Efficient Methods for Learning Visual Multi-object 6D Pose Estimation and Tracking

DISSERTATION
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von
Arul Selvam Periyasamy
aus
Kangeyam, Indien

Bonn, June 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter/Betreuer:	Prof. Dr. rer. nat. Sven Behnke
Gutachter:	Prof. Dr. rer. nat. Jörg-Dieter Stückler

Tag der Promotion:	22.01.2026
Erscheinungsjahr:	2026

Dedicated to my parents Perma and Periyasamy, and my wife Thendral Jaya Paramasamy.

Abstract

Object detection and 6D object pose estimation are foundational components in a visual scene understanding system. Despite the intertwined nature of these tasks, the standard methods for scene understanding decouple object detection and pose estimation. They perform object detection in the first stage and process only the crops containing the target object to estimate pose parameters. In this thesis, we present an alternate approach called multi-object pose estimation, in which we perform joint object detection and pose estimation in a single step for all the objects in the scene. We formulate multi-object pose estimation as a set prediction problem. We utilize the permutation invariant nature of the recent Transformer architecture to generate a set of object predictions for a given single-view RGB image. Our model achieves accuracy comparable to the state-of-the-art models while being significantly faster. Video sequences contain rich temporal information that offer additional context than single-view images. To take advantage of the temporal information contained in the video sequences, we develop an enhanced version of our multi-object pose estimation model by incorporating temporal fusion modules and demonstrate improved pose estimation accuracy as well as improved object detection accuracy. In general, datasets are crucial for learning-based perception methods. The most commonly used datasets for object-centric scene understanding feature static scenes. To enable dynamic scene understanding, we introduce a photo- and physically-realistic dataset featuring simulations of commonly occurring bin-picking scenarios. We use this dataset to evaluate the temporal fusion approach we present in this thesis. Moreover, ability to refine less accurate pose predictions is an important attribute in building robust scene understanding systems. We introduce pose and shape parameter refinement pipelines based on iterative render and compare optimization. However, comparing rendered and observed images in the RGB color space is error-prone. Thus, we propose image comparison in learned feature space that are invariant to secondary lighting effects. To facilitate time efficient iterative refinement, we develop a lightweight differentiable renderer. Furthermore, real-world objects exhibit symmetry. The standard pose estimation models are designed to estimate a single plausible pose among a set of symmetrical poses. Thus, they are not suitable for inferring symmetry. To this end, we model object symmetries using implicit probability density networks and present an automatic ground-truth annotation scheme to train such implicit networks without manual symmetry labels.

Acknowledgments

First, I would like to thank Prof. Dr. Sven Behnke for giving me the opportunity to do my Ph.D. research in the Autonomous Intelligent Systems group. I am grateful for his guidance and insightful discussions that shaped my research over the last few years.

I would like to express my deep gratitude to Max Schwarz, with whom I collaborated on several research projects and robotic competitions. A special thanks also goes to Michael Schreiber for his assistance in the lab, as well as for extending his support in matters beyond the lab.

In addition, I sincerely thank my colleagues Jan Quenzel, Christian Lenz, Radu Alexandru Rosu, Dmytro Pavlichenko, Malte Splietker, Grzegorz Ficht, Jan Nogga, Angel Villar-Corrales, Malte Mosbach, André Rochow, Dr. Simon Bultmann, Dr. Philipp Allgeuer, Dr. Tobias Klamt, Dr. David Droeschel, Dr. Marius Beul, and Dr. Hafez Farazi for the stimulating environment in the lab. The numerous robotic competitions we participated in over the eight years I was at the group have been a great learning experience and wonderful memories for me. Also, I would like to thank my students, Arash Amini, Luis Denninger, Vladimir Tsaturyan, and Patrick Löwen for collaborating with me on their bachelor/master's thesis.

The research conducted during this thesis was partially funded by the University of Bonn's "MBZIRC Preisgeld" scholarship programme, German Ministry of Education and Research (BMBF), grant no. 01IS21080, project "Learn2Grasp: Learning Human-like Interactive Grasping based on Visual and Haptic Feedback", Competence Center for Machine Learning Rhine Ruhr (ML2R), which is funded by the Federal Ministry of Education and Research of Germany (grant no. 01—S18038A), and multiple Amazon Research Awards. These funds have been invaluable in enabling me to pursue my research activities.

I would also like to thank my parents, Prema and Periyasamy, for their continued support. Additionally, I would like to thank my wife, Thendral Jaya Paramasamy, for her unwavering support, patience, and encouragement over the years. Finally, I extend my gratitude to Prof. Dr. Jörg-Dieter Stückler, Professor for Intelligent Perception in Technical Systems, University of Augsburg, for agreeing to be the second reviewer for this thesis.

Contents

Introduction

The ability of autonomous robots to interact with their environment greatly depends on their scene understanding capabilities. The environments in which the robots operate determine the exact definition of scene understanding. In the context of autonomous bin-picking, scene understanding involves detecting and recognizing the objects in the scene, estimating their position and orientation with respect to a well-defined coordinate system (shown in Fig. 1.1), reasoning about their symmetries, and estimating object shape parameters. Bin-picking environments are challenging due to the presence of a wide range of objects and lighting conditions, a high degree of occlusion, and the dynamic nature of the environment (see Fig. 1.2). Thus, they serve as a perfect setting for evaluating the progress in scene understanding algorithms. In this thesis, we present methods for learning and refining object poses for scene understanding. Selecting a suitable representation greatly influences the learning potential of a machine learning model. This motivates us to investigate different representations for 6D pose parameters and object symmetries. We evaluate our methods on the standard scene understanding datasets featuring static scenes as well as a dynamic bin-picking synthetic dataset created as a part of this thesis.

Krizhevsky, Sutskever, and Hinton (2012) introduced AlexNet, a convolutional neural network (CNN), which performed significantly better than the state-of-the-art object recognition models on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). CNNs use a series of convolutional layers that are designed to automatically detect features such as edges, textures, objects, and other important aspects of images. The hierarchical nature of the CNNs enables learning low-level image features like edges and corners in the initial layers and aggregating these features to learn high-level concepts like objects, humans, and other complex visual elements in the final layers. Although earlier versions of CNNs have been used successfully in applications like hand-written digits recognition and automatic meter reading (LeCun et al., 1998; Behnke, 2003b), the computational

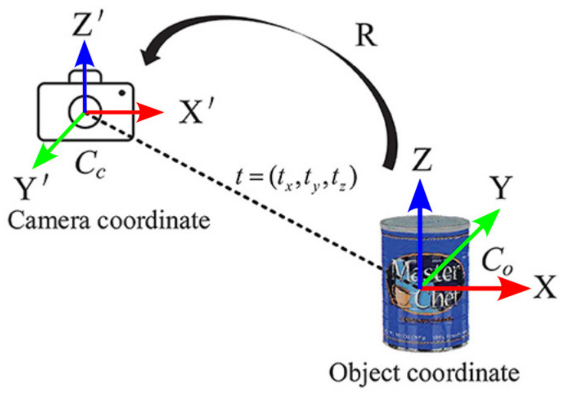


Figure 1.1: Object pose definition. 6D object pose is defined as the translation and the rotation of an object with respect to the camera coordinate frame. Both the translation and the rotation components consist of three degrees of freedom each.

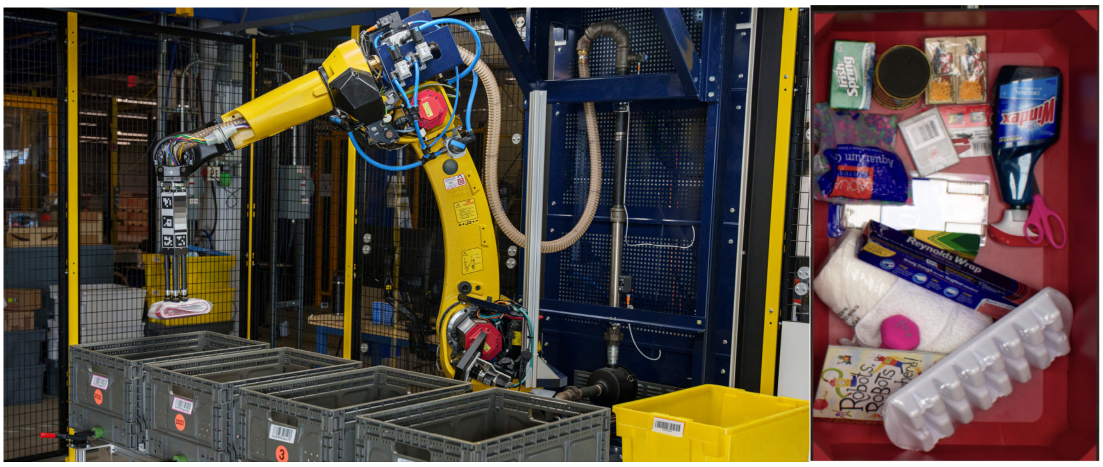


Figure 1.2: Left: An industrial bin-picking setup: Amazon Robotics Sparrow system[1].
Right: A typical bin featuring a diverse set of objects (Schwarz et al., 2018a)

and dataset limitations hampered the broader adoption of CNNs. With the availability of general purpose graphics processing units (GPGPUs) and improvements in distributed multi-GPU training schemes, training deep CNNs was made possible (Cireşan et al., 2011; Cireşan, Meier, and Schmidhuber, 2012; Schulz and Behnke, 2012). Moreover, algorithmic improvements like activation functions (Nair and Hinton, 2010; Jarrett et al., 2009), pooling mechanisms (LeCun, Kavukcuoglu, and Farabet, 2010; LeCun, Huang, and Bottou, 2004), and regularization techniques (Hinton et al., 2012) made AlexNet successful. It demonstrated the capabilities of deep learning and kick-started widespread adoption of deep learning for computer vision. Since then, deep learning has become the predominant machine learning method for solving computer vision tasks. Deep learning architectures like deep residual learning (He et al., 2016), fully convolutional neural networks (Long, Shelhamer, and Darrell, 2015), attention-based vision transformers (Vaswani et al., 2017; Dosovitskiy et al., 2021) have greatly increased the capabilities of deep learning models. In an orthogonal direction, large-scale datasets featuring diverse environments also made training deep learning models feasible on a wide range of tasks. Furthermore, improvements to training schemes emerged, such as fine-tuning. In this approach, only a few final layers of the models are trained for specific tasks after initially training the model from scratch on large-scale datasets, which removed the barriers for applying deep learning in data-limited domains.

Starting in 2015, Amazon Robotics organized the annual Amazon Robotics Challenge (ARC) for three editions with an emphasis on bin-picking scenarios (Correll et al., 2016; Schwarz et al., 2017; Schwarz et al., 2018a). In the first two editions of the competition, the competing teams were given the actual set of objects weeks in advance. In the final edition, only half the objects were given in advance, and the rest were given to the teams 40 minutes before the start of the competition. This increase in complexity was set to encourage the research community to explore data- and compute-efficient perception algorithms. The performance of the participating teams demonstrated the progress made in scene understanding, which was mostly driven by deep learning. Moreover, it showcased the impact of physically realistic data augmentation and synthetic data generation pipelines on efficient training of deep learning models. However, it also highlighted the limitations of the prevalent perception systems at that time. Most teams opted for 2D perception tasks, i. e. object detection and/or semantic segmentation. Only a few teams incorporated 3D tasks like pose estimation in their perception pipelines. Even the teams that included 3D perception components used them as a supplement to the predominant 2D perception pipelines in limited scenarios. This is due to several factors. First, acquiring 3D annotations is time- and resource-consuming. Second, developing automated annotation pipelines for 3D is significantly more complex than for 2D. Finally, the object pose estimation methods at that time performed poorly under occlusion and were decoupled from the object detection modules. Despite, the remarkable progress made by pose estimation methods in recent years, the standard perception pipelines still follow a decoupled design in which object detection is performed in the first stage and only the crops containing the target object are processed by pose estimation models in the second stage (Hodaň et al., 2020; Sundermeyer et al., 2023). Such pipelines necessitate complex specialized layers like *non maximum suppression* (NMS), *region of interest* (RoI) pooling, and *anchor boxes.* In cluttered environments, this decoupling leads to poor results. In this thesis, we address these limitations by introducing multi-object pose estimation as a set prediction formulation, in which we perform object detection and pose estimation jointly in a single step.

We incorporate ideas from well-established sub-fields of computer vision and computer graphics like render-and-compare (Nair, Susskind, and Hinton, 2008; Krull et al., 2015; Moreno et al., 2016), differentiable rasterization (Loper and Black, 2014; Liu et al., 2019), and neural implicit models (Xie et al., 2022; Mildenhall et al., 2021; Murphy et al., 2021) to predict and refine multi-object poses, and model object symmetries. Render-and-compare is a special case of the analysis-by-synthesis paradigm in which an image rendered according to an initial estimate of scene parameters is compared with the observed image of the scene. By iteratively adjusting the parameters to minimize the differences between the rendered and the observed images, we generate more accurate scene parameters. The pose and shape refinement methods we discuss in Chapter 4 follow the render-and-compare approach. We utilize a differentiable rasterizer, which renders an image given the scene parameters in the synthesis step and provides gradients of the scene parameters with respect to the rendered image during the optimization step in our pipeline for pose and shape refinement. Symmetry is a common feature of objects found in nature and human-made environments. However, the standard pose estimation models estimate a single pose given an input image. This does not reveal any information about the symmetry. To capture symmetry, we investigate the object pose representations based on implicit probability functions (Murphy et al., 2021)

Microsoft Kinect spurred the era of inexpensive depth cameras (Zhang, 2012). Robotics and computer vision research greatly benefited from the availability of affordable depth cameras (Endres et al., 2013; Schwarz et al., 2018b; Zollhöfer et al., 2018; Gupta et al., 2014). However, depth cameras have a limited *effective measurement range*, a narrow field of view (FOV), and operate at lower frames per second (fps) compared to RGB cameras (Robinson and Contributors, 2020; Jing et al., 2017). Industrial depth cameras offer improved capabilities, but they are significantly more expensive than RGB cameras. Moreover, depth cameras do not work well on highly reflective and transparent surfaces. Thus, in this thesis, we only consider RGB input during inference. However, we assume that high-quality 3D object meshes are available during training.

1 https://www.aboutamazon.com/news/operations/amazon-robotics-robots-fulfillment-center

1.1 Summary of Contributions

In this thesis, we present novel approaches for learning multi-object pose estimation single-view RGB images and video sequences, refining pose and shape parameters using abstract render and compare framework, implicit neural representations for modeling object symmetries, and introduce a photo- and physically-realistic dataset and its generator. The contributions we present in this thesis are as follows:

- *Simulated dynamic bin-picking dataset.* A photo- and physically-realistic dataset together with its generator. The dataset features commonly occurring bin-picking scenarios.
- *Multi-object pose estimation models.* In contrast to the standard pose estimation models, which are multi-staged, we present single-stage multi-object pose estimation models that jointly perform object detection and 6D pose parameter estimation in a single step.
- *Multi-object pose tracking using temporal fusion.* Multi-object pose estimation model utilizing temporal fusion for improved object detection and pose estimation accuracy from video sequences.
- *Object pose and shape refinement using abstract render-and-compare.* Methods for object pose and shape parameters refinement based on the abstract render-and-compare approach. We also present a lightweight differentiable renderer to facilitate gradient-based optimization in the iterative render-and-compare framework.
- *Implicit probability density networks for modeling object symmetries without explicit symmetry labels.* A neural implicit probability density function to model object symmetry and an automatic ground-truth generation pipeline to train the implicit model without explicit symmetry labels.

1.1.1 Publications

Parts of this thesis have been published in peer-reviewed conference proceedings and journals. The most relevant publications covering the chapters of this thesis are listed below in chronological order:

Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke:

Refining 6D object pose predictions using abstract render-and-compare

In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, Toronto, Canada, 2019. DOI:10.48550/arXiv.1910.03412

Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke:

SynPick: A dataset for dynamic bin picking scene understanding

In: *17th International Conference on Automation Science and Engineering (CASE)*, Lyon, France, 2021. DOI:10.48550/arXiv.2107.04852

Arash Amini, Arul Selvam Periyasamy, and Sven Behnke

T6D-Direct: Transformers for multi-object 6D pose direct regression

In: *43rd DAGM German Conference on Pattern Recognition (GCPR)*, Bonn, Germany, 2022. DOI:10.48550/arXiv.2109.10948

Arul Selvam Periyasamy, Max Schwarz, and Sven Behnke:

Iterative 3D deformable registration from single-view RGB images using differentiable rendering

In: *17th International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal, 2022. DOI:10.5220/0010817100003124

Arash Amini*, Arul Selvam Periyasamy*, and Sven Behnke:

YOLOPose: Transformer-based multi-object 6D pose estimation using keypoint regression

In: *17th International Conference on Intelligent Autonomous Systems (IAS)*, Zagreb, Croatia, 2022. DOI:10.48550/arXiv.2205.02536

Best Paper Award

Arul Selvam Periyasamy*, Luis Denninger*, and Sven Behnke:

Learning implicit probability distribution functions for symmetric orientation estimation from RGB images without pose labels

In: *6th IEEE International Conference on Robotic Computing (IRC)*, Naples, Italy, 2022. DOI:10.48550/arXiv.2211.11394

Arul Selvam Periyasamy, Arash Amini, Vladimir Tsaturyan, and Sven Behnke:

YOLOPose V2: Understanding and improving transformer-based 6D pose estimation

In: *Robotics and Autonomous Systems (RAS)*, Volume 168, pp 104490, 2023. DOI:10.48550/arXiv.2307.11550

Arul Selvam Periyasamy*, Vladimir Tsaturyan*, and Sven Behnke:

Efficient multi-object pose estimation using multi-resolution deformable attention and query aggregation

In: *7th IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, USA, 2023. DOI:10.48550/arXiv.2312.08268

Arul Selvam Periyasamy, and Sven Behnke:

MOTPose: Multi-object 6D pose estimation for dynamic video sequences using attention-based temporal fusion

In: *IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024. DOI:10.48550/arXiv.2403.09309

The following publications, listed in chronological order, are related to the topics presented in this thesis and were written during the time the presented research was conducted. They are cited as external literature in this thesis and do not cover significant parts of the chapters:

Arul Selvam Periyasamy, Catherine Capellen, Max Schwarz, and Sven Behnke:

ConvPoseCNN2: Prediction and refinement of dense 6D object pose

In: *Communications in Computer and Information Science (CCIS)*, Volume 1474, pp 353-371, 2020. DOI:10.48550/arXiv.2205.11124

Arul Selvam Periyasamy, Max Schwarz, and Sven Behnke:

Towards 3D scene understanding using differentiable rendering

In: *SN Computer Science*, Volume 3, Issue 3, pp 245, 2023. DOI:10.1007/s42979-022-01663-3

1.1.2 Open Source Releases

The following projects were developed to facilitate the research presented in this thesis and were made open source.

- *StillllebenDR*. A lightweight differentiable renderer built on Stillleben[2].
- *SynPick Generator and Dataset*. A photo- and physically-realistic dataset together with its generator[3].

1.2 Outline

This thesis is structured as follows.

In Chapter 2, we present SynPick, a dataset for dynamic scene understanding and discuss our implementation of the dataset generator.

In Chapter 3, we introduce a pipeline for automatic ground-truth symmetry pose annotation based on render-and-compare. Using the generated annotations, we train an implicit probability density network to model object symmetries without any explicit symmetry annotation.

2 https://github.com/AIS-Bonn/stillleben/blob/master/python/stillleben/diff.py
3 https://www.ais.uni-bonn.de/datasets/synpick/

We present novel pipelines for object pose and shape refinement based on the abstract render-and-compare framework employing our implementation of a lightweight GPU-powered differentiable renderer in Chapter 4.

In Chapter 5, we introduce T6D-Direct, a multi-object pose estimation model based on the set prediction formulation that performs joint object detection and pose parameter regression.

We extend the T6D-Direct model to include keypoint regression and investigate fully vision transformer models for multi-object pose estimation in Chapter 6.

In Chapter 7, we introduce MOTPose, a multi-object pose estimation model that fuses temporal information from the video sequences to improve pose estimation accuracy.

Finally, in Chapter 8, we conclude the thesis and discuss future research directions.

2 SynPick: A Dataset for Dynamic Scene Understanding

Datasets serve as the foundation upon which the machine learning models are trained and validated. Several large scale datasets for object perception exist. However, these datasets consist of predominantly static scenes. Manually annotating dynamic scenes is prohibitively expensive. Modern physics engines provide an alternative option to generate dynamic datasets for object perception with minimal manual effort. In this Chapter, we present SynPick, a photo-realistic physically-plausible dataset of dynamic bin-picking scenes.

Statement of Personal Contribution

The SynPick dataset and generator presented in this Chapter are adapted from the following publication.

Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke:

SynPick: A dataset for dynamic bin picking scene understanding

In: *17th International Conference on Automation Science and Engineering (CASE)*, Lyon, France. 2021.

The author of this thesis substantially contributed to all aspects of the publication, including the literature survey, the conception, formalization, design, and implementation of the dataset generator, the generation of the dataset, the evaluation of baseline method on the generated dataset, the preparation of the manuscript, as well as the revision and final editing of the version published.

2.1 Introduction

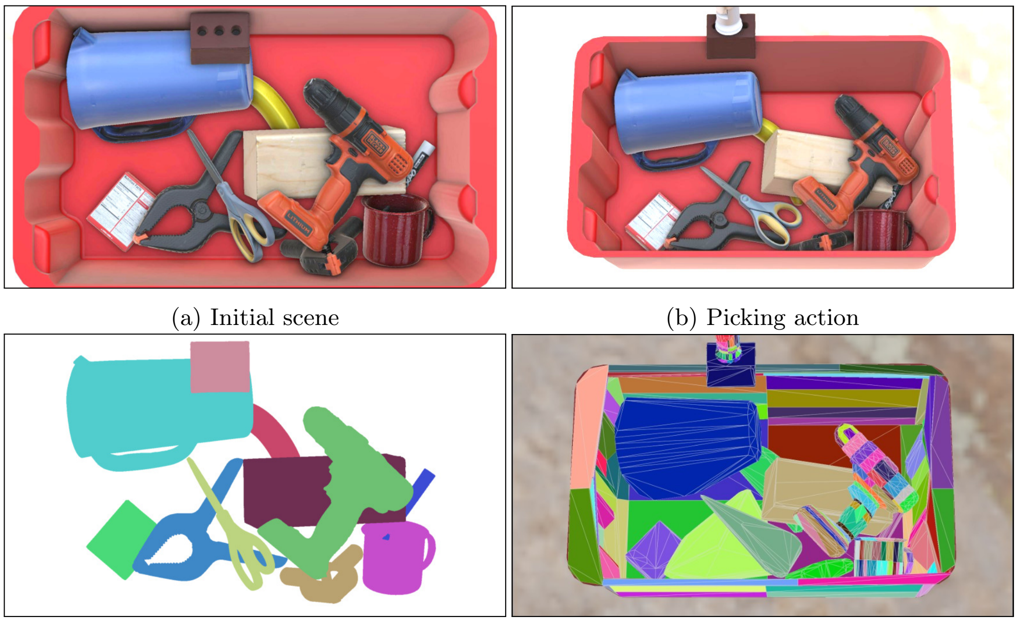
Large-scale datasets are one of the major factors driving the success of deep learning models. Creation of the ImageNet dataset (Deng et al., 2009) marked an important milestone in computer vision research. AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) kick-started the widespread adoption of deep learning in computer vision and eventually in many other domains (Popel et al., 2020; Mehrish et al., 2023; Deng and Liu, 2018). The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) version of the ImageNet with 1000 classes, referred to as ImageNet-1K, consists of 1,281,167 training images, 50,000 validation images and 100,000 test images. The complete version dataset is referred to as ImageNet-21K. It consists of 14,197,122 images with 21,841 classes. ILSVRC served as the standard benchmark for object recognition models. Over the years, several influential datasets like Microsoft Common Objects in Context (COCO) (Lin et al., 2014) targeting object detection and semantic segmentation, KITTI Vision Benchmark Dataset (Geiger, Lenz, and Urtasun, 2012) and Cityscapes (Geiger, Lenz, and Urtasun, 2012) focusing on autonomous driving, Open Images (Kuznetsova et al., 2020) with 9.2 million images and 30.1 million image-level labels for object detection and visual relationship learning, YouTube-8M with 6.1 video segments targeting video understanding were introduced. Halevy, Norvig, and Pereira (2009) in their article titled "The Unreasonable Effectiveness of Data" highlighted the effectiveness of large-scale automatically-curated low-quality annotations over manually-annotated high-quality but rather small-scale data on a variety of task. Although, this article was published before the era of deep learning, the follow-up article by Sun et al. (2017) titled "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era" noted that the phenomenon continuous in the deep learning era as well. While the exact number of human hours spent on creating these datasets is not explicitly documented in publicly available sources, creating large-scale datasets is time- and resource-consuming. Moreover, the complexity of dataset creation also depends on the nature of the annotations generated. Generating 3D bounding box and 6D pose annotations is relatively harder than 2D annotations like class labels, bounding box, or semantic and instance segmentation masks. To address these limitations, data augmentation techniques have been proposed. The data augmentation techniques are easier to implement and require minimal resource- and time-overhead during the neural network training. These techniques can be broadly classified into *data warping* and *synthetic data generation* (Wong et al., 2016; Shorten and Khoshgoftaar, 2019). Data warping techniques like color space transformations, noise injection, and planar affine transformations have been widely used since the early years of neural networks (LeCun et al., 1998; Moreno-Barea et al., 2018; Shorten and Khoshgoftaar, 2019). Synthetic data generation techniques involve overlaying object crops (see Fig. 2.2) on to random backgrounds or existing annotated images (Schwarz et al., 2018a; Takahashi, Matsubara, and Uehara, 2019; Pérez-García, Sparks, and Ourselin, 2021). Although data augmentation techniques have been applied successfully in many computer vision tasks, the main limitation of such techniques in the context of object pose estimation is the lack of photo-realism and physically-plausibleness. To address this limitation, we follow the simulation and rendering approach (depicted in Fig. 2.3) to develop SynPick, a dataset consisting of bin-picking actions simulated using Stillleben (Schwarz and Behnke, 2020). Exemplar image and ground-truth annotations are shown in Fig. 2.1.

2.2 Related Work

In this section, we briefly review the commonly used datasets for object pose estimation. See Table 2.1 for an overview.

YCB-Video

The YCB-Video dataset (Xiang et al., 2018) consists of 92 RGB-D video sequences of static scenes with 6D pose labels. Out of the 92 video sequences, 12 are used for testing and the rest are used for training and validation. Overall, 133,827 images are contained in the dataset. For each scene, a random subset of 21 objects are arranged in arbitrary pose configurations. These 21 objects are selected from the YCB objects dataset (Calli et



(a) Initial scene (b) Picking action

(c) Semantic annotation (d) Physics simulation

Figure 2.1: SynPick features typical dynamic bin picking sequences along with 6D pose & semantic segmentation annotations.

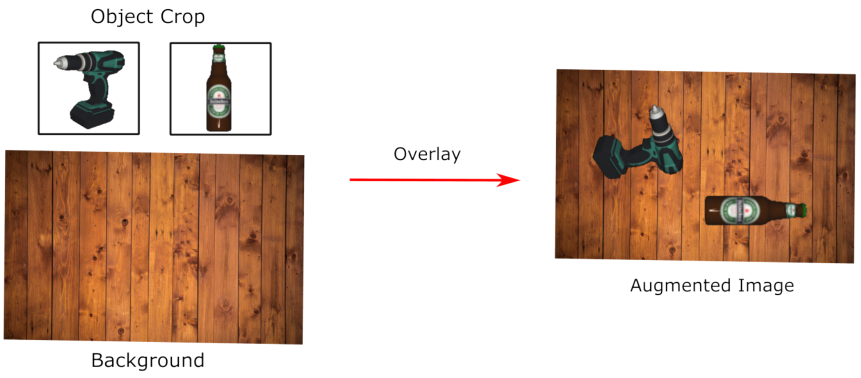


Figure 2.2: Data augmentation using object overlay. Crops containing objects are overlaid on to the background. The augmentation process is simple but the augmented images are neither photo-realistic nor physically-plausible.

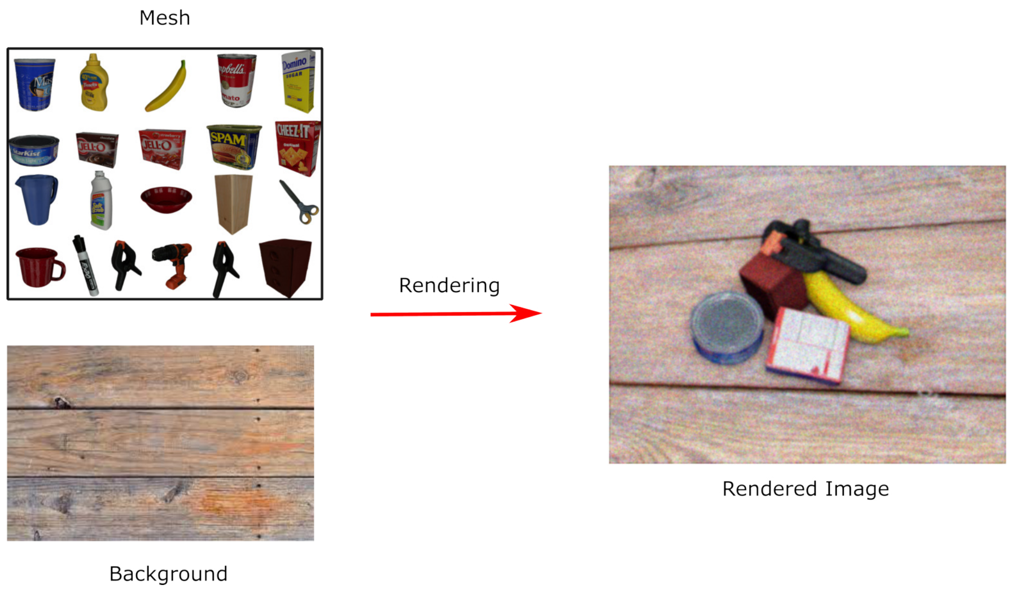


Figure 2.3: Data augmentation using simulation & rendering. Physically-plausible photo-realistic renderings are generated. The physics simulation and the rendering process is time- and resource-consuming.

al., 2015). High quality 3D meshes are also provided with the dataset. The ground-truth labels are generated using a semi-automated pipeline. The first frame a video sequence is manually annotated. These pose annotations are further refined by registering the signed distance function (SDF) representation of the objects with the depth image. For the subsequent frames, the pose from the first frame is extrapolated using the camera trajectory obtained using visual odometry techniques.

Linemod-Occluded

Hinterstoisser et al. (2013) introduced the Linemod dataset consisting of 15 RGB-D video sequences featuring 13 texture-less objects. In total, the dataset consists of 1,214 images. In each video sequence, annotations are provided for only one object. (Brachmann et al., 2014) extended the dataset with annotations for all the *eight* objects in one of the video sequences (`benchwise`). This version of the Linemod is called Linemod-Occluded. The texture-less nature of the objects make the Linemod-Occluded challenging. Since, the size of the dataset is significantly smaller in comparison to YCB-Video, for example, a supplementary synthetic version of Linemod-occluded is used for training and the real version of the dataset is often used exclusively for testing.

HomebrewedDB

HomebrewedDB (Kaskman et al., 2019) consists of 13 RGB-D video sequences of static scenes captured using a moving camera, each containing 1340 frames. Two of video sequences feature drastic lighting and objects with alternate textures. The scenes are static and 33 objects were used to construct the dataset and high-quality 3D models of the object provided with the dataset. The objects are placed on markerboard with ArUco

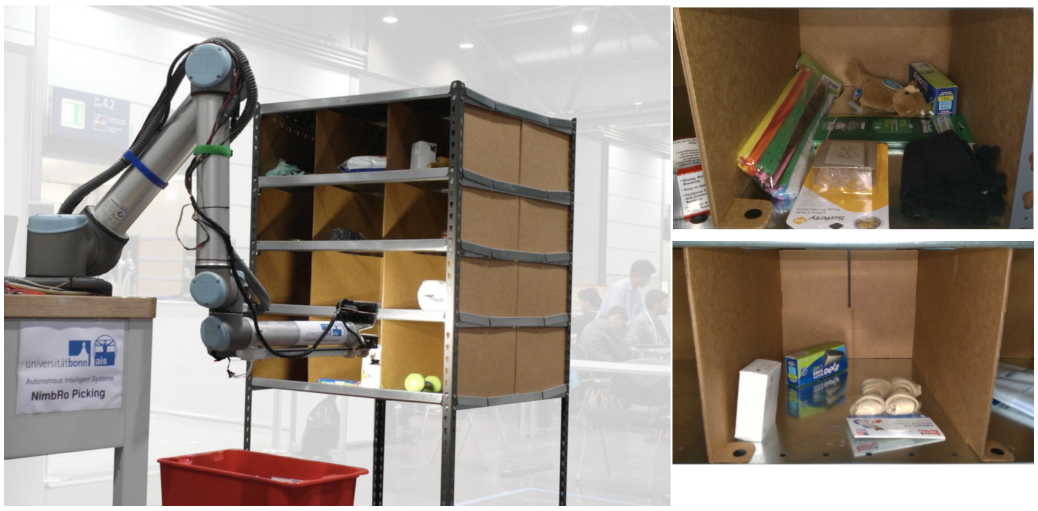


Figure 2.4: Amazon Picking Challenge (APC) setup (Schwarz et al., 2018b). Left: Amazon-Kiva Pod and the robotic arm with the RGB-D camera and the suction gripper. Right: Exemplary bin layouts.

markers to facilitate camera localization. Similar to the YCB-Video dataset, only the first frames of the video sequences are manually annotated. For the rest of the frames, the object poses are propagated using the marker-aided camera pose estimates.

Rutgers APC

In contrast to the standard pose estimation datasets which predominantly feature table-top scenes, Rennie et al. (2016) introduced the Rutgers APC dataset with objects on the Amazon-Kiva Pod (See Fig. 2.4). The scenes were captured using the RGB-D camera mounted into a Motoman SDA10F robot. Each bin is captured from three different positions. To ensure constant scene lighting, the camera was equipped with LED strips. After an initial robot-pod calibration step, forward kinematics of the robot arm assists in propagating pose annotations from the initial frame to the others. Overall, 10,368 images featuring 24 objects are provided in the dataset.

TU Dresden Light & Toyota Light

Both TU Dresden Light (TUD-L) and Toyota Light (TYO-L) (Hodan et al., 2018) datasets feature varying lighting and ambient settings TUD-L contains limited occlusion and clutter. It consists of three objects captured under eight different lighting conditions. TYO-L features 21 objects on table-top settings with four different table cloths and five different lighting conditions. TUD-L and TYO-L consists of 23,914 and 1,670 images, respectively.

Falling Things Dataset

The most related to SynPick we present in this Chapter is the Falling Things Dataset (FAT) (Tremblay, To, and Birchfield, 2018). It consists of renderings of randomly sampled subsets of YCB objects placed in different indoor and outdoor scenes. Unlike most other

Table 2.1: 6D Pose Estimation and Tracking Datasets

Name	Type	Objects	#Frames	Annotation	Diverse Lighting	Dynamics	Multi-View
YCB-Video	I	21	133,827	Semi-auto	Yes	Static	Moving cam
Linemod-Occluded	I	8	1,214	Semi-auto	No	Static	Moving cam
TUD-L	I	3	23,914	Semi-auto	Yes	Moving object	No
TYO-L	II	21	1,670	Manual	Yes	Static	No
HomebrewedDB	I	33	17,420	Semi-auto	No	Static	Moving cam
BlenderProc4BOP	IV	flexible	50,000	Automated	Yes	Static	25 views
FAT	III	21	61,500	Automated	Yes	Falling	Stereo
ObjectSynth	IV	39	600,000	Automated	Yes	Static	200 views
SynPick (ours)	III	21	503232	Automated	Yes	Pick/Move	3 views

I - real videos. II - real images.
III - synthetic videos. IV - synthetic images.

datasets, FAT does not include tabletop scenes where the objects are nicely arranged without significant occlusions. Instead, the objects are dropped from a height onto the scene, and using a physics engine the object interactions modeled. However, both the context (kitchen/temple/forest) as well as the dynamics (objects falling on the background geometry) do not really fit a bin picking application. In comparison, our dataset features standard picking tote, and pick and move dynamics.

Self-supervised Pose Estimation

An interesting orthogonal approach to supervised pose estimation is the self-supervised approach proposed by Deng et al. (2020). In this approach, a 6D pose estimation module trained in a supervised manner is used to initialize the 6D poses for objects in the scene. A robot then changes the configuration of the objects by random pick and place actions. By capturing and propagating the initial pose estimate to the objects in new configuration using forward kinematics, the authors generate new training data and refine the pose estimator actively. While this method provides a scalable approach to train pose estimators, it is limited by the simple object manipulation actions the robot can perform without breaking the object pose tracking module. Our proposed method does not suffer from this limitation and can model complex object interactions accurately. Furthermore, our data generator runs faster than real-time and can be easily parallelized.

2.3 Dataset Generation

The dataset generator is built using the Stillleben library (Schwarz and Behnke, 2020). Stillleben is a GPU-compatible fast rendering engine capable of generating photo-realistic images for training machine learning models on the fly. It also has built-in support for physics simulation through NVIDIA PhysX[1] library integration. The intended use of the Stillleben library is to generate tabletop scenes with objects in random but physically plausible pose configurations. The random pose configurations are generated by dropping objects from a fixed height onto a tabletop and simulating physical interactions

1 https://github.com/NVIDIAGameWorks/PhysX

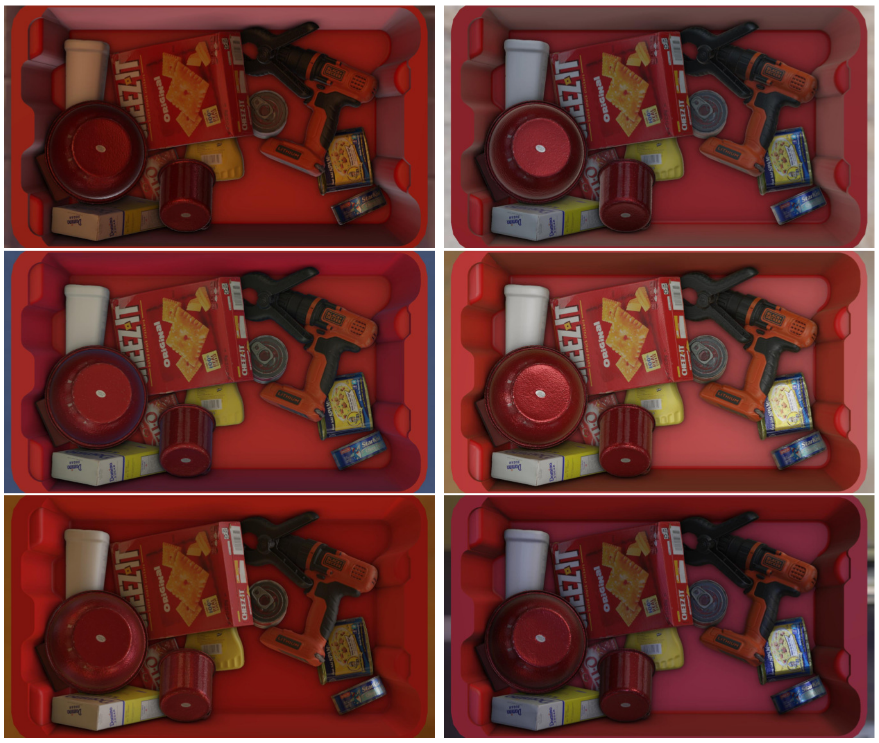


Figure 2.5: Lighting variants. We show the same object arrangement in different sIBL light maps.

between the objects until the all the objects come to rest. The resulting static scene is rendered once. In contrast, for generating the SynPick dataset, we need to render the scene continuously to simulate dynamic bin picking scenes. To this end, we make following modifications to the Stillleben library.

