence is improved and training time reduced. Defining full convergence as encountering two loss plateaus, with a loss plateau being two consecutive epochs without reduction in overall loss, COPE only requires 120 epochs for full convergence as compared to 200 required by PyraPose. The smaller output space also leads a training time of $\sim$ 55 minutes per epoch with COPE, as compared to $\sim$ 70 minutes with PyraPose. Both of these aspects considered Cope requires only half of the training time. Furthermore, COPE estimates poses with negligible runtime increase for up to 90 object instances. Per-instances geometric correspondences have to be processed separately with PyraPose to derive object poses. This requires running RANSAC-PnP on each set of hypotheses. Thus COPE provides a natural and simple formulation for object pose estimation, with ease of use and scalability comparable to approaches for the lower-dimensional problem of object detection.

Chapter 5

Conclusion

Scenarios involving robotic object manipulation provide manifold challenges depending on the scene setup, such as diverse object sets, clutter, occlusion, illumination and object material properties. Object pose estimators overfit to the training data with respect to these characterisitcs, thus generalisation to arbitrary places of testing is limited. Since capturing and annotating data in and retraining the pose estimator for each distinct place of deployment is infeasible, it is desired to train on synthetic data to simulate arbitrary scene complexity and variation while retaining correct annotations. Especially given that it cannot be assumed that data or precise information of the target domain is available beforehand. Therefore, object localisation is exclusively enabled from geometric objects priors.

5.1 Problems

This section outlines the problems tackled in this thesis and restates the connected research questions.

Using synthetic data comes with major advantages over using annotated real-world data. Large amounts of data can easily be created without requiring manual labour and geometric object priors are sufficient for data generation, as is the case for classical methods. Furthermore, using real-world data leads to a bias of trained models with respect to the training data characteristics such as camera in- and extrinsics, imaging noise, scene context and photometric attributes. These variations are controllable and can be arbitrarily complex when using synthetic data. However, the major challenge when using rendered data is overcoming the domain gap, i.e. the adaptation or generalisation to the domain of deployment. As such, the first research question to answer is how to adapt or generalise trained models to novel domains.

Recently learning-based object pose estimation approaches started to catch up and surpass classical ones with respect to pose estimation performance. The clear advantages of these approaches is that CNNs encode functions, instead of templates like classical approaches do. These resulting continuous output spaces of learning-based approaches enable effective interpolation between the available training samples, and efficient multi-task approach formulations. Thus, learning-based object pose estimation is theoretically well suited for multi-object problems. Nonetheless, predicting poses of

multiple distinct objects simultaneously comes with challenges, namely finding adequate pose representation for training target sampling and handling objects of multiple scales to avoid inducing a bias. Hence, many approaches are multi-staged, using separately trained networks for detection and pose estimation, which limits efficiency. As a consequence, the second research question is how to enable simultaneous object detection and correspondence estimation.

Detecting objects in the image and subsequently processing the resulting sparse location priors for each object separately, is detrimental to runtime and scalability. Apart from that, most pose estimation approaches adopt surrogate training targets in the form of geometric correspondences to formulate easier-to-solve problems, since direct pose regression has been shown to result in worse performance. Outputting surrogate representations for the 6D pose requires PnP to subsequently derive the object pose. This leads to shortcomings because a) adopting surrogate training targets decouples pose estimation from the training process and thus limits learning and b) running inference for multi-instance scenarios leads to a computational complexity of at least $O(n)$ with respect to the number of objects n for the pose estimation stage. Thus, this type of approach has severely diminishing applicability for realistic scenarios. In order to overcome these shortcomings the last relevant research question for this work is how to formulate scalable, single-staged object pose estimation.

5.2 Summary

This section reviews the contributions of this thesis, summarises our findings and acknowledges limitations.