Arrangement & Rendering Engine

We use the high-quality 3D meshes provided by the YCB-Video dataset (Xiang et al., 2018) together with the physics-based scene arrangement engine described above with minor changes to generate a realistic-looking pile of objects in a tote. Simulating collisions with high-quality meshes is time consuming. However, convex hull approximations for the mesh geometry enables faster simulation without physical-plausibleness of the simulation. For the objects that are highly non-convex `cup`, `bowl`, and `power drill`, we compute the smallest convex meshes describing the concave geometry accurately using the V-HACD convex decomposition algorithm (Mamou, Lengyel, and Peters, 2016). We randomly sample a set of objects such that the total volume does not exceed a threshold of 71. We drop from the objects from a height into the tote and simulate effect of gravity and inter-object collisions. After the initial scene arrangement step, we simulate different pin picking scenarios and render the scene from three different view points. The physics simulation is run with a very small step size of 2 ms to ensure realistic behavior. The renderings are done at Full HD resolution (1920×1080) with a rate of 15 Hz (relative to simulated time). To generate varying, yet realistic lighting conditions (see Fig. 2.5), we use image-based lighting (IBL) maps from the sIBL archive[2].

2 http://www.hdrlabs.com/sibl/archive.html

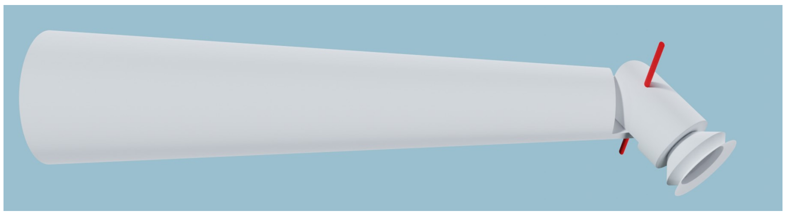


Figure 2.6: Articulated gripper. The cone-shaped finger ends in an actuated tiltable suction cup (joint axis marked in red).

Suction Simulation

We use an articulated gripper with a suction end effector with a cone-shaped finger shown in Fig. 2.6 in our simulations. The finger ends in an actuated tiltable suction cup. In real world, a successful pick using a suction gripper depends on how well the suction cup seals onto the objects. When the suction cup seals onto the object surface tightly, the objects will remain attached to gripper even under mild perturbation. In contrast, when the seal is partial, the object might fall down due to gravitational influence or collisions. We simulate the degree of sealing using raycasting. We cast ten rays from the gripper perimeter in the direction of the gripper. Every object hit by the raycast (with a maximum distance of 3 cm) is considered caught. For every caught object, a PhysX joint is created between it and the suction cup, simulating a strong force pulling the object against the gripper as well as limiting its orientation relative to the gripper. The joints have a force limit of 40 N or 2 N , depending on whether all rays found a target (indicating a good vacuum seal). If the force required to keep the object at the suction cup exceeds this limit, the joint breaks and the object is dropped back into the tote.

Bin Picking Scenarios

As a part of SynPick, we simulate three common real-world picking scenarios: `untargeted pick`, `targeted pick`, and `move`.

Untargeted Pick

In the untargeted pick scenario, the objects are picked one at a time in any order with the target of emptying the tote and stowing the all objects into a different target tote. At any point during the untargeted picking process, the object with the most successful pick probability is attempted. The object with the least occlusion often has the highest probability of a being successfully picked. Starting from the RGB image, we follow the planning and grasp heuristics selection strategy proposed by Schwarz et al. (2018a). We use ground-truth segmentation masks rendered by Stillleben directly in place of a semantic segmentation module. After extracting object contours, ideal suction points are found inside the contour. Depending on object weight, either the pole of inaccessibility (Garcia-Castellanos and Lombardo, 2007), i.e. the point with maximum distance to the contour is found to minimize the chance of catching other objects, or the center of mass is computed from the contour to ensure good mass distribution. To determine the object with the highest probability of being picked, we utilize the clutter graph, which represents the hierarchy of the objects' positioning relative to the other objects; the object correspond-

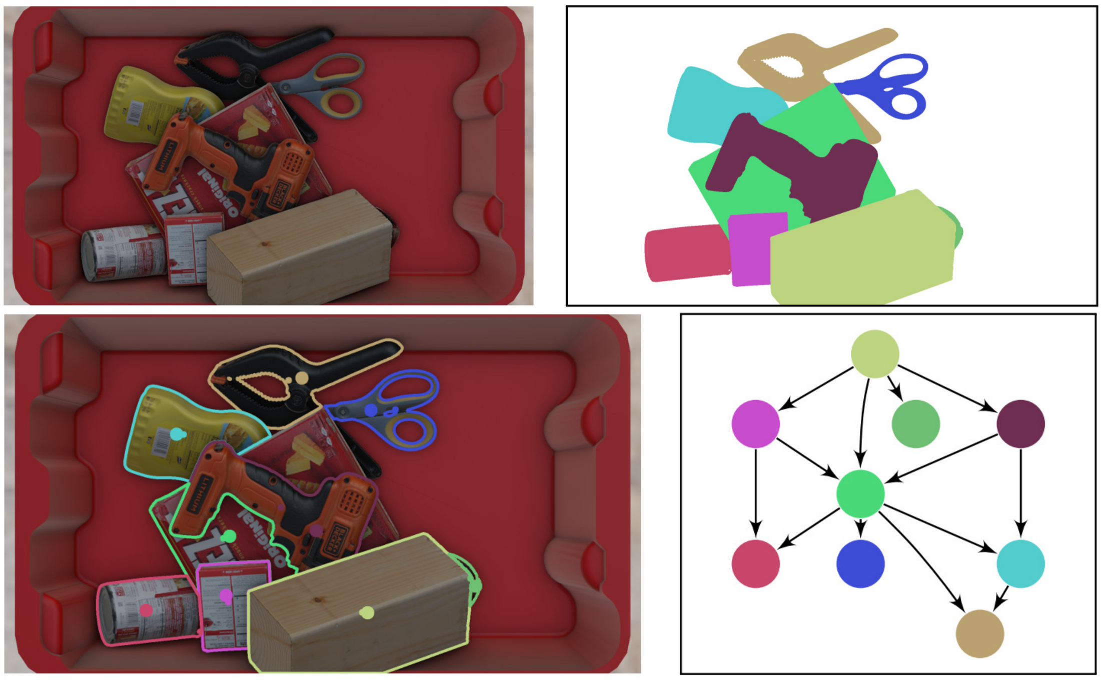


Figure 2.7: Picking heuristic. Given an RGB image and the ground-truth segmentation, we follow Schwarz et al. (2018a) to generate object contours with suction grasp points, as well as a clutter graph describing the scene layout. Each edge in the graph flows from the object on top to the object below.

ing the node with the least incoming edges is the least occluded. The complete pipeline is shown in Fig. 2.7.

Once the target object and the corresponding suction point are determined, we find an inverse kinematics solution for the articulated gripper, which places the suction cup orthogonally on the local surface, as estimated by a local average of the pixel-wise normals. A gripper trajectory is then computed to bring the gripper to the target pose, apply suction, and lift the object out of the tote. The gripper is moved along the trajectory using Cartesian impedance control, with a stiffness of 2500 $\frac{N}{m}$ and a spring damping of 200 $\frac{N\,s}{m}$. The force exerted by the impedance control is limited to $200N$. The high stiffness simulates an industrial robotic arm holding the gripper in place. The picking process is repeated until the tote is empty. If three picking attempts have failed, we also stop the sequence to prevent infinite loops. An example untargeted pick sequence is shown in Fig. 2.8 (a).

Targeted Pick

In the second scenario, we simulate targeted picking, where a specific object needs to be extracted. We run the same pipeline as above, with one key difference: We choose the object which is occluded most, i.e. is at the bottom of the clutter graph. This choice should lead to more complex object interactions during picking. Unsurprisingly, the success rate of the targeted pick action is low compared to that of the untargeted pick.

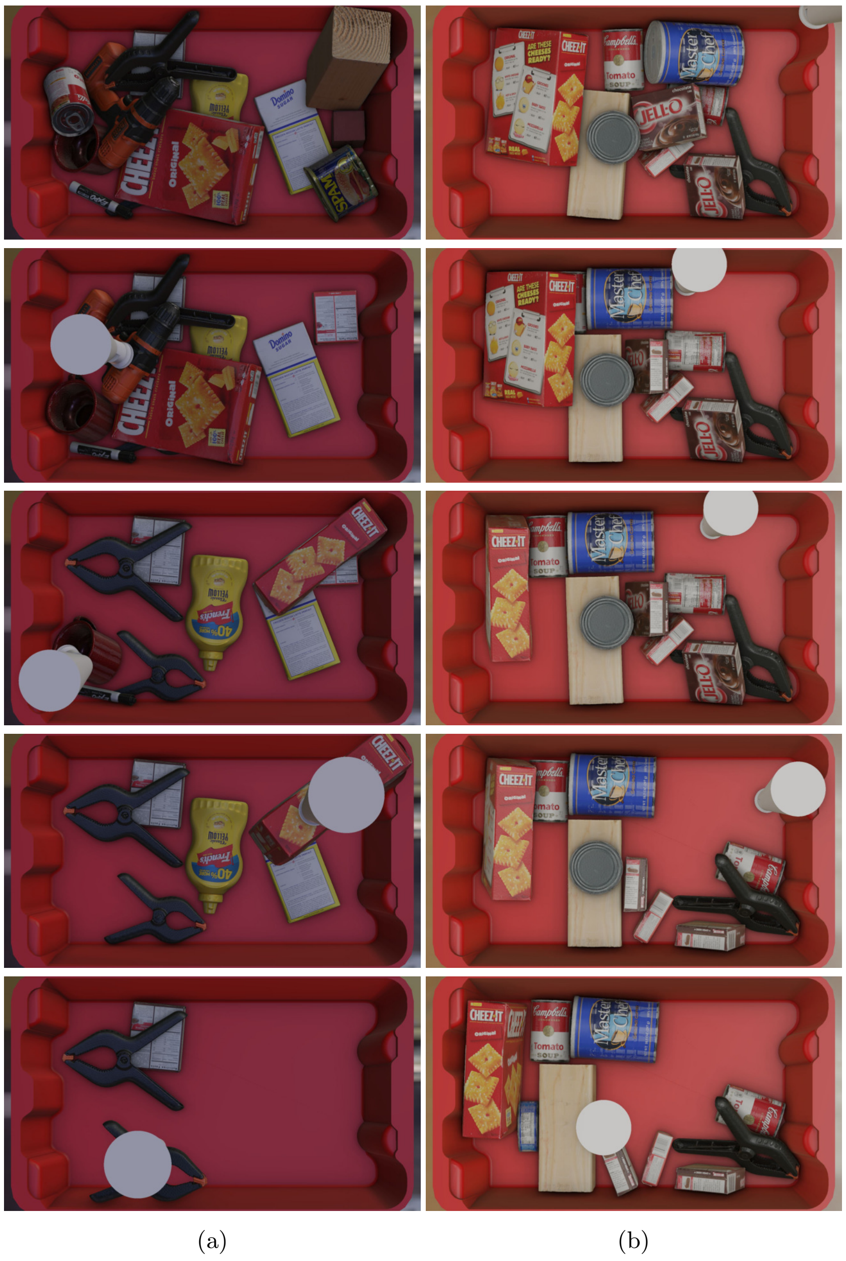


Figure 2.8: Exemplary scenes from the dataset demonstrating the evolution of the scene while the gripper is performing picking (a) and moving (b) actions. Objects in the tote are covered/uncovered as the scene evolves.

Table 2.2: SynPick statistics

Mode	Split	Frames	Object visibility
Untargeted pick	train	137,544	0.77
Move	train	99,786	0.67
Targeted pick	train	164,991	0.77
Untargeted pick	test	31,119	0.77
Move	test	23,910	0.69
Targeted pick	test	45,882	0.75
Total		503232	0.74

Number of frames in each SynPick split along with the corresponding mean visibility fraction.

Move

As a third possible scenario, we perform a non-picking manipulation sequence. The goal is to disturb the object arrangement so that other occluded objects become visible. We simulate the move action by moving the gripper, starting from one corner inside the tote, to all four corners in random order. The gripper is moved at a fixed velocity of 0.1 $\frac{m}{s}$. In this mode, the gripper is operated with a lower stiffness of 1000 $\frac{N}{m}$ and a force limit of 30 N . This ensures we do not squeeze objects against the immovable tote too much, which could result in instability of the physics simulation. An exemplary scene for the move action is shown in Fig. 2.8 (b).

2.4 Dataset Statistics

Number of Frames

For each of the actions simulated, SynPick provides 300 video sequences; 240 sequences are used for training and 60 sequences are used for testing. Each sequence has varying number of frames. In Table 2.2, we present the total number of images per simulated action. Overall, SynPick consists of 503,232 images. Compared to most commonly used object perception datasets, SynPick provides significantly large number of annotated images, and covers diverse lighting and object interactions. See Table 2.1 for a detailed comparison.

Mean Visibility Fraction

Like in any computer vision task, occlusions present a significant hindrance for 6D pose estimation. A prerequisite for training robust 6D pose estimation models is a dataset that captures real-world occlusion scenarios. The quality of the annotations generated by semi-automated pipelines often drops significantly in highly-cluttered environments. Thus, most of the standard object perception datasets do not feature high degree of occlusion. Physically realistic simulation of dynamic bin picking scenes captures more realistic occlusion scenarios that are not captured in a static scene. To analyze the degree of occlusions present in the SynPick dataset, we present in Table 2.2, the mean visibility fraction for the different SynPick splits.

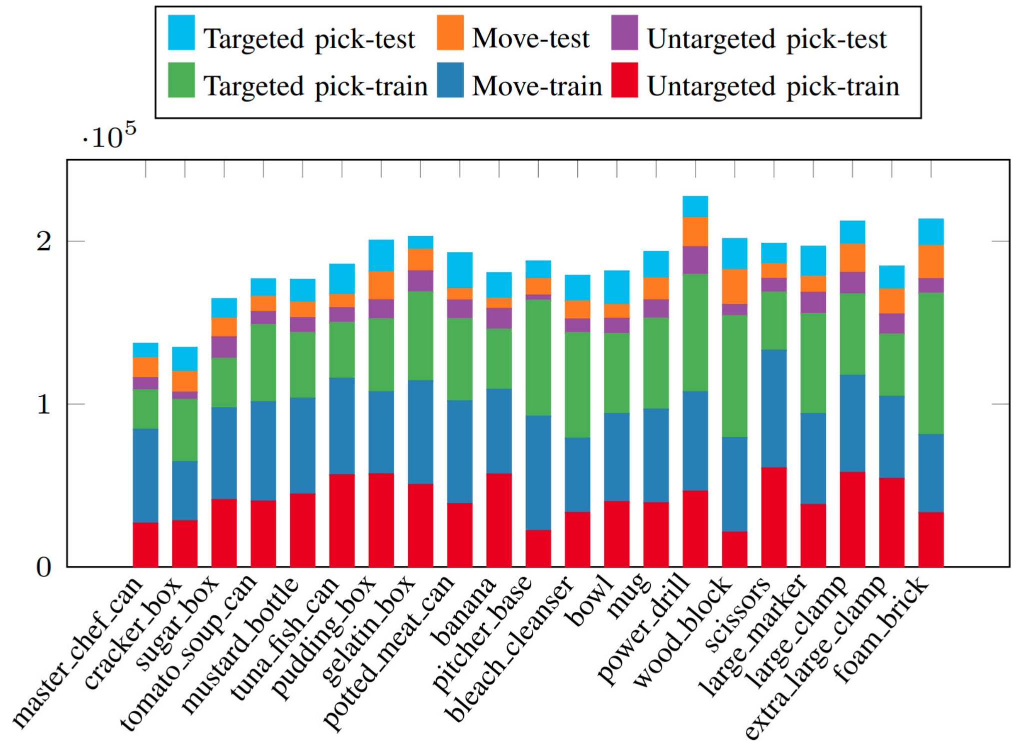


Figure 2.9: Number of 6D pose annotations for each object category present in SynPick dataset splits.

Object Distribution

In Fig. 2.9, we present the distribution of object categories across various splits of the SynPick dataset. The random object selection scheme discussed in Section 2.3 selects objects at random until the total volume exceeds a threshold. This ensures an even distribution of the objects across the scenes. However, two of the biggest objects in the dataset, `master chef can` and `cracker box` are featured less compared to other objects. These two objects exhibit simple geometry with unambiguous texture. Thus, they are easy to learn and the pose estimation methods perform well on these objects. Therefore, we do not explicitly force these objects to appear equally often as the other objects.

2.5 Motivation for Temporal Integration

We present a simple motivating case for temporal integration in object pose estimation methods based on the CosyPose (Labbe et al., 2020) model. This experiment is intended to demonstrate the low-hanging fruit which can be reached by temporal filtering. We implement an Exponentially Moving Average (EMA) with recursive filter coefficient α:

$$\hat{t}_n = (1-\alpha)\hat{t}_{n-1} + \alpha t_n, \tag{2.1}$$

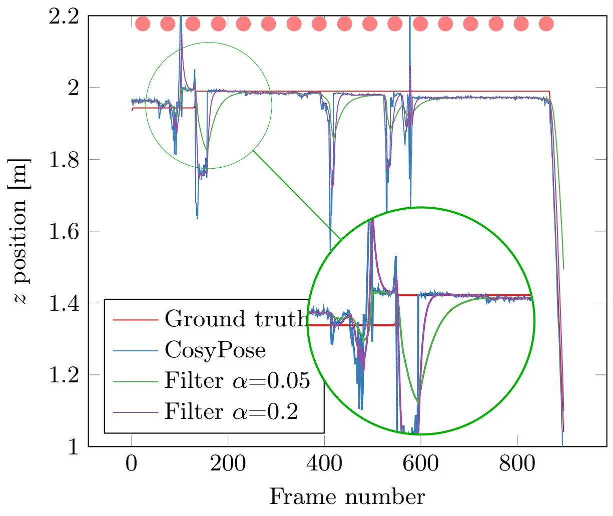


Figure 2.10: Position trajectory (z axis) of the `scissors` object in the first test scene. The raw CosyPose (Labbe et al., 2020) predictions, an exponentially moving average with different recursive filter coefficients α, and the ground truth are shown. The time of each pick attempt has been marked with a red circle.

where t_n is the CosyPose translation estimate at frame n of the sequence and $\hat{t}_n$ is the filtered output. The object orientations are filtered very similarly, but in the quaternion space in order to interpolate the orientations correctly:

$$\hat{q}_n = \text{slerp}(\hat{q}_{n-1}, q_n, \alpha), \tag{2.2}$$

where slerp is the spherical linear interpolation function, which interpolates with $0 < \alpha < 1$ between the two given rotations.

Figure 2.10 shows a sequence of raw CosyPose translation predictions in the z axis (into the image) for one exemplary object. It can be seen that CosyPose exhibits both stationary noise as well as large deviations, which are mostly caused by temporary occlusions—either by the gripper or other objects. While our naive filtering cannot address steady-state errors, it does smoothen the stationary noise and softens the large jumps caused by occlusions. The single-view 6D pose estimation methods are useful, but does not really capture a real bin picking situation. In our picking scenario, which is typical for industrial applications, a tote of objects is emptied completely, object by object. It is certainly beneficial to monitor the object poses over time, to make use of dependencies between frames. This way, not only temporary effects such as occlusions by the gripper or other objects can be mitigated, but noisy predictions can also be smoothed to obtain a more precise estimate than from a single frame alone.

2.6 Discussion & Conclusion

In this Chapter, we presented SynPick, a photo- and physically-realistic dataset of dynamic bin picking scenes. We built the SynPick generator on top of the openly available Stillleben renderer and simulated three commonly occurring pin picking actions: **`untargeted pick`**, **`targeted pick`**, and **`move`**. For physics simulation, we utilize the PhysX library from Nvidia. Our dataset features the same 21 objects from the YCB-Video dataset (Xiang et al., 2018). Overall, the dataset consists of more than half a million images with pose annotations. The robotic arm used in the dataset generation consists of an articulated suction gripper. We model the suction characteristics of the gripper using ray casting. This enables simulating complex scenarios where objects are dropped due to collision while moving them out of the tote, even though the pick action itself was successful. We make the dataset, as well as the generator, public. We hope that the availability of SynPick will encourage the research community to explore dynamic scene perception algorithms. Furthermore, we presented a motivating case for incorporating temporal information into the pose estimation models. Using a simple recursive filtering, approach we demonstrate the improvement in the accuracy of a leading pose estimation model on a SynPick sequence. In Chapter 7, we use SynPick to train and evaluate temporal information fusion in the multi-object pose estimation model we present in Chapter 6.

3 Implicit Probability Distribution Functions for Object Pose Estimation

The standard object pose estimation models output a single 6D pose. Thus, they lack the ability to reason about symmetries. The parametric distributions used for modeling symmetry are often computationally intractable and not suitable for learning-based approaches. In this Chapter, we present an efficient alternative approach that models symmetries implicitly using neural networks. Moreover, we present a render-and-compare based automatic pseudo ground-truth generation pipeline to train the implicit probability distribution network without explicit symmetry labels.

Statement of Personal Contribution

The contents presented in this Chapter is adapted from the following publication.

Arul Selvam Periyasamy*, Luis Denninger*, and Sven Behnke:

Learning implicit probability distribution functions for symmetric orientation estimation from RGB images without pose labels

In: *6th IEEE International Conference on Robotic Computing (IRC)*, Naples, Italy, 2022.

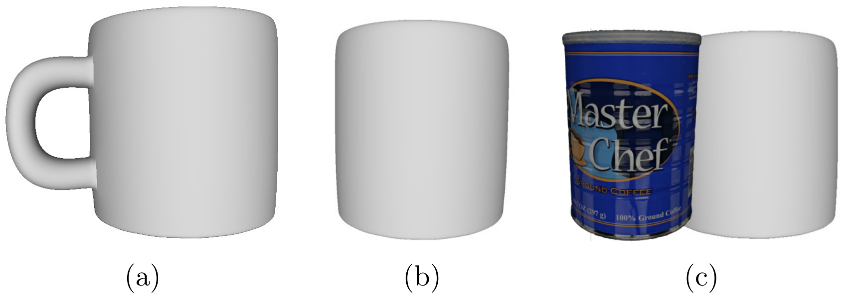
The author of this thesis substantially contributed to all aspects of the publication, including the literature survey, the conception, formalization, design, and implementation of the proposed method, the creation of the Tabletop dataset used for evaluating the proposed method, the preparation and conduct of experiments, the analysis and interpretation of the experimental results, the preparation of the manuscript, as well as the revision and final editing of the version published.

3.1 Introduction

Objects in our daily lives and industrial setups exhibit symmetries. Symmetries refer to the characteristics of an object that make it appear unchanged when subjected to certain transformations, such as reflections, rotations, or translations.

Vetter, Poggio, and Bülthoff (1994) studied the ability of human visual system to exploit symmetry to recognize objects in novel orientations, even when they have previously seen only a relatively small number of different views. In their experiment, 14 human subjects were presented with single training views of 32 symmetric and 32 non-symmetric objects. The participants were able to recognize symmetric objects with higher accura-

Figure 3.1: Pose uncertainty due to occlusion. (a) `Coffee mug` does not exhibit pose ambiguity when the handle is visible. However, due to self-occlusion (b) or inter-object occlusion (c), the pose is ambiguous



cies than the non-symmetric objects from novel test views. Thus, understanding and exploiting symmetry forms a core attribute of the human visual system.

However, in the context of computational object pose estimation, symmetries introduce multiple challenges including:

1. *Ambiguity in pose representation.* Due to symmetry, one image observation might correspond to multiple poses. The standard neural network training schemes to estimate a single pose employing discriminative loss functions are no longer applicable in the presence of symmetry.

2. *Limited discriminative features.* Symmetrical parts of an object may share similar visual features, making it difficult for algorithms to distinguish between different poses. This lack of discriminative features can affect the accuracy and robustness of pose estimation methods.

3. *Computational complexity.* Training model to understand the symmetries needs more training data and training time than for the non-symmetric objects.

4. *Limited generalization.* Models trained on symmetric objects may have difficulty generalizing to objects with different symmetries or non-symmetric objects. This lack of generalization can be a significant limitation in real-world scenarios where objects exhibit diverse symmetries.

In addition to symmetries, uncertainties in object pose can also occur due to occlusion (See Fig. 3.1). Modeling such uncertainties is vital for many autonomous systems.

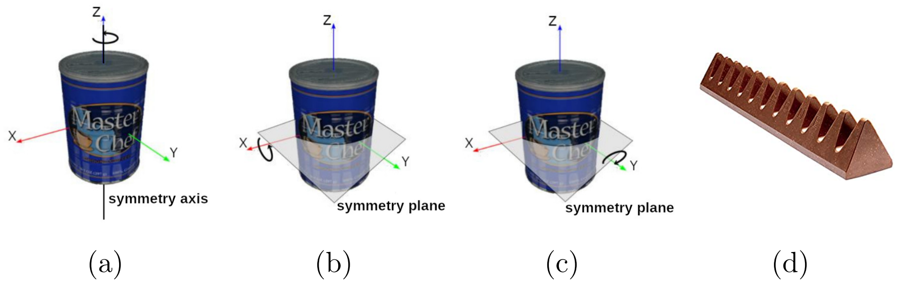
Symmetries can be classified into visual symmetries and geometric symmetries. Visual symmetries, as defined in Eq. (3.1), arise due to the lack of distinctive visual features, whereas even in the presence of symmetry-breaking visual features, objects may exhibit symmetries in terms of their geometry.

Formally, ambiguities occur when an object $\mathcal{O}$ appears similar under at least two different poses $\mathbf{P}_i$ and $\mathbf{P}_j$, i.e., we obtain the same image $\mathcal{I}$ when object $\mathcal{O}$ is in pose $\mathbf{P}_i$ and $\mathbf{P}_j$:

$$\mathcal{I}(\mathcal{O}, \mathbf{P}_i) = \mathcal{I}(\mathcal{O}, \mathbf{P}_j). \tag{3.1}$$

Knowing the object axes and type of symmetries, we can describe symmetries—despite the presence of textures—with respect to the object's geometry as discrete or continuous

Figure 3.2: Objects exhibiting geometric symmetries. (a-c): *Can* object symmetries. Due to the presence of visual-symmetry-breaking texture, the *can* object exhibits only geometric symmetries, namely a continuous rotational symmetry along the z axis and a discrete flip symmetry along the x and y axes. (d): Example of an object exhibiting translation symmetry under partial occlusion.



rotations around the object axes, as shown in Fig. 3.2 (a-c). Moreover, objects with repetitive geometric structures (shown in Fig. 3.2 (d)) exhibit translational symmetry under partial occlusion. Formally, an object $\mathcal{O}$ consisting of n 3D points $\mathbf{x}$ can be considered to exhibit geometric symmetries when there are at least two poses $\mathbf{P}_i$ and $\mathbf{P}_j$ that have a small mean closest point distance:

$$\frac{1}{n}\sum_{\mathbf{x}_1\in\mathcal{O}}\min_{\mathbf{x}_2\in\mathcal{O}}||\mathbf{P}_i\mathbf{x}_1-\mathbf{P}_j\mathbf{x}_2||\approx 0. \tag{3.2}$$

Enumerating all sources of symmetries is not tractable, making an approach as shown in Fig. 3.2 to describe arbitrary object symmetries not scalable.

Following the terminology introduced by Brégier et al. (2018), we define *proper symmetries* $\mathcal{M}$ as the group of poses that exhibit geometric symmetries:

$$\mathcal{M}=\{\mathbf{m}\in\mathbf{SE}(3)\ \ \forall\mathbf{P}\in\mathbf{SE}(3),$$
$$\frac{1}{n}\sum_{\mathbf{x}_1\in\mathcal{O}}\min_{\mathbf{x}_2\in\mathcal{O}}||\mathbf{P}_i\mathbf{x}_1-\mathbf{m}\cdot\mathbf{P}_j\mathbf{x}_2||\approx 0\}, \tag{3.3}$$

with $\mathbf{m}$ being the transformations rotating the object around its symmetry axes in discrete steps.

The methods for pose estimation for symmetric objects can be classified into two families. The first family of methods estimates a single valid pose $\mathbf{P}\in\mathbf{SE}(3)$ corresponding to the RGB-(D) image. One major advantage of this approach is that the model architectures remain the same for symmetric and non-symmetric objects. In both cases, given an RGB-(D) image, the network generates a single 6D pose prediction. The only difference lies in the loss function used to train the model. Thus, in terms of the neural network architecture, training scheme, and inference, both symmetric and non-symmetric objects are treated the same.

The second family of methods estimate the complete set of *proper symmetries* $\mathcal{M}$. Modeling symmetries explicitly provides benefits that are twofold: i) facilitate models in learning better visual representations and ii) better integration with downstream tasks.

In this Chapter, we build on the implicit probability distribution ImplicitPDF models for rotation manifolds introduced by Murphy et al. (2021). Provided with an RGB image and a pose hypothesis, we train a CNN to estimate the likelihood of the pose hypothesis given the RGB image. The novelty of our approach is that we do not need explicit ground-truth pose annotations to train the ImplicitPDF CNN model, eliminating the bottleneck of acquiring large-scale ground-truth annotated dataset.

Given RGB-D images of symmetric objects without pose annotations, we propose an automatic pose labeling scheme and train the ImplicitPDF model using the generated pseudo ground-truth pose labels. The model trained with the pseudo ground-truth labels is able to express the complete set of *proper symmetries* $\mathcal{M}$ without prior knowledge about object symmetries.

3.2 Related Work

The study of symmetries has a rich history that spans multiple disciplines, including mathematics, physics, art, and philosophy.

An entire line of work were dedicated to detecting rotational symmetries from images (Cho and Lee, 2009; Flynn, 1994; Labonte, Shapira, and Cohen, 1993; Wang and Huang, 2017; Thrun and Wegbreit, 2005). These approaches identify similar local patches using handcrafted features, which are then grouped to predict the rotational symmetry orders along different axes. While these methods work well in detecting symmetries, their usability in pose estimation is rather limited.

Saxena, Driemeyer, and Ng (2009) proposed invariant representations for specific symmetry classes and trained a probabilistic Gaussian model to predict object orientation from image inputs. Compared to such an approach, ImplicitPDFs allow us to model symmetry of any arbitrary kind.

In the deep learning era, Wohlhart and Lepetit (2015) introduced the first metric learning-based approach to deal with object symmetries. They learned symmetry-aware descriptors using a triplet loss that minimizes/maximizes the Euclidean distance between similar/dissimilar object orientations. Their training approach needed explicit symmetry labels. During inference, the pose of the target object is determined using the nearest neighbor search in the descriptor space.

Many of the state-of-the-art methods for predicting a single 6D pose are trained using ShapeMatch-Loss (Xiang et al., 2018) for symmetric objects (Hodaň et al., 2020; Labbe et al., 2020; Xu et al., 2022; Amini, Periyasamy, and Behnke, 2022). Employing ShapeMatch-Loss implicitly selects one pose closest to the current pose prediction from the *proper symmetry* set as ground-truth. While ShapeMatch-Loss does not need explicit definition of symmetry, during training, the ground-truth pose depends on the current pose prediction and this variability in ground-truth pose might hamper the models' learning ability. In contrast, Pitteri et al. (2019) and Periyasamy, Schwarz, and Behnke (2018) mapped the symmetrical rotations to a single "canonical" rotation. One disadvantage of these methods is the requirement of explicit definition of object symmetry. Moreover, modeling pose estimators as single-pose predictors is neither efficient nor informative.

In addition to the single pose estimator, Rad and Lepetit (2017) added an auxiliary task to classify the type of symmetry an object exhibits. The authors argued that the auxiliary task helped the model to learn additional properties of the object's symmetry, which, in turn, benefits 6D pose estimation. While the auxiliary task helped improving the single pose prediction accuracy, the formulation of the auxiliary task as a classification task limits its scope in modeling *proper symmetries*. Gilitschenski et al. (2019) modeled multiple pose hypotheses as Bingham distributions and trained a CNN model to estimate the distribution parameters given an RGB-D input.

Corona, Kundu, and Fidler (2018) sidestepped the problem of estimating 6D poses and modeled pose estimation as a task of image comparison. They trained a model to estimate a similarity score of two RGB images and used the similarity score to select the image from a codebook of images that best matches the test image during inference. The pose corresponding to the matched image is considered as the pose of the target object in the test image. In the case of symmetric objects, multiple images from the codebook have a high similarity score. Since the inference involves comparing against a large-size codebook of images, the inference time requirement is high. Sundermeyer et al. (2020) addressed this issue by using *augmented autoencoder* AAE, a variant of *denoising autoencoder* DAE, to learn a low-dimensional latent space representation for images and uses the latent space for image comparison. Despite the speed-up achieved in the codebook comparison, discretization of $\mathbf{SO}(3)$ is still needed to make the model real-time capable. Manhardt et al. (2019) trained their model to generate a set of predictions given an RGB image with an intent to cover multiple possible poses.

3.3 Implicit Probability Distribution for 6D Pose Estimation

Inspired by the success of *neural fields* in modeling shape, scene, and rotation manifold (Xie et al., 2022; Mescheder et al., 2019; Mildenhall et al., 2021; Murphy et al., 2021), we model the possible 6D object poses as a conditional probability distribution of the likelihood $\mathcal{P}(\mathbf{P} \mid \mathcal{I})$ of the pose hypothesis $\mathbf{P}$ given an image $\mathcal{I}$ implicitly using a neural network. To this end, we train a neural network $\mathcal{F}$ to predict the unnormalized joint log probability $\mathcal{F}(\mathbf{P}, \mathcal{I})$ of the pose hypothesis $\mathbf{P}$ and image $\mathcal{I}$ as shown in Fig. 3.3. Let α be the normalization constant such that

$$\mathcal{P}(\mathbf{P}, \mathcal{I}) = \alpha \exp(\mathcal{F}(\mathbf{P}, \mathcal{I})). \tag{3.4}$$

Using the product rule,

$$\mathcal{P}(\mathbf{P} \mid \mathcal{I}) = \frac{\mathcal{P}(\mathbf{P}, \mathcal{I})}{\mathcal{P}(I)}, \tag{3.5}$$

where

$$\mathcal{P}(I) = \int_{\mathbf{P} \in \mathbf{SE}(3)} \mathcal{P}(\mathbf{P}, \mathcal{I}) d\mathcal{I}. \tag{3.6}$$

We decouple the 6D object pose manifold $\mathbf{P} \in \mathbf{SE}(3)$ into separate translation $\mathbf{t} \in \mathbb{R}^3$ and rotation $\mathbf{R} \in \mathbf{SO}(3)$ manifolds:

$$\mathcal{P}(\mathbf{P}, \mathcal{I}) = \mathcal{P}(\mathbf{R}, \mathcal{I})\mathcal{P}(\mathbf{t}, \mathcal{I}). \tag{3.7}$$

For simplicity, we consider only the object orientation $\mathbf{R} \in \mathbf{SO}(3)$ instead of the 6D pose $\in \mathbf{SE}(3)$. To make computing marginal probabilities tractable, we replace the contin-

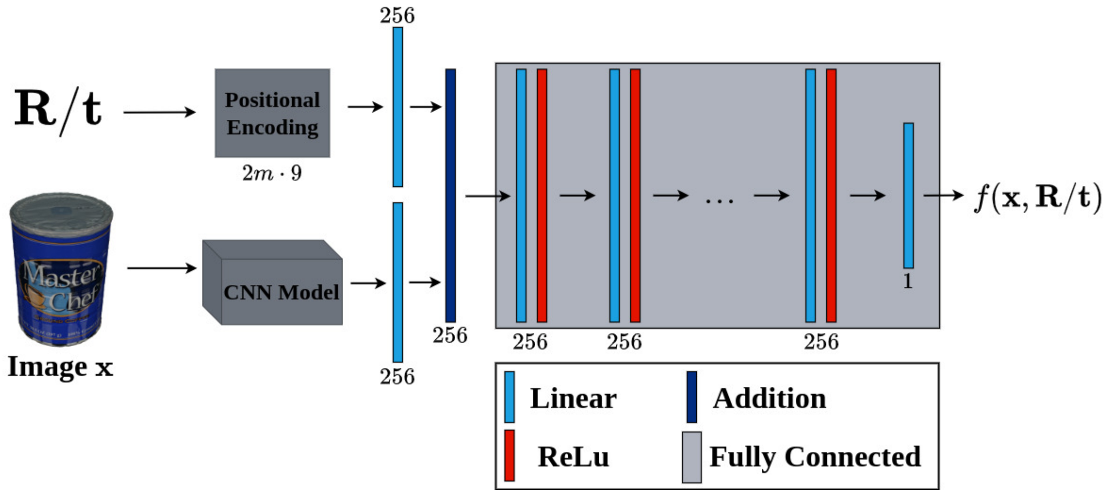


Figure 3.3: Learning ImplicitPDF. Given an image RGB $\mathbf{I}$ and an orientation hypothesis $\mathbf{R}$ or a translation hypothesis $\mathbf{t}$, the CNN model is trained to generate the unnormalized joint log probability of the hypothesis and the image.

uous integral in Eq. (3.6) with a discrete summation over a equivolumetric partitioning of $\mathbf{SO}(3)$ with N partitions of volume $V = {}^{\pi^2}/_{N}$, and cancel out the normalization constant α:

$$\mathcal{P}(\mathbf{R}\,|\,\mathcal{I}) = \frac{1}{V}\frac{\exp(\mathcal{F}(\mathbf{R},\mathcal{I}))}{\sum_i^N \exp(\mathcal{F}(\mathbf{R}_i,\mathcal{I}))}. \tag{3.8}$$

For a detailed derivation of Eq. (3.8), we refer to (Murphy et al., 2021). Note that Eq. (3.8) can be adapted to object translation $\mathbf{t} \in \mathbb{R}^3$ with $V = {}^{1}/_{N}$.