Domain Adaptation and Generalisation For object localisation in the depth domain we present a strategy that renders and augments synthetic data in a way that effectively transitions trained models to real-world depth data. During synthetic scene creation, non-textured models and random distractor objects are physically sampled to create realistic scene setups and object interactions. Images are rendered with a camera setup that resembles the imaging geometry of the expected physical sensor. A realistic sensor model in combination with simplex noise augments the synthetic depth images to enable domain adaptation. Experiments are presented showing that the resulting data improves detection performance over training with available real-world data, since it is easier to create more diverse variations and object interactions in the virtual scene. These data are also suited for object pose estimation and results in state-of-the-art performance on two standard datasets. The findings are limited to structured light depth sensors. Further experiments are needed to establish the requirements to generalise to depth sensors that use other physical principles for data capturing.

For object localisation in the RGB domain we employ a combination of strategies. Training images are generated using physically-based rendering, sampling physically plausible poses, and object interactions and randomising material properties. Domain generalisation of the trained models is encouraged through the application of randomised image augmentations, pretraining the backbone and freezing the parameters of the

convolutions of early backbone stages and the Batch normalization layers. We present results showing that the inclusion of different types of contrast noise in the data augmentation improves generality of encoded representations.

Multi-object Pose Estimation We demonstrate that employing the strategies described above in feature pyramid-based network training leads to pose estimators that generalise well in RGB. The presented network design effectively shares internal representations over multiple objects through standardising training targets using anchors and a specialised feature pyramid. Objects are detected and geometric correspondences are predicted simultaneously for multiple objects. Poses are estimated from the geometric correspondences using PnP. The network is thus end-to-end trainable and only 1 network is required per dataset. Results are presented that show that this leads to state-of-the-art performance, suitability for robotic grasping and pose estimation in novel domains, i.e. with a difference in object placement, illumination and with a different camera.

Scalable Object Pose Estimation The state of the art for object pose estimation adopts surrogate training targets, such as geometric correspondences, since directly regressing the 6D pose leads to worse results. The feasibility of regressing the 6D pose from these intermediate correspondences has recently been demonstrated. We directly regress the 6D pose from the predicted 2D geometric correspondences. This is done simultaneously for all feature map locations in the multi-scale feature map of feature pyramids. Feature map training locations are sampled and standardised using the object shape. For testing, instance-specific output is computed by clustering the set of hypotheses using their mutual IoUs of the predicted 2D bounding boxes. As such, we handle multi-object multi-instance cases in a single stage, directly outputting 6D poses. This efficient formulation is end-to-end trainable and results in practically constant runtime for up to 90 object instances, due to the negligible runtime increase with respect to the number of object instances in the test image.

5.3 Outlook

This section acknowledges the general limitations of the presented thesis and proposes untouched future research tasks to progress the state of the art in promising directions.

Texture-agnostic Pose Estimation This thesis provides solutions for object localisation using depth data. As such, objects where no texture is available can be handled. However, it is desired to solve the problem of pose estimation in RGB, since this modality provides richer information on transparent or strongly specular reflecting objects, and is better suited to overcome ambiguities resulting from geometric symmetries and heavy clutter. Additionally, using object textures for synthetic-to-real object pose estimation is disadvantageous. Reconstructing the texture of physically available objects often results in artefacts on either the texture or the object's geometry. Then again CNN's are biased towards these textural cues [154], which might be detrimental for the generalisation of the trained models, especially considering the resulting bias due to imperfect reconstructions. Ongoing research tackles the problem of training object pose estimators in RGB from meshes with no texture priors available, which has not

been addressed in this thesis. Random textures are assigned to the objects instead of using the vanilla reconstructed ones. Initial experiments show that for LM's [112] ape 5 different textures per training view and for T-less' [119] object number 6 already 3 different textures are sufficient to improve over training with the reconstructed object texture.