3.3.1 Training

We train our model to minimize the negative log-likelihood NLL of the ground-truth pose $\mathbf{P}_{GT}$:

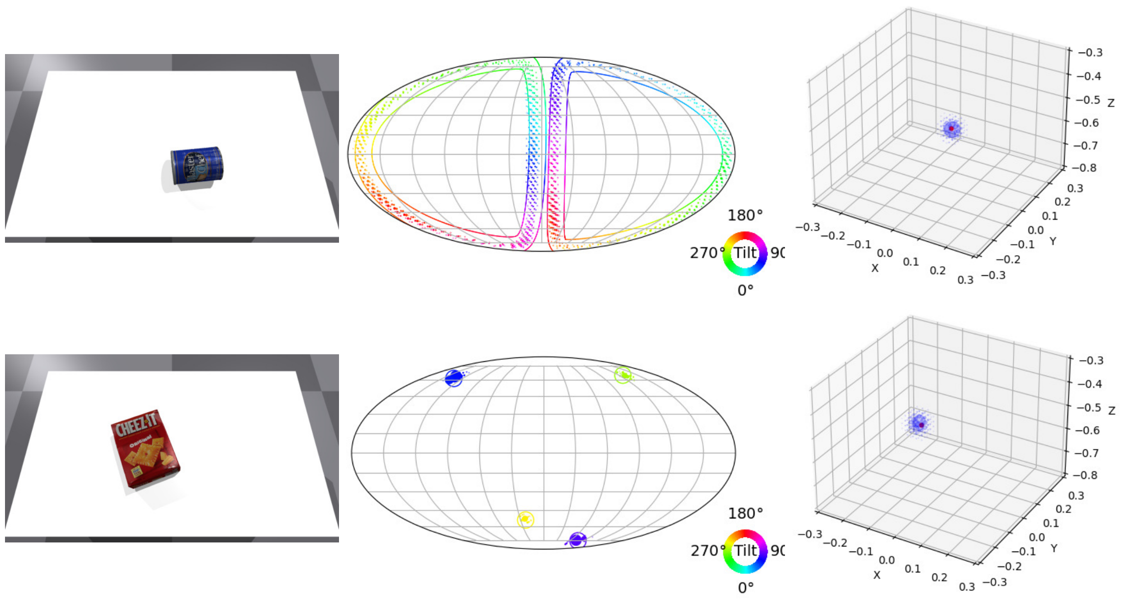
$$\mathcal{L}(\mathcal{I}, \mathbf{P}_{GT}) = -\log(\mathcal{P}(\mathbf{P}_{GT}|\mathcal{I})). \tag{3.9}$$

Following Eq. (3.8), we approximate the computation of the distribution $\mathcal{P}(\mathbf{R}_i|\mathcal{I})$ using $\mathbf{R}_i \in \{\mathbf{R}^0\}$, an equivolumetric grid covering $\mathbf{SO}(3)$ as in Section 3.3.3. The orientation hypothesis $\mathbf{R}$ and the translation hypothesis $\mathbf{t}$ given to the model as input is represented using *positional encoding* (Vaswani et al., 2017).

3.3.2 Inference

During inference, given an image $\mathcal{I}$, we predict the (single) most plausible pose $\mathbf{P}^*_I$ using gradient ascent starting from a set of initial hypotheses $\{\mathbf{P}^0\}$:

$$\mathbf{P}^*_I = \underset{\mathbf{P}\in\mathbf{SE}(3)}{\arg\max}\,\mathcal{F}(\mathbf{P},\mathcal{I}). \tag{3.10}$$



(a) (b) (c)

Figure 3.4: Visualization of the ground-truth, the orientations (column b) and the translations (column c) predicted by the ImplicitPDF model for `can` and `box` objects. Given an RGB image and an orientation hypothesis or a translation hypothesis, the ImplicitPDF model estimates the likelihood. The continuous lines and circles represent the ground-truth symmetries and the dots represent the hypotheses with a high estimated likelihood. The third degree of freedom is represented using a point on the color wheel. In column c, the translation component $\mathbf{t} \in \mathbb{R}^3$ is visualized using the 3D grid. The red circle represents the ground-truth translation.

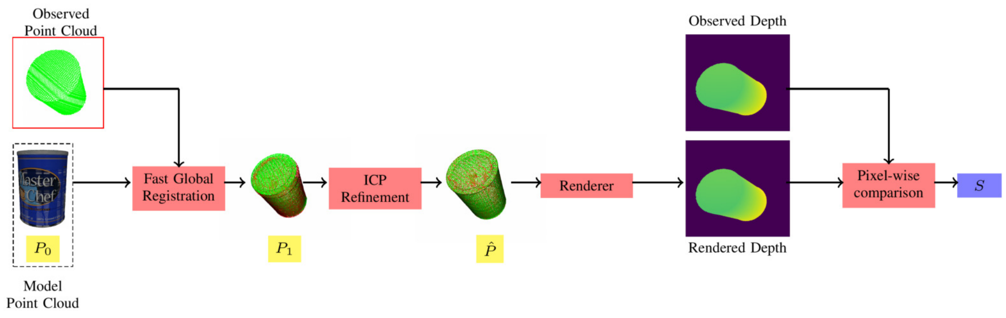


Figure 3.5: Pseudo ground-truth generation process. Given an RGB-D image without pose annotation, we register the model point cloud in a random pose P_0 against the observed point cloud using the *fast global registration* algorithm (Zhou, Park, and Koltun, 2016) to generate hypothesis P_1, which is further refined using the ICP algorithm utilizing the depth information to generate $\hat{P}$. We then render the model according to $\hat{P}$ to generate the depth image. The rendered depth image is compared pixel-wise with the observed depth image to compute the comparison score S. For a given RGB-D image we run this process multiple times and select the $\hat{P}$ with the smallest S.

Additionally, to generate the full orientation and translation distributions, we evaluate $\mathcal{P}(\mathbf{R}j \,|\, \mathcal{I})$:$\mathbf{R}_j \in \{\mathbf{R}^n\}$ sampled equivolumetrically over $\mathbf{SO}(3)$ and $\mathcal{P}(\mathbf{t}j \,|\, \mathcal{I})$:$\mathbf{t}_j \in \{\mathbf{t}^n\}$ sampled equivolumetrically over $\mathbb{R}^3$, respectively.

3.3.3 Equivolumetric Sampling and Visualization of $\mathbf{SE}(3)$

The translation component $\mathbf{t}$ is sampled uniformly in $\mathbb{R}^3$ and it is visualized using a 3D grid. To sample the rotation component $\mathbf{R}$, We follow the equivolumetric sampling of the rotation manifold approach proposed by Murphy et al. (2021) to generate $\{\mathbf{R}^0\}$ and $\{\mathbf{R}^n\}$ to cover $\mathbf{SO}(3)$ at different resolutions. Using the HEALPix algorithm (Gorski et al., 2005) as a starting step, we generate equal area grids on the 2-sphere and iteratively use Hopf fibration (Lyons, 2003) to follow a great circle through each point on the surface of a 2-sphere to cover $\mathbf{SO}(3)$. We also use the visualization method proposed by Murphy et al. (2021) to visualize distributions of object orientations on the $\mathbf{SO}(3)$ manifold. Rotation matrices in $\mathbf{SO}(3)$ have three degrees of freedom—two of the degrees of freedom are represented as a 2-sphere and projected on to a plane using Mollweide projection. The third degree of freedom is represented using Hopf fibration by a great circle of points to each point on the 2-sphere. The location of a point on the great circle is represented using a color wheel as shown in Fig. 3.4. The number of samples generated in iteration S_i is given by $72 \cdot 8^{S_i}$.

3.4 Learning Without Ground-Truth Annotations

Murphy et al. (2021) introduced the SYMSOL I and SYMSOL II datasets to benchmark symmetry learning methods. The datasets consist of renderings of platonic solids (tetrahedron, cube, icosahedron), cone, and cylinder and corresponding ground-truth symmetries. In real-world applications, acquiring ground-truth symmetry annotations is prohibitively expensive. To address this issue, we propose a two-stage automatic pose labeling scheme as illustrated in Fig. 3.5.

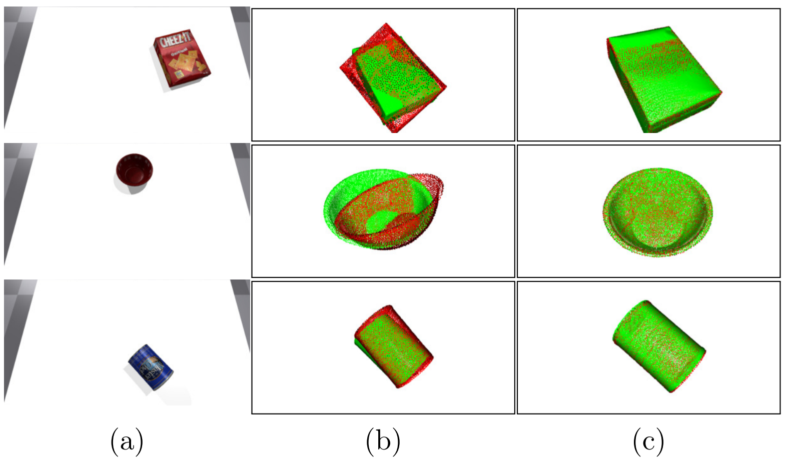


Figure 3.6: Visualization of pseudo ground-truth generation. (a) RGB image of the scene. (b) Pose generated by the fast global registration (Zhou, Park, and Koltun, 2016) algorithm. (c) Final pseudo ground-truth pose generated by ICP optimization. In (b) and (c), the point clouds in ground-truth poses are visualized in green and the point clouds in generated pseudo ground-truth poses are visualized in red.

Given an RGB-D image of a scene and the 3D mesh of the target object, we start with unprojecting the depth map into 3D to generate the observed point cloud C_{obs}. We transform the model point cloud C_{model} according to a random initial pose P_0 and perform global registration of the transformed model point cloud against the observed point cloud to generate a pose hypothesis P_1. We employ the *fast global registration* algorithm (Zhou, Park, and Koltun, 2016) in conjunction with *Fast Point Feature Histogram* (FPFH) feature (Rusu, Blodow, and Beetz, 2009) to perform global registration. We refine P_1 using the *Iterative Closest Point* (ICP) algorithm to generate $\hat{P}$. The results of the different stages of the pipeline are visualized in Fig. 3.6. Transforming C_{model} to a random initial pose P_0 at the beginning ensures that we generate a different $\hat{P}$ every time we run the process for an RGB-D image. This way, we generate a set of pseudo ground-truth pose labels for each image in the training set. The variability in the set of generated pseudo ground-truth pose labels for an image is vital for the model in learning the complete set of *proper symmetries*. Due to self-occlusion, an object is only partially visible in an RGB-D image. Without the knowledge of camera view direction, registering the complete model point cloud C_{model} against the partial observed point cloud C_{obs} might result in bad registrations and it is not possible to detect the bad registrations based on the standard ℓ_2 distance metric. To address this issue, we utilize the render-and-compare framework. We render the depth map according to $\hat{P}$ and compare it pixel-wise with the observed depth map. In the ideal scenario, the render-and-compare difference should be close to zero. To generate one pseudo ground-truth label, we repeat the process multiple times and select the $\hat{P}$ with the smallest comparison score.

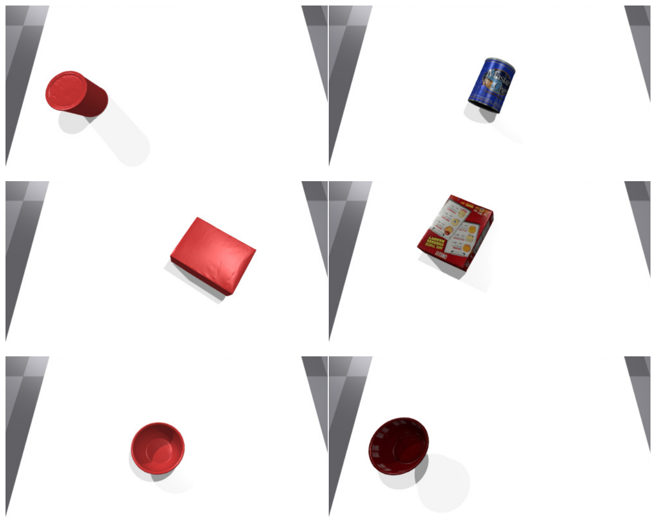


Figure 3.7: Exemplar RGB images from the Uniform and Texture dataset. First column: Uniform dataset. Second column: Texture dataset.

3.5 Evaluation

3.5.1 Tabletop Dataset Generation

To evaluate the proposed method, we generate a photo-realistic Tabletop dataset using the Isaac GYM framework (Makoviychuk et al., 2021). The dataset consists of three objects—`can`, `box`, and `bowl` (shown in Fig. 3.7)—placed in randomly-sampled physically-plausible poses on a tabletop. The translation varies approximately one meter in the x- and y-dimension and 30 centimeters in the depth dimension z. Each frame in the dataset consists of an RGB-D image, 6D object pose (single pose used to render the image) and segmentation ground-truth. Moreover, to evaluate the impact of the object texture over the model's ability to learn symmetries, we generate two versions of the dataset: using a uniform colored texture and using the original material texture. We call these datasets *Uniform* and *Texture*, respectively. Both datasets consist of 20K images. 20% of the samples in each dataset are used for validation and the remaining are used for training.

3.5.2 Implementation Details

We train our model for 50 epochs with 200 iterations per epoch. Each iteration consists of a batch of 64 images. We assume that the object bounding box and the object segmentation mask are available to us. Using the bounding box, we extract a crop of 560×560 and resize it to the standard ResNet/ConvNeXt input size 224×224. Additionally, we mask the background pixels using the segmentation mask. To compute $\mathcal{P}(\mathbf{R_{GT}} \,|\, \mathcal{I})$ we use a grid $\{\mathbf{R^0}\}$ of cardinality 4,608 (S_i=2) and to compute $\mathcal{P}(\mathbf{t_{GT}} \,|\, \mathcal{I})$ we discretize $\mathbf{t} \in \mathbb{R}^3$ into 4,913 bins. During testing, we compute the evaluation metrics using $\{\mathbf{R^n}\}$ of cardinality 294,912 (S_i=4) and 97,336 translation vectors.

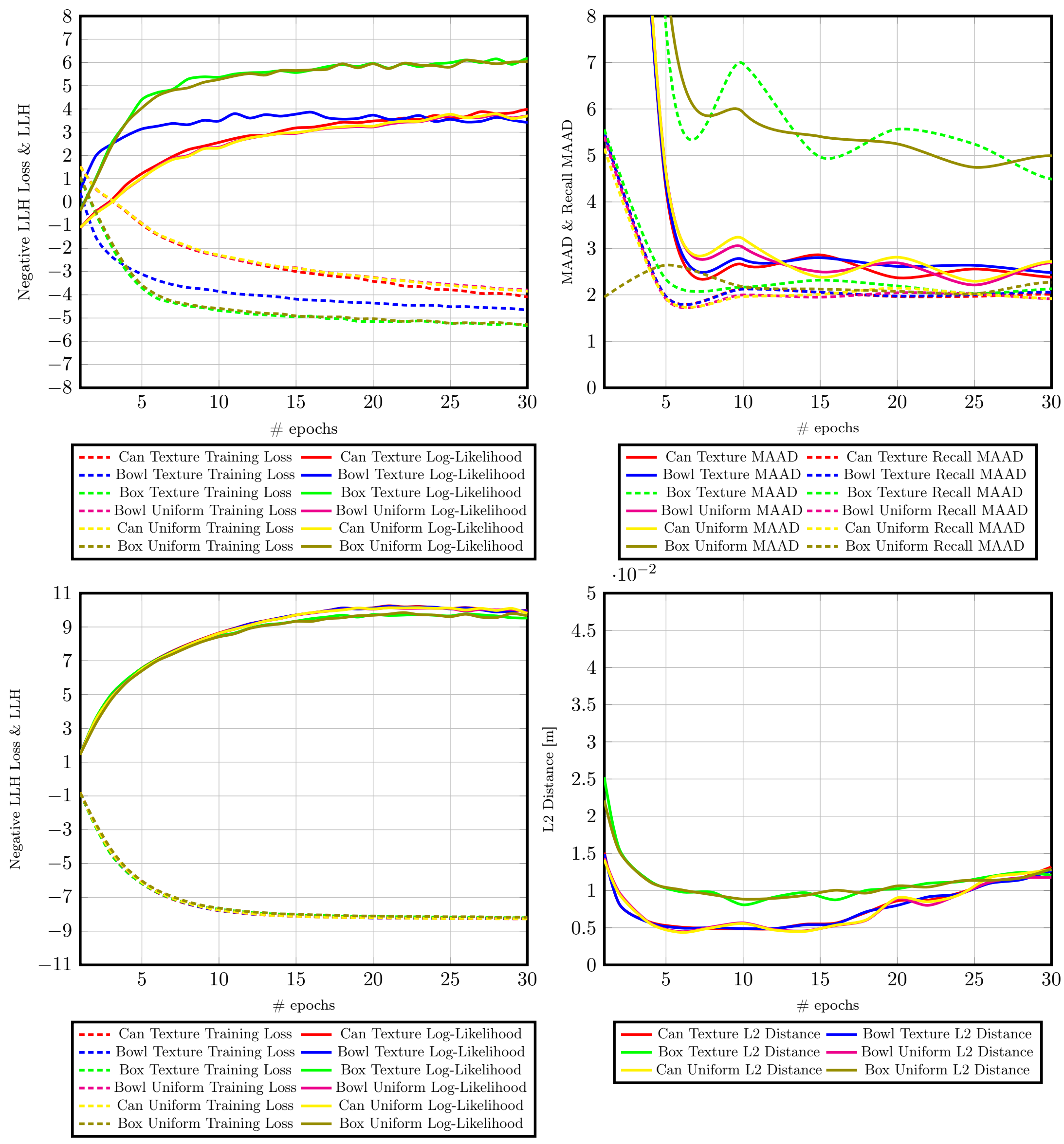
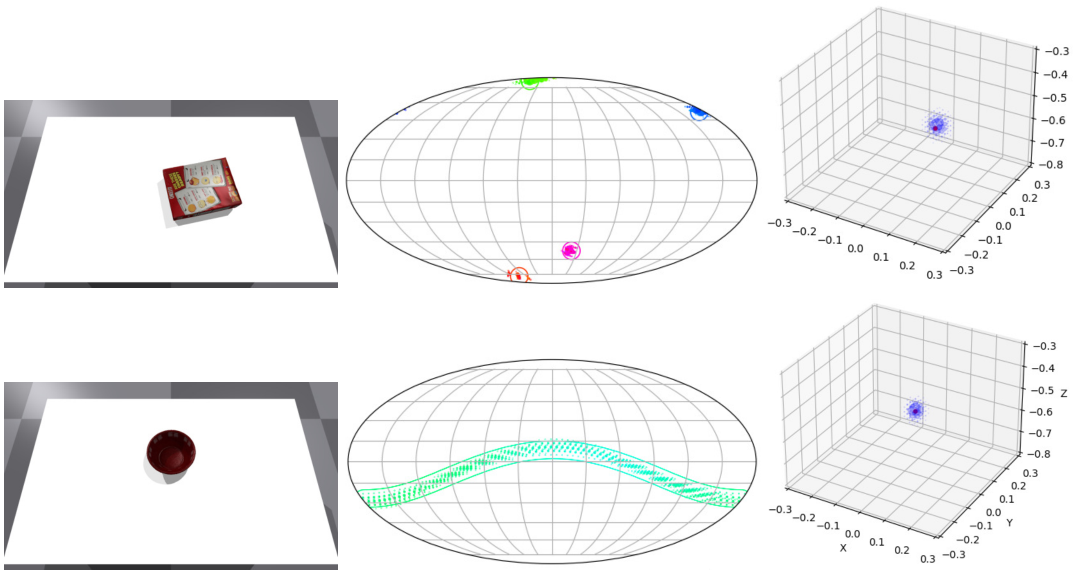
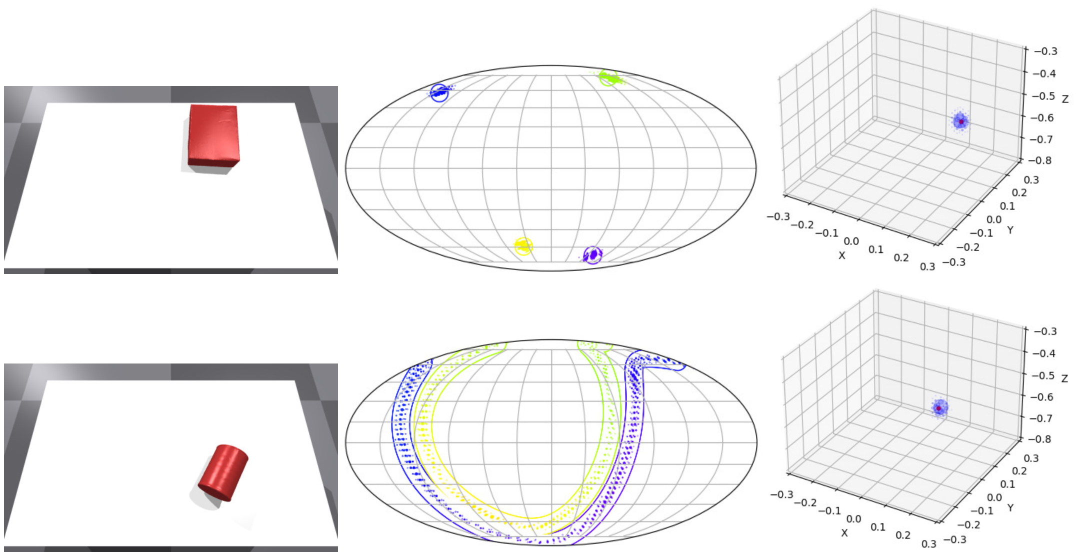


Figure 3.8: The training loss, the log-likelihood, the MAAD and Recall MAAD metrics, and the ℓ_2 distance metric on the validation set during the training process on the Uniform and Texture dataset. We early stop the training after 30 epochs when the log-likelihood starts stagnating.



(a) `Texture` dataset.



(b) `Uniform` dataset.

Figure 3.9: Pose distributions predicted by our model. (a): Texture dataset. (b): Uniform dataset. The visualizations are generated using 294,912 orientation hypotheses (S_i=4) and 97,336 translation hypotheses.

3.5.3 Evaluation Metrics

To evaluate the performance of the proposed method for orientation estimation, we use the *log-likelihood* (LLH) and the *mean absolute angular error* (MAAD) metrics. Given a set of ground-truth annotations $\mathbf{R}_{GT}$, the LLH metric measures the likelihood of the ground-truth orientations:

$$\mathrm{LLH}(\mathbf{R}) = \mathbb{E}_{I\sim\mathcal{P}(I)}\,\mathbb{E}_{\mathbf{R}\sim\mathcal{P}_{GT}(\mathbf{R}|\mathcal{I})}\log(\mathcal{P}(\mathbf{R}|\mathcal{I})).$$

To compute log-likelihood, we do not need the complete set of *proper symmetries*. The standard *mean absolute angular deviation* (MAAD) is defined as:

$$\mathrm{MAAD}(\mathbf{R}) = \mathbb{E}_{\mathbf{R}\sim\mathcal{P}(\mathbf{R}|\mathcal{I})}[\min_{\mathbf{R}'\in\mathbf{R}_{GT}} d(R, R')],$$

where d is the geodesic distance between rotations.

We report Recall MAAD as a measure of recall. We extract a set of orientations $\{\hat{\mathbf{R}}\}$ with a predicted probability threshold of 1e-3. For each orientation in $\{\mathbf{R}_{GT}\}$, we find the closest orientation in $\{\hat{\mathbf{R}}\}$ in terms of the geodesic distance and report the mean of the shortest angular distance over the set $\{\mathbf{R}_{GT}\}$. In the case of continuous symmetries, we discretize the symmetries into 200 orientations for computing the Recall MAAD metric.

Furthermore, to evaluate the overall pose estimation accuracy of the proposed method, we report the area under the curve (AUC) of the ADD-S metric (Xiang et al., 2018):

$$\text{ADD-S} = \frac{1}{m}\sum_{\mathbf{x}_1\in\mathcal{O}_{model}}\min_{\mathbf{x}_2\in\mathcal{O}_{model}}||(R_{GT}\mathbf{x}_1 + \mathbf{t}_{GT}) - (\hat{R}\mathbf{x}_2 + \hat{\mathbf{t}})||. \tag{3.11}$$

3.5.4 Experimental Results

Our model learns to predict the 6D pose of the object on both Uniform and Texture datasets. In Fig. 3.8, we show the training loss and the log-likelihood metric on the validation set during the training process. We early stop the training when the log-likelihood metric starts to stagnate. Qualitative samples of the predicted pose distributions are shown in Fig. 3.9. From the visualizations, we observe that the model learns to predict the complete set of *proper symmetries*. In Tables 3.1 and 3.2, we report the validation of the orientation set prediction in terms of LLH and Recall MAAD and the accuracy of single pose estimation in terms of the AUC of the ADD-S metric, ℓ_2 mesh point distance and the MAAD metric for the orientation estimation, respectively. Overall, in the absence of occlusions, our model achieves a MAAD score of ~3.38°, a Recall MAAD score of ~2.03°, and an average ℓ_2 translation error of 0.62. In terms of the AUC of ADD-S metric, our model achieves an impressive 87.4 and 86.86 on the Texture and the Uniform dataset, respectively. A lower MAAD indicates high accuracy of the orientation predictions and a lower Recall MAAD shows that the model learns the complete set of *proper symmetries*. Moreover, the LLH scores for the rotation and the translation predictions are 4.64 and 9.29, respectively. A higher LLH for the translation prediction compared to the rotation prediction reflects the lack of ambiguity in the translation estimation task.

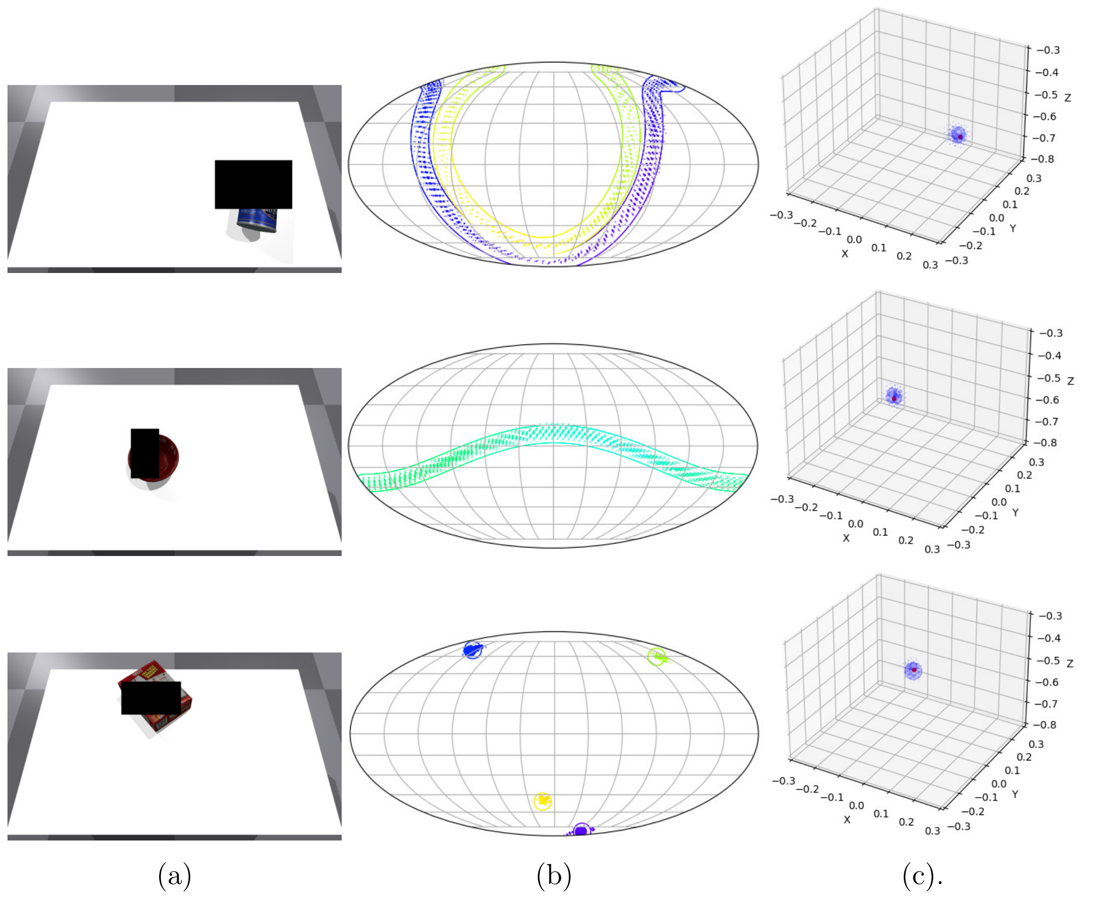


Figure 3.10: Pose distributions predicted by our model in the presence of occlusion. (a): RGB image of the scene augmented with occlusion. (b) & (c): predicted orientation and translation distributions, respectively.

3.5.5 Effect of Occlusion

Occlusion increases the complexity of computer vision problems. Pose estimation, in particular, is heavily impacted by the presence of occlusion. To make our model robust against occlusion, we train our model by masking out random crops in the input images. We augment 80% of images in each training batch randomly. Between 10% and 50% of the image portions are occluded. In Fig. 3.10, we present qualitative samples of the poses predicted by our model in the presence of occlusion. Despite the presence of occlusion, our model learns the pose distribution well on both Texture and Uniform dataset. Our model performs only slightly worse compared to the occlusion-free model. Quantitatively, in the presence of occlusion, our model achieves a MAAD score of $\sim$5.57°, a Recall MAAD score of $\sim$2.16°, and an average ℓ_2 translation error of 0.86. Interestingly, in terms of the Recall MAAD metric, the model trained using the single ground-truth annotation performs significantly better than in the case of no occlusions. This can be attributed to the uncertainty in the orientation estimate introduced by occlusion. Nevertheless, the model does not learn the correct pose distribution from a single ground-truth label. Moreover, LLH scores for the occlusion dataset are similar to the non-occlusion dataset.

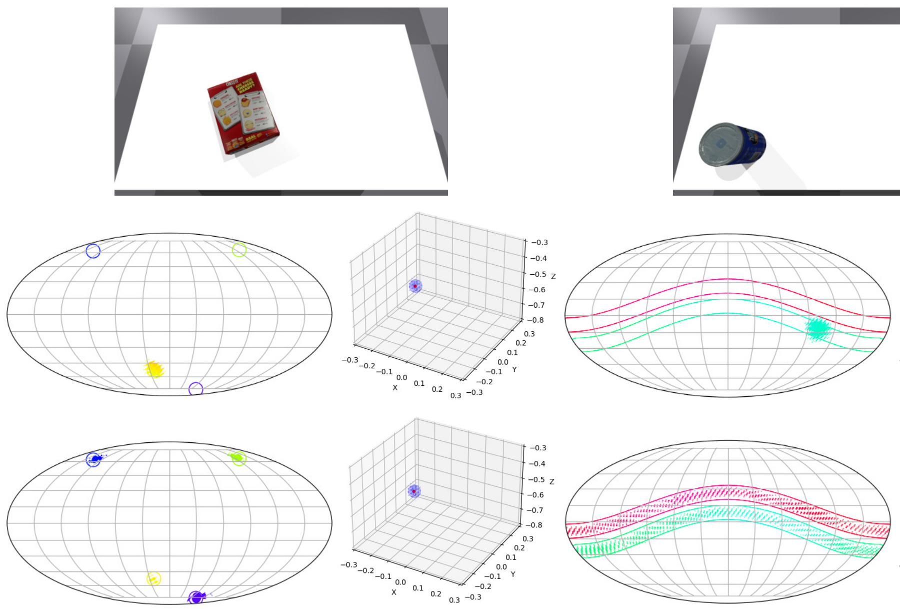


Figure 3.11: Comparison of training with multiple pseudo ground-truth annotations and single ground-truth pose. Top: RGB image of the scene. Middle: Learning with a single ground-truth pose. Bottom: Learning with multiple pseudo ground-truth poses. The model trained using pseudo ground-truth annotations learns the complete pose distribution, whereas the model trained using a single ground-truth annotation learns only the single ground-truth pose but does not learn the complete pose distribution.

This suggests that our models learn to deal with occlusion and the confidence in pose predictions is not influenced by the presence of occlusion.

3.5.6 Comparison with Training Using Different Ground-Truths

To evaluate the effectiveness of the pseudo ground-truth pose labeling scheme, we trained the implicit pose estimation model using three different types of ground-truth poses: single ground-truth pose used to render the image, complete set of *proper symmetry* ground-truth poses generated analytically, and the pseudo ground-truth poses generated using our pipeline. The qualitative results are presented in Fig. 3.11. In Table 3.1, we report the quantitative comparison results. The single ground-truth model achieves a higher LLH score and a similar MAAD score for all objects, compared to both analytical and pseudo ground-truth models, i.e. the single ground-truth model learns to estimate one single orientation precisely but fails to learn the symmetric orientations, whereas the other two models manage to learn the complete set of symmetric orientations. Moreover, the pseudo ground-truth model achieves results similar to the analytical model on all three metrics. Based on these results, we can conclude that the automatic pose labeling scheme is able generate pose labels with high accuracy and covers a sufficiently big portion of the set of *proper symmetries* for the model to learn the symmetries. Furthermore, as a measure of accuracy of the generated pseudo ground-truth orientation labels, we report in Table 3.3 the MAAD metrics of the generated pseudo ground-truth orientation labels for all three objects. Overall, the average error rate of the generated pseudo ground-truth labels is ∼2.7°. Among the three objects present in the dataset, the *box* object has the

Table 3.1: Results of models trained on different ground truths.

	GT	Without Occlusion			With Occlusion		
		LLH (Rot.) ↑	LLH (Trans.) ↑	Recall MAAD [°] ↓	LLH (Rot.) ↑	LLH (Trans.) ↑	Recall MAAD [°] ↓
can	Single	6.46	9.17	120.7	5.347	8.77	97.99
	Analytic	3.78	9.22	1.87	3.47	8.81	1.95
	Pseudo	3.99	9.19	1.86	3.55	8.83	1.99
box	Single	6.56	9.59	123.61	6.76	9.35	119.63
	Analytic	5.78	9.53	2.17	5.75	9.68	2.08
	Pseudo	6.09	9.49	2.17	5.68	9.56	1.95
bowl	Single	6.49	9.19	119.28	5.36	9.28	97.12
	Analytic	3.95	9.23	2.12	3.93	9.27	1.88
	Pseudo	3.85	9.19	2.05	3.76	9.21	2.54
Pseudo Avg.		4.64	9.29	2.03	4.33	9.2	2.16

↑ indicates higher value better, whereas ↓ indicates lower value better.

Object		Pseudo Ground-Truth	ImplicitPosePDF				
			Texture	Uniform	Single	Analytic	Occlusion
Can	MAAD [°]	1.44	2.44	2.81	3.11	2.5	4.16
	ℓ_2 [cm]	0.33	0.48	0.38	0.45	0.46	0.74
	AUC	93.40	91.61	91.61	91.63	91.42	91.94
Box	MAAD [°]	3.09	5.24	5.25	5.2	5.35	5.66
	ℓ_2 [cm]	0.82	0.88	0.85	0.61	0.62	0.88
	AUC	81.94	79.15	78.32	83.81	80.87	76.12
Bowl	MAAD [°]	1.32	2.47	2.65	2.77	2.43	6.89
	ℓ_2 [cm]	0.23	0.49	0.45	0.45	0.46	0.97
	AUC	96.37	91.43	90.66	93.93	93.57	85.63
Average	MAAD [°]	1.95	3.38	3.57	3.69	3.43	5.57
	ℓ_2 [cm]	0.46	0.62	0.56	0.5	0.51	0.86
	AUC	90.57	87.4	86.86	89.79	88.62	84.56

Table 3.2: Accuracy of single 6D pose estimation on the Tabletop dataset.

Table 3.3: Evaluation of the pseudo ground-truth orientation labels

Object	Dataset	MAAD[°]
can	*Texture*	1.44
	Uniform	3.12
box	*Texture*	4.14
	Uniform	4.3
bowl	*Texture*	1.42
	Uniform	1.89
Average		2.72

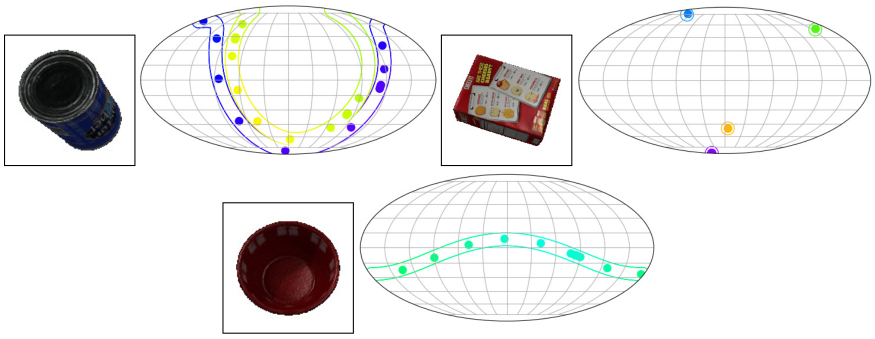


Figure 3.12: Visualizing the generated pseudo ground-truth orientation labels. Dots represent the generated pseudo ground-truth orientation labels, whereas the circles and the continuous lines represent the ground-truth orientations. The generated pseudo ground-truth orientation distribution corresponds to the ground-truth orientation distribution with high accuracy.

highest MAAD error. This can be attributed to the fact that *box* exhibits discrete symmetry. Generating pose labels for discrete symmetry is more difficult than for continuous symmetry. As shown in Fig. 3.12, the pseudo ground-truth pose labels correspond to the ground-truth orientation distribution with a high degree of accuracy.

3.5.7 Backbone Ablations

CNN models learn features that generalize well across datasets. However, the degree of generalization varies across different architectures. In order to find the architecture best suited for usage as backbone feature extractor in the ImplicitPDF model, we experimented with ResNet (He et al., 2016), and ConvNeXt (Liu et al., 2022b) architectures. Convolutional neural networks, in general, learn low-level image features at a high resolution in the initial layers and high-level features at low resolution in the final layers (Behnke, 2003b; Schulz and Behnke, 2012; LeCun, Bengio, and Hinton, 2015). For many computer vision tasks like object classification and object detection, high-level features—despite being low resolution—are ideal, whereas tasks like semantic segmentation benefit from access to low-level features (Lin et al., 2017; Schwarz et al., 2018b; Ronneberger, Fischer, and Brox, 2015). To evaluate the effectiveness of different models as feature extractors, we experimented with two versions of ResNet—ResNet-18 and ResNet-50—and three variants the ConvNeXt model (Liu et al., 2022b). The ConvNeXt model in the Tiny and Small configuration extracts a feature vector of size 768 and in the Base configuration, a vector of size 1024. Furthermore, to evaluate the effectiveness of features extracted from different ConvNeXt layers, we experimented with two additional variants of the ConvNeXt Tiny backbone model—Tiny-1 and Tiny-2. Tiny-1 has the last average-pooling layer removed, resulting in a feature vector of size 768×7×7, and Tiny-1 additionally has the last ConvNeXt block removed, with a feature vector of size 384×14×14. In Section 3.5.7, we present the quantitative results of the comparison. In terms of both LLH and MAAD metrics on the Uniform and Texture datasets, all models perform similarly, whereas, in terms of the ℓ_2 error metric, ConvNeXt Tiny models performed better than the other models. Based on these results, we use ConvNeXt Tiny models in all our experiments.

Table 3.4: Comparison of different models as the feature extractor.

	Model	Metric	ResNet (He et al., 2016)		ConvNeXt (Liu et al., 2022b)				
			18	50	Tiny	Tiny-1	Tiny-2	Small	Base
FLOPs[†]			1.8G	3.8G	4.5G	-	-	8.7G	15.4G
can	**R**	LLH	3.76	3.86	**3.99**	3.83	1.09	3.77	3.72
		MAAD [°]	2.24	2.46	**2.44**	2.52	21.5	2.55	2.79
	y	LLH	9.19	9.31	**9.19**	9.34	9.13	9.56	9.43
		ℓ_2 [cm]	0.71	0.79	**0.48**	0.52	0.53	0.6	0.57
box	**R**	LLH	5.75	5.98	**6.09**	5.75	3.02	5.996	5.649
		MAAD [°]	5.2	4.73	**5.24**	5.97	34.27	4.28	8.609
	y	LLH	8.86	8.97	**9.49**	9.03	8.79	9.29	9.24
		ℓ_2 [cm]	1.07	1.1	**0.88**	1.04	1.02	0.94	0.99
bowl	**R**	LLH	3.3	3.17	**3.85**	3.86	1.05	3.21	3.24
		MAAD [°]	2.96	3.17	**2.47**	2.7	22.84	2.88	3.22
	y	LLH	9.1	9.25	**9.19**	9.52	9.45	9.56	9.6
		ℓ_2 [cm]	0.58	0.63	**0.49**	0.53	0.53	0.67	0.7

R: Orientation IPDF model. **y**: Translation IPDF model.
[†] FLOP values are taken from (He et al., 2016) and (Liu et al., 2022b).
Best values are shown in bold.