Self-supervised Pose Estimation Recent works adopt geometric correspondences as surrogate training targets to maximise pose estimation performance. Though these achieve tremendous results, the hand-crafted nature does not present an optimal solution. Future work should thus investigate self-supervised approaches for descriptor learning. Self-supervised descriptor space encoding using Vision Transformers and contrastive learning might be a promising direction [155]. Especially, since self-supervised learning already shows promising results for encoding general objects representations, and thus might be well suited for pose estimation under domain shift [100]. Additionally, self-supervised learning allows using setup-specific real-world data without the requirement of annotations. Including pose annotations, synthetic samples are mapped to the same locations in descriptor space as real-world ones, hence improving domain adaptation.

Transparent Object Pose Estimation Grasping or estimating poses of transparent objects is a very challenging problem due to their material properties. The material's transmissivity of incoming light rays makes it difficult to produce robust estimates with standard approaches. Recently [156] showed that it is feasible to learn object matting, i.e. light ray refraction and attenuation, with CNNs. Future work will thus investigate if encoding such photometric representations is better suited for transparent object pose estimation, than using geometric correspondences.

List of Figures

List of Tables

Bibliography

[1] D. Bauer, T. Patten, and M. Vincze, "VeREFINE: Integrating object pose verification with physics-guided iterative refinement," *IEEE Robotics Automation Letters*, vol. 5, no. 3, pp. 4289–4296, 2020 (cit. on pp. 1, 5, 50).

[2] T. Patten, K. Park, and M. Vincze, "Dgcm-net: Dense geometrical correspondence matching network for incremental experience-based robotic grasping," *Frontiers in Robotics and AI*, p. 120, 2020 (cit. on p. 1).

[3] S. Thalhammer, M. Leitner, T. Patten, and M. Vincze, "Pyrapose: Feature pyramids for fast and accurate object pose estimation under domain shift," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2021 (cit. on p. 1).

[4] K. Kleeberger, C. Landgraf, and M. F. Huber, "Large-scale 6d object pose estimation dataset for industrial bin-picking," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 2573–2578 (cit. on p. 1).

[5] K. Kleeberger and M. F. Huber, "Single shot 6d object pose estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 6239–6245 (cit. on p. 1).

[6] J. Sock, K. Kim, C. Sahin, and T. Kim, "Multi-task deep networks for depth-based 6d object pose and joint registration in crowd scenarios," English, in *Proceedings of British Machine Vision Conference*, Jul. 2018 (cit. on pp. 1, 5, 6, 18, 31, 36, 59).

[7] T. Hou, A. Ahmadyan, L. Zhang, J. Wei, and M. Grundmann, "Mobilepose: Real-time pose estimation for unseen objects with weak shape supervision," *arXiv preprint arXiv:2003.03522*, 2020 (cit. on p. 1).

[8] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 12, pp. 2633–2651, 2015 (cit. on p. 1).

[9] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang, "Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020 (cit. on p. 1).

[10] K. Park, T. Patten, and M. Vincze, "Neural object learning for 6d pose estimation using a few cluttered images," in *Proceedings of the European Conference on Computer Vision*, Springer, 2020, pp. 656–673 (cit. on p. 1).

[11] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6D pose estimation," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 683–698 (cit. on pp. 1, 5).

[12] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking," in *Robotics: Science and Systems (RSS)*, 2019 (cit. on p. 1).

[13] C. Wang, R. Martín-Martín, D. Xu, *et al.*, "6-pack: Category-level 6d pose tracker with anchor-based keypoints," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 059–10 066 (cit. on p. 1).

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105 (cit. on pp. 1, 2, 29, 35, 43).

[15] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448 (cit. on pp. 1, 3).

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241 (cit. on p. 1).

[17] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *European Conference on Computer Vision*, Springer, 2020, pp. 574–591 (cit. on pp. 1, 5, 6, 19, 43, 49, 71, 72).

[18] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410 (cit. on p. 2).

[19] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Interpretable transformations with encoder-decoder networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5726–5735 (cit. on p. 2).

[20] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019 (cit. on pp. 2, 4–6, 62, 65, 66).

[21] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019 (cit. on pp. 2, 4).

[22] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DoF pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570 (cit. on pp. 2, 6, 14, 18, 19, 31, 38, 40).

[23] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of European Conference on Computer Vision*, 2018, pp. 699–715 (cit. on pp. 2, 6, 13–15, 18, 26, 28, 31, 33, 36, 40, 43, 59, 61, 63, 64, 67, 68, 71).

[24] T. Hodaň, X. Zabulis, M. Lourakis, Obdržálek, and J. Matas, "Detection and fine 3d pose estimation of texture-less objects in rgb-d images," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2015, pp. 4421–4428 (cit. on pp. 3, 5, 17, 35).

[25] S. Hinterstoisser, S. Holzer, C. Cagniart, *et al.*, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*, IEEE, 2011, pp. 858–865 (cit. on pp. 3, 5, 13, 17, 35).

[26] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 998–1005 (cit. on pp. 3, 5, 17, 35, 57, 59–61).

[27] J. Vidal, C.-Y. Lin, and R. Marti, "6d pose estimation using an improved method based on point pair features," Apr. 2018, pp. 405–409 (cit. on pp. 3, 5, 17, 60).

[28] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector," in *European conference on computer vision*, Springer, 2016, pp. 21–37 (cit. on pp. 3, 6, 9, 15–17, 30, 39, 46).

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788 (cit. on pp. 3, 4, 6, 30).

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015 (cit. on pp. 3, 10, 15, 16, 19, 30, 41, 44, 46–48, 53, 78).

[31] A. G. Howard, M. Zhu, B. Chen, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017 (cit. on pp. 4, 6).

[32] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125 (cit. on pp. 4, 6, 9, 15, 16, 30, 39, 40, 46, 65).

[33] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 778–10 787 (cit. on pp. 4, 6, 15, 16, 30, 39, 41).

[34] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common objects in context," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 740–755 (cit. on pp. 4, 13, 15, 20, 27, 28, 40, 41, 70).

[35] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4 (cit. on p. 4).

[36] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009 (cit. on pp. 4, 6, 9, 19, 34, 35, 43).

[37] Y. Di, F. Manhardt, G. Wang, X. Ji, N. Navab, and F. Tombari, "So-pose: Exploiting self-occlusion for direct 6d pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 396–12 405 (cit. on pp. 5, 7, 19, 43, 44, 48, 70, 71).

[38] R. L. Haugaard and A. G. Buch, "Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings," *arXiv preprint arXiv:2111.13489*, 2021 (cit. on pp. 5, 6, 19, 33, 43, 71, 72).

[39] Z. Li, G. Wang, and X. Ji, "Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7678–7687 (cit. on pp. 5, 6, 18, 19, 28, 31, 38, 39, 43, 48, 49, 63–65, 67, 68, 71, 72).

[40] K. Park, T. Patten, and M. Vincze, "Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7668–7677 (cit. on pp. 5, 6, 14, 18, 19, 28, 31, 33, 38, 40, 43, 48, 49, 63–65, 67, 68, 71).

[41] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 611–16 621 (cit. on pp. 5, 7, 19, 44, 48).

[42] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2017, p. 7 (cit. on pp. 5, 14, 15).

[43] K. Park, T. Patten, J. Prankl, and M. Vincze, "Multi-task template matching for object detection, segmentation and pose estimation using depth images," 2019 (cit. on pp. 5, 13, 14, 18).

[44] B. Planche, Z. Wu, K. Ma, *et al.*, "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition," in *3D Vision (3DV), 2017 International Conference on*, IEEE, 2017, pp. 1–10 (cit. on pp. 5, 13, 25).

[45] S. Zakharov, B. Planche, Z. Wu, A. Hutter, H. Kosch, and S. Ilic, "Keep it unreal: Bridging the realism gap for 2.5d recognition with geometry priors only," in *Proceedings of International Conference on 3D Vision*, 2018, pp. 1–11 (cit. on pp. 5, 8, 15, 26, 36, 54).