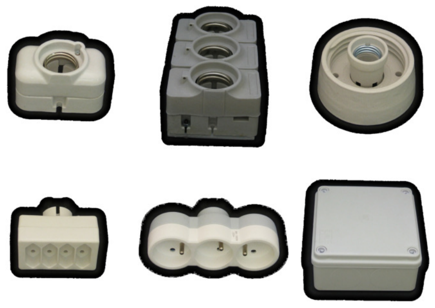


Figure 3.13: The T-Less dataset objects used for evaluating our method.

3.5.8 Evaluation on T-Less Dataset

The T-Less Dataset consists of RGB-D images of texture-less objects of varying sizes along with 6D pose annotations. Training data consists of RGB-D images of individual objects placed in isolation with a black background. We evaluate our method on a subset of T-Less objects that exhibit geometric symmetries (shown in Fig. 3.13). We use the variant of the T-Less dataset proposed by Gilitschenski et al. (2019) in which the training images provided in the original T-Less are split into training, validation, and test sets. In Fig. 3.14, we present exemplar qualitative visualizations of the predicted orientations. In Table 3.5, we report quantitative results. Our method achieves a MAAD score of ∼3.22°and an LLH score of 6.09. Since we report the metrics only for the objects that exhibit geometric symmetries, uncertainty always exists in terms of the object orientation. Thus, our model performs worse in terms of the LLH metrics, compared to other

Table 3.5: Comparison results on the T-Less dataset.

Method	LLH ↑	MAAD[°] ↓
Deng et al. (2022)	5.3	23.1
Gilitschenski et al. (2019)	6.9	3.4
Prokudin, Gehler, and Nowozin (2018)	8.8	34.3
Murphy et al. (2021)	9.8	4.1
Analytic	6.2	1.7
Ours*	6.09	3.22

* Results from a subset of the T-Less objects (shown in Fig. 3.13).

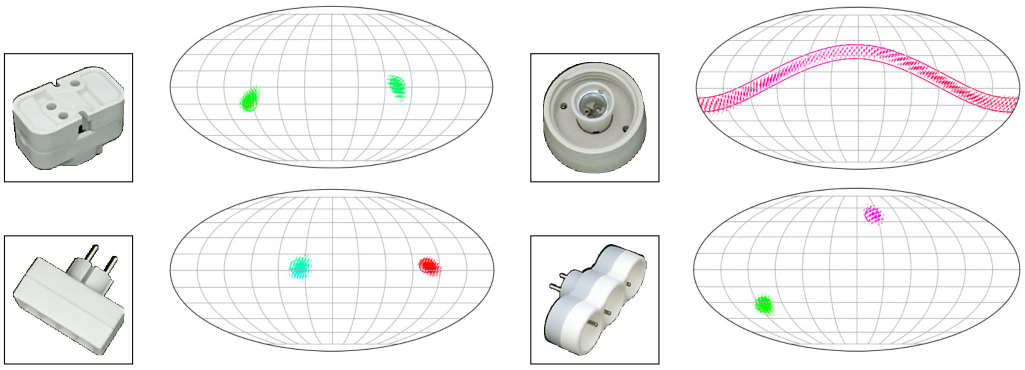


Figure 3.14: Orientation distributions on the T-Less Dataset. Orientation distributions predicted by the IPDF model trained with pseudo ground-truth labels on the T-Less dataset.

methods, but this does not affect the accuracy in terms of the MAAD metrics. Moreover, compared to the model trained using analytically generated ground-truth orientation labels, which achieves a MAAD score of ∼1.7°and an LLH score of 6.2, our method performs only slightly worse. This indicates that analytically generating ground-truth orientation labels for objects with complex geometric symmetry is not trivial. Moreover, the pose labeling scheme generates pseudo pose labels ($\mathbf{SE}(3)$), whereas the analytical ground-truth generation is possible only for orientation labels ($\mathbf{SO}(3)$). Having access to pose labels enables training complete pose estimation models. Thus, the proposed automated pose labeling scheme serves as an efficient alternative to generating ground-truth orientation labels analytically.

3.6 Discussion & Conclusion

In this Chapter, we presented the ImplicitPDF model for learning 6D pose estimation for symmetrical objects. We proposed a pseudo ground-truth labeling scheme to generate pose annotations and used it to train the ImplicitPDF model without any manual pose annotations. We quantified the accuracy of the pose labeling scheme and demonstrated the advantages of multiple pseudo ground-truth labels over the single ground-truth pose label for training the ImplicitPDF model. Moreover, by comparing with the models trained using analytically generated ground-truth orientations, we demonstrated the effectiveness of the automatic pose labeling scheme. Overall, our method predicts the object pose as well as the complete set of *proper symmetries* for uniform color objects, as well as for objects with texture with a high degree of accuracy. In future, incorporating the knowledge about symmetry provided by the ImplicitPDF models presented in this Chapter into manipulation and affordance planning algorithms will improve these algorithms significantly and enhance autonomous manipulation of real-world objects.

4 Object Pose and Shape Refinement Using Abstract Render-and-Compare

Refining initial scene parameter predictions aids in improving the robustness of the perception systems. Render-and-compare offers an elegant framework for scene parameter refinement in which the initial parameters are iteratively refined to minimize the pixel-wise differences between the rendered and the observed images. In this Chapter, we introduce a lightweight differentiable renderer that computes the gradient of rendered image with respect to the scene parameter and employ it in a gradient-based optimization scheme to refine 6D object pose and shape parameters. Moreover, to alleviate the complexities of pixel-wise comparison of the rendered and the observed images in the RGB space, we propose image comparison in a learned descriptor space.

Statement of Personal Contribution

The differentiable renderer, pose and shape refinement pipelines presented in this Chapter are adapted from the following publications.

Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke:

Refining 6D object pose predictions using abstract render-and-compare

In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, Toronto, Canada. 2019.

Arul Selvam Periyasamy, Max Schwarz, and Sven Behnke:

Iterative 3D deformable registration from single-view RGB images using differentiable rendering

In: *17th International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal, 2022.

The author of this thesis substantially contributed to all aspects of the publication with the exception of the dense descriptors discussed in Section 4.4, including the literature survey, the conception, formalization, design, and implementation of the differentiable renderer, the preparation and conduct of experiments for the evaluation of the proposed pipelines, the analysis and interpretation of the experimental results, the preparation of the manuscript, as well as the revision and final editing of the version published. Conception, implementation, and training of the dense descriptors and the mesh representation for surface features discussed in Section 4.4 are individual contributions of the co-authors. The author of this thesis acknowledges the significance of these contributions to the proposed pipelines and expresses deep gratitude for these contributions.

4.1 Introduction

Robust robotic interaction in environments made for humans is an open research field. An important prerequisite in this context is scene perception, yielding the necessary information such as detected objects and their poses or affordances for later manipulation actions. While there are various high-accuracy methods for scene understanding, the problem becomes significantly harder in the presence of clutter and inter-object effects. As such, current works in autonomous manipulation that require precise grasping are often limited to non-cluttered or even isolated scenes (e.g. Pavlichenko et al. (2018) and Klamt et al. (2018)). While the manipulation action itself and planning for it is certainly more difficult in cluttered scenes, robust 6D object pose estimation is a necessary prerequisite.

An interesting approach in this context is the idea of viewing computer vision as an inverse graphics process (Grenander, 1976; Grenander, 1978). It promises to perform scene analysis by inverting the rasterization process, which sounds highly promising—today's rendering techniques are capable of producing convincing photo-realistic renderings of highly complicated scenes, so inversion of the process should yield high-quality scene analysis. However, the problem plaguing the inverse graphics field is that the rendering process is largely unidirectional, with complex physical effects such as lighting, surface scattering, transparency, and so on. Furthermore, scene analysis is especially in demand for cluttered scenes, e.g., in warehouse automation contexts, but occlusion effects caused by clutter are among the most difficult to invert or differentiate.

To take a step towards a solution to this problem, we propose to first remove most *secondary* rendering effects from the scene using abstract surface features learned in an unsupervised manner. This way, only the *primary* effects remain—occlusion and projection. These effects can then be explained and analyzed by a simpler differentiable rendering component.

We apply our render-and-compare framework for two scene understanding tasks. First, the task of monocular 6D pose estimation, specifically pose refinement, where initial pose guesses are available. In our approach, 6D pose predictions from two different pose estimation methods are refined by minimizing the pixelwise difference between the rendered and observed images in a novel abstract descriptor space invariant to secondary rendering effects. Second, the shape refinement task. We iteratively refine the 3D shape of objects by minimizing the pixelwise difference in the CNN feature space, which is also invariant to secondary rendering effects. Finally, we perform joint refinement of pose and shape of objects using the render-and-compare framework.

The render-and-compare framework requires several iterations of parameter optimization. Each iteration involves rendering the scene according to the current parameter estimate and computing gradients through the rendering process. This necessitates an efficient differentiable renderer. To this end, we develop StilllebenDR, a high-performant differentiable renderer. We discuss StilllebenDR and use it to design render-and-compare pipelines for object pose and shape refinement tasks.

4.2 Related Work

Vision as inverse graphics aims at inferring object parameters like shape, illumination, reflectance, and pose, scene parameters like camera parameters, lighting, and secondary reflections by inverting the rendering process. The process of rendering 3D scene to discrete 2D pixels involves discretization steps that are not differentiable. However, several approximation methods have been proposed to realize a differentiable renderer. Loper and Black (2014) proposed OpenDR, a generic differentiable renderer that can compute gradients with respect to object and scene parameters. Kato, Ushiku, and Harada (2018) introduced a differentiable renderer that is suited for neural networks. Rezende et al. (2016) treat forward rendering as a black-box and used REINFORCE (Williams, 1992) to compute gradients. Li et al. (2018a) proposed edge sampling algorithm to differentiate ray tracing that can handle secondary effects such as shadows or global illumination. Liu et al. (2019) proposed a differentiable probabilistic formulation instead of discrete rasterization. The differentiable renderer used in this work is closely modeled after OpenDR but tailored for object pose refinement with a strong focus on speed.

Vision as inverse graphics is most often formulated as a render-and-compare approach, where model parameters are optimized by minimizing the difference between rendered and observed images. Zienkiewicz, Davison, and Leutenegger (2016) used render-and-compare for real-time height mapping fusion. Several recent works used render-and-compare for solving a wide range of vision problems: Kundu, Li, and Rehg (2018) introduced a framework for instance-level 3D scene understanding; Moreno et al. (2016) estimated 6D object pose in cluttered synthetic scenes. More closely related is the DeepIM method by Li et al. (2018b), who formulated 6D object pose estimation as an iterative pose refinement process that refines the initial pose by trying to match the rendered image with the observed image. In contrast to our approach, they avoid the need for backpropagating gradients through the renderer by training a neural network to output pose updates. While the method yields very promising results, it is not directly clear how to apply this method to symmetric objects without specifying symmetry axes, whereas our method inherently optimizes to a suitable pose. We also note that DeepIM is object-centric, refining each object's pose separately. In contrast, our method retains the entire scene, refining all object poses simultaneously and thus is able to account for inter-object effects.

In this work, we use render-and-compare to refine 6D object poses in cluttered real-world scenes. Instead of comparing the rendered and observed RGB images, we propose to use an abstraction network to deal with the difficulties in comparing images from two different modalities.

4.3 StillebenDR

Rasterization is the process of generating 2D images given the 3D scene description. Libraries like OpenGL (Segal and Akeley, 1999), Vulkan (Khronos, 2018), and DirectX (Microsoft, 2019) offer optimized rasterization implementations. Although the standard formulation of rendering 3D faces of object meshes into discrete pixels is not differentiable, probabilistic formulations like SoftRas (Liu et al., 2019), PyTorch3D (Ravi et al., 2020), and DIB-R (Chen et al., 2019) allow for differentiable rendering. Most of the commonly used differentiable renderers are implemented using CUDA with programming interfaces to neural network libraries like TensorFlow (Abadi et al., 2016) or PyTorch (Paszke et al., 2019).

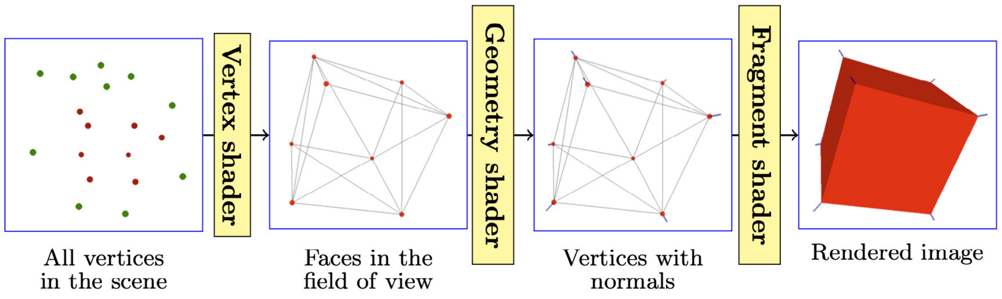


Figure 4.1: OpenGL shader pipeline. A typical use-case of a shader pipeline involving the geometry shader. All the valid vertices in the world coordinates are transferred into the camera coordinates and *line primitives* are generated connecting the vertices for a face in the vertex shader. For each vertex, a *line primitive* corresponding to the normal direction is generated in the geometry shader. Faces are rasterized into discrete pixels in the fragment shader.

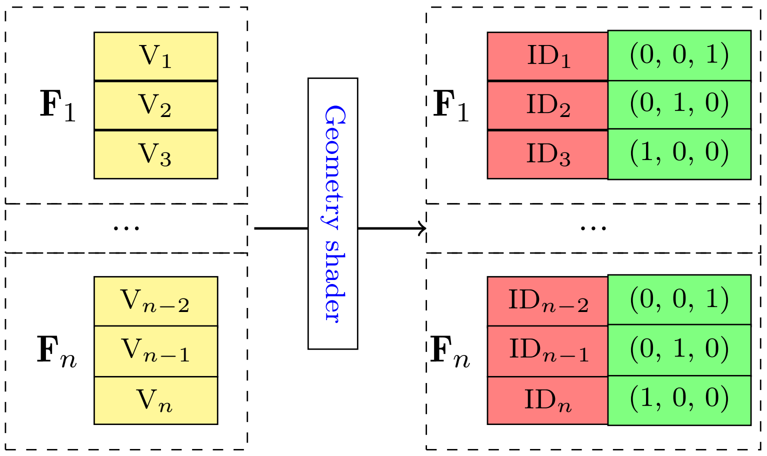


Figure 4.2: Geometry shader configuration for barycentric coordinates generation. For each vertex, we generate a global vertex (in 🟥) and a local vertex index that uniquely corresponds to a vertex for each face (in 🟩). Global vertex indices are immutable, whereas local vertex indices are mutable.

StilllebenDR is built as an extension to the Stillleben library (Schwarz and Behnke, 2020), a synthetic data generation pipeline designed to generate training data for deep learning models on the fly. It provides PyTorch interface [1] and uses OpenGL for rendering and PhysX [2] for physics simulation. OpenGL provides an optimized implementation of a standard rasterization pipeline.

StilllebenDR enables differentiation support with only a minimal overhead to the OpenGL rasterization pipeline. During the rasterization step, a face F constituting of vertices V with colors C is projected on a pixel I. The pixel color I is computed as

$$\mathrm{I_{rgb}} = \sum_i \mathrm{b}_i \mathrm{C}_i, \tag{4.1}$$

1 https://pytorch.org/
2 https://github.com/NVIDIAGameWorks/PhysX

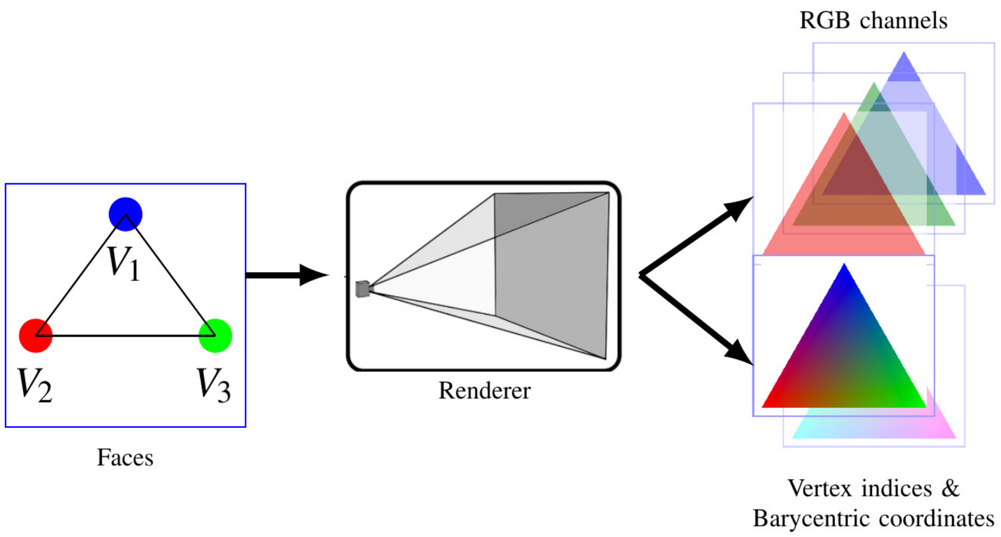


Figure 4.3: Forward rendering. In addition to the RGB channels, we also render vertex indices and barycentric coordinates per pixel as separate channels and store them for backward computations.

where b_i are the barycentric coordinates and $\sum_i \mathrm{b}_i = 1$. We simplify the notation I_{rgb} and use I instead. OpenGL allows users to write shaders, specialized light-weight programs that are designed to run at specific stages of a graphics pipeline.

Breaking down the graphics pipeline into a sequence of shaders enables parallelization. In addition to the standard RGB-D channels, we utilize the flexibility of the shaders to render additional channels containing information like barycentric coordinates and vertex indices, as shown in Fig. 4.3. The *vertex shader* and the *fragment shader* are the only two mandatory shaders in an OpenGL pipeline. In the *vertex shader*, vertices in the mesh coordinate are projected into the clip space, the space covered by the camera frustum. In the *fragment shader*, the color for rasterized pixels are computed. In order to generate barycentric coordinates and vertex indices as additional output channels, we make use of one of the less commonly used shaders, namely the *geometry shader*. The *geometry shader* is invoked at the end of the *vertex shader* and is used to generate additional primitives like vertices, lines, or faces. A common use-case for an OpenGL pipeline consisting of the *geometry shader* is to visualize vertex normals (see Fig. 4.1). For each vertex, a line corresponding to the normal direction in the *geometry shader* is generated. We adapt the *geometry shader* to generate barycentric coordinates and vertex indices (see Fig. 4.2). For each vertex, a global vertex index and local vertex are generated. The global vertex index is marked as immutable and is rendered as constant values. For each face, one of *(0, 0, 1), (0, 1, 0), (1, 0, 0)* mutable vector values are assigned to each vertex uniquely. The vector values are interpolated in the later stages of the shader pipeline using the barycentric coordinates. The immutable global vertex indices and the interpolated local vertex indices reflect the corresponding barycentric coordinates and are rendered as additional output channels.

Given the loss L, computed pixel-wise between the rendered and the observed images, the gradient of the loss with respect to different scene parameters can be decomposed into the gradient of the loss with respect to the rendered image, and the gradient of

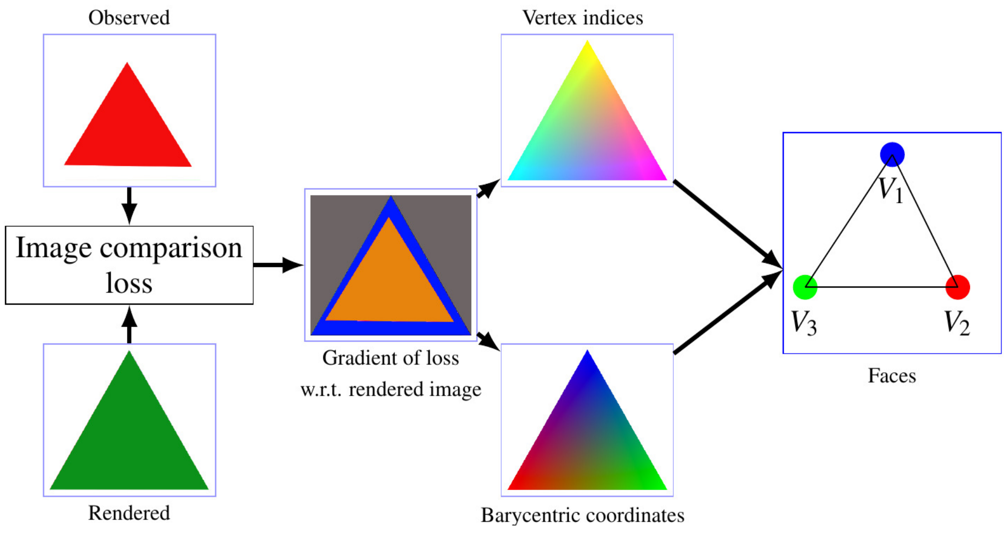


Figure 4.4: Backward rendering. The gradient of the image comparison loss function is propagated to the vertices by differentiating through the renderer using the vertex indices and barycentric coordinates information stored during the forward rendering step.

the rendered image with respect to the scene parameters following the chain rule. For example, the gradient of the loss with respect to the vertex V_i is computed as

$$\frac{\partial \mathrm{L}}{\partial \mathrm{V}_i} = \frac{\partial \mathrm{I}}{\partial \mathrm{V}_i} \cdot \frac{\partial \mathrm{L}}{\partial \mathrm{I}}, \tag{4.2}$$

where $\frac{\partial \mathrm{I}}{\partial \mathrm{V}_i}$ is computed automatically by PyTorch autograd. Using the barycentric weights and the vertex indices stored during the forward rendering step, $\frac{\partial \mathrm{I}}{\partial \mathrm{V}_i}$ is computed as

$$\frac{\partial \mathrm{I}}{\partial \mathrm{V}_i} = \mathrm{C}_i. \tag{4.3}$$

Similarly, we break down the gradient of the loss function with respect to object pose P as follows:

$$\frac{\partial \mathrm{L}}{\partial \mathrm{P}} = \frac{\partial \mathrm{I}}{\partial \mathrm{P}} \cdot \frac{\partial \mathrm{L}}{\partial \mathrm{I}}. \tag{4.4}$$

The backward pass of the rendering process is depicted in Fig. 4.4.

4.4 Pose Refinement using Abstract Render and Compare

Real scenes exhibit a large variety of secondary effects such as lighting, camera noise, reflections, and so on. All of these effects are very difficult to model and severely constrain the applicability of differentiable rendering methods. We propose an additional abstraction module $f : \mathbb{I} \to \mathbb{A}$, mapping the RGB image space I to an abstract feature space A. Ideally, the mentioned secondary effects lie in the null space of f. For convenience, we require that A is also image-like so that pixels in I correspond to feature vectors in A.

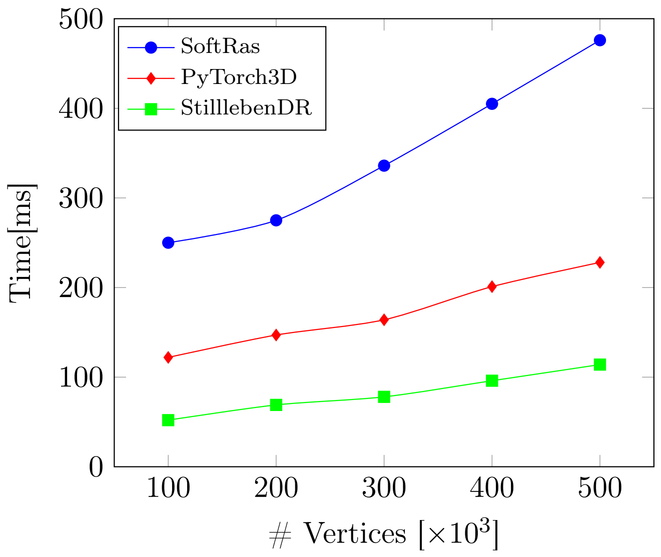


Figure 4.5: Runtime comparison between SoftRas (Liu et al., 2019), PyTorch3D (Ravi et al., 2020), and StilllebenDR (ours). We report the average time taken by different differentiable rendering approaches to perform forward rendering (1024×1024 pixels) and backward gradient computations.

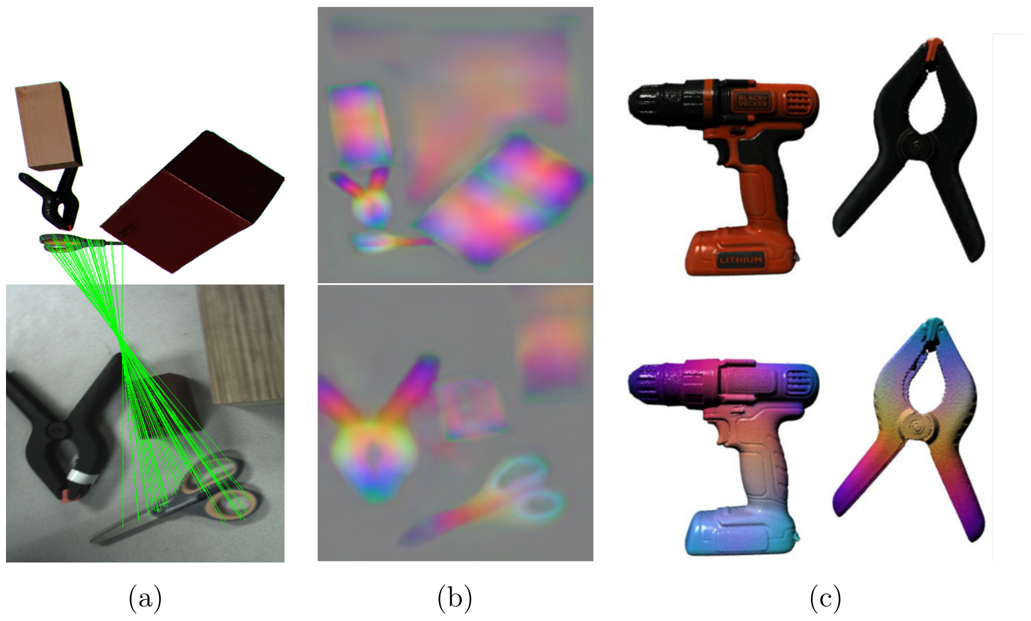


Figure 4.6: Learning dense descriptors from real-synthetic correspondences. (a): Real and synthetic input frames, respectively. Positive correspondences matches are shown in green for one object. (b): Learned dense abstract representation for the corresponding input frames in (a). (c): Learned surface features, projected and fused onto the mesh. The 3D feature vectors are visualized directly as RGB colors.

The very difficult problem of decomposing an image into its different intrinsic components, such as shading, reflectance, and shape, has been studied extensively (Barrow et al., 1978; Tappen, Freeman, and Adelson, 2003; Finlayson, Drew, and Lu, 2004). However, in this application such a complex and physically accurate decomposition is not required. In order to be usable for differentiable rendering, we only require that features are similar for corresponding points on the same object (under varying lighting conditions etc), and dissimilar for non-corresponding objects. Many traditional feature extractors exhibit this property (e.g. SIFT). Of particular interest, however, are feature extractors designed for dense output. Recently, (Schmidt, Newcombe, and Fox, 2017) showed that highly precise feature extractors can be trained in a self-supervised way from ground-truth correspondences. The learned descriptors outperform sparse feature extractors by a large margin.

4.4.1 Learning Dense Descriptors

In order to leverage this idea in the differentiable rendering setting, we propose to learn descriptors from the object meshes in conjunction with a training dataset for pose estimation. For a pose-annotated real dataset frame A, we render a synthetic frame B with the same object set, but using different poses (see Fig. 4.6 (a)). Corresponding points in both RGB frames can be easily determined from the object poses through projective geometry. We then minimize the pixel-wise contrastive loss function between the positive correspondences and the negative correspondences. To encourage the network to disregard clutter in the background, we also introduce a loss on background pixels for a single frame. We refer the reader to Periyasamy, Schwarz, and Behnke (2019) for additional details related to network architecture and training of the dense descriptor model.

4.4.2 Mesh Representation for Surface Features

As discussed in Section 4.3, the simplest representation for a 3D polygon mesh consists of set of triangles called *faces* defined by their *vertices*. Faces are connected by their common edges or vertices. Each vertex corresponds to a position $\in \mathrm{R}^3$ and a RGB color $\in [0, 1] \subset \mathrm{R}^3$. 3D meshes can be simplified into volumetric representations by approximating vertices in a neighborhood using voxels. Since the learned features should be constant for each local surface patch, independent of the viewing pose, we can *fuse* the descriptor information onto the mesh representation. To this end, we render N views (50 in our experiments) of the object from randomly sampled viewing directions and viewing distances. After feature extraction using the learned network, the resulting point-feature pairs are aggregated in the object frame. A voxel grid downsampling is applied, where descriptors and positions are each averaged inside each voxel. The voxel size is determined heuristically from the object bounding box s.t. the voxel count is constant—this results in constant-size output. This method is robust and easy to tune in case more complex geometry needs to be supported. In our experiments, we use 5000 voxels. Finally, each vertex of the object mesh is assigned an interpolated descriptor from the four nearest voxels using inverse-distance weighting. Figure 4.6 (c) shows exemplary meshes and corresponding feature visualizations. Rendering fused meshes does not resemble a typical RGB textured image. However, some geometric information of object surface can deduced from the rendered images. Moreover, fushed meshes help us in ensuring gradient flow in the pose refinement pipeline we introduce in Section 4.4.6.

4.4.3 Iterative Pose Refinement

The pose refinement problem is an optimization problem. In our case, we assume that we start with a pose initialization of reasonable quality, such that local optimization methods can find the optimum solution. In this context, it is very favorable to be able to compute derivatives of the rendering process, since the number of parameters grows linearly with the number of objects in the scene (at least six parameters per object). Optimization without gradient information quickly becomes infeasibly slow.

We base our differentiable rendering module on the method of OpenDR (Loper and Black, 2014), which is able to approximate gradients with respect to lighting parameters, camera parameters, object poses, etc. We note that in our setting, only pose parameters need to be optimized, because lighting and other surface effects are removed by the abstraction network and camera parameters are assumed to be fixed in monocular pose estimation.

The OpenDR method is built around the screen-space approximation of the derivative of the rendering process. Gradients due to occlusion effects during this 3D-2D reduction are approximated from the local intensity gradient. In essence, this idea assumes that occluded pixels are similar to their visible neighbors.

In order to simplify gradient computation, we locally linearize the pose:

$$T(\alpha,\beta,\gamma,a,b,c) = T_0 \begin{pmatrix} 1 & -\gamma & \beta & a \\ \gamma & 1 & -\alpha & b \\ -\beta & \alpha & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

The rotation part of T is orthonormalized after each optimization step.

4.4.4 Gradients on Occlusion Boundaries

The scenes we are interested in feature high levels of occlusion between the individual objects. (Loper and Black, 2014) makes several assumptions in computation of the screen-space gradients. While these assumptions help in simplifying the computation, in real world scenarios, they are often violated. We discuss the assumptions involved, scenarios where these assumptions are violated, and propose solutions for better approximation of the pose gradients. Occlusion boundaries are highly important for pose refinement, since they offer much information about the scene layout. On the occlusion boundary pixels, OpenDR uses Sobel kernel ($\frac{1}{2}[-1,0,1]$) and its transpose to compute the gradient along the horizontal axis and the vertical axis respectively, with the underlying assumption that a shift in the occlusion boundary can be approximated by the replacement of the current pixel by the neighboring pixel (of the other object). However, this assumption is valid only if the occlusion boundary pixel belongs to the object in the foreground. Fig. 4.7 (a), (b) depict the front view and top view of an example scene where the mug is occluding the can. Translating the mug results in covering or uncovering of can pixels, which is well approximated using the local Sobel gradient. Conversely, translating the can in the

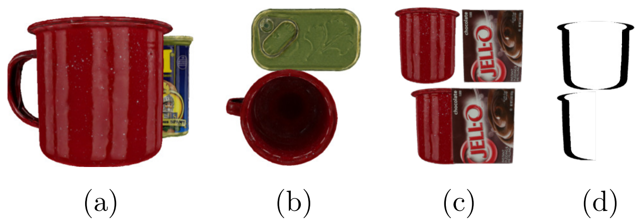


Figure 4.7: Corner cases for renderer gradient estimation. (a) and (b): Front and top view of an exemplary scene with occlusion. (c) Top: Occlusion-free scene; Bottom: Scene with mug occluded. (d) Magnitude of pixel-wise loss corresponding to scenarios depicted in (c) when the mug is translated. Black depicts: high loss magnitude. When the rendered mug is moved behind the occluder, all pixels with high loss (and thus gradient information) lie outside of the rendered object mask.

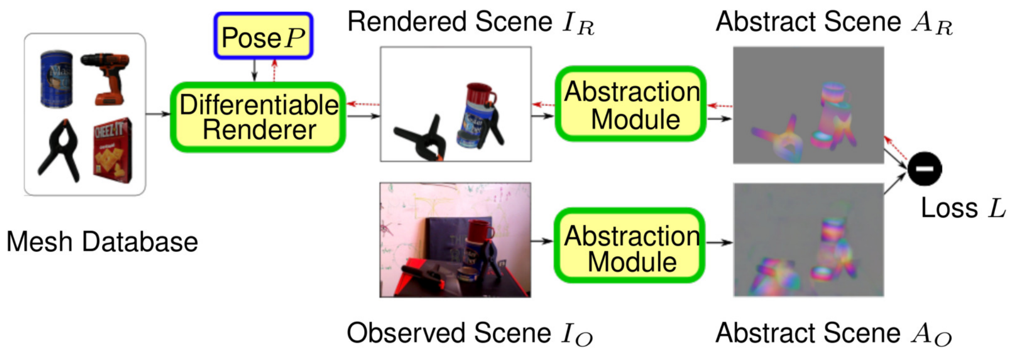


Figure 4.8: Render-Abstraction pipeline. The renderer produces an RGB image of the scene, which is then mapped into the abstract feature space. The loss gradient (red dotted line) is propagated back through the abstraction module and the differentiable renderer.

background does not result in covering/uncovering mug pixels, rather, more can pixels will become visible or become covered. Thus using the Sobel derivative is incorrect in this case. To address this issue, we detect such cases using the Z-buffer during rendering suppress the Sobel gradient on these pixels. We note that the occluded pixel belongs to the same object in this case, so that zero gradient should be a good approximation.

4.4.5 Propagating Image-space Gradients to Object Coordinates

While propagating the image-space gradients to the object coordinates, only gradient from the pixels belonging to the object needs to be propagated. The naïve way to do this is to mask the image-space gradient with rendered object mask. However, in certain situations this means we are ignoring exactly the pixels where a pixel-wise loss function generates high gradients, namely just outside of the rendered object boundary. Figure 4.7(c-d) illustrates this point.

To address this issue, we propose a dilation of the rendered object mask by one pixel, in order to include gradient information directly outside of the object boundary.

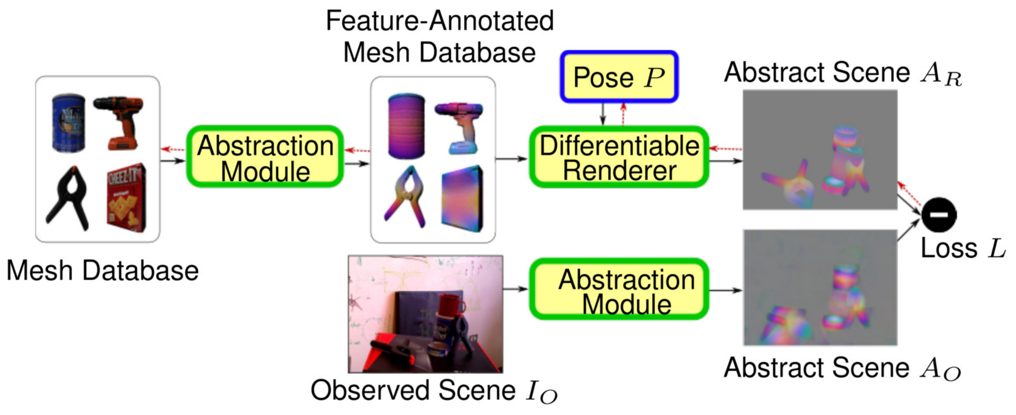


Figure 4.9: Scene analysis using differentiable rendering and learned abstraction module. An abstract representation is extracted both for the input scene and the mesh database. The differentiable rendering module then tries to match the abstract scene representation with the feature-annotated meshes. The loss gradient (red dotted line) only needs to be backpropagated through the differentiable renderer.

4.4.6 Evaluation

Armed with the abstraction module and our differentiable renderer, we can tackle the 6D pose refinement problem in cluttered real-world scenes. We first experimented with the architecture depicted in Fig. 4.8. The 3D scene with the objects in the current estimated pose P is rendered to generate image I_R. The abstraction module (see Section 4.4.1) is used to generate abstract representations A_R and A_O from the images I_R and I_O. The loss L is computed as the pixel-wise loss between A_R and A_O. Finally, we can derive the gradient of L with respect to the poses P:

$$\frac{\partial L}{\partial P} = \frac{\partial I_R}{\partial P} \cdot \frac{\partial A_R}{\partial I_R} \cdot \frac{\partial L}{\partial A_R}. \tag{4.6}$$

As depicted in Fig. 4.8, $\frac{\partial I_R}{\partial P}$ is approximated by the differentiable renderer and $\frac{\partial A_R}{\partial I_R} \cdot \frac{\partial L}{\partial A_R}$ is computed by standard backpropagation. During experiments, we noticed that the latter gradient is rather sparse and focuses most of its magnitude on few spatial locations in the image (see Fig. 4.10). This effect is well-known, for example in the field of adversarial example generation for CNNs (Goodfellow, Shlens, and Szegedy, 2015), (Palacio et al., 2018). Here, it is highly undesirable, since the differentiable rendering process works best with smooth, uniform gradients.

To mitigate this issue, we investigated a second pipeline shown in Fig. 4.9. We use the method described in Section 4.4.2 to create meshes with fused surface descriptors. Rendering these meshes directly results in the abstract rendered A_R. In this case, $\frac{\partial L}{\partial P}$ is simpler:

$$\frac{\partial L}{\partial P} = \frac{\partial L}{\partial A_R} \cdot \frac{\partial A_R}{\partial P}. \tag{4.7}$$

Here, $\frac{\partial A_R}{\partial P}$ is approximated directly by the differentiable renderer. An additional benefit of this variant is that only one forward pass of the abstraction module for A_O is required. This directly translates to significant reduction in the optimization process runtime.