[46] C. Wang, D. Xu, Y. Zhu, *et al.*, "DenseFusion: 6D object pose estimation by iterative dense fusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3343–3352 (cit. on pp. 6, 18, 19, 40).

[47] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *The IEEE International Conference on Computer Vision*, Oct. 2017 (cit. on pp. 6, 18, 31, 33, 35, 36, 38, 40, 43, 59).

[48] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Proceedings of Robotics: Science and Systems*, 2018 (cit. on pp. 6, 18, 26, 31, 33–36, 42, 62, 67, 68).

[49] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: 6D pose object detector and refiner," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950 (cit. on pp. 6, 15, 18, 19, 28, 29, 31, 38, 40, 43, 61, 63–67, 70, 71).

[50] Z. Li, Y. Hu, M. Salzmann, and X. Ji, "Robust rgb-based 6-dof pose estimation without real pose annotations," *arXiv preprint arXiv:2008.08391*, 2020 (cit. on pp. 6, 28).

[51] M. Rad, M. Oberweger, and V. Lepetit, "Feature mapping for learning fast and accurate 3d pose inference from synthetic images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2018 (cit. on p. 6).

[52] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, "Self6d: Self-supervised monocular 6d object pose estimation," in *Proceedings of the European Conference on Computer Vision*, Springer, 2020, pp. 108–125 (cit. on pp. 6, 15, 49, 61, 63, 64, 66).

[53] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6D object pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3385–3394 (cit. on pp. 6, 18, 19, 31, 38, 41, 42, 47, 59).

[54] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3D object pose estimation," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 119–134 (cit. on pp. 6, 18, 19, 31, 35, 36, 38, 41–43, 59).

[55] T. Hodaň, F. Michel, E. Brachmann, *et al.*, "Bop: Benchmark for 6d object pose estimation," in *Computer Vision – ECCV*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 19–35 (cit. on pp. 6, 35, 60, 62).

[56] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *2017 IEEE International Conference on Computer Vision*, pp. 2999–3007, 2017 (cit. on pp. 6, 15, 16, 18, 19, 30, 33, 38, 39, 43, 45–48, 64, 65, 70).

[57] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020 (cit. on pp. 6, 15).

[58] T. Hodan, D. Baráth, and J. Matas, "EPOS: Estimating 6D pose of objects with symmetries," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11 700–11 709, 2020 (cit. on pp. 6, 18, 19, 28, 29, 31, 33, 43, 49, 70, 71).

[59] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301 (cit. on pp. 6, 18, 31, 35, 36, 38, 59, 62).

[60] D. Barath and J. Matas, *Graph-cut ransac*, 2018 (cit. on pp. 6, 19, 49).

[61] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6dpose: Recovering 6d object pose from a single rgb image," in *ECCV*, 2018 (cit. on pp. 6, 18, 31).

[62] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529 (cit. on pp. 6, 13, 15, 17, 18, 26, 31, 59, 61, 62).

[63] B. Chen, A. Parra, J. Cao, N. Li, and T.-J. Chin, "End-to-end learnable geometric vision by backpropagating pnp optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8100–8109 (cit. on pp. 7, 19, 44).

[64] Y. Hu, P. Fua, W. Wang, and M. Salzmann, "Single-stage 6D object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2927–2936 (cit. on pp. 7, 19, 44, 48).

[65] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981 (cit. on pp. 9, 35).

[66] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. on pp. 9, 19, 34, 43).

[67] F. M. Carlucci, P. Russo, and B. Caputo, "A deep representation for depth images from synthetic data," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 1362–1369 (cit. on pp. 13, 14).

[68] M. Denninger, M. Sundermeyer, D. Winkelbauer, *et al.*, "Blenderproc," *CoRR*, vol. abs/1911.01911, 2019. arXiv: 1911.01911. [Online]. Available: http://arxiv.org/abs/1911.01911 (cit. on pp. 13, 14, 24, 27, 28, 40, 62, 69).

[69] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1301–1310 (cit. on pp. 13, 14).

[70] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, "Understanding real world indoor scenes with synthetic data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4077–4085 (cit. on p. 13).

[71] T. Hodaň, V. Vineet, R. Gal, *et al.*, "Photorealistic image synthesis for object instance detection," *Proceedings of the IEEE International Conference on Image Processing*, 2019 (cit. on pp. 13, 14, 26, 40, 62, 69).

[72] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 23–30 (cit. on pp. 13, 15, 25).

[73] A. Rozantsev, V. Lepetit, and P. Fua, "On rendering synthetic images for training an object detector," *Computer Vision and Image Understanding*, vol. 137, pp. 24–37, 2015 (cit. on pp. 13, 25).

[74] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953 (cit. on p. 13).

[75] T. Hodaň, M. Sundermeyer, B. Drost, *et al.*, "BOP challenge 2020 on 6D object localization," *Proceedings of the European Conference on Computer Vision Workshops*, 2020 (cit. on pp. 14, 19, 21, 27, 31, 65, 66, 69).

[76] G. Csurka, "A comprehensive survey on domain adaptation for visual applications," in *Domain Adaptation in Computer Vision Applications*, G. Csurka, Ed. Cham: Springer International Publishing, 2017, pp. 1–35, ISBN: 978-3-319-58347-1. [Online]. Available: `https://doi.org/10.1007/978-3-319-58347-1_1` (cit. on p. 14).

[77] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *2013 IEEE International Conference on Computer Vision*, Dec. 2013, pp. 2960–2967 (cit. on p. 14).

[78] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 1180–1189. [Online]. Available: `http://dl.acm.org/citation.cfm?id=3045118.3045244` (cit. on p. 14).

[79] M. Long and J. Wang, "Learning transferable features with deep adaptation networks," in *ICML*, 2015 (cit. on p. 14).

[80] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 regularization for learning kernels," *arXiv preprint arXiv:1205.2653*, 2012 (cit. on p. 14).

[81] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014 (cit. on p. 14).

[82] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456 (cit. on p. 14).

[83] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016 (cit. on p. 14).

[84] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017 (cit. on p. 14).

[85] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *CoRR*, vol. abs/1611.04201, 2017 (cit. on p. 14).

[86] K. Perlin, "Improving noise," in *ACM Transactions on Graphics*, ACM, vol. 21, 2002, pp. 681–682 (cit. on pp. 15, 25, 26, 54).

[87] M. Rad, M. Oberweger, and V. Lepetit, "Domain transfer for 3d pose estimation from color images without manual annotations," in *Asian Conference on Computer Vision*, Springer, 2018, pp. 69–84 (cit. on pp. 15, 36, 61, 63, 64).

[88] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578 (cit. on pp. 15, 16, 30, 45, 46).

[89] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision*, 2018, pp. 734–750 (cit. on pp. 15, 16, 30, 45, 46).

[90] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9627–9636 (cit. on pp. 15, 16, 19, 30, 45–47, 71, 72).

[91] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045 (cit. on pp. 15, 16, 30, 39–41).

[92] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969 (cit. on pp. 15, 16, 30, 36, 71, 72).

[93] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768 (cit. on pp. 16, 30, 39).

[94] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *International Conference on Learning Representations*, 2016 (cit. on pp. 16, 40).

[95] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye, "Freeanchor: Learning to match anchors for visual object detection," *Advances in neural information processing systems*, vol. 32, 2019 (cit. on pp. 16, 30, 46).

[96] A. Aldoma, F. Tombari, L. D. Stefano, and M. Vincze, "A global hypotheses verification method for 3d object recognition," in *European conference on computer vision*, Springer, 2012, pp. 511–524 (cit. on p. 17).

[97] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, and M. Vincze, "Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation," in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 2104–2111 (cit. on p. 17).

[98] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," in *2011 18th IEEE international conference on image processing*, IEEE, 2011, pp. 809–812 (cit. on p. 17).