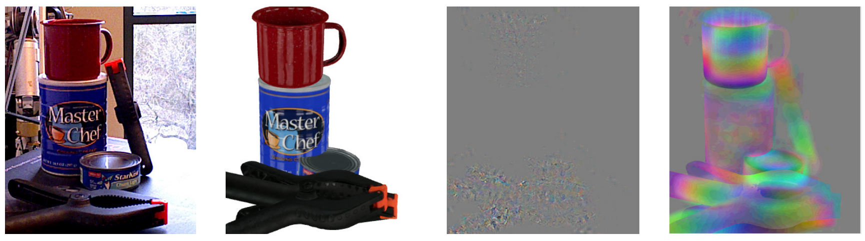


Figure 4.10: Gradient of pixel-wise loss w.r.t. the rendered scene in the two pipeline variants. (a) Observed scene. (b) Rendered scene. (c) Render-Abstraction pipeline (see Fig. 4.8). (d) Abstraction-Render pipeline (see Fig. 4.9). The actual gradient magnitudes are scaled for better visualization. Gray corresponds to zero gradient.

We optimize the object poses with the AdaGrad optimization scheme. This avoids manual tuning of learning rates for translation and rotation parameters—which differ largely in scale. In our experiments, we use a learning rate $\lambda = 1e - 2$ with a decay of 0.99. The optimization runs for 50 iterations, which corresponds to roughly 2 s per frame.

We perform our experiments on the YCB Video dataset (Xiang et al., 2018), which consists of 133,936 images extracted from 92 videos, showing 21 different objects in cluttered arrangements. Importantly, the dataset comes with high-quality meshes, which are also used for synthetic data generation by most of the pose estimation methods applied to the dataset (Xiang et al., 2018; Oberweger, Rad, and Lepetit, 2018).

To be able to control primary and secondary effects separately, we rely on our rendering pipeline and generate multiple scenes with random secondary parameters as detailed in Section 4.4. Our dense descriptor model is able to effectively suppress the background pixels and produces robust, consistent output under changing lighting conditions and camera model parameters.

For a quantitative analysis, we measure the ADD and ADD-S metrics as in (Xiang et al., 2018) for each object occurrence, which measure average point-wise distances between transformed objects and the ground truth, for non-symmetric and symmetric objects, respectively:

$$\text{ADD} = \frac{1}{m} \sum_{x \in \mathbb{M}} ||(Rx + T) - (\tilde{R}x + \tilde{T})||, \tag{4.8}$$

$$\text{ADD-S} = \frac{1}{m} \sum_{x_1 \in \mathbb{M}} \min_{x_2 \in \mathbb{M}} ||(Rx_1 + T) - (\tilde{R}x_2 + \tilde{T})||, \tag{4.9}$$

where R and T are the ground-truth rotation and translation, $\tilde{R}$ and $\tilde{T}$ denote the estimated pose, and $\mathbb{M}$ is the set of model points as included in the YCB Video dataset. We aggregate all results and measure the area under the threshold-accuracy curve for distance thresholds from zero to 0.1 m, which is the same procedure as in (Xiang et al., 2018).

We demonstrate pose refinement from the initialization of PoseCNN (Xiang et al., 2018) and the newer method by Oberweger, Rad, and Lepetit (2018). Figure 4.11 displays qualitative refinement examples, while Table 4.1 gives quantitative results. In our experiments, we assume that objects were correctly detected so that we can focus on the

Table 4.1: Pose refinement results on the YCB Video Dataset

	PoseCNN refined (ours)		HeatMaps refined (ours)	
Object	ADD(Δ)	ADD-S(Δ)	ADD(Δ)	ADD-S(Δ)
master_chef_can	63.3(+13.1)	91.7(+7.8)	76.7(−5.1)	90.2(−1.2)
cracker_box	65.3(+12.2)	81.7(+4.9)	82.9(−0.7)	89.4(−0.6)
sugar_box	85.3(+16.9)	92.0(+7.8)	86.4(+4.3)	92.2(+2.4)
tomato_soup_can	59.4(−6.8)	79.9(−1.1)	57.4(−22.4)	78.2(−11.3)
mustard_bottle	86.5(+5.5)	92.3(+1.9)	86.7(−4.7)	92.6(−2.4)
tuna_fish_can	81.1(+10.4)	94.3(+6.3)	69.7(+21.0)	85.7(+14.0)
pudding_box	71.1(+8.4)	83.1(+4.1)	68.8(−21.4)	80.7(−13.4)
gelatin_box	81.5(+6.3)	89.1(+1.9)	73.0(−20.7)	82.8(−13.1)
potted_meat_can	63.7(+4.2)	80.3(+1.8)	74.6(−4.5)	87.6(−2.4)
banana	82.1(+9.8)	91.8(+5.8)	68.8(+17.1)	81.0(+13.2)
pitcher_base	85.1(+31.8)	92.7(+15.7)	83.8(+14.4)	92.1(+7.1)
bleach_cleanser	65.0(+14.7)	80.4(+8.9)	78.3(+2.0)	87.6(+2.2)
bowl	6.5(+3.1)	75.5(+5.9)	1.5(−2.1)	66.4(−11.6)
mug	65.9(+7.4)	84.0(+5.9)	57.9(+4.0)	78.9(+3.1)
power_drill	73.7(+18.4)	85.9(+13.2)	81.5(−1.3)	90.4(−0.4)
wood_block	45.5(+18.9)	73.3(+9.0)	0.0(+0.0)	60.3(+3.3)
scissors	40.0(+4.1)	58.6(+1.7)	75.4(+10.1)	85.4(+5.8)
large_marker	63.9(+5.6)	77.3(+5.6)	59.8(+3.3)	70.2(+0.0)
large_clamp	37.0(+12.4)	65.1(+15.0)	75.3(+18.1)	85.6(+12.5)
extra_large_clamp	25.4(+9.3)	63.7(+19.6)	20.4(−3.1)	58.3(+3.7)
foam_brick	43.3(+3.1)	90.8(+2.8)	37.0(+5.0)	92.1(+3.2)
ALL	62.8(+9.1)	82.4(+6.6)	67.0(+0.7)	83.5(+1.1)

We report the area under the accuracy curve (AUC) for varying error thresholds on the ADD and ADD-S metrics.

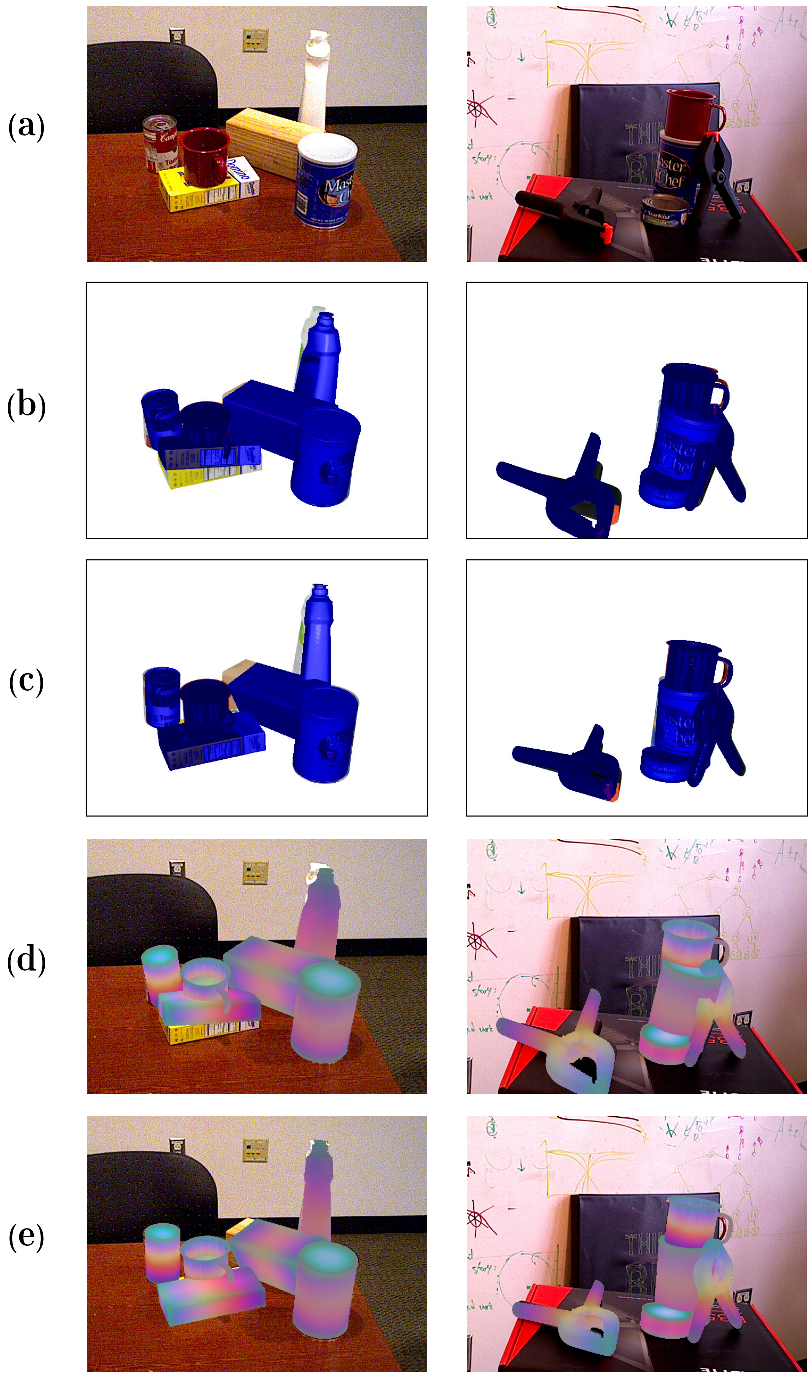


Figure 4.11: Qualitative examples from the YCB Video Dataset. (a): Observed scene. (b) and (c): Renderings (blue) with initial and optimized pose parameters, respectively. (d) and (e): Renderings of the feature-annotated meshes in initial and optimized poses, respectively.

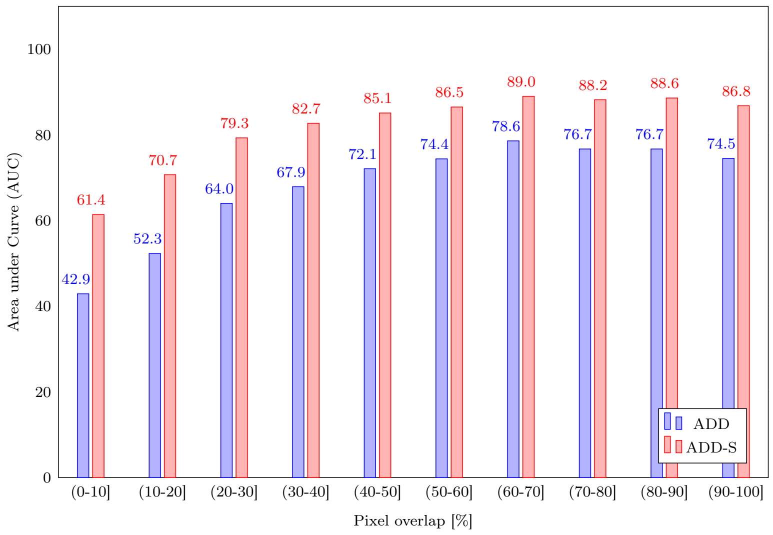


Figure 4.12: Basin of attraction in translation dimensions. We show the resulting ADD/ADD-S metrics for varying initial 2D overlap of ground-truth pose and initial estimate.

problem of refining poses rather than correcting detections. Our pipeline gives consistent improvements across nearly all objects of the dataset for the PoseCNN initialization. Note that we do not compare against the PoseCNN variant with ICP post-refinement, since our pipeline works with RGB only and ICP requires depth measurement. The improvement is especially significant for large and textured objects. On the initializations of Oberweger, Rad, and Lepetit (2018), which are already of very high quality, our gains are smaller. We hypothesize that our approach is currently limited by the spatial resolution of the computed feature representation.

Finally, compared to DeepIM (Li et al., 2018b), our method almost reaches the same overall performance. We note that the experiments performed in (Li et al., 2018b) apparently started from a better PoseCNN initialization than what was available to us, though the difference seems small. Interestingly, our method obtains significantly better results on a few object classes—suggesting that a combination of the techniques (e.g. by making the abstract representation and computed pose updates accessible to the DeepIM network) could yield further improvements.

To quantify the robustness of our render-and-compare pipeline to the quality of the initialization, we analyzed the basin of attraction of the refinement process. We experimented with 295 scenes from the validation set of YCB-Video dataset ($\sim$10 % of the total validation scenes) by randomly perturbing the translation and rotation components of the ground-truth poses to varying degrees and optimizing the perturbed poses. The translation perturbations were uniformly sampled in a range of $\pm$5 centimeters. Since the impact the translation perturbations has for an object depends of the size of the object, we compute the percentage of pixel overlap between the observed image and the rendered image for an object.

Similarly, we uniformly sample an axis of rotation and a rotation angle in the range $\pm$45 degrees. The AUC of the optimized pose with respect to different overlaps is shown in Fig. 4.12 and the rotation angle is shown in Fig. 4.13. Our method is able to robustly handle translation perturbations with almost no loss in accuracy down to 30% remaining overlap. In the rotation experiment, the ADD-S metric is almost unaffected by rotations

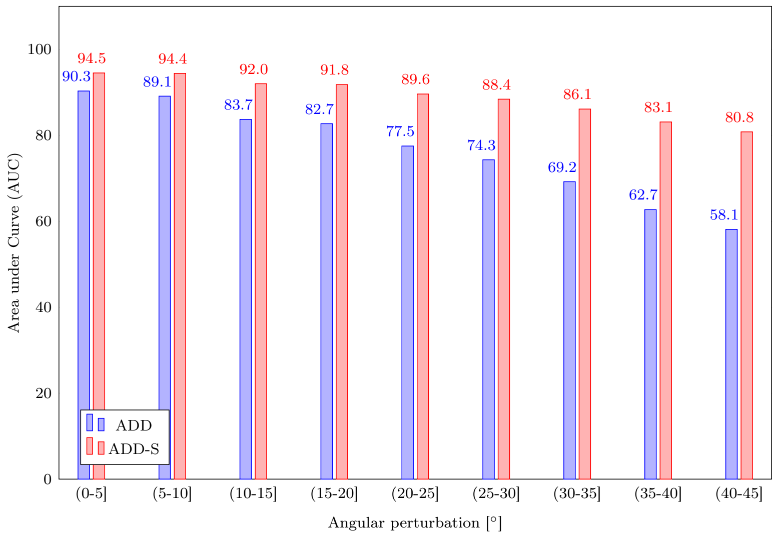


Figure 4.13: Basin of attraction in rotation dimensions. We show the resulting ADD/ADD-S metrics for varying initial angular perturbations from the ground-truth pose.

of up to 45°. The ADD metric drops off more steeply—this is caused by the entirely symmetric objects, where the system has no chance of correcting the perturbation around the symmetry axis.

4.5 Joint Object Shape and Pose Refinement using Render-and-Compare

Given a canonical model of an object category, 3D deformable registration aims at deforming the canonical model to match an observed instance while maintaining the geometric structure of the category. Our approach for solving deformable registration is closely related to DeepCPD (Rodriguez, Huber, and Behnke, 2020). DeepCPD uses *coherent point drift* (CPD) to create a low-dimensional shape-space of the object category and employs a CNN to estimate 3D deformation from single-view RGB images. Our pipeline presented in Section 4.5.2 uses differentiable rendering to estimate a 3D deformation field instead.

4.5.1 Latent Shape-Space

Given an object category with multiple instances and a canonical model, we use the *coherent point drift* (CPD) registration algorithm to learn a low-dimensional latent shape-space. The deformation τ_i of the canonical model $\mathbf{C}$ to an instance i is modeled as

$$\tau_i(\mathbf{C}_i, \mathbf{W}_i) = \mathbf{C} + \mathbf{G}(\mathbf{C}, \mathbf{C})\mathbf{W}_i. \tag{4.10}$$

where $\mathbf{W}$ is the deformation field and $\mathbf{G}$ is the Gaussian Kernel matrix defined element-wise as

$$\mathbf{G}(y_i, z_i) = g_{ij} = \exp(-\frac{1}{2\beta^2}||y_i - z_i||^2). \tag{4.11}$$

W is estimated in the M-step of the EM algorithm. Since the shape of $\mathbf{W}_i$ depends on the canonical model **C** and not the instance i, we reduce the dimension of $\mathbf{W}_i$ using *principle component analysis* (PCA). We use latent shape-space of dimension five for all the object categories in our experiments. We refer the reader to Rodriguez, Huber, and Behnke (2020), and Rodriguez et al. (2018) for a detailed explanation of the latent shape-space.

4.5.2 Deformable Registration Pipeline

In Fig. 4.14, we present a pipeline to deform the canonical model **C** to fit the observed image I_{obs} of a novel object instance. We generate the deformation field from the latent shape-space parameters $\mathcal{S}$ as described in Section 4.5.1 and render the deformed canonical model. We denote the rendered image as I_{rnd} using StilllebenDR. The rendered and the observed images are compared pixel-wise using the image comparison function described in Section 4.5.3. We implement the image comparison function and mesh deformation generation from $\mathcal{S}$ using PyTorch. The PyTorch autograd engine enables gradient propagation through these steps automatically. The gradient of the mesh parameters with respect to I_{rnd} is provided by StilllebenDR. As shown in Fig. 4.4, the gradient can be computed only for the faces that are visible in I_{rnd}. However, instead of applying the gradients directly to the mesh parameters, we propagate the gradient to the latent shape-space $\mathcal{S}$ and generate the mesh deformation from $\mathcal{S}$. This step ensures that all vertices deform coherently and maintain the geometry of the object category. We perform the forward rendering and gradient-based $\mathcal{S}$ update iteratively until the image comparison error is negligible.

4.5.3 Image Comparison

Comparing images pixel-wise in the RGB color space is error-prone. Zhang et al. (2018) and Zagoruyko and Komodakis (2015) demonstrated the effectiveness of the convolutional neural network feature space for image comparison. Inspired by the *learned perceptual image patch similarity metric* (LPIPS) (Zhang et al., 2018), we construct the image comparison function as shown in Fig. 4.15. We extract the features from the last layer before the output layer of the U-Net model (Ronneberger, Fischer, and Brox, 2015) for both rendered and observed images. We normalize the features between -1 and 1 and aggregate the features along the channel dimension. Finally, we compute mean squared error (MSE) of the aggregated features from the rendered and the observed images.

4.5.4 Evaluation

We evaluate the proposed 3D deformable registration on the DeepCPD dataset (Rodriguez, Huber, and Behnke, 2020). The dataset consists of four object categories: bottles, cameras, drills, and sprays (shown in Fig. 4.16). For each object category, the dataset provides a canonical model and a varying number of instances. Two instances per category are used for testing and the rest are used for training. All the instances of an object category are aligned to have a common coordinate frame. Since the proposed method does not involve training a separate model for modeling the deformation, we use the training instances only to train the U-Net model for semantic segmentation. We compare our

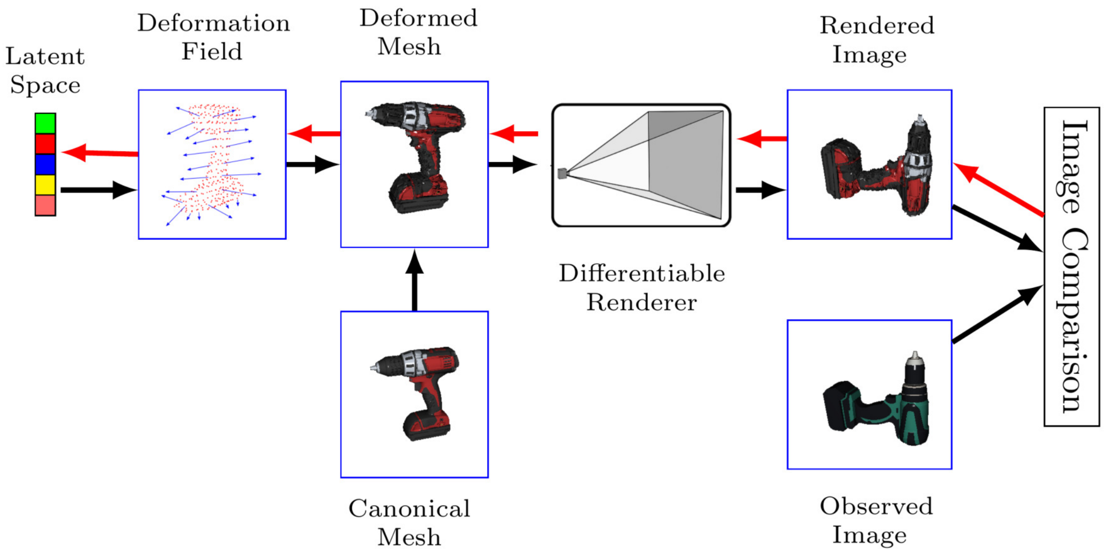


Figure 4.14: Proposed deformable registration pipeline. Latent shape-space parameters are optimized to minimize the difference between rendered image of the deformed mesh and the observed mesh. Image comparison loss is minimized using gradients obtained by differentiating through the rendering process. Black arrows indicate the forward rendering process and red arrows indicate the backward gradient flow.

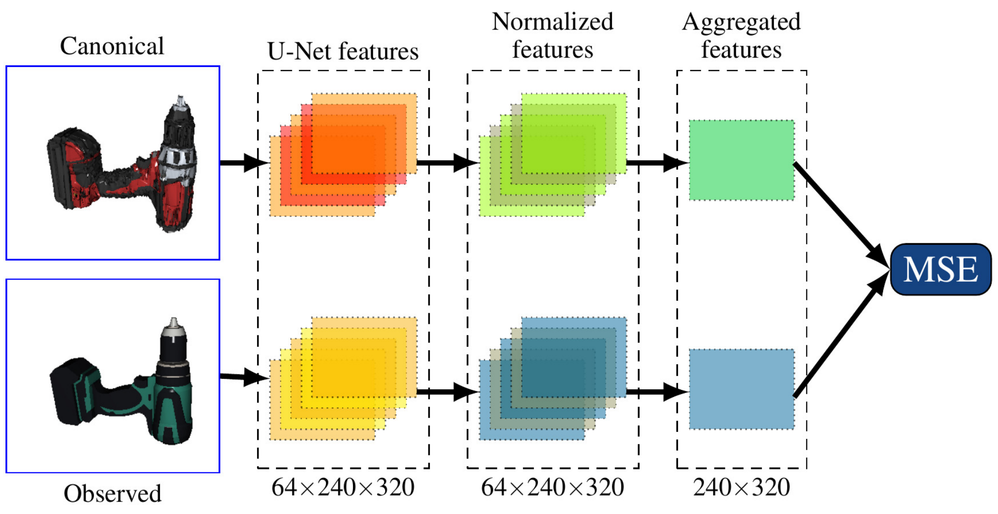


Figure 4.15: Image comparison operation. We compare the rendered canonical and the observed image using U-Net (Ronneberger, Fischer, and Brox, 2015) features. We normalize the extracted U-Net features and normalize them between -1 and 1 and aggregate the features along the channel dimension. Finally, we compute the mean-squared error pixel-wise between the aggregated features.

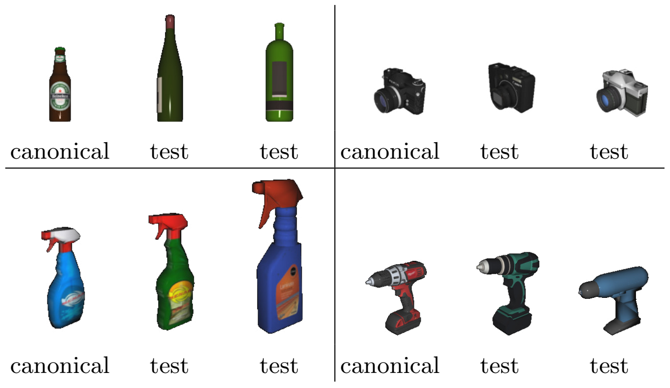


Figure 4.16: DeepCPD dataset with canonical instances and exemplary test instances.

method with CLS (Myronenko and Song, 2010) and DeepCPD (Rodriguez, Huber, and Behnke, 2020). CLS works with point clouds and thus needs depth information, whereas DeepCPD is RGB only.

Deformable Registration with Known Poses

We employ the proposed end-to-end-differentiable pipeline for deformable registration iteratively. We use stochastic gradient descent (SGD) with a momentum of 0.9 and exponential weight decay of 0.95. The meshes provided by the DeepCPD dataset are not watertight. Since the goal of our method is to deform the canonical model to fit the observed instance while maintaining the geometry of the object category, our pipeline does not benefit from having vertex colors. Thus, we use uniform red color for all the vertices in the canonical mesh. The tiny invisible holes on the surface of the meshes develop into larger visible holes during the iterative deformable registration process. This results in not only the rendered image looking unrealistic but also the image comparison being harder. To alleviate this issue, we use the *ManifoldPlus* algorithm (Huang, Zhou, and Guibas, 2020) to generate watertight meshes.

In Table 4.2, we report the average ℓ_2 distance between subsampled points of the canonical mesh and the test instances. Some qualitative visualizations are shown in Fig. 4.17. From the visualizations, one can observe that the rendered deformed mesh fits the observed mesh nicely. Our method not only works for objects with simple geometry like `bottles` but also for objects with complex geometry like `drills` and `sprays`. Quantitatively, our method performs only slightly worse than DeepCPD despite not employing any specialized learnable components to model the deformation.

Joint Deformable Registration and Pose Optimization

The accuracy of the deformable registration greatly depends on the quality of the pose estimation. Under the assumption that the exact pose of the observed instance is known, the competing methods CLS and DeepCPD perform slightly better than the proposed method. When the pose estimation is not accurate enough, the accuracy of the deformable registration degrades as well. In contrast to the methods in comparison, our end-to-end-differentiable pipeline can jointly optimize for 6D object pose along with deformable registration. To demonstrate this feature, we randomly sample offsets in the range of [-0.05, 0.05] m for the x and y translation components and [-15°and 15°] for the rotation

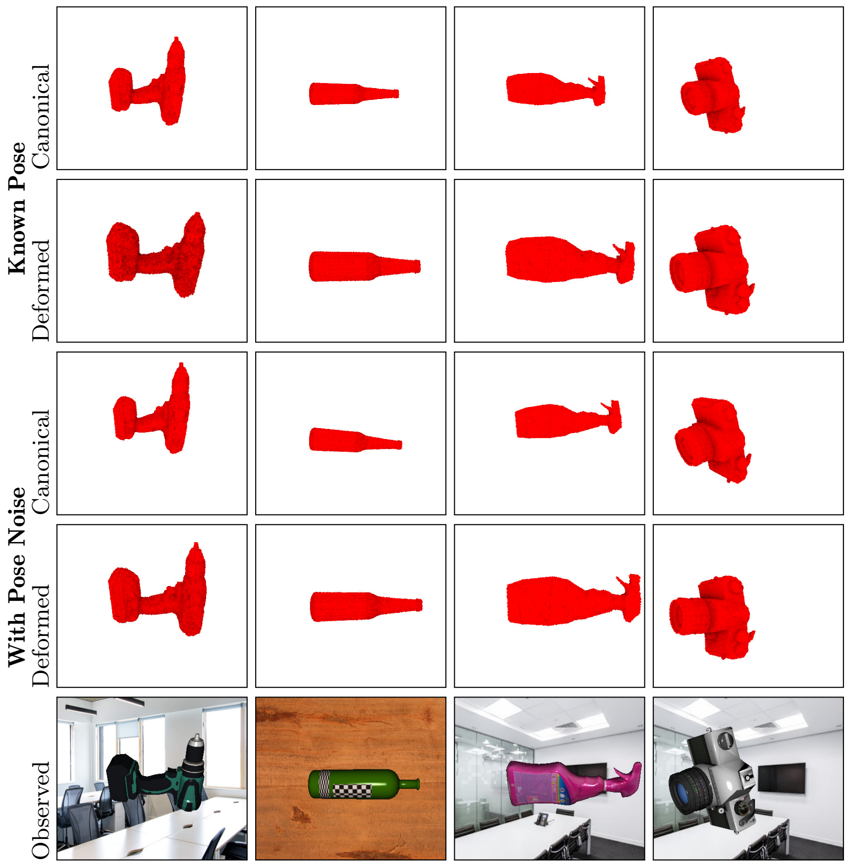


Figure 4.17: Visualization of 3D deformation. The canonical mesh is deformed to fit the observed mesh iteratively using differentiable rendering.

Table 4.2: Comparison of our approach with CLS (Myronenko and Song, 2010) and DeepCPD (Rodriguez, Huber, and Behnke, 2020). Mean and (standard deviation) error values in $10^{-5}m$.

Instance	Ground Truth	Known Pose			With Pose Noise		
		CLS (3D)	DeepCPD (RGB)	Ours (RGB)	CLS (3D)	DeepCPD (RGB)	Ours (RGB)
Camera T1	34.61	**51.93**	102.17	122.43	168.54	**105.26**	126.65
	(1.97)	(10.45)	(47.89)	(22.86)	(357.8)	(64.21)	(28.31)
Camera T2	16.45	**19.87**	18.80	66.54	406.45	306.96	**89.65**
	(1.61)	(4.59)	(5.11)	(29.73)	(492.03)	(127.89)	(33.54)
Bottle T1	23.25	**25.92**	45.21	52.63	297.79	227.90	**75.41**
	(2.34)	(5.18)	(9.75)	(19.45)	(579.49)	(146.0)	(34.23)
Bottle T2	90.42	**72.33**	88.35	112.84	852.40	289.36	**112.76**
	(28.54)	(11.35)	(18.39)	(25.78)	(1818)	(147.68)	(31.76)
Spray T1	29.84	**30.78**	47.87	77.74	1035	146.89	**89.59**
	(1.42)	(1.89)	(12.99)	(26.95)	(406.69)	(117.57)	(33.75)
Spray T2	111.94	**121.19**	154.97	151.21	1488	255.69	**178.42**
	(14.29)	(19.16)	(82.34)	(79.76)	(554.33)	(167.32)	(88.14)
Drill T1	21.18	**28.86**	52.71	71.54	232.35	92.96	**84.34**
	(0.949)	(1.42)	(23.54)	(34.56)	(1325)	(58.23)	(43.56)
Drill T2	63.95	**58.50**	119.88	134.21	215.54	262.31	**157.27**
	(5.23)	(21.51)	(107.43)	(89.16)	(565.48)	(228.40)	(96.36)

components. Although our method can optimize z translation along with other pose parameters, optimizing both z translation and vertex position jointly is an ill-posed problem. Thus, we include offsets only for x and y translation components. In our experiments, we observed that the pose parameters require fewer updates to converge than the shape parameters. Therefore, we update the shape parameters at a higher frequency than the pose parameters, i.e., we update the pose parameters once per three shape parameter updates. Quantitative results of joint pose and shape optimization are presented in Table 4.2. The mean error only increases marginally when pose noise is injected, indicating that our method is less susceptible to pose initialization errors than the competing methods.

4.6 Limitations

Large-scale deployment of the abstract render-and-compare framework proposed in this Chapter is limited by the following factors.

1. *Availability of high-quality meshes.* The abstract render-and-compare method is based on the assumption that a scene—excluding the secondary lighting effects—can be faithfully rendered using modern computer graphics pipelines. One of the prerequisites for an acceptable quality rendering is the availability of high-quality meshes. Both the dense descriptor learning approach discussed in Section 4.4.1 and the iterate pose and share refinement steps discussed in Sections 4.4.6 and 4.5 make use the 3D meshes. Despite the progress in 3D object reconstruction from RGB/D images, obtaining large-scale 3D meshes can be prohibitively expensive (Chang et al., 2015; Han, Laga, and Bennamoun, 2019; Choy et al., 2016; Fu et al., 2021).

2. *Complexity of the image comparison operation.* Comparing images pixel-wise is a complex operation. Moreover, gradient flow to the pose and shape parameters originates from the image comparison operator. Thus, image comparison operator not only needs to good at comparison but also differentiable and guarantee a smooth gradient flow. In this Chapter, we proposed the abstract space image comparison techniques instead of the standard RGB pixel space. The abstract space comparison alleviates many common issues in image comparison. However, due to the inherent complexity of image comparison, reliability remains open question in the pipelines we proposed. For example, in the pose refinement experiment discussed in Section 4.4.6, for some objects, the refinement step resulted in degraded accuracy. Similarly, in the joint pose and share refinement experiment discussed in Section 4.5, we noted that a clear background-foreground separation is needed for the proposed pipeline to work.

4.7 Discussion & Conclusion

In this Chapter, we introduced abstract render-and-compare pipelines for object pose refinement, and joint pose and shape refinement. To facilitate an efficient scene rendering, we introduced StilllebenDR. StilllebenDR augments the openly available Stillleben with differentiable rendering capabilities without adding a large overhead to the forward rendering process. We proposed a method to learn dense descriptors based on a pixel-wise contrastive loss function. The learned dense descriptor space is invariant to the secondary lighting effects in the scene. Equipped with the differentiable renderer and the dense descriptor, we proposed a pipeline for multi-object 6D object refinement. Moreover, we introduce a pipeline for joint pose and share refinement based on the latent shape space model and image comparison in neural network feature space. Furthermore, we highlighted major limitations facing our model. Here, we discuss recent works that provide opportunities to address these limitations. Li et al. (2018b) and Labbe et al. (2020) propose to learn pose updates directly given the rendered and the observed images. This circumvents the complexities in image comparison and analytical gradient flow through the rendering process. Also, the recent developments in differentiable volume rendering methods like NeRF (Mildenhall et al., 2021), 3D Gaussian Splatting (Kerbl et al., 2023; Lassner and Zollhofer, 2021) enable learning implicit object representations, removing the need for 3D object meshes. Yen-Chen et al. (2021) proposed inverting NeRFs for pose refinement. Deng et al. (2023) introduced a method for shape refinement based on NeRF. Shao et al. (2020) introduced a reinforcement learning based pose refinement as an alternative to differentiable rendering pipelines. Overall, we conclude that render-and-compare is a powerful technique for scene parameter refinement, and differentiable rendering is an integral component in realizing render-and-compare refinement pipelines.

5

Multi-Object Pose Estimation Using Direct Regression

Object detection and pose estimation are intertwined. The standard multi-staged methods do not capture the coupled nature of these tasks. To enable a single-staged pipeline, we formulate joint multi-object detection and pose estimation as a set prediction problem, and employ direct regression to estimate pose parameters. Moreover, we take advantage of the permutation-invariant nature of the attention mechanism to design our architecture for set prediction.

Statement of Personal Contribution

The contents presented in this Chapter is adapted from the following publication.

Arash Amini, Arul Selvam Periyasamy, and Sven Behnke

T6D-Direct: Transformers for multi-object 6D pose direct regression

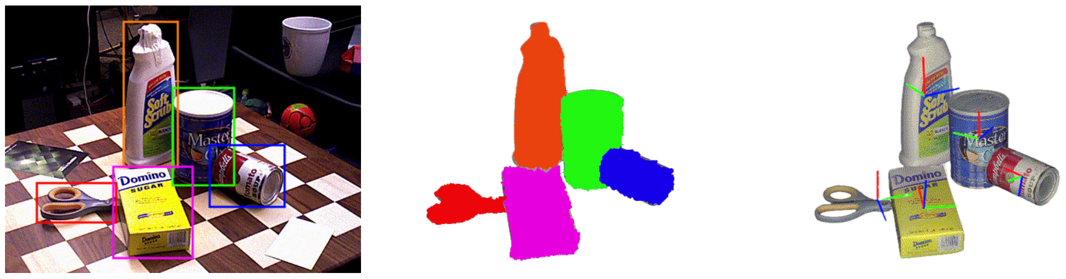
In: *43rd DAGM German Conference on Pattern Recognition (GCPR)*, Bonn, Germany, 2022.

The author of this thesis substantially contributed to all aspects of the publication, including the literature survey, the conception, formalization, design, and implementation of the proposed method, the preparation and conduct of experiments for the evaluation of the proposed approach, the analysis and interpretation of the experimental results, the preparation of the manuscript, as well as the revision and final editing of the version published.

5.1 Introduction

Object pose estimation is a long-standing problem in computer vision and it serves as a necessary pre-requisite for autonomous robots. Given an RGB input, the task aims at estimating the position and the orientation of the target objects in the camera coordinate frame. Pose of rigid body objects with respect to a specific frame of reference can be represented using six parameters—three for translation $\mathbf{t}$ and three for rotation R. The special Euclidean group SE(3) expressed using 4×4 matrices represents the set of rigid body transformations in three-dimensional Euclidean space.

$$\mathrm{SE}(3) = \left\{ A | A = \begin{bmatrix} R & t \\ 0_{1\times3} & 1 \end{bmatrix}, R \in \mathrm{R}^{3\times3}, t \in \mathrm{R}^3, R^T R = RR^T = I \right\}$$



(a) object detection (b) semantic segmentation (c) pose estimation

Figure 5.1: Subtasks in a standard pose estimation pipeline. Object detection and/or semantic segmentation is performed first and only the crops containing the target objects are processed by the pose estimation models. Employing modules like NMS, RoI pooling, and anchor boxes the pipeline is made end-to-end differentiable.

The translation component is commonly represented using three dimensional scalars, whereas several representations exists for orientation. The choice of the orientation representation is dependent on the application domain.

The standard methods for object pose estimation are multi-staged. In the first stage, object detection is performed. Object detection can either be formulated as joint bounding box and class probabilities prediction (shown in Fig. 5.1 a) or as semantic/instance segmentation-driven (shown in Fig. 5.1 b) crop extraction. In the subsequent stage, the crop containing the target object is processed by the pose estimation module to generate the 6D pose parameters (shown in Fig. 5.1 c). The ground-truth object class labels are represented using *one-hot encoding*. The class probabilities prediction is formulated as multinomial logistic regression employing the *softmax function*. The 2D bounding box are modeled using four parameters, which can be represented using two different standard conventions: the pixel coordinates of the object center along with the height and the width of the bounding box and the pixel coordinates of the top left and bottom right corners. The semantic segmentation task is formulated as pixel-wise classification. The multi-staged pipelines have two major shortcomings. Firstly, realizing an end-to-end differentiable multi-staged pipeline is not trivial. Special modules like *non maximum suppression* (NMS),*region of interest* (RoI) pooling, and *anchor boxes* are needed to make the multi-staged pipelines end-to-end differentiable. Secondly, the sequential nature of the multi-staged pipelines do not exploit the intertwined nature of the subtasks. To this end, we introduce the T6D-Direct model, a single-stage multi-object pose estimation model.

5.2 Related Work

In accordance with many of the computer vision problems, the deep learning methods dominate the state-of-the-art landscape for object pose estimation. Prior to the advent of learning-based methods, methods for pose estimation can be broadly classified into template-matching-based and sparse-feature-based. The early machine learning methods for pose estimation based on Hough forests, regression forests, gradient boosted decision trees laid the foundations for deep learning methods. We review these algorithms in the following section briefly.

Template-matching-based Methods

In template-matching-based methods (Cao, Sheikh, and Banerjee, 2016; Hinterstoißer et al., 2012; Hinterstoisser et al., 2013), a rigid template of the target object is constructed offline and matched against different image locations to compute a similarity score. The pose corresponding to the best-matched template is assigned to the target object. By formulating the pose estimation as a similarity search problem, the template-matching-based methods benefited from the developments made in the similarity search research field—particularly in efficient inference. However, the *curse of dimensionality* severely hindered the applicability of similarity search for computer vision problems. To counter this, dimensionality reduction techniques like *principal component analysis* (PCA) and parallel processing using GPGPUs were employed to speed up the template-matching process. Despite these improvements, the template-matching-based methods performed poorly under occlusion. Also, variations due to view point, lighting conditions, and secondary lighting effects had an adverse effect on performance of the template-based-methods.