[99] S. V. Alexandrov, T. Patten, and M. Vincze, "Leveraging symmetries to improve object detection and pose estimation from range data," in *International Conference on Computer Vision Systems*, Springer, 2019, pp. 397–407 (cit. on p. 17).

[100] V. N. Nguyen, Y. Hu, Y. Xiao, M. Salzmann, and V. Lepetit, "Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6771–6780 (cit. on pp. 17, 82).

[101] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *European Conference on Computer Vision*, Springer, 2014, pp. 462–477 (cit. on pp. 17, 58, 60).

[102] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3364–3372 (cit. on pp. 17, 62).

[103] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015 (cit. on p. 18).

[104] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778 (cit. on pp. 18, 28–30, 41, 43, 53).

[105] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "A novel representation of parts for accurate 3d object detection and tracking in monocular images," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 4391–4399 (cit. on pp. 18, 36, 38).

[106] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6d pose refinement in rgb," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 800–815 (cit. on pp. 18, 29, 63).

[107] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 629–11 638 (cit. on pp. 18, 19).

[108] K. Park, A. Mousavian, Y. Xiang, and D. Fox, "Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 710–10 719 (cit. on p. 19).

[109] G. Terzakis and M. Lourakis, "A consistently fast and globally optimal solution to the perspective-n-point problem," in *Proceedings of the European Conference on Computer Vision*, Springer, 2020, pp. 478–494 (cit. on pp. 19, 34, 43).

[110] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 431–440 (cit. on pp. 19, 48).

[111] J. Liu, Z. Zou, X. Ye, *et al.*, "Leaping from 2d detection to efficient 6dof object pose estimation," in *Proceedings of the European Conference on Computer Vision*, Springer, 2020, pp. 707–714 (cit. on pp. 19, 71).

[112] S. Hinterstoisser, V. Lepetit, S. Ilic, *et al.*, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Proceedings of the Asian Conference on Computer Vision*, 2012, pp. 548–562 (cit. on pp. 20, 32, 36, 40, 52, 57, 58, 61, 63–67, 69, 72, 77, 78, 82).

[113] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "Blensor: Blender sensor simulation toolbox," in *International Symposium on Visual Computing*, Springer, 2011, pp. 199–208 (cit. on p. 25).

[114] M. J. Landau, B. Choo, and P. A. Beling, "Simulating kinect infrared and depth images.," *IEEE Trans. Cybernetics*, vol. 46, no. 12, pp. 3018–3031, 2016 (cit. on p. 25).

[115] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, IEEE, 2012, pp. 524–530 (cit. on p. 25).

[116] F. Alhwarin, A. Ferrein, and I. Scholl, "Ir stereo kinect: Improving depth images by combining structured light with ir stereo," in *Pacific Rim International Conference on Artificial Intelligence*, Springer, 2014, pp. 409–421 (cit. on p. 25).

[117] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 67–77, 2019 (cit. on p. 25).

[118] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012 (cit. on p. 25).

[119] 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 *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2017, pp. 880–888 (cit. on pp. 26, 82).

[120] Y. Nakagawa, H. Uchiyama, H. Nagahara, and R.-I. Taniguchi, "Estimating surface normals with depth image gradients for fast and accurate registration," in *3D Vision (3DV), 2015 International Conference on*, IEEE, 2015, pp. 640–647 (cit. on pp. 26, 53, 57).

[121] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258 (cit. on pp. 28, 29).

[122] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114 (cit. on p. 28).

[123] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4918–4927 (cit. on p. 28).

[124] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Proceedings of the European Conference on Computer Vision Workshops*, 2018, pp. 0–0 (cit. on pp. 29, 43, 70).

[125] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019 (cit. on pp. 29, 70).

[126] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015 (cit. on pp. 30, 53, 70).

[127] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020 (cit. on pp. 30, 46).

[128] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015 (cit. on pp. 33, 57, 70).

[129] F. Manhardt, D. M. Arroyo, C. Rupprecht, *et al.*, "Explaining the ambiguity of object detection and 6D pose from visual data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6841–6850 (cit. on pp. 33, 40, 48, 63, 64).

[130] K. Park, J. Prankl, M. Zillich, and M. Vincze, "Pose estimation of similar shape objects using convolutional neural network trained by synthetic data," in *Proceedings of the OAGM-ARW Joint Workshop*, May 2017, pp. 87–91 (cit. on p. 33).

[131] J. Richter-Klug and U. Frese, "Handling object symmetries in cnn-based pose estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 850–13 856 (cit. on p. 33).

[132] Y. Shi, J. Huang, X. Xu, Y. Zhang, and K. Xu, "Stablepose: Learning 6d object poses from geometrically stable patches," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 222–15 231 (cit. on pp. 33, 48).

[133] E. Corona, K. Kundu, and S. Fidler, "Pose estimation for objects with rotational symmetry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 7215–7222 (cit. on p. 33).

[134] A. Glent Buch, L. Kiforenko, and D. Kraft, "Rotational subgroup voting and pose clustering for robust 3d object recognition," Oct. 2017, pp. 4137–4145 (cit. on p. 35).

[135] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *Proceedings of European Conference on Computer Vision*, 2014, pp. 536–551 (cit. on pp. 36, 42, 53, 57, 60–62, 65, 67, 69, 70, 72, 77, 78).

[136] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2 (cit. on pp. 38, 48).

[137] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2403–2412 (cit. on p. 40).

[138] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010 (cit. on p. 42).

[139] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753 (cit. on p. 46).

[140] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2018, pp. 3559–3568 (cit. on p. 48).

[141] M. R. Loghmani, B. Caputo, and M. Vincze, "Recognizing objects in-the-wild: Where do we stand?" In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2170–2177 (cit. on p. 50).

[142] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 1378–1385 (cit. on p. 50).

[143] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of the research platform of a domestic mobile manipulator utilized for international competition and field test," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 7675–7682 (cit. on pp. 50, 67, 74).

[144] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of human support robot as the research platform of a domestic mobile manipulator," *ROBOMECH Journal*, vol. 6, no. 4, pp. 1–15, 2019 (cit. on pp. 50, 67, 74).

[145] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, IEEE, vol. 1, 2001, pp. I–I (cit. on pp. 53, 57).

[146] Y. Litvak, A. Biess, and A. Bar-Hillel, "Learning pose estimation for high-precision robotic assembly using simulated depth images," in *Proceedings of International Conference on Robotics and Automation*, 2018, pp. 3521–3527 (cit. on p. 59).

[147] F. Bottarel, G. Vezzani, U. Pattacini, and L. Natale, "GRASPA 1.0: GRASPA is a robot arm grasping performance benchmark," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 836–843, 2020 (cit. on pp. 62, 67).

[148] *BOP: Benchmark for 6d object pose estimation*, `https://bop.felk.cvut.cz/home/`, Accessed: 2020-10-14 (cit. on pp. 63, 64).

[149] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2016, pp. 3583–3592 (cit. on pp. 69, 70).

[150] L. Lipson, Z. Teed, A. Goyal, and J. Deng, "Coupled iterative refinement for 6d multi-object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6728–6737 (cit. on pp. 70, 71).

[151] Y. Hu, P. Fua, and M. Salzmann, "Perspective flow aggregation for data-limited 6d object pose estimation," *arXiv preprint arXiv:2203.09836*, 2022 (cit. on pp. 70, 71).

[152] Y. Su, M. Saleh, T. Fetzer, *et al.*, "Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6738–6748 (cit. on pp. 70–72).

[153] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019 (cit. on p. 77).

[154] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018 (cit. on p. 81).

[155] M. Caron, H. Touvron, I. Misra, *et al.*, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9650–9660 (cit. on p. 82).

[156] G. Chen, K. Han, and K.-Y. K. Wong, "Tom-net: Learning transparent object matting from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9233–9241 (cit. on p. 82).

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, August 2022

Stefan Thalhammer M.Sc.