Sparse-feature-based Methods

Lowe (2004) introduced the landmark work *scale invariant feature transforms* (SIFT). The SIFT algorithm detects and describes distinctive features in an image that are invariant to scale, rotation, and affine transformations. It led to significant improvements in many computer vision tasks. Many of the follow-up works including SURF (Bay, Tuytelaars, and Gool, 2006) BRISK (Leutenegger, Chli, and Siegwart, 2011), ORB (Rublee et al., 2011), HOG (Dalal and Triggs, 2005) proposed improvements to SIFT in terms of the robustness and execution speed. Using 2D-3D correspondences between image features and model keypoints, 6 DoF object poses are estimated employing the Perspective-n-Point (PnP) algorithm. The putative 2D-3D matching is susceptive to outliers. To deal with outliers, PnP algorithm is combined with the RANSAC paradigm (Hao et al., 2013; Peñate Sanchez et al., 2013; Chum and Matas, 2005; Sattler, Leibe, and Kobbelt, 2011). Similar to the template-matching-based methods, occlusion hindered the performance sparse-feature-based methods as well.

Learning-based Methods

Tejani et al. (2014) introduced *latent-class Hough Forests* by integrating scale-invariant patch descriptors into regression forest using a template-based split function. Schulter et al. (2013) proposed *alternating regression forests* for object detection and pose estimation employing random forest by optimizing a global loss function over all trees. Lai et al. (2011) presented scalable approach for object pose estimation based on gradient boosted decision trees. Hara and Chellappa (2014) introduced a node splitting method for regression trees and incorporated it into the regression forest framework for car direction estimation. Although the early learning-based methods made considerable progress in terms of pose estimation accuracy and inference speed, compared to the deep learning methods they scaled poorly with data. With the emergence of large-scale datasets and GPGPU compute power, these methods where phased-out in favour of deep learning methods.

CNN-based Pose Estimation

Modern deep-learning-based methods augment the classical methods with learned features and components. While some of the modern methods strive for end-to-end learnable pipelines, some others combine learnable modules with classic computer vision algorithms. In this section, we review some of the prominent CNN-based works in recent years.

Xiang et al. (2018) introduced PoseCNN, an end-to-end differentiable pipeline for object pose estimation from RGB images. Additionally, they introduced the YCB-Video dataset, which we use for evaluating the methods we propose in this thesis. PoseCNN jointly learned semantic segmentation and 6D pose estimation using a common convolutional neural network CNN backbone. The segmentation branch utilized fully convolutional layers to learn pixel-wise class probability distribution and the pose estimation branch had two stages. In the first stage, for each object pixel, a direction vector pointing towards the pixel onto which the center of the object is projected, and the distance between the camera and the object center is estimated. In the second stage, using the segmentation estimate from the first stage, crops containing the target objects are determined. The orientation regression module estimates the orientation component of the object pose, which is represented as quaternions. The translation component is estimated from the dense direction vector and the depth estimate using a novel differentiable hough voting layer. Utilizing RoI pooling layers while cropping, PoseCNN is designed to be end-to-end differentiable. Kehl et al. (2017) observed that the gradient flow is well-behaved in the case of classification tasks compared to the regression tasks. Based on this observation, instead of regressing the orientation parameters, Kehl et al. (2017) discretized the orientation space and performed classification. Oberweger, Rad, and Lepetit (2018) introduced a sparse-keypoint-based approach, in which the pixel projections of the 3D bounding box corners were estimated using fully convolutional networks. Knowing the 2D-3D correspondences, the 6D object pose is estimated employing the perspective n points (PnP) algorithm. Peng et al. (2019) argued that estimating pixel projections of 3D bounding box corners is harder since the projections, often, lie outside of target object pixels. And to address this issue, they proposed to select keypoints on the surface of the object employing the *farthest point sampling* (FPS) algorithm. Sundermeyer et al. (2018) formulated the problem of pose estimation as nearest neighbor search in the Augmented Autoencoder (AAE) latent space. Similar to the training regime of Denoising Autoencoder (DAE) (Vincent et al., 2010), in which a CNN is trained to recreate the noise-free image from the corrupted RGB input, AAE is trained to recreate noise-free occlusion-free secondary-light-effects-invariant output image from the augmented input image. The latent space of AAE serves as a low-dimensional representation of the input image. After training the AAE model, a codebook of latent spaces corresponding to images covering the discretized $SE(3)$ manifold is created. During inference, the latent space of the input image is extracted and the nearest neighbor from the codebook is found. The 6D pose corresponding to the nearest neighbor is assigned to the input image. An orthogonal class of methods to the ones we discussed so far are the refinement-based methods (Labbe et al., 2020; Li et al., 2018b; Periyasamy, Schwarz, and Behnke, 2019), which formulate the pose estimation task as a problem of pose refinement. Given an observed RGB image, an initial pose estimate, and the 3D model of the target object, an image is rendered according to the current pose estimate. The pose refinement methods estimate a pose update that minimizes the differences between the rendered and the observed images.

The pose update step is repeated iteratively until the pose update becomes negligibly small.

Vision Transformer Models

Deep learning methods for computer vision tasks typically utilized CNNs. Vaswani et al. (2017) introduced the Transformer architecture based on multi-head attention for learning long-term dependencies in natural language processing tasks. Since the attention mechanism is permutation-invariant, to preserve the order of input sequences, the author proposed positional encodings. Cordonnier, Loukas, and Jaggi (2020) studied the relationship between the convolutional operation and attention mechanism and mathematically showed that a special case of attention mechanism in conjunction relative positional embedding can be cast as the convolution operation. Dosovitskiy et al. (2021) introduced vision transformer (ViT), a transformer architecture for computer vision without any convolutional layers that performed comparably or better than CNN-based architectures in many computer vision applications. Several subsequent works utilized transformer architecture to supplement CNNs or to completely replace CNNs and achieved state-of-the-art results in many computer vision tasks (Khan et al., 2022; Wang et al., 2020; Carion et al., 2020).

Multi-Object Pose Estimation Models

Several *fully convolutional neural networks* (Long, Shelhamer, and Darrell, 2015) were proposed for multi-object pose estimation (Hu et al. (2019), Zakharov, Shugurov, and Ilic (2019), Capellen, Schwarz, and Behnke (2019), Periyasamy et al. (2020)). These models utilize a multi-branch feature maps for generating segmentation masks and pixel-wise dense 2D-3D correspondences. During training, the feature maps are supervised using ground-truth segmentation and dense correspondence maps. During inference, from these maps, the 6D object poses for all the target objects are generated using PnP+RANSAC scheme. Given the ground-truth maps, training these models adds only a small overhead compared to the standard semantic segmentation models. The major limitation of these models is that the PnP+RANSAC scheme to recover 6D object pose from dense pixel correspondences is not differentiable. Moreover, to ensure robust 6D pose predictions from the dense 2D-3D correspondences, multiple RANSAC evaluation are needed. Thus, the overall inference time is much larger.

Object Detection as Set Prediction

Carion et al. (2020) introduced DETR, an end-to-end differentiable architecture that combined CNN and the attention mechanism for object detection. An interesting aspect of the DETR architecture is the formulation of object pose estimation as a set prediction problem. Exploiting the permutation-invariant nature of the attention operation, DETR generates a set of predictions. Each element in the set is a tuple of class probabilities and 2D bounding box parameters. Using bipartite matching, the model is trained end-to-end without any special layers like NMS, RoI pooling, or anchor boxes. We take inspiration from the DETR architecture in proposing our architecture for 6D object pose estimation, which we discuss in the following sections.

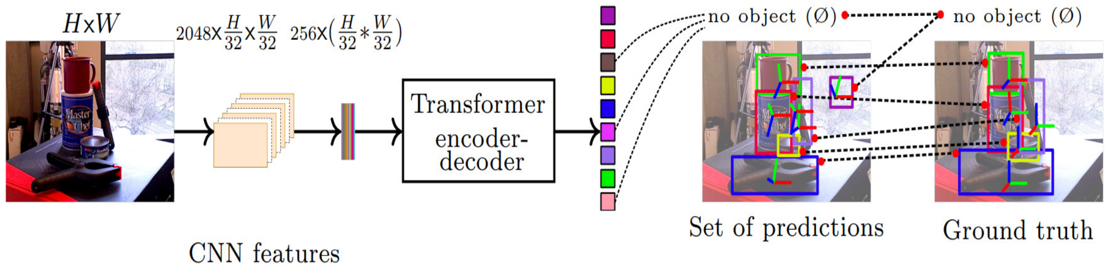

HxW
$2048X\frac{H}{32}X\frac{W}{32}$
$256X(\frac{H}{32}*\frac{W}{32})$
Transformer encoder-decoder
no object (Ø)
no object (Ø)
Set of predictions
Ground truth
CNN features

Figure 5.2: T6D-Direct overview. Given an RGB image, we use a CNN backbone to extract lower-resolution image features and flatten them to create feature vectors suitable for a standard Transformer model. The Transformer model generates a set of predictions with a fixed cardinality N. To facilitate the prediction of a varying number of objects in an image, we choose N to be much larger than the expected number of objects in an image and pad the rest of the tuples in the set with Ø object predictions. We perform bipartite matching between the predicted and ground truth set to find the matching pairs and train the pipeline to minimize the Hungarian loss between the matched pairs.

5.3 T6D-Direct: Multi-Object 6D Pose Using Direct Regression

In this section, we introduce T6D-Direct, a transformer model for multi-object for 6D pose direct regression. We discuss its architecture, the pose estimation as set prediction formulation and the differentiable matching procedure, the representations for direct regression of pose parameters, the loss function used to train the model.

5.3.1 Multi-Object 6D Pose Regression as Set Prediction

Inspired by DETR (Carion et al., 2020), we formulate multi-object pose estimation as a set prediction problem. An overview of the T6D-Direct model is shown in Fig. 5.2. Given an RGB input, our model generates a set of predictions of cardinality N. Each element in the set is a tuple consisting of class probability estimation, 2D bounding box detection and 6D object pose parameters. We employ a ResNet backbone for feature extraction. Positional encoding compensates for the loss of spatial information in the permutation-invariant attention computation. Combined image features and positional encodings are provided to the encoder module, which uses the multi-head self-attention mechanism to generate encoder feature embeddings. In the decoder, the cross-attention mechanism is employed between the encoder feature embeddings and a set of N learned embeddings called object queries to generate N object embeddings, which are then processed by feed-forward networks (FFNs) to generate 2D bounding box and class probabilities, in parallel. Since the model is designed to generate a set of predictions of fixed cardinality N, it is trained to predict Ø classes after detecting all the target objects present in the image. By associating predictions and ground truth objects with a bipartite matching algorithm (Kuhn, 1955), our model is trained end-to-end and jointly detects all objects and estimates their pose in the given image in one forward pass without the need for specialized layers.

In Fig. 5.3, we present the detailed architecture of the T6D-Direct model. Given an RGB input of height H and width W, the ResNet50 (He et al., 2016) backbone model ex-

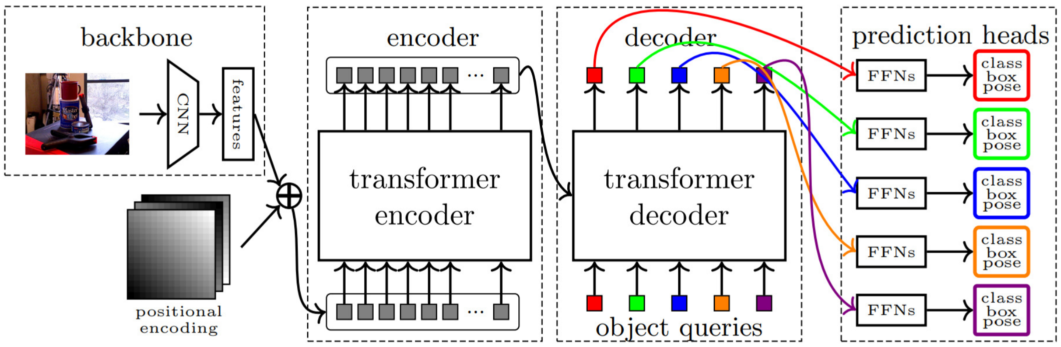


Figure 5.3: T6D-Direct architecture in detail. Flattened positional encoded image features from a backbone model are made available to each layer of the transformer encoder. The output of the encoder is provided as input to the decoder along with positional encoding. However, unlike the encoder that takes fixed sine & cosine positional encoding, we provide learned positional encoding to the decoder. We call these learned positional encoding *object queries*. Each output of the decoder is processed independently in parallel by shared prediction heads to generate a set of N tuples each containing the class probabilities, the bounding boxes, and the translation and the orientation components of the 6D pose. Since the cardinality of the set is fixed, after predicting all the objects in the given image, we train the model to predict Ø object for the rest of the tuples.

tracts feature maps of dimensions $2048{\times}H/32{\times}W/32$. Using $1{\times}1$ convolutions we reduce the dimension of the features and vectorize them to $d{\times}(H/32{*}W/32)$. To compensate for the loss of spatial information in the permutation-invariant attention computation, we use absolute positional encodings (P.E.) (Vaswani et al., 2017; Carion et al., 2020). The pixel coordinates are represented as sine and cosine functions of different frequencies:

$$\text{P.E.}_{(pos,p)} = sin(pos/10000^{\frac{2p}{d}}),$$

$$\text{P.E.}_{(pos,p+1)} = cos(pos/10000^{\frac{2p+1}{d}}),$$

where pos is the pixel coordinate (either width or height), d is the embedding dimension, and p is the index of the positional encoding. The positional embeddings are added to the backbone feature vectors before feeding them to the transformer encoder as input.

Multi-Head Attention (MHA)

Given a query token z belonging to a set of query tokens Ω_z and a key token x belonging to a set of key tokens Ω_x, the multi-head attention for the element with index q is computed as:

$$\text{MHA}(z_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k\in\Omega_k} A_{mqk} \cdot W^{'}_m x_k \right], \tag{5.1}$$

where $m \in M$ represents the attention head, $W^{'}_m \in \mathbf{R}^{d_v \times d}$ and $W_m \in \mathbf{R}^{d \times d_v}$ are learnable projection parameters, $d_v = d/M$ and A represents the normalized attention weight. MHA is a core component of the transformer architecture. In contrast to the convolution operation, which limits the receptive field to a small neighborhood, self-attention enables a receptive field of the size of the whole image. Note that the convolution operation can be cast as a special case of self-attention (Cordonnier, Loukas, and Jaggi, 2020).

Transformer Encoder and Decoder

The Transformer encoder module consists of six encoder layers with skip connections. Each layer performs multi-head self-attention of the input vectors, i.e. the image features act as both the query and the key. The output of the Encoder is referred as *encoder embeddings*. The decoder also consists of six decoder layers with skip connections, which perform cross-attention between the *encoder embeddings* and the learned embeddings referred to as *object queries* to generate object embeddings. Object Queries randomly initialized during training, learned jointly with the model parameters during training, and remain fixed during inference.

FFNs

From the N object embeddings, we use feed-forward networks (FFNs) to generate a set of N output tuples independently. Each tuple consists of the class probability, the bounding box, and the pose parameters. Prediction heads are fully-connected three-layer MLPs with hidden dimension of 256 and ReLU activation in each layer.

Bipartite Matching

Given n ground truth objects $y_1, y_2, ..., y_n$, we pad $\varnothing$ objects to create a ground truth set y of cardinality N. To match the predicted set $\hat{y}$, generated by our T6D-Direct model, with the ground truth set y, we perform bipartite matching. Formally, we search for the permutation of elements between the two sets $\sigma \in \mathfrak{S}_N$ that minimizes the matching cost:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}), \tag{5.2}$$

where $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$ is the pair-wise matching cost between the ground truth tuple y_i and the prediction at index $\sigma(i)$. DETR model included bounding boxes b_i and class probabilities p_i in their cost function. In the case of T6D-Direct model, we have two options for defining $\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$. One option is to use the same definition used by the DETR model, i.e., we include only bounding boxes and class probabilities and ignore pose predictions in the matching cost. We call this variant of matching cost as $\mathcal{L}_{match\_object}$.

$$\mathcal{L}_{match\_object}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{c_i \neq \varnothing} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{c_i \neq \varnothing} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}). \tag{5.3}$$

The second option is to include the pose predictions in the matching cost as well. We call this variant $\mathcal{L}_{match\_pose}$.

$$\begin{aligned} \mathcal{L}_{match\_pose}(y_i, \hat{y}_{\sigma(i)}) = \mathcal{L}_{match\_object}(y_i, \hat{y}_{\sigma(i)}) + \\ \mathcal{L}_{rot}(R_i, \hat{R}_{\sigma(i)}) + \mathcal{L}_{trans}(t_i, \hat{t}_{\sigma(i)}), \end{aligned} \tag{5.4}$$

where $\mathcal{L}_{rot}$ is the angular distance between the ground truth and predicted rotations, and $\mathcal{L}_{trans}$ is the ℓ_2 loss between the ground truth and estimated translations.

In our experiments, both the variants performed equally well. However, the first variant was comparatively faster since computing $\mathcal{L}_{match\_pose}$ was significantly expensive. Thus, we used the first variants in the rest of the experiments.

5.3.2 Loss Function

After establishing the matching pairs using the bipartite matching, the T6D-Direct model is trained to minimize the *Hungarian loss* between the predicted and ground truth target sets consisting of probability loss, bounding box loss, and pose loss:

$$\mathcal{L}_{Hungarian}(y,\hat{y}) = \sum_{i}^{N}[-\log\hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{c_i\neq\varnothing}\mathcal{L}_{box}(b_i,\hat{b}_{\hat{\sigma}(i)})+ \\ \lambda_{pose}\mathbb{1}_{c_i\neq\varnothing}\mathcal{L}_{pose}(R_i,t_i,\hat{R}_{\hat{\sigma}(i)},\hat{t}_{\hat{\sigma}(i)})]. \quad (5.5)$$

Class Probability Loss

The first component in the *Hungarian loss* is the class probability loss. We use the standard negative log-likelihood (NLL) loss as the class probabilities loss function. Additionally, the number of ∅ classes in a set is significantly larger than the other object classes. To counter this class imbalance, we weight the log probability loss for the ∅ class by a factor of 0.4.

Bounding Box Loss

The second component in the *Hungarian loss* is bounding box loss $\mathcal{L}_{box}(b_i,\hat{b}_{\sigma(i)})$. We use a weighted combination of generalized IoU ((Rezatofighi et al., 2019)) and ℓ_1 loss.

$$\mathcal{L}_{box}(b_i,\hat{b}_{\sigma(i)}) = \alpha\mathcal{L}_{iou}(b_i,\hat{b}_{\sigma(i)}) + \beta||b_i - \hat{b}_{\sigma(i)}||, \quad (5.6)$$

$$\mathcal{L}_{iou}(b_i,\hat{b}_{\sigma(i)}) = 1 - \left(\frac{|b_i \cap \hat{b}_{\sigma(i)}|}{|b_i \cup \hat{b}_{\sigma(i)}|} - \frac{|B(b_i,\hat{b}_{\sigma(i)}) \setminus b_i \cup \hat{b}_{\sigma(i)}|}{|B(b_i,\hat{b}_{\sigma(i)})|}\right), \quad (5.7)$$

where α, β are hyperparameters and $B(b_i,\hat{b}_{\sigma(i)})$ is the largest box containing both the ground truth b_i and the prediction $\hat{b}_{\sigma(i)}$.

Pose Loss

The third component of the *Hungarian loss* is the pose loss. Inspired by Wang et al. (2021), we use the disentangled loss to individually supervise the translation t and rotation R via employing symmetry aware loss (Xiang et al., 2018) for the rotation, and ℓ_2 loss for the translation.

$$\mathcal{L}_{pose}(R_i, t_i, \hat{R}_{\sigma(i)}, \hat{t}_{\sigma(i)}) = \mathcal{L}_R(R_i, \hat{R}_{\sigma(i)}) + ||t_i - \hat{t}_{\sigma(i)}||, \tag{5.8}$$

$$\mathcal{L}_R = \begin{cases} \frac{1}{|\mathcal{M}|} \sum\limits_{\mathrm{x}_1 \in \mathcal{M}} \min\limits_{\mathrm{x}_2 \in \mathcal{M}} ||(R_i \mathrm{x}_1 - \hat{R}_{\sigma(i)} \mathrm{x}_2)|| & \text{if symmetric,} \\ \frac{1}{|\mathcal{M}|} \sum\limits_{\mathrm{x} \in \mathcal{M}} ||(R_i \mathrm{x} - \hat{R}_{\sigma(i)} \mathrm{x})|| & \text{otherwise,} \end{cases} \tag{5.9}$$

where $\mathcal{M}$ indicates the set of 3D model points. Here, we subsample 1500 points from provided meshes. R_i is the ground truth rotation and t_i is the ground truth translation. $\hat{R}_{\sigma(i)}$ and $\hat{t}_{\sigma(i)}$ are the predicted rotation and translation, respectively.

5.3.3 Direct Regression of Pose Parameters

The ability of a machine learning model to learn a task greatly depends on the representation used to encode the task output. For each object, our model directly regresses the 6D pose parameter. The natural choice of representation for the translation component is three dimensional vector (R^3). However, representation for the orientation component is not straightforward. The orientation parameters belong to the *special orthogonal group*, (SO(3)). Several orientation representations exist in the literature. We review these representations briefly and discuss our representation of choice.

Euler Angles

Euler angles represent the orientation of an object using three angles, each corresponding to a rotation around one of the principal axes. The sequence of these rotations can vary, leading to different conventions such as XYZ, ZYX, etc. Euler angles are easy to understand, visualize, and only need three parameters. However, they suffer two major limitations. Firstly, Euler angles are not unique. Non-unique representations are not suitable for machine learning applications. Secondly, they suffer from a phenomenon known as *gimbal lock* in which two of the three axes align, causing a loss of one degree of freedom and making it impossible to represent certain rotations.

Axis-angle Representation

The axis-angle representation defines a rotation by specifying an axis around which the rotation occurs and the magnitude of the rotation about that axis. Thus, it has four parameters. However, both the axis and the angle can represented by a three dimensional vector (referred as *rotation vector* or the *Euler vector*) co-directional with the rotation axis whose length is the rotation angle. Although the axis-angle representation does not suffer from gimbal lock, it is also not a non-unique representation, since rotation by an angle θ is same as the rotation by the angle -θ.

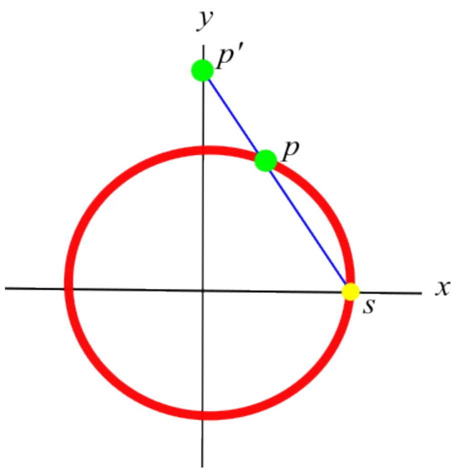


Figure 5.4: Stereographic projection in 2D. The point p on the unit circle is projected on to the y-axis using the fixed projection point s.

Quaternions

Complex numbers $\mathbb{C}$ represent rotation in 2D. Quaternions q represent rotations in 3D by extending complex numbers to one real part and three imaginary parts i, j, k, such that

$$i^2 = j^2 = k^2 = ijk = -1,$$

and take a general form

$$q = a + bi + cj + dk,$$

where a, b, c, d form the four parameters of quaternions. While quaternions provide an elegant and computationally efficient representation for working with rotations, it is also a non-unique representation. An unit quaternion q and its conjugate $-q$ represent the same rotation.

5D & 6D Continuous Representations

The standard representations we discussed so far are widely used in various scientific disciplines serving a wide range of use cases. However, the results from neural network approximation theory showed that to learn discontinuous functions the neural networks need more number of neurons as well as training iterations and the resulting accuracy is inferior compared to learning continuous functions (Xu and Cao (2005), Xu and Cao (2004), LeCun, Bengio, and Hinton (2015)). Moreover, empirical evaluation of the standard rotation representations also shows larger errors in the vicinity of discontinuity (Zhou et al. (2022)). Motivated by these results, Zhou et al. (2022) argued that *uniformity* and *continuity* are prerequisites for any neural network-compatible representation. Representations that are four dimensional or fewer do not satisfy this criteria. To this end, they proposed 5D & 6D continuous rotation representations. The 6D representation is formed by dropping the last column vector of the 3×3 rotation matrix. The reverse mapping from the 6D representation to corresponding the 3×3 rotation matrix is defined by the Gram-schmidt orthogonalization procedure (Leon, Björck, and Gander (2013)). Stereographic projection allows dimensionality reduction from R^m to R^{m-1}. In Fig. 5.4, we illustrate the stereographic projection in 2D. A point p on the unit circle is project

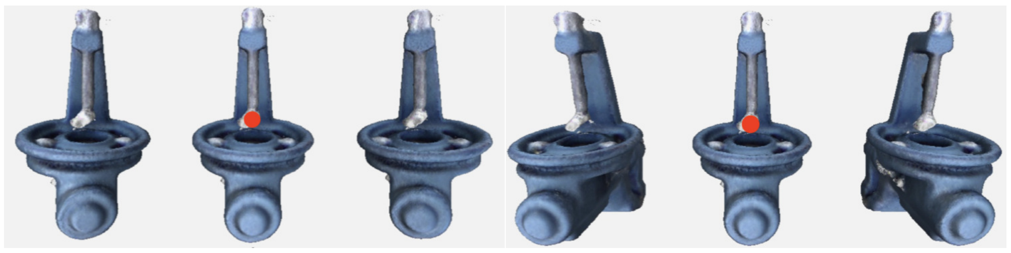


Figure 5.5: Allocentric & egocentric representations. The intersection of the camera ray with the image plane is shown using red dots. Object appears different under egocentric representation despite having the same orientation parameters due to translation lateral to the image plane. Camera intrinsics are chosen to exaggerate visual differences. This phenomenon is not observed under allocentric representation. Figure adapted from Manhardt et al. (2020).

on to the y-axis using a ray from the fixed projection point $s{=}(1,0)$. Formally, a vector $u \in \mathrm{R}^m$ is projected onto to $v \in \mathrm{R}^{m-1}$ as follows.

$$v = \left[\frac{w_2}{1-w_1}, \frac{w_2}{1-w_1}, ..., \frac{w_m}{1-w_1}\right]^T, w = \frac{u}{||u||}.$$

The vector $v \in \mathrm{R}^{m-1}$ can be projected back to R^m using stereographic un-projection as follows.

$$u = \frac{1}{||v||}\left[\frac{1}{2}\left(||v||^2 - 1\right), u_1, ..., u_{m_1}\right]^T.$$

The stereographic projection and un-projection enables conversion between 6D and 5D continuous rotation representations. We utilize these representation for directly regressing the pose parameters in the T6D-Direct model.

Allocentric & Egocentric Representations

Pose of an object can be encoded using either allocentric or egocentric representations. In Fig. 5.5, we depict the differences between the two representations. Egocentric representation encodes pose relative to the current position and orientation of the camera, whereas allocentric representation encodes spatial position based on a fixed reference frame that is independent of the camera pose. Under egocentric representation, mere translation lateral to the camera plane results in vastly different visual appearance of the object. This might hinder the learning ability of a model since visually different observations correspond the same orientation parameters. Allocentric representation does not exhibit this phenomenon. We evaluate both representations for direct regression in the T6D-Direct model.

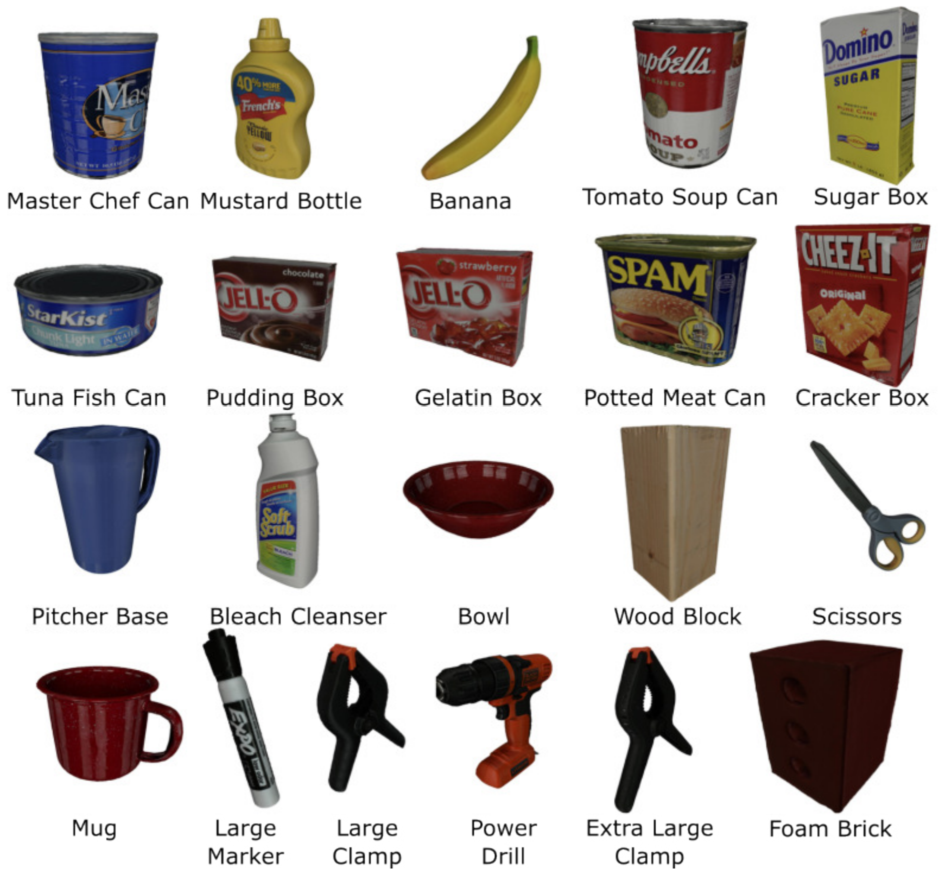


Figure 5.6: YCB-Video dataset objects. The dataset consists of 21 objects. The standard evaluation protocol proposed by Xiang et al. (2018) considers five objects to be symmetric: `bowl`, `wood block`, `large clamp`, `extra large clamp`, and `foam brick`. `Large clamp` and `extra large clamp` differ only by size.

5.4 Evaluation

5.4.1 Dataset

We use the challenging YCB-Video (Xiang et al., 2018) dataset to evaluate the performance of our model. It provides bounding box, segmentation, and 6D pose annotations for 133,936 RGB-D images. Since our model is RGB-based, we do not use the provided depth information. The dataset is generated by capturing video sequences of a random subset of objects from a total of 21 objects (shown in Fig. 5.6) placed in tabletop configuration. From the 92 video sequences, twelve are used for testing and 80 are used for training. The objects used exhibit varying geometric shapes, reflectance properties, and symmetry. Thus, YCB-Video is a challenging dataset for benchmarking 6D object pose estimation methods. YCB-Video also provides high-quality meshes for all 21 objects. Mesh points from these objects are used in computing the evaluation metrics that we describe in Section 5.4.2. Hodaň et al. (2020) provided a variant of YCB-Video[1] as part of the BOP challenge in which the centers of the 3D bounding boxes are aligned with the origin of the model coordinate system and the ground-truth annotations are converted correspondingly. We use the BOP variant of the YCB-Video dataset. In addition to the YCB-Video dataset images, we use the synthetic dataset provided by PoseCNN (Xiang et al., 2018) for training our model. Moreover, we initialize our model using the pre-trained weights on the COCO dataset (Lin et al., 2014) for the task of object detection.

5.4.2 Evaluation Metrics

Xiang et al. (2018) proposed area under the curve (AUC) of ADD and ADD-S metrics for evaluating the accuracy of non-symmetric and symmetric objects, respectively. Given the ground-truth 6D pose annotation with rotation and translation components R and $\mathbf{t}$, and the predicted rotation and translation components $\hat{R}$ and $\hat{\mathbf{t}}$, ADD metric is the average ℓ_2 distance between the subsampled mesh points $\mathcal{M}$ in the ground truth and the predicted pose. In contrast, the symmetry-aware ADD-S metric is the average distance between the closest subsampled mesh points $\mathcal{M}$ in the ground-truth and predicted pose. Following the standard procedure proposed by Xiang et al. (2018), we aggregate the results and report the area under the threshold-accuracy curve for distance thresholds from zero to 0.1m.

$$\text{ADD} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} \|(Rx + \mathbf{t}) - (\hat{R}x + \hat{\mathbf{t}})\|, \tag{5.10}$$

$$\text{ADD-S} = \frac{1}{|\mathcal{M}|} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|(Rx_1 + \mathbf{t}) - (\hat{R}x_2 + \hat{\mathbf{t}})\|. \tag{5.11}$$

The ADD and ADD-S metrics are combined into one metric by using ADD for non-symmetric objects and ADD for symmetric objects. This combined metric is denoted as ADD-(S).

1 https://bop.felk.cvut.cz/datasets/

Table 5.1: Quantitative comparison on the YCB-Video Dataset.

Metric	AUC of ADD-S			AUC of ADD-(S)			
Object	PoseCNN	T6D-Direct	DeepIM	PoseCNN	PVNet	T6D-Direct	DeepIM
master_chef_can	84.0	91.9	**93.1**	50.9	**81.6**	61.5	71.2
cracker_box	76.9	86.6	**91.0**	51.7	80.5	76.3	**83.6**
sugar_box	84.3	90.3	**96.2**	68.6	84.9	81.8	**94.1**
tomato_soup_can	80.9	88.9	**92.4**	66.0	78.2	72.0	**86.1**
mustard_bottle	90.2	94.7	**95.1**	79.9	88.3	85.7	**91.5**
tuna_fish_can	87.9	92.2	**96.1**	70.4	62.2	59.0	**87.7**
pudding_box	79.0	85.1	**90.7**	62.9	85.2	72.7	**82.7**
gelatin_box	87.1	86.9	**94.3**	75.2	88.7	74.4	**91.9**
potted_meat_can	78.5	83.5	**86.4**	59.6	65.1	67.8	**76.2**
banana	85.9	**93.8**	72.3	**91.3**	51.8	87.4	81.2
pitcher_base	76.8	92.3	**94.6**	52.5	**91.2**	84.5	90.1
bleach_cleanser	71.9	83.0	**90.3**	50.5	74.8	65.0	**81.2**
bowl*	69.7	91.6	**81.4**	69.7	89.0	**91.6**	81.4
mug	78.0	89.8	**91.3**	57.7	**81.5**	72.1	81.4
power_drill	72.8	88.8	**92.3**	55.1	83.4	77.7	**85.5**
wood_block*	65.8	**90.7**	81.9	65.8	71.5	**90.7**	81.9
scissors	56.2	**83.0**	75.4	35.8	54.8	59.7	**60.9**
large_marker	71.4	74.9	**86.2**	58.0	35.8	63.9	**75.6**
large_clamp*	49.9	**78.3**	74.3	49.9	66.3	**78.3**	74.3
extra_large_clamp*	47.0	54.7	**73.2**	47.0	53.9	54.7	**73.3**
foam_brick*	87.8	**89.9**	81.9	87.8	80.6	**89.9**	81.9
MEAN	75.9	86.2	**88.1**	61.3	73.4	74.6	**81.9**

Symmetric objects are denoted by *. The best results are shown in bold.

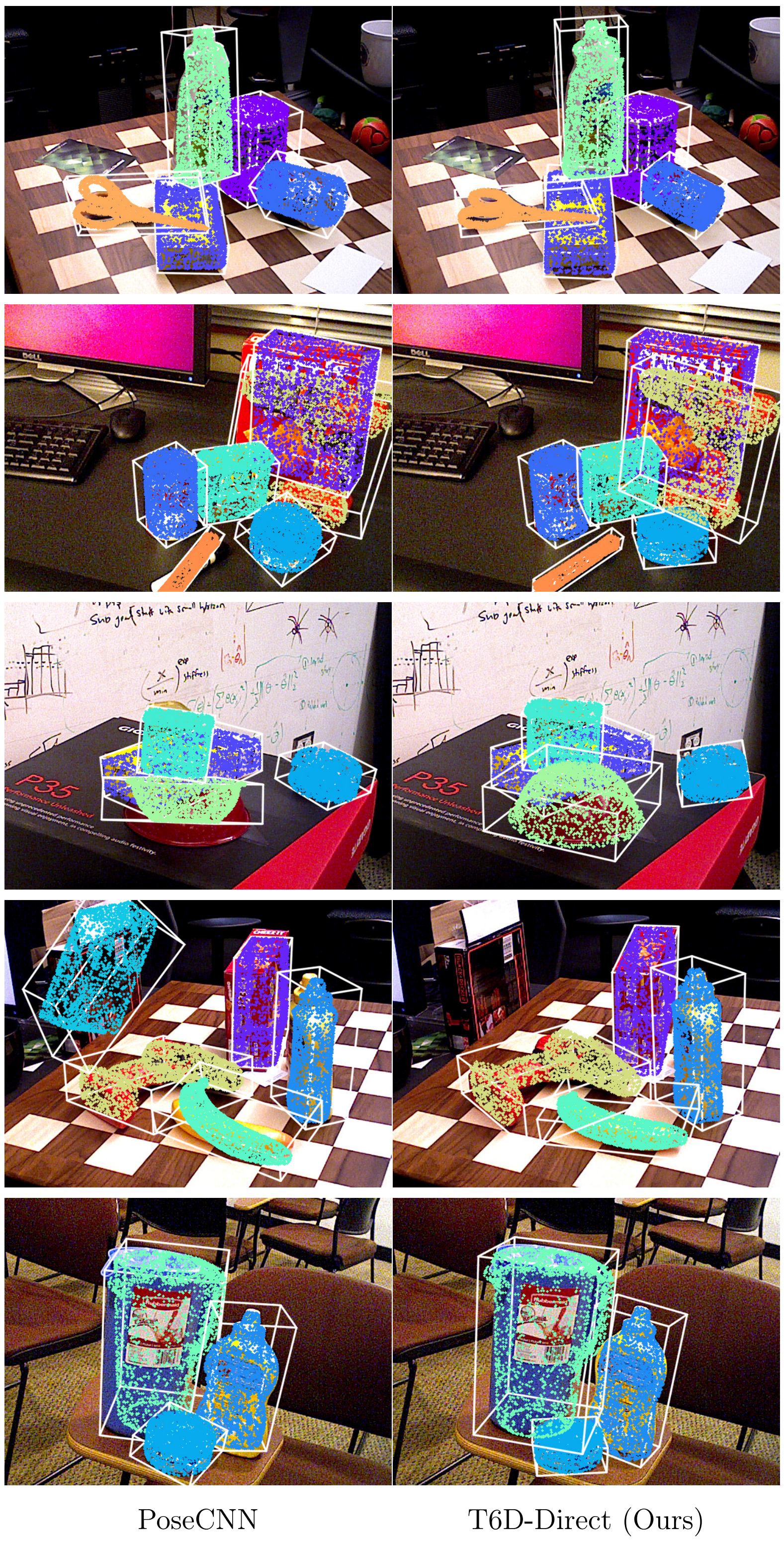


Figure 5.7: Qualitative examples from the YCB-Video dataset. Left: PoseCNN (Xiang et al., 2018). Right: T6D-Direct (Ours) predictions. The pose predictions are visualized using 3D bounding boxes and the object mesh points in predicted pose are overlaid on top of the image pixels.

Table 5.2: Comparison with leading pose estimation methods on the YCB-Video dataset. † indicates refinement-based methods. Inference time is the average time taken for processing all objects in an image.

Method	ADD-(S)	AUC of ADD-S	AUC of ADD-(S)	fps
CosyPose† (Labbe et al., 2020)	-	**89.8**	**84.5**	2.5
PoseCNN (Xiang et al., 2018)	21.3	75.9	61.3	4
Single-Stage (Hu et al., 2020)	**53.9**	-	-	45
GDR-Net (Wang et al., 2021)	49.1	89.1	80.2	15.3
T6D-Direct (Ours)	48.7	86.2	74.6	58

5.4.3 Experimental Results

In Table 5.1, we present the per object quantitative results of T6D-Direct on the YCB-Video dataset. For a fair comparison, we follow the same object symmetry definition and evaluation procedure described by the YCB-Video dataset (Xiang et al., 2018). We compare our results with PoseCNN (Xiang et al., 2018), PVNet (Peng et al., 2019) and DeepIM (Li et al., 2018b). In terms of the approach, T6D-Direct is comparable to PoseCNN; both are direct regression methods, whereas PVNet is an sparse keypoint-based method, and DeepIM is a refinement-based approach. In terms of both AUC of ADD-S and AUC of ADD-(S) metrics, T6D-Direct outperforms PoseCNN and outperforms the AUC of ADD-(S) results of PVNet. Some qualitative results are shown in Fig. 5.7. Comparing the ADD-S and ADD-(S) per object, we observe that for non-symmetric objects `master chef can`, `tuna fish can`, and `scissors`, the ADD-(S) accuracy drops significantly. For the rest of the objects the difference between ADD-S and ADD-(S) score are in line with the competing methods. In the following sections, we analyze the inference time of our method and present the results of the ablation study.

5.4.4 Inference Time Analysis

In Table 5.2, we present the inference time of the T6D-Direct model in comparison with the state-of-the-art CNN-based pose estimation model. Being a single-stage model allows faster inference speed for the T6D-Direct model. Among the models compared, Single-Stage (Hu et al., 2020) also performs object pose estimation in a single step. It uses a segmentation-driven CNN to predict pixel-wise 2D-3D correspondences followed by a RANSAC step to generate 6D poses. PoseCNN (Xiang et al., 2018) uses a multi-stage approach in which objects are segmented and the corresponding crops are extracted in the first stage. In the second stage, the translation component is estimated using a novel differentiable Hough voting layer and the orientation component is directly regressed in the form of quaternions. GDR-Net (Wang et al., 2021) introduced Patch-PnP, a light-weight CNN module that directly regresses 6D pose parameters from the 2D-3D correspondence estimations. CosyPose (Labbe et al., 2020), a pose refinement approach based on render-and-compare framework refines PoseCNN (Xiang et al., 2018) predictions iteratively to generate more accurate pose predictions. Although CosyPose achieves high pose prediction accurate, it is comparatively slower at 2.5 frames per second. Among the methods

Table 5.3: Ablation study on YCB-Video. We provide results of our method trained using different loss functions and training schemes.

Row	Method	ADD-(S)	AUC of ADD-(S)
1	T6D-Direct with Point Matching loss	47.0	**75.6**
2	1 + multi-stage training	20.5	59.1
3	1 + pose matching cost component	42.8	71.7
4	1 + allocentric R_{6d}	42.9	74.4
5	T6D-Direct with PLoss	45.8	74.4
6	T6D-Direct	**48.7**	74.6

in comparison, only Single-Stage (Hu et al., 2020) has an inference speed closer to the T6D-Direct model. At 58 fps, T6D-Direct is suitable for real-time applications.

5.4.5 Effectiveness of Loss Functions

Loss functions quantify the difference between the predicted output of the neural network and the actual target values. They guide the model in learning the objective. In case of object pose estimation, depending on the pose representation there exists several loss functions. The simplest loss function is the ℓ_2 loss for both the translation and rotation components. The main advantage of the ℓ_2 loss, it does not need the 3D object mesh or the point cloud to compute in the loss. The major down-side of the ℓ_2 loss is that it does not capture the object symmetry. Loss functions like Point Matching loss (Li et al., 2018b; Xiang et al., 2018), disentangled SLoss (Xiang et al., 2018) requires object mesh or point cloud data, but captures the object symmetry. In Table 5.3, we examine the performance of our model using the symmetry aware version of Point Matching loss with ℓ_2 norm (Li et al., 2018b; Xiang et al., 2018) which, in contrast to the disentangled loss presented in Section 5.3.2, couples the rotation and translation components. This loss function results in the best AUC of ADD-(S) metric. Moreover, since the symmetry aware SLoss component of the Point Matching loss is computationally expensive, we experimented with training our model using only the PLoss component. Interestingly, the ADD-(S) accuracy of the model trained using only the PLoss component (row 5) is only slightly worse than the model trained using the both components (row 1).

5.4.6 Effectiveness of Training Strategies

As discussed in Section 5.4.6, there are two training schemes: single-stage and multi-stage. In the multi-stage scheme, we train the Transformer model for object detection and only train the FFNs for pose estimation, whereas in the single-stage scheme, we train the complete model in one stage. In our experiments, as shown in Table 5.3, multi-stage training (row 2) yielded inferior results, although both schemes were pretrained on the COCO dataset. This demonstrates that the Transformer model is learning the features specific to the 6D object pose estimation task on YCB-Video, and COCO fine-tuning mainly helps in faster convergence during training and not in more accurate pose estimations. We thus believe that most large-scale image datasets can serve as

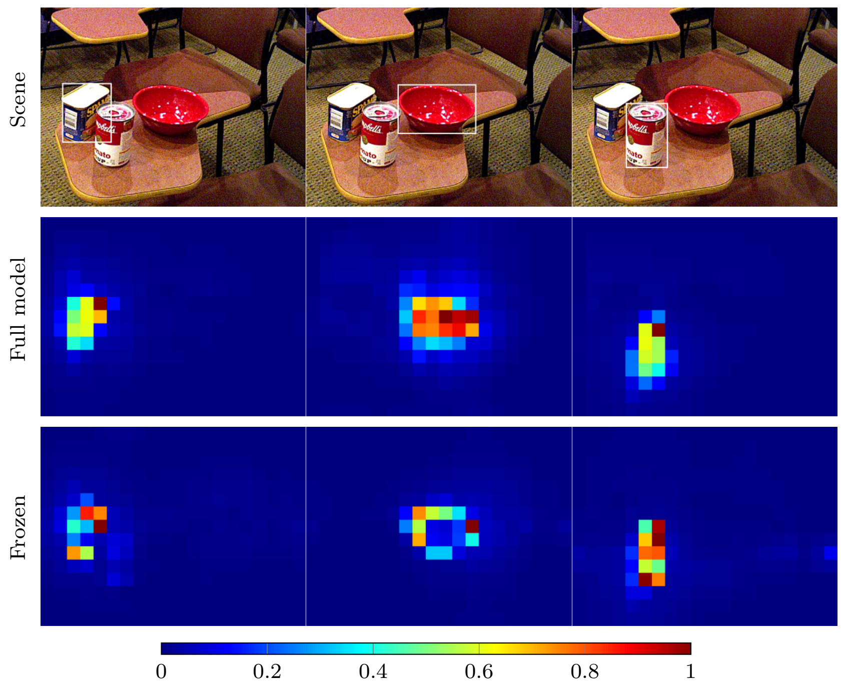


Figure 5.8: Decoder cross-attention. Object predictions of a given image (first row) and decoder attention maps for the object queries (second and third rows). Training the complete model for both object detection and pose estimation tasks (second row). Training the model first on the object detection task, and then training the frozen model on the pose estimation task (third row). Attention maps are visualized using the jet color map (shown above for reference).

pretraining data source. We also provide the results of including the pose component in the bipartite matching cost mentioned in Section 5.3.2. Including the pose component (row 3) does not provide any considerable advantage; thus, we include only the class probability and bounding box components in the bipartite matching cost in all further experiments. Further, egocentric rotation representation (row 1) performed slightly better than allocentric representation (row 4). We hypothesize that supplementing RGB images with positional encoding allows the Transformer model to learn spatial features efficiently. Therefore, the allocentric representation does not have any advantage over the egocentric representation.

5.4.7 Analyzing Attention Mechanism

The attention mechanism allows for the model to learn to focus on to the pixels relevant for the joint object detection and the pose estimation task. Visualizing the attention maps generated in the encoder and the decoder modules help us in understanding the inner working of the T6D-Direct model better. In the decoder module, we perform cross attention between the object queries and the encoder embeddings. For each object queries, the decoder module and the subsequent FFNs generate an object prediction. In Fig. 5.8-second row, we visualize the attention maps corresponding to object queries that resulted in object predictions (excluding $\emptyset$ predictions) made by the T6D-Direct model. Note that the attention maps highly correlate with the semantic masks of the corresponding objects

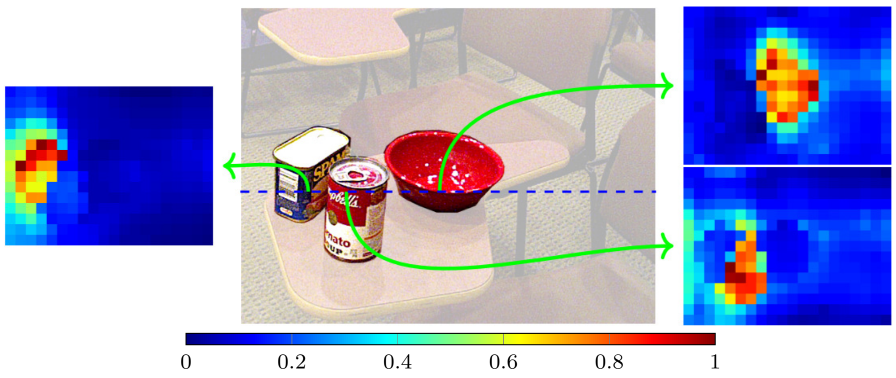


Figure 5.9: Encoder self-attention. We visualize the self-attention maps for three pixels belonging to three objects in the image. All three pixels lie on the same horizontal line but attend to different parts of the image. Attention maps are visualized using the jet color map (shown above for reference).

in the input image. Moreover, we also experimented with fine tuning only the FFNs for the joint object detection and pose estimation task. We started with training the T6D-Direct model for the object detection task on the COCO dataset and while fine tuning the model for the joint object detection and pose estimation task, we freeze the encoder and the decoder modules. In Fig. 5.8-third row, we visualize the attention maps generated by the partial frozen model. The fully fine-tuned T6D-Direct model performed better than the partial frozen model. A closer look at the attention maps reveals that the frozen model attends to the object boundary pixels more, whereas the fully fine-tuned model attends to all the object pixels.

Similarly, in Fig. 5.9, we visualize the attention maps in the encoder modules corresponding to three pixels that belong to the three objects in the image and lie in the same horizontal line. The object pixels and the pixels in the immediate vicinity of attended more, whereas the background pixels are completely ignored. From these visualizations we can conclude that the attention mechanism learns to focus on the pixels relevant for the task without any explicit guidance.

5.5 Discussion & Conclusion

In this Chapter, we introduced the multi-object pose estimation as set prediction formulation and discussed the vision transformer based single-stage T6D-Direct model. Our model is based on DETR (Carion et al., 2020) a vision transformer model for object detection. Given an RGB input, our model generates a set of tuples of fixed cardinality. Each tuple consists of class probabilities, 2D bounding box parameters, and the position and orientation parameters. Since the cardinality of the set of fixed, our model learns to predict ∅ class for the remaining elements of the set after predicting all the objects in the given image. Taking advantage of the permutation invariant nature of the attention mechanism and the bipartite matching algorithm, the T6D-Direct model is trained to localize and generate 6D pose for a varying number of objects in one pass. Moreover, direct regression of the pose parameters enables a straightforward and fast feed forward prediction head implementation. We evaluated different orientation parameter representations. The 6D continuous parameterization produced the best results. Employing the symmetry-aware Point Matching loss (Li et al., 2018b; Xiang et al., 2018) for orientation loss, our model learns to handle symmetry implicitly. Our model achieves impressive accuracy while running at much higher fps compared to other state-of-the-art methods. Analyzing the encoder self-attention maps reveals that the attention mechanism learns to focus on pixels belonging to the same object and the boundary pixels of the neighboring objects. Similarly, the attention maps corresponding the object detections resembles semantic segmentation masks. From these attention maps we can conclude that the attention mechanism learns to focus on the relevant pixels without any external supervision. Although our model inherits the object detection branch from the DETR model, the experimental results demonstrates that freezing the object detection branch and only training the pose estimation branch performs poorly compared to training the complete model. The pose estimation as set prediction formulation and the T6D-Direct forms the foundation for the methods we present in the next chapters.

6 Multi-Object Pose Estimation Using Keypoint Regression

Multi-object pose estimation as set prediction offers an elegant formulation to realize single-stage models. However, direct regression of pose parameters is less robust against occlusion. In this Chapter, we address this limitation by regressing an intermediate keypoint representation and employ a learned PnP module to recover 6D pose. This allows our model to benefit from the keypoint representation, while retaining the simplicity of direct regression. Furthermore, we explore different vision transformer architectures for improved accuracy and faster inference.

Statement of Personal Contribution

This Chapter draws from material originally presented in the following publications:

Arash Amini*, Arul Selvam Periyasamy*, and Sven Behnke:

YOLOPose: Transformer-based multi-object 6D pose estimation using keypoint regression

In: *17th International Conference Intelligent Autonomous Systems (IAS)*, Zagreb, Croatia, 2022.

Best Paper Award

Arul Selvam Periyasamy, Arash Amini, Vladimir Tsaturyan, and Sven Behnke:

YOLOPose V2: Understanding and improving transformer-based 6D pose estimation

In: *Robotics and Autonomous Systems (RAS)*, Volume 168, pp 104490, 2023.

Arul Selvam Periyasamy*, Vladimir Tsaturyan*, and Sven Behnke:

Efficient multi-object pose estimation using multi-resolution deformable attention and query aggregation

In: *7th IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, USA. 2023.

The author of this thesis substantially contributed to all aspects of these publications, including the literature survey, the conception, formalization, design, and implementation of the proposed methods, the preparation and conduct of experiments for the evaluation of the proposed approach, the analysis and interpretation of the experimental results, the preparation of the manuscript, as well as the revision and final editing of the version published.

6.1 Introduction

Selective attention plays an important role in human vision system (Ungerleider and G, 2000; Luck and Ford, 1998; Serences and Yantis, 2006), by which it attends to specific regions or features of a scene. Keypoints act as salient features for selective attention guiding the focus towards important elements in the scene. They are crucial in both human and computer vision systems, highlighting their importance in visual perception. With the introduction of SIFT (Lowe, 2004), many of the computer vision tasks were formulated as keypoint detection (Calonder, Lepetit, and Fua, 2008; Li and Allinson, 2008). In the previous Chapter, we reviewed the keypoint-based methods prior for deep learning for object pose estimation (see Section 5.2). In this Chapter, we extend the T6D-Direct model introduced in Chapter 5 with keypoint regression. We explore different keypoint representations and by employing a learned rotation estimation module that predicts the orientation parameters from the intermediate keypoint predictions, the model is end-to-end differentiable. Furthermore, based on experimental evaluation, we show that the learned rotation estimation module is robust against noisy keypoint predictions than the analytical PnP algorithm.

6.2 Related Work

YOLOPose and its variants we introduce in this Chapter also follow the multi-object pose estimation as set prediction formulation discussed in Section 5.3.1. However, instead of the directly regressing the pose parameter we first predict the location of keypoints and employ a novel pose estimation module to generate pose prediction from the detected keypoints. In this section, we review different families of keypoint-driven pose estimation methods.

Keypoints as Sparse Heat Maps

One of the standard formulations for keypoint prediction is keypoints as sparse heat maps. One output feature map corresponds to a specific keypoint. Only the target pixel or the target pixel along with its neighborhood have a higher value and all other pixel are zeros (see Fig. 6.1 (a)). The models are trained to minimize *mean squared error* (MSE) loss between the ground-truth and predicted feature maps. Behnke (2003a) and Behnke (2005) performed face localization and tracking using the pair of eyes as sparse keypoints employing the novel Neural Abstraction Pyramid architecture. Oberweger, Rad, and Lepetit (2018) predicted the pixel projections of the 3D bounding box corners and recovered the 6D object pose using the PnP algorithm. Peng et al. (2019) preferred keypoints that lie on the object surface such that the pixel projections of the keypoints always lie inside of the target object pixels. They employed *farthest point sampling* (FPS) algorithm to automatically select a set of keypoints from the 3D object model.

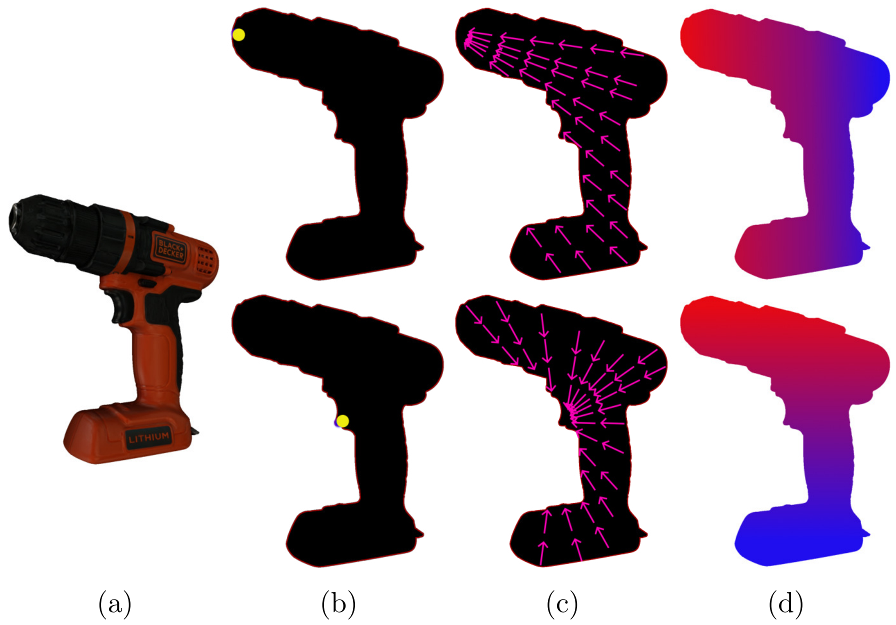


Figure 6.1: Different sparse and dense formulations. (a): RGB input. (b): Keypoints as sparse heat maps. Keypoints are depicted using yellow circles (c): Keypoints as dense 2D vectors (d): Dense 3D object coordinates; only x (top) and y (bottom) coordinates are shown.

Keypoints as Dense 2D Vectors

Keypoints as sparse heat maps formulation lends itself to straightforward fully connected convolutional neural network design and loss functions. However, this formulation suffers severely under occlusion. Xiang et al. (2018), Peng et al. (2019), He et al. (2021), and He et al. (2020) predicted keypoints using dense 2D vectors. For each object pixel, the model predicts a 2D vector representing the direction of the keypoints as shown in Fig. 6.1 (b). From these direction vectors, the 6D pose is recovered employing differentiable Hough transformations (Xiang et al., 2018) or RANSAC combined with the PnP algorithm (Peng et al. (2019) and He et al. (2021)).

Dense 3D Object Coordinates

The accuracy and robustness of the keypoint-based pose estimation increase with the number of keypoints. However, the size of the model also increases with number of keypoints. Predicting the corresponding 3D object coordinates for each object pixel (depicted in Fig. 6.1 (c)) does not suffer from this limitation (Wang et al. (2021) and Zakharov, Shugurov, and Ilic (2019)). From the dense 3D object coordinates predictions, 6D pose is recovered using the PnP algorithm.

In contrast to these family of methods, we do not represent keypoints using feature maps. Instead, we regress to an intermediate keypoint representation (discussed in Section 6.3.2) and employ a learned rotation estimation module to recover 6D pose from the predicted keypoints.

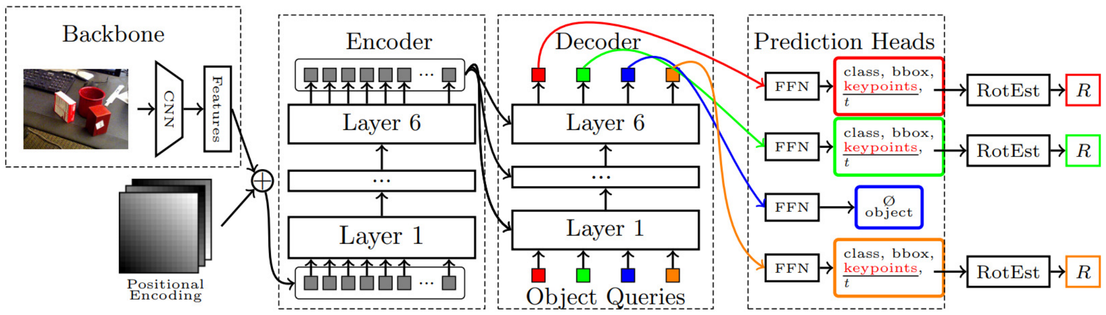


Figure 6.2: YOLOPose architecture in detail. Given an RGB input, we extract features using the ResNet backbone. The extracted features are supplemented with positional encoding and provided as input to the Transformer encoder. The encoder module consists of six standard encoder layers with skip connections. The output of the encoder module is provided to the decoder module along with N object queries. The decoder module also consists of six standard decoder layers with skip connections generating N output embeddings. The output embeddings are processed with FFNs to generate a set of N elements in parallel. Each element in the set is a tuple consisting of the bounding box, the class probability, the translation, and the interpolated bounding box keypoints. A learnable rotation estimation module is employed to estimate object orientation R from the predicted 2D keypoints.

6.3 YOLOPose

YOLOPose follows the multi-object pose estimation as set prediction formulation introduced in Chapter 5. Similar to the T6D-Direct model, YOLOPose is also a single-stage model, i.e., YOLOPose jointly detects and estimates pose for all objects in the give input without intermediate NMS, RoI, or anchor boxes modules. Unlike T6D-Direct, YOLOPose does not regress 6D pose directly. Instead, YOLOPose predicts object keypoints and employs a novel learnable PnP module to estimate object pose. Our model outputs a set of elements with a fixed cardinality N. Each element in the set is a tuple containing the 2D bounding boxes, the class probability, the translation, and the keypoints. 2D bounding boxes are represented with the center coordinates, height, and width proportional to the image size. The class probability is represented with the standard one-hot encoding. To estimate translation $\mathbf{t} = [t_x, t_y, t_z]^T \in \mathbb{R}^3$ defined as the coordinate of the object origin in the camera coordinate system, we follow the method proposed by PoseCNN (Xiang et al., 2018) which decouples $\mathbf{t}$ into the object's distance from the camera t_z and the 2D location of projected 3D object's centroid in the image plane $[c_x, c_y]^T$. Finally, having the intrinsic camera matrix, we can recover t_x and t_y. The exact choice of the keypoints is discussed in Section 6.3.2. The number of objects present in an image varies; therefore, to enable output sets with fixed cardinality, we choose N to be larger than the expected maximum number of objects in an image in the dataset and introduce a no-object class Ø. This Ø class is analogous to the background class used in semantic segmentation models. In addition to predicting the corresponding classes for objects present in the image, our model is trained to predict Ø for the rest of the elements in the set.

6.3.1 Model Architecture

The proposed YOLOPose architecture is shown in Fig. 6.2. The model consists of a ResNet backbone followed by Transformer-based encoder-decoder module and MLP prediction heads to predict a set of tuples. CNN architectures have several inductive biases designed into them (LeCun, Bengio, et al., 1995; Cohen and Shashua, 2017). These strong biases enable CNNs to learn efficient local spatial features in a fixed neighborhood defined by the receptive field to perform well on many computer vision tasks. In contrast, Transformers, aided by the attention mechanism, are suitable for learning spatial features over the entire image. This makes the Transformer architecture suitable for multi-object pose estimation. In this section, we describe the individual components of the YOLOPose architecture.

Backbone Network

We use a ResNet50 backbone for extracting features from the given RGB image. For an image size of height H and width W, the backbone network extracts 2048 low-resolution feature maps of size H/32×W/32. We then use 1×1 convolution to reduce the 2048 feature dimensions to smaller d=256 dimensions. The standard Transformer models are designed to process vectors. Hence, to enable processing the d×H/32×W/32 features, we vectorize them to $d\times\frac{H}{32}\frac{W}{32}$.

Encoder

The Transformer encoder module consists of six encoder layers with skip connections. Each layer performs multi-head self-attention of the input vectors. Given pixel with embedding x of dimension d, the embedding is split into h chunks, or “heads" and for each head i, the scaled dot-product attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d/h}})V,$$

where Q, K, and V are the query, key, and value matrices for the head i, respectively, and are computed by linearly projecting x using projection parameter matrices W^q, W^k, W^v, respectively. The attention outputs of the heads are concatenated and transformed linearly to compute multi-head self-attention:

$$\text{MultiHead(Q,K,V)} = \underset{i\in h}{\text{concat}}(\text{Attention}(Q_i, K_i, V_i))W^O,$$

where $W^O \in \mathbb{R}^{d\times d}$ is also a projection parameter matrix, and concat denotes concatenation along the embedding dimension. In contrast to the convolution operation, which limits the receptive field to a small neighborhood, self-attention enables a receptive field of the size of the whole image. Note that the convolution operation can be cast as a special case of self-attention (Cordonnier, Loukas, and Jaggi (2020)).

Positional Encodings

The multi-head self-attention operation is permutation-invariant. Thus, the Transformer architecture ignores the order of the input vectors. Similar to the T6D-Direct model we

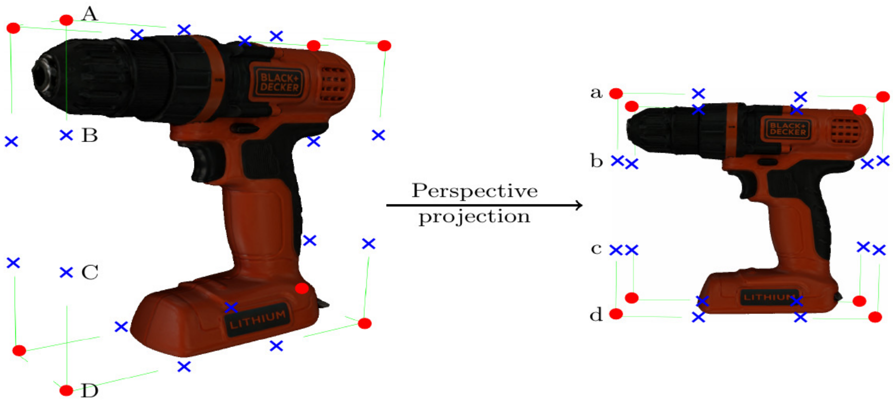


Figure 6.3: Interpolated bounding box points. Bounding box points are indicated with red dots, and the interpolated points are indicated with blue crosses. The cross-ratio of every four collinear points is preserved during perspective projection, e.g., the cross-ratio of points A, B, C, and D remains the same in 3D and, after perspective projection, in 2D.

discussed in Section 5.3.1, We employ the standard solution of supplementing the input vectors with absolute positional encoding following Carion et al. (2020) to provide the Transformer model with spatial information of the pixels. We encode the pixel coordinates as sine and cosine functions of different frequencies:

$$\text{P.E.}_{(pos,p)} = sin(pos/10000^{\frac{2p}{d}}),$$
$$\text{P.E.}_{(pos,p+1)} = cos(pos/10000^{\frac{2p+1}{d}}),$$

where *pos* is the pixel coordinate (either width or height), d is the embedding dimension, and p is the index of the positional encoding.

DECODER

On the decoder side, we compute cross-attention between the encoder output embeddings and N learnable embeddings, referred to as *object queries*, to generate decoder output embeddings, where N is the cardinality of the predicted set. The decoder consists of six decoder layers and the object queries are provided as input to each decoder layer. Unlike the fixed positional encoding used in the encoder, the object queries are learned jointly with the original learning objective—joint object detection and pose estimation, in our case—from the dataset. At the start of the training process, the object queries are initialized randomly, and during inference, the object queries are fixed. In Section 6.3.8, we investigate the role of object queries generating object predictions. The embeddings used in our model—both learned and fixed—are 256-dimensional vectors.

FFN

Feed-forward networks (FFNs) are fully-connected three-layer MLPs with hidden 256 hidden units in each layer and ReLU activation. From the N decoder output embeddings, we use (FFNs) to generate a set of N output tuples independently. Each tuple consists of the class probability, the bounding box, the keypoints, and the 6D pose parameters.

6.3.2 Keypoints Representation

An obvious choice for selecting 3D keypoints is the eight corners of the 3D bounding box (Oberweger, Rad, and Lepetit, 2018). Peng et al. (2019) instead used the farthest point sampling (FPS) algorithm to sample eight keypoints on the surface of the object meshes, which are also spread out on the object to help the PnP algorithm find a more stable solution. Li et al. (2021) defined the 3D representation of an object as sparse interpolated bounding boxes (IBB), shown in Fig. 6.3, and exploited the property of perspective projection that cross-ratio of every four collinear points in 3D (A, B, C, and D as illustrated in Fig. 6.3) is preserved under perspective projection in 2D (Hartley and Zisserman, 2004). The cross-ratio consistency is enforced by an additional component in the loss function that the model learns to minimize during training. We further investigate these keypoints representations in Section 6.3.6 and present our results in Table 6.4.

6.3.3 RotEst

The standard solution for the perspective geometry problem of recovering 6D object/-camera pose given 2D-3D correspondences and a calibrated camera is the PnP algorithm. The minimum number of correspondences needed for employing PnP is 4. However, the accuracy and the robustness of the estimated pose increase with the number of correspondences. Moreover, PnP is used in conjecture with RANSAC to increase the robustness. Although PnP is a standard and well-understood solution, incorporating it in neural network pipelines introduces two drawbacks. First, it is not trivially differentiable. Second, PnP combined with RANSAC needs multiple iterations to generate highly accurate pose predictions. These drawbacks hinder us in realizing end-to-end differentiable pipelines with a single step forward pass for pose estimation. To this end, we introduce the RotEst module. For each object, from the estimated pixel coordinates onto which the 32 keypoints (the eight corners of the 3D bounding box and the 24 intermediate bounding box keypoints) are projected, the RotEst module predicts the object orientation represented as the 6D continuous representation in SO(3) Zhou et al. (2019). Furthermore, we experimented with providing additional inputs to the FFNs. We created three variants of the YOLOPose model: variants A, B, and C (shown in Fig. 6.4). In variant A, in addition to the estimated IBB keypoints, we provide the output embedding of the object query to the FFNs. In variant B, IBB keypoints, object query output embedding, and the canonical 3D bounding box points (based on the predicted object class) are provided to the FFNs, whereas in variant C, estimated IBB keypoints and class probabilities are fed as input to FFNs. Note that the size of the embedding used in YOLOPose model is 256. Thus, the number of parameters used in FFNs of the three variants is larger than that of the YOLOPose model. We implement the RotEst module using six fully connected layers with a hidden dimension 1024 and a dropout probability of 0.5.

6.3.4 Loss Function

Our model is trained to minimize the Hungarian loss between the predicted and the ground-truth sets. Computing the Hungarian loss involves finding the matching pairs in the two sets. We use bipartite matching (Kuhn, 1955; Stewart, Andriluka, and Ng, 2016; Carion et al., 2020) to find the permutation of the predicted elements that minimize the

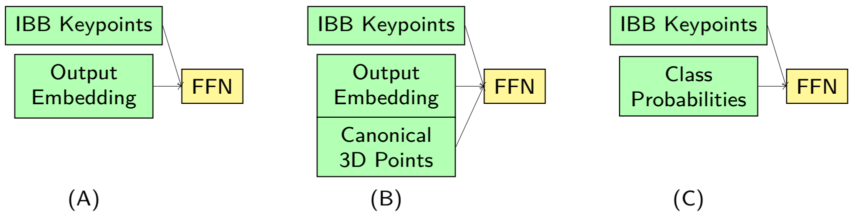


Figure 6.4: Variants of the YOLOPose model. All three variants are derived from the YOLOPose model and differ in the inputs provided to the pose estimation FFNs.

matching cost. Given the $\varnothing$ class padded ground-truth set $\mathcal{Y}$ of cardinality N containing labels $y_1, y_2, ..., y_N$, the predicted set denoted by $\hat{\mathcal{Y}}$, we search for the optimal permutation $\hat{\sigma}$ among the possible permutations $\sigma \in \mathfrak{S}_N$ that minimizes the matching cost $\mathcal{L}_{match}$. Formally,

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}). \tag{6.1}$$

Although each element of the set is a tuple containing four components, bounding box, class probability, translation, and keypoints, we use only the bounding box and the class probability components to define the matching cost function. In practice, omitting the other components in the cost function definition does not hinder the model's ability in learning to predict the keypoints and keeps the computational cost of the matching process minimal.

Given the matching ground-truth and predicted sets $\mathcal{Y}$ and $\hat{\mathcal{Y}}_\sigma$, respectively, the Hungarian loss is computed as:

$$\mathcal{L}_{Hungarian}(\mathcal{Y}, \hat{\mathcal{Y}}_\sigma) = \sum_{i}^{N}[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{c_i \neq \varnothing}\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) + \lambda_{kp}\mathbb{1}_{c_i \neq \varnothing}\mathcal{L}_{kp}(k_i, \hat{k}_{\hat{\sigma}(i)}) + \lambda_{pose}\mathbb{1}_{c_i \neq \varnothing}\mathcal{L}_{pose}(R_i, t_i, \hat{R}_{\hat{\sigma}(i)}, \hat{t}_{\hat{\sigma}(i)})]. \tag{6.2}$$

Class Probability Loss

The class probability loss function is the standard negative log-likelihood NLL loss. Since we choose the cardinality of the set to be higher than the expected maximum number of objects in an image, the $\varnothing$ class appears disproportionately often. Thus, we weigh the loss for the $\varnothing$ class with a factor of 0.1.

Bounding Box Loss

The 2D bounding boxes are represented as (c_x, c_y, w, h) where (c_x, c_y) are 2D pixel coordinates and w and h are object width and height, respectively. To train the bounding box prediction head, we use a weighted combination of the Generalized IoU (GIoU) (Rezatofighi et al., 2019) and ℓ_1-loss with 2 and 10 factors, respectively.

$$\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \alpha\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \beta||b_i - \hat{b}_{\sigma(i)}||, \tag{6.3}$$

$$\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) = 1 - \left(\frac{|b_i \cap \hat{b}_{\sigma(i)}|}{|b_i \cup \hat{b}_{\sigma(i)}|} - \frac{|B(b_i, \hat{b}_{\sigma(i)}) \setminus b_i \cup \hat{b}_{\sigma(i)}|}{|B(b_i, \hat{b}_{\sigma(i)})|} \right), \tag{6.4}$$

and $B(b_i, \hat{b}_{\sigma(i)})$ is the largest box containing both the ground truth b_i and the prediction $\hat{b}_{\sigma(i)}$.

Keypoint Loss

Having the ground truth K_i and the model output $\hat{K}_{\hat{\sigma}(i)}$, the keypoints loss can be represented as:

$$\mathcal{L}_{kp}(K_i, \hat{K}_{\hat{\sigma}(i)}) = \gamma ||K_i - \hat{K}_{\hat{\sigma}(i)}||_1 + \delta \mathcal{L}_{\mathcal{CR}}, \tag{6.5}$$

where γ and δ are hyperparameters. The first part of the keypoints loss is the ℓ_1 loss, and for the second part, we employ the cross-ratio loss $\mathcal{L}_{\mathcal{CR}}$ defined in Equation 6.7 to enforce the cross-ratio consistency in the keypoint loss as proposed by Li et al. (2021). This loss is self-supervised by preserving the cross-ratio (CR) of each line to be 4/3. The reason is that after the camera projection of the 3D bounding box on the image plane, the cross-ratio of every four collinear points remains the same.

$$\text{CR} = \frac{||C - A||\ ||D - B||}{||C - B||\ ||D - A||} = \frac{4}{3}, \tag{6.6}$$

$$\mathcal{L}_{\mathcal{CR}} = Smooth\ell_1(\text{CR}^2 - \frac{||c - a||^2 ||d - b||^2}{||c - b||^2 ||d - a||^2}), \tag{6.7}$$

where CR^2 is chosen since $||.||^2$ can be easily computed using vector inner product. A, B, C, and D are four collinear points and their corresponding predicted 2D projections are a, b, c, and d, respectively.

Pose Loss

We supervise the rotation R and the translation $\mathbf{t}$ individually via employing PLoss and SLoss from (Xiang et al., 2018) for rotation, and ℓ_1 loss for translation.

$$\mathcal{L}_{pose}(R_i, \mathbf{t}_i, \hat{R}_{\sigma(i)}, \hat{\mathbf{t}}_{\sigma(i)}) = \mathcal{L}_{rot}(R_i, \hat{R}_{\sigma(i)}) + ||\mathbf{t}_i - \hat{\mathbf{t}}_{\sigma(i)}||_1, \tag{6.8}$$

$$\mathcal{L}_{rot} = \begin{cases} \frac{1}{|\mathcal{M}_i|} \sum\limits_{\text{x}_1 \in \mathcal{M}_i} \min\limits_{\text{x}_2 \in \mathcal{M}_i} ||R_i \text{x}_1 - \hat{R}_{\sigma(i)} \text{x}_2||_1 & \text{if symmetric,} \\ \frac{1}{|\mathcal{M}_i|} \sum\limits_{\text{x} \in \mathcal{M}_i} ||R_i \text{x} - \hat{R}_{\sigma(i)} \text{x}||_1 & \text{otherwise,} \end{cases} \tag{6.9}$$

where $\mathcal{M}_i$ indicates the set of 3D model points.

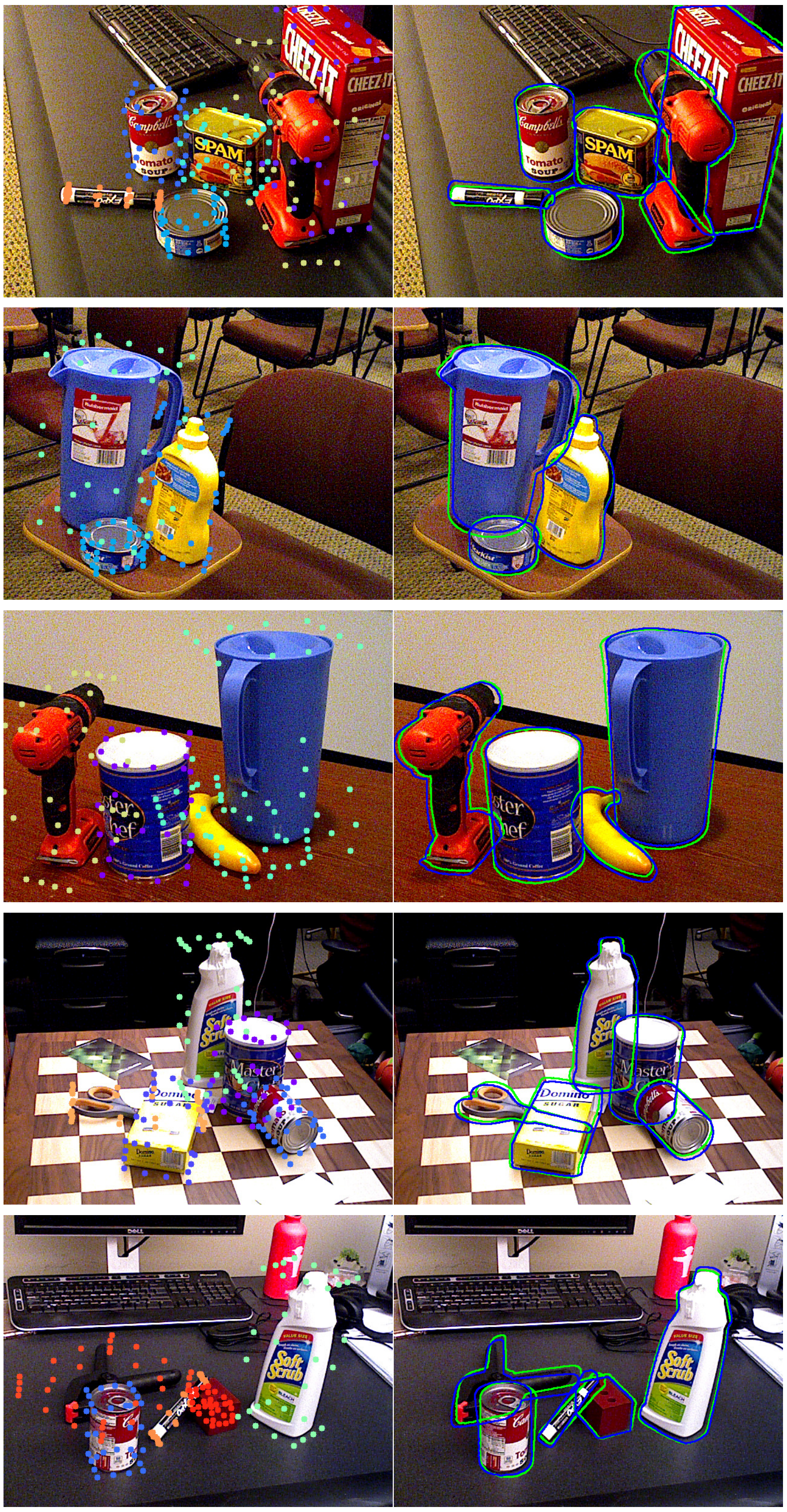


Figure 6.5: Qualitative results on YCB-Video test set. Left: The predicted IBB keypoints overlaid on the input images. Right: Ground-truth and predicted object poses are visualized as object contours in green and blue colors, respectively.

Table 6.1: Comparison of YOLOPose with other keypoint-based methods on the YCB-Video dataset.

Method	GDR-Net		YOLOPose (Ours)		YOLOPose-A (Ours)		DeepIM		YOLOPose-A (Ours)	
Metric	AUC of ADD-S	AUC of ADD-(S)	AUC of ADD-S	AUC of ADD-(S)	AUC of ADD-S	AUC of ADD-(S)	AUC of ADD-S	AUC of ADD-(S)	AUC of ADD-S @0.1d	AUC of ADD-(S) @0.1d
master_chef_can	**96.6**	71.1	91.3	64.0	91.7	**71.3**	93.1	71.2	71.3	36.6
cracker_box	84.9	63.5	86.8	77.9	**92.0**	83.3	91.0	**83.6**	83.3	71.1
sugar_box	**98.3**	93.2	92.6	87.3	91.5	83.6	96.2	**94.1**	83.6	59.5
tomato_soup_can	**96.1**	**88.9**	90.5	77.8	87.8	72.9	92.4	86.1	72.9	29.8
mustard_bottle	**99.5**	**93.8**	93.6	87.9	96.7	93.4	95.1	91.5	93.4	93.4
tuna_fish_can	95.1	85.1	94.3	74.4	94.9	70.5	**96.1**	**87.7**	70.5	17.4
pudding_box	**94.8**	86.5	92.3	87.9	92.6	**87.0**	90.7	82.7	87.0	70.9
gelatin_box	**95.3**	88.5	90.1	83.4	92.2	85.7	94.3	**91.9**	85.7	23.4
potted_meat_can	82.9	72.9	85.8	**76.7**	85.0	71.4	**86.4**	76.2	71.4	31.2
banana	**96.0**	85.2	95.0	88.2	95.8	**90.0**	91.3	81.2	90.0	83.1
pitcher_base	**98.8**	**94.3**	93.6	88.5	95.2	90.8	94.6	90.1	90.8	90.1
bleach_cleanser	**94.4**	80.5	85.3	73.0	83.1	70.8	90.3	**81.2**	70.8	62.9
bowl*	84.0	84.0	92.3	92.3	**93.4**	**93.4**	81.4	81.4	93.4	87.4
mug	**96.9**	87.6	84.9	69.6	95.5	**90.0**	91.3	81.4	90.0	71.0
power_drill	91.9	78.7	**92.6**	**86.1**	92.5	85.2	92.3	85.5	85.2	73.6
wood_block*	77.3	77.3	84.3	84.3	**93.0**	**93.0**	81.9	81.9	93.0	93.0
scissors	68.4	43.7	**93.3**	**87.0**	80.9	71.2	75.4	60.9	71.2	42.5
large_marker	**87.4**	76.2	84.9	76.6	85.2	**77.0**	86.2	75.6	77.0	14.7
large_clamp*	69.3	69.3	92.0	92.0	**94.7**	**94.7**	74.3	74.3	94.7	94.1
extra_large_clamp*	73.6	73.6	**88.9**	**88.9**	80.7	80.7	73.3	73.3	80.7	65.7
foam_brick*	90.4	90.4	90.7	90.7	**93.8**	**93.8**	81.9	81.9	93.8	78.9
MEAN	89.1	80.2	90.1	82.6	**91.2**	**83.3**	88.1	81.9	83.3	61.4

Symmetric objects are denoted by *. The best results are shown in bold.

6.3.5 Experimental Results

In this section, we evaluate the performance of our proposed YOLOPose model and its variants and compare it with other keypoint-based pose estimation 6D pose estimation methods. We report the quantitative results using YCB-Video dataset discussed in Section 5.4.1 in terms of the ADD and the ADD-S metrics presented in Section 5.4.2. The γ and δ hyperparameters in $\mathcal{L}_{kp}$ (Eq. (6.5)) are set to 1 and 10, respectively. While computing the Hungarian loss, the pose loss component is weighted down by a factor of 0.05. The cardinality of the predicted set N=20. The model takes images of the size 640×480 as input and is trained using the AdamW optimizer (Loshchilov and Hutter, 2017) with an initial learning rate of 10^{-4} for 150 epochs. Afterward, the model is trained additionally for 50 epochs, with a reduced learning rate by a factor of 0.1. The batch size is 32. Gradient clipping with a maximal gradient norm of 0.1 is applied. We present exemplar qualitative results in Fig. 6.5. In Table 6.1, we provide the quantitative per class area under the accuracy curve (AUC) of the ADD-S and ADD-(S) metrics discussed in Section 5.4.2, in detail. Both YOLOPose and YOLOPose-A perform well across all object categories and achieve higher AUC scores than the methods in comparison. YOLOPose-A achieves an impressive AUC of ADD-S and ADD-(S) score of 91.2 and 83.3, respectively, which is an improvement of 1.1 and 0.7 over the YOLOPose model. In terms of the individual objects, YOLOPose-A performs significantly better than the mean on `mustard bottle`, `bowl`, `large clamp`, and `foam brick`, while performing worse than the mean on `master chef can`, `tuna fish can`, `bleach cleanser`, and `scissors`. Interestingly, our methods perform well on identical `large clamp` and `extra large clamp`, whereas both the competing methods perform poorly on these objects. Real-world robotic applications require handling objects of different sizes and this necessitates highly accurate pose estimates. The standard procedure of reporting the AUC of ADD-(S) and ADD-S metrics with a fixed threshold of 0.1m does not take the object size into account. To better reflect the performance of our method on smaller objects, we present the AUC of ADD-(S) and ADD-S metric with a threshold of 10% of the object diameter. We denote this metric as AUC of ADD-(S) and ADD-S @0.1d. The accuracy of the proposed method drops significantly for smaller objects while using the object-specific threshold. In particular, the AUC of ADD-(S)@0.1d score for `tuna fish can`, `gelatin box`, and `large marker` are less than 30. This could be due to the fact that the Pose Loss discussed in Section 6.3.4 is computed using the subsampled model points and smaller objects contribute less to the overall loss. In Table 6.2, we also present a comparison of the ADD-S and the mean AUC ADD-S and ADD-(S) scores of the predominant RGB as well as RGB-D methods. Benefiting from the geometric features imparted by the depth information, RGB-D methods outperform RGB-only methods. However, RGB-only methods are catching up with the RGB-D methods fast (Sundermeyer et al., 2023).

The FFNs in our model generate the set predictions from the decoder output embeddings, which are the result of cross-attention between the object queries and the encoder output embeddings. Each encoder output embedding corresponds to a specific image pixel. This allows us to investigate the pixels that contribute the most to each object prediction. In Fig. 6.6, we visualize the decoder cross-attention corresponding to four different object detections, where the attended regions correspond to the object's spatial position in the image very well. Moreover, looking closely at the pixels with the highest attention score reveals the object parts that contribute most to the object predictions. For example, in Fig. 6.6(a), the tip and base of the `power drill` contribute the most

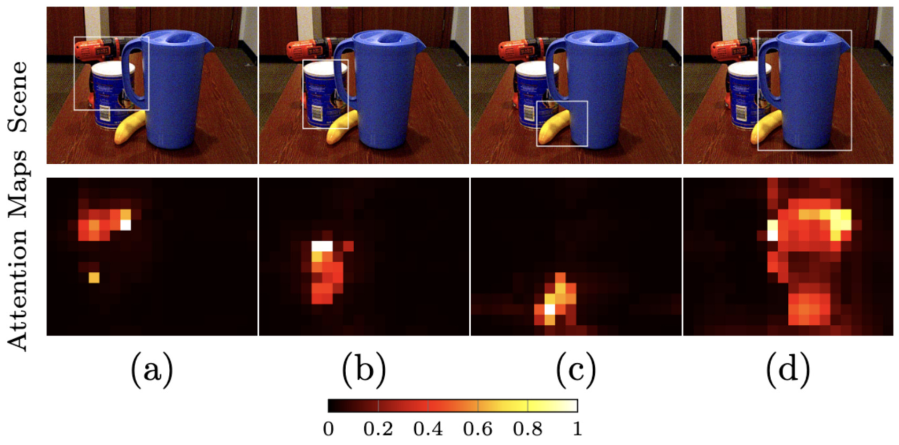


Figure 6.6: Top: Object detections predicted by bounding boxes in the given image. Bottom: Decoder cross-attention maps for the object queries corresponding to the predictions in the first row.

and in Fig. 6.6(d), the spout and the handle `pitcher base` contribute the most. Note that in Fig. 6.6(a), the base is severely occluded and the base barely visible. Despite being occluded, the attention mechanism focuses on the base heavily, which demonstrates the significance of the base in `power drill` pose estimation.

In Table 6.3, we present a quantitative comparison of the YOLOPose variant discussed in Section 6.3.7. Variant A performs the best among the variants. This can be attributed to the additional object-specific information contained in the output embedding.

6.3.6 Effectiveness of Keypoint Representations

We compare various keypoints representations, namely 3D *bounding box* (BB) keypoints, random keypoints sampled using the FPS algorithm, and our representation of choice, the *interpolated bounding box* (IBB) keypoints. We use the OpenCV implementation of the RANSAC-based EPnP algorithm with the same parameters to recover the 6D object pose from the predicted keypoints. Since EPnP does not contain any learnable components, this experiment serves the goal of evaluating the ability of the YOLOPose model to estimate different keypoint representations in isolation. YOLOPose is trained using only the ℓ_1 loss in the case of BB and FPS representations, whereas for the IBB representation, ℓ_1 is combined with the cross-ratio loss described in Section 6.3.4. Table 6.4 reports object pose estimation performance for the different representations. When used in conjecture with the EPnP solver, the FPS keypoints performed worse than all other representations. The reason is that the locations of FPS keypoints are less intuitive, making them more difficult to predict, especially for our proposed model that needs to deal with all objects in the YCB-Video dataset. In contrast, the IBB representation yields the best performance. We conjecture that as the cross-ratio loss based on the prior geometric knowledge preserves the keypoints geometrically, this representation is the appropriate choice for our method where a single model is trained for all objects.

Table 6.2: Comparison of inference time and accuracy on the YCB-Video dataset.

Input	Method	ADD-(S)	AUC of ADD-S	AUC of ADD-(S)	Inference Time [ms/frame]
RGB	CosyPose[†] (Labbe et al., 2020)	-	89.8	**84.5**	395
	PoseCNN (Xiang et al., 2018)	21.3	75.9	61.3	-
	GDR-Net (Wang et al., 2021)	49.1	89.1	80.2	65
	YOLOPose (Ours)	65.0	90.1	82.6	**17**
	YOLOPose-A (Ours)	**69.0**	**91.2**	83.3	22
RGB-D	PVNet3D (He et al., 2020)	-	95.5	91.8	170
	PVNet3D+ICP (He et al., 2020)	-	96.1	92.3	190
	FFB6D (Xiang et al., 2018)	-	96.6	92.7	**75**
	FFB6D+ICP (Xiang et al., 2018)	-	**97.0**	**93.7**	95

[†] indicates the refinement-based method.

Table 6.3: Quantitative comparison of the YOLOPose variants.

Method	AUC of ADD-(S)	AUC of ADD-S	Parameters $\times 10^6$
YOLOPose	82.6	90.1	**48.6**
Variant A	**83.3**	**91.2**	53.2
Variant B	82.8	91.0	53.4
Variant C	82.8	90.9	52.8

Table 6.4: Comparison of different keypoint representations.

Method	ADD-(S)	AUC of ADD-(S)
FPS + PnP	31.4	56.9
handpicked + PnP	31.5	55.7
IBB + PnP	**56.0**	**74.7**
IBB + PnP for R; head for t	63.9	82.3
IBB + heads for R and t	**65.0**	**82.6**

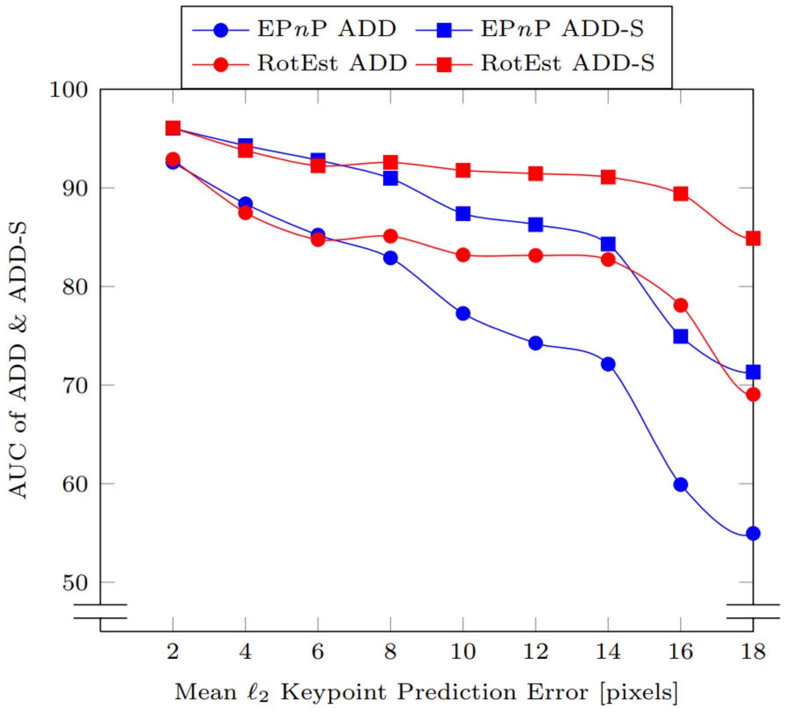
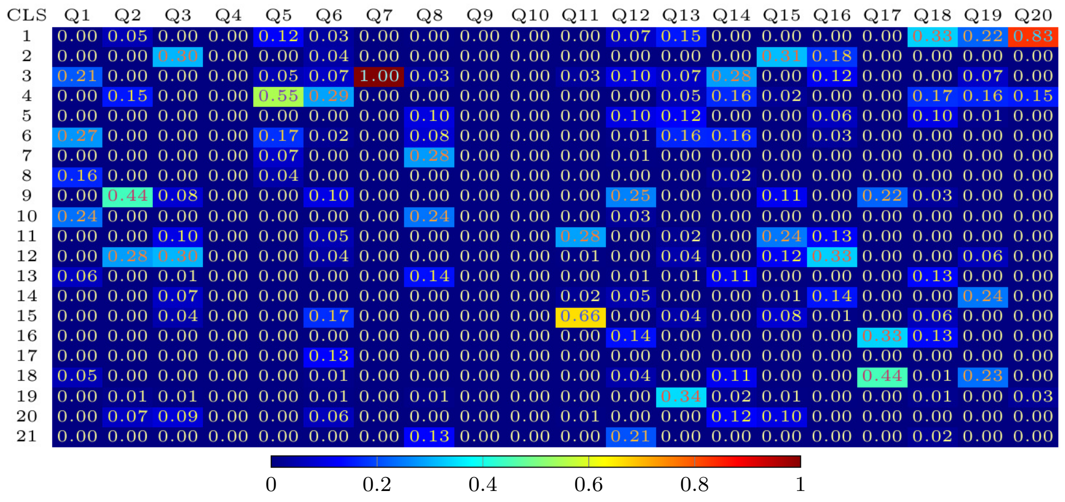


Figure 6.7: Comparison of the pose estimation accuracy with respect to the keypoint estimation accuracy between EPnP and RotEst. In the case of highly accurate keypoint estimation, EPnP performs comparably to RotEst. However, the RotEst module is more robust against inaccuracies in keypoint estimation. Overall, RosEst performs better than EPnP.

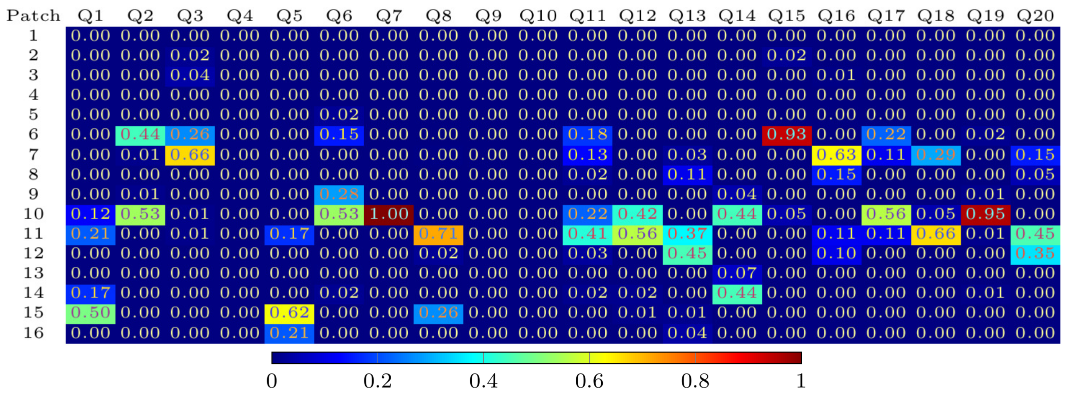


CLS	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
1	0.00	0.05	0.00	0.00	0.12	0.03	0.00	0.00	0.00	0.00	0.00	0.07	0.15	0.00	0.00	0.00	0.00	0.33	0.22	0.83
2	0.00	0.00	0.30	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.18	0.00	0.00	0.00	0.00
3	0.21	0.00	0.00	0.00	0.05	0.07	1.00	0.03	0.00	0.00	0.03	0.10	0.07	0.28	0.00	0.12	0.00	0.00	0.07	0.00
4	0.00	0.15	0.00	0.00	0.55	0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.16	0.02	0.00	0.00	0.17	0.16	0.15
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.10	0.12	0.00	0.00	0.06	0.00	0.10	0.01	0.00
6	0.27	0.00	0.00	0.00	0.17	0.02	0.00	0.08	0.00	0.00	0.00	0.01	0.16	0.16	0.00	0.03	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.28	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.16	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00
9	0.00	0.44	0.08	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.11	0.00	0.22	0.03	0.00	0.00
10	0.24	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	0.00	0.00	0.10	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.28	0.00	0.02	0.00	0.24	0.13	0.00	0.00	0.00	0.00
12	0.00	0.28	0.30	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.01	0.00	0.04	0.00	0.12	0.33	0.00	0.00	0.06	0.00
13	0.06	0.00	0.01	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.01	0.01	0.11	0.00	0.00	0.00	0.13	0.00	0.00
14	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.05	0.00	0.00	0.01	0.14	0.00	0.00	0.24	0.00
15	0.00	0.00	0.04	0.00	0.00	0.17	0.00	0.00	0.00	0.00	0.66	0.00	0.04	0.00	0.08	0.01	0.00	0.06	0.00	0.00
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.33	0.13	0.00	0.00
17	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18	0.05	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.11	0.00	0.00	0.44	0.01	0.23	0.00
19	0.00	0.01	0.01	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.34	0.02	0.01	0.00	0.00	0.01	0.00	0.03
20	0.00	0.07	0.09	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.12	0.10	0.00	0.00	0.00	0.00	0.00
21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.00	0.21	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00

Figure 6.8: Correlation between object queries and the detected object classes. Except for queries 7 and 20, the correlation is weak.

6.3.7 Effectiveness of RotEst

After deciding on the keypoint representation, we compare the performance of the learnable feed-forward rotation and translation estimators against the analytical EPnP algorithm. The factors that impact rotation and translation components are different Li, Wang, and Ji (2019). The rotation is highly affected by the object's appearance in a given image. In contrast, the translation is more vulnerable to the size and the location of the object in the image. Therefore, we decide to estimate rotation and translation separately. In Table 6.4, we report the quantitative comparison of the different variants. One can observe that using only the rotation from EPnP and directly regressing the translation improved the accuracy significantly. In general, RotEst performs slightly better than using EPnP orientation and direct translation estimation. Furthermore, the RotEst module and the translation estimators are straightforward MLPs and thus do not add much execution time overhead. This enables YOLOPose to perform inference in real-time. Moreover, to quantify the robustness of the RosEst module compared to the EPnP algorithm against the inaccuracies in keypoint estimation. We exclude the symmetric objects in the comparison. Figure 6.7, we present the comparison between the AUC of ADD and ADD-S scores achieved by using the RotEst module and using the EPnP algorithm for recovering 6D pose from the estimated IBB keypoints. We discretize the average ℓ_2 pixel error in keypoint point estimation into bins of size two and average the AUC scores for all predictions corresponding to each bin. EPnP performs equally well in terms of both the AUC of ADD-S metrics compared to the RotEst module when the keypoint estimation accuracy is high. In the case of large keypoint estimation errors, the RotEst module demonstrates a significantly higher degree of robustness compared to the EPnP algorithm.



Patch	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.44	0.26	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.18	0.00	0.00	0.00	0.93	0.00	0.22	0.00	0.02	0.00
7	0.00	0.01	0.66	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.03	0.00	0.00	0.63	0.11	0.29	0.00	0.15
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.11	0.00	0.00	0.15	0.00	0.00	0.00	0.05
9	0.00	0.01	0.00	0.00	0.00	0.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.01	0.00
10	0.12	0.53	0.01	0.00	0.00	0.53	1.00	0.00	0.00	0.00	0.22	0.42	0.00	0.44	0.05	0.00	0.56	0.05	0.95	0.00
11	0.21	0.00	0.01	0.00	0.17	0.00	0.00	0.71	0.00	0.00	0.41	0.56	0.37	0.00	0.00	0.11	0.11	0.66	0.01	0.45
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.03	0.00	0.45	0.00	0.00	0.10	0.00	0.00	0.00	0.35
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00
14	0.17	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.02	0.02	0.00	0.44	0.00	0.00	0.00	0.00	0.01	0.00
15	0.50	0.00	0.00	0.00	0.62	0.00	0.00	0.26	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 6.9: Correlation between object queries and the image patch in which the object is detected. The images are divided into 4×4 patches. Compared to the correlation between object queries and the detected object classes shown in Fig. 6.8, the correlation between object queries and image patches is stronger.

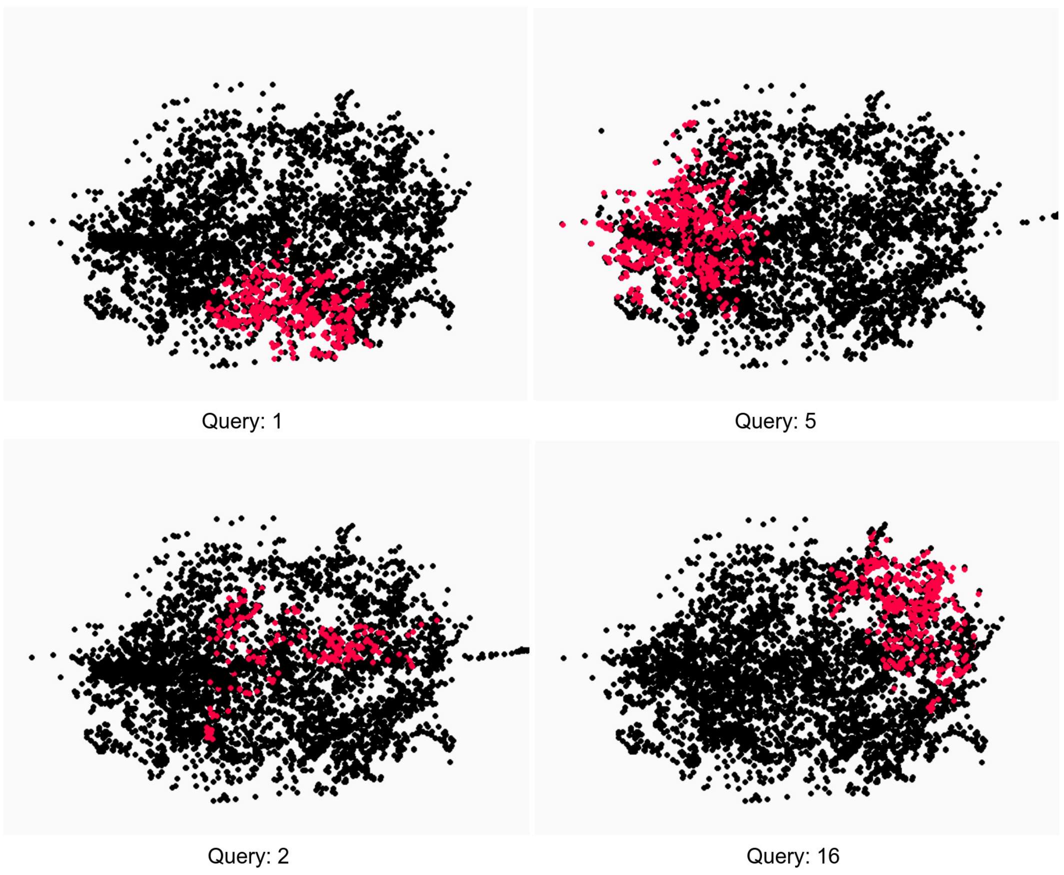


Figure 6.10: Visualization of the center of the bounding boxes predicted by an object query. Black dots represent all the spatial positions of the ground-truth bounding boxes normalized to the image size present in the test dataset. Red dots represent the bounding boxes predicted by an object query. Object queries specialize in detecting objects in specific regions of the image.

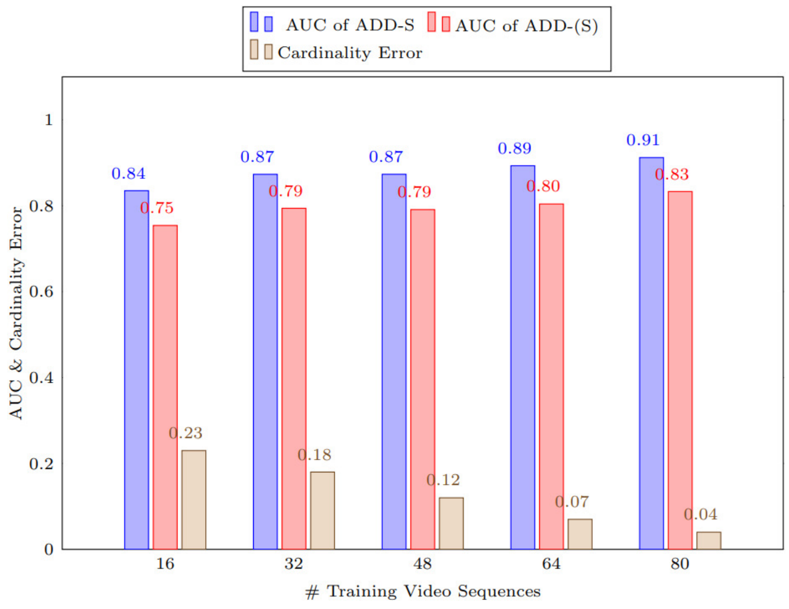


Figure 6.11: Comparison of pose estimation and object detection accuracies using a different number of video sequences for training. The AUC scores are normalized to the range [0, 1].

6.3.8 Understanding Object Queries

To understand the role of the learned object queries in the YOLOPose architecture, we analyzed the correlation between the object queries and the detected object class ids as well as the object bounding boxes. Since we use only 20 object queries in the YOLOPose architecture—compared to 100 in the DETR (Carion et al., 2020) architecture—we can investigate the object queries individually in detail. To this end, we compute the correlation between object queries and class ids, and image patches that form a 4×4 grid. In Fig. 6.8, we visualize the correlation between the object queries and class ids. Except for queries 7 and 20, the correction is weak. In contrast, the correlation between the object queries and the image patch of the detected object is stronger (see Fig. 6.9). Note that queries 4, 9, and 10 do not correspond to any objects. This is the case only for the test dataset. In the case of the training dataset, all the object queries correspond to object detections. Moreover, we visualized the spatial location of the center of the bounding boxes predicted by object queries. In Fig. 6.10, we show exemplar visualizations. The visualizations also reveal that the object queries specialize in object detection in specific regions of the image.

6.3.9 Dataset Size-Accuracy Trade-off

Vision Transformer models match or outperform CNN models in many computer vision tasks, but they require large datasets for pre-training (Wang et al., 2022; Cao, Yu, and Wu, 2022; Gani, Naseer, and Yaqub, 2022). Furthermore, obtaining large-scale 3D annotations are significantly harder than 2D annotations. Thus, the 3D datasets are supplemented with easy-to-acquire synthetic datasets. The YOLOPose architecture consists of a CNN backbone model for feature extraction and attention-based encoder-decoder module for

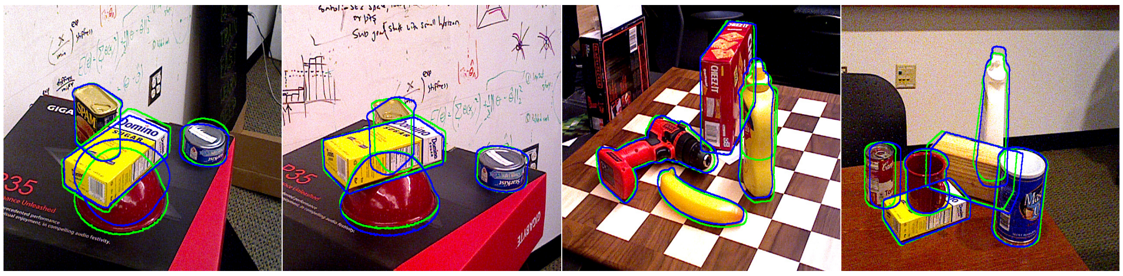

(a) (b) (c) (d)

Figure 6.12: Typical failure cases for the YOLOPose model. The pose estimation accuracy of our approach is hampered by occlusion. Ground-truth and predicted object poses are visualized as object contours in green and blue colors, respectively.

set prediction. Learning set prediction is significantly challenging due to the additional overhead of finding the matching pairs between the ground-truth and the predicted sets and results in a low convergence rate. To mitigate this issue, we pre-train our model on the COCO dataset (Lin et al., 2014) for the task of object detection formulated as set prediction. The COCO dataset comprises 328,000 images with bounding annotations for 80 object categories. The COCO dataset pre-training enables faster convergence while training on the YCB-Video dataset. To quantify the dataset size-accuracy trade-off in training our model for the task of joint object detection and pose estimation formulated as set prediction, we train our model with different subsets of the YCB-Video dataset of varying sizes. As discussed in Section 5.4.1, YCB-Video consists of 92 video sequences. 80 of which are used for training and the rest of them are used for testing. Additionally, Xiang et al. (2018) provide 80,000 synthetic images for training as well. We created five different variants of the training set by using only a subset of the 80 training sequences. The first variant consists of only 16 video sequences and each subsequent variant consists of 16 additional video sequences added to the previous variant progressively. All five variants are supplemented with the complete set of synthetic images. We train one YOLOPoseA model for each of the dataset variants and evaluate the performance of the models on the test set consisting of twelve video sequences. In Fig. 6.11, we present the AUC of ADD-S and ADD-(S) scores as well as the *cardinality error* (CE), which is defined as the ℓ_1 error between the cardinality of the ground-truth and the predicted set. The model trained with the smallest training set variant consisting of only 16 video sequences achieves an AUC of ADD-S and ADD-(S) score of 83.5 and 75.4, respectively, whereas the model trained using the complete training videos achieves an AUC of ADD-S and ADD-(S) score of 91.2 and 83.3, respectively. The difference between the models trained using the smallest training set variant and the largest is even more significant in terms of the cardinality error—0.23 compared to 0.04. This demonstrates the need for large datasets with a wide range of scene configurations. Presented with smaller datasets with less variability in scene configuration, the YOLOPose model not only performs poorly in terms of pose estimation accuracy but also in terms of object detection accuracy.

6.3.10 Typical Failure Cases

In Fig. 6.12, we show examples of low-accuracy pose predictions—particularly in the case of partially-occluded objects. One of the commonly observed failure cases is the *bowl* ob-

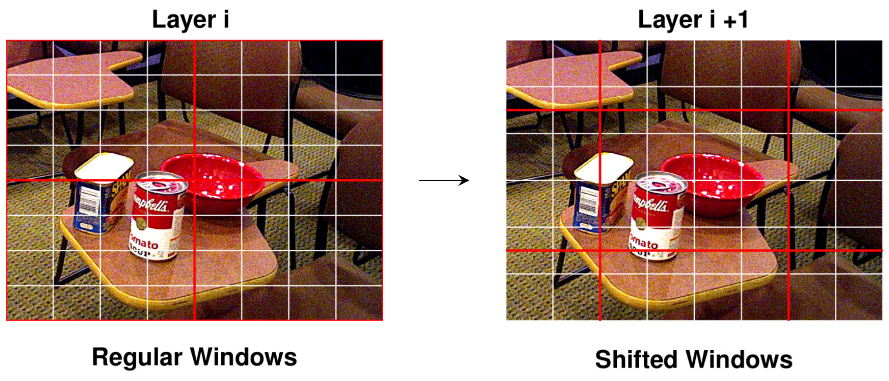


Figure 6.13: Shifting window local attention mechanism. Self-attention is restricted to the local window shown in red grids. The windows are shifted between the layers enabling global interaction within each block.

ject often predicted facing upwards even though *bowl* is placed downwards (See Fig. 6.12a). This is due to the limitation of the symmetry-aware SLoss (Eq. (6.9)). The SLoss is defined as the ℓ_2 distance between the closest model points of the object in the predicted and the ground-truth poses. For some objects—`bowl`, for example—the 180° flip error is not penalized enough during training.

6.4 Fully Vision-Transformer Models

While YOLOPose achieves impressive results, it relies solely on the attention mechanism to learn joint object detection and pose estimation. Thus, YOLOPose does not benefit from incorporating inductive biases into its architecture. Convolutional neural networks, on the other hand, hard-wire various inductive biases into their architecture. The success of CNNs is largely attributed to the inductive biases incorporated into their architectural design (LeCun, Bengio, et al., 1995; Behnke, 2003b; Cohen and Shashua, 2017; Schulz and Behnke, 2012). In this section, we discuss various design strategies for imbuing inductive biases into the YOLOPose model.

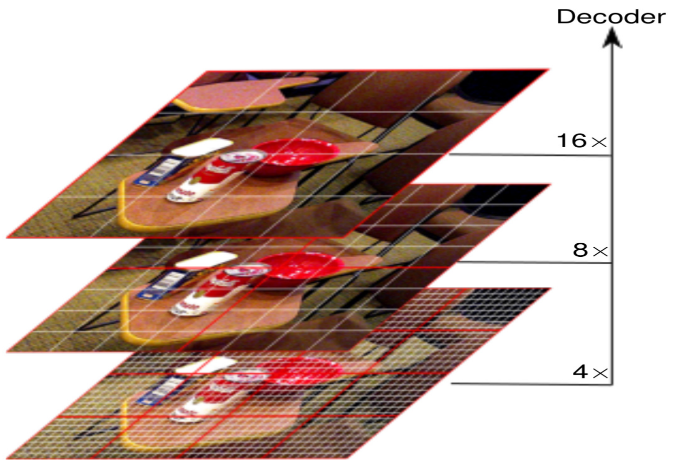


Figure 6.14: Hierarchical processing of image features. White grids represent the image patch and red grids represent the self-attention window. The size of the self-attention window and the number of feature maps increase through the hierarchy.

6.4.1 Local Hierarchical Shifting Window Attention

The encoder module in YOLOPose utilizes self-attention between the image features extracted by a CNN backbone. This enables aggregating information from all spatial locations in the encoder module. In terms of the design, one of the main shortcomings of YOLOPose is that the backbone module is exclusively CNN-based and the encoder module is exclusively attention-based. Thus, the backbone module does not benefit from the self-attention mechanism. *Vision Transformer models* (ViT) (Dosovitskiy et al., 2021) addressed this issue by designing a backbone model based exclusively on the transformer architecture. Raghu et al. (2021) noted that in ViT, the similarity of the lower layer features and the higher layer features is stronger than in the case of CNN-based ResNet model. Based on this observation, the authors concluded that the self-attention mechanism along with the skip connections enables lower layers in the ViT model to learn global features. However, ViT suffers from two major limitations:

1. feature maps used in the model are low resolution and
2. complexity of the attention mechanism increases quadratically.

These factors limit the suitability of ViT as a backbone model. To address this issue, Liu et al. (2021) proposed to incorporate hierarchical processing and local sliding window attention in the design of the transformer model. The pixels are divided into crops. All pixels belonging to an image crop are projected linearly to features of dimension d. The hierarchical processing of the features is shown in Fig. 6.14. Unlike ViT, in which all the image patches in any particular layer interact with all other patches, the interaction is restricted to a local window (shown in Fig. 6.13). The layers of the model are grouped into blocks. The size of the attention window and the number of feature maps increases progressively higher in the hierarchy. The attention window is shifted between the layers in the same block. This ensures the global interaction of the features within each block. The hierarchical processing of features and limiting the attention to a local window enables linearly scaling computational complexity. We refer to our implementation of the pose estimation model with local hierarchical shifting window attention as YOLOPose-LHSA.

6.4.2 Multi-resolutional Deformable Attention

The attention mechanism offers a simple yet effective mechanism to model long-range dependencies in input tokens. Despite the advantages the attention mechanism offers for computer vision tasks, the computational cost of MHA, especially for high-resolution images, remains high. To address this issue, Zhu et al. (2021) introduced *deformable multi-head attention* (DMHA). In Fig. 6.15, we depict the DMHA model for pose estimation. For a query token z, instead of computing attention over all the key tokens Ω_x, DMHA computes attention over a small set of 2D reference points $p \in \Omega_p$. The reference points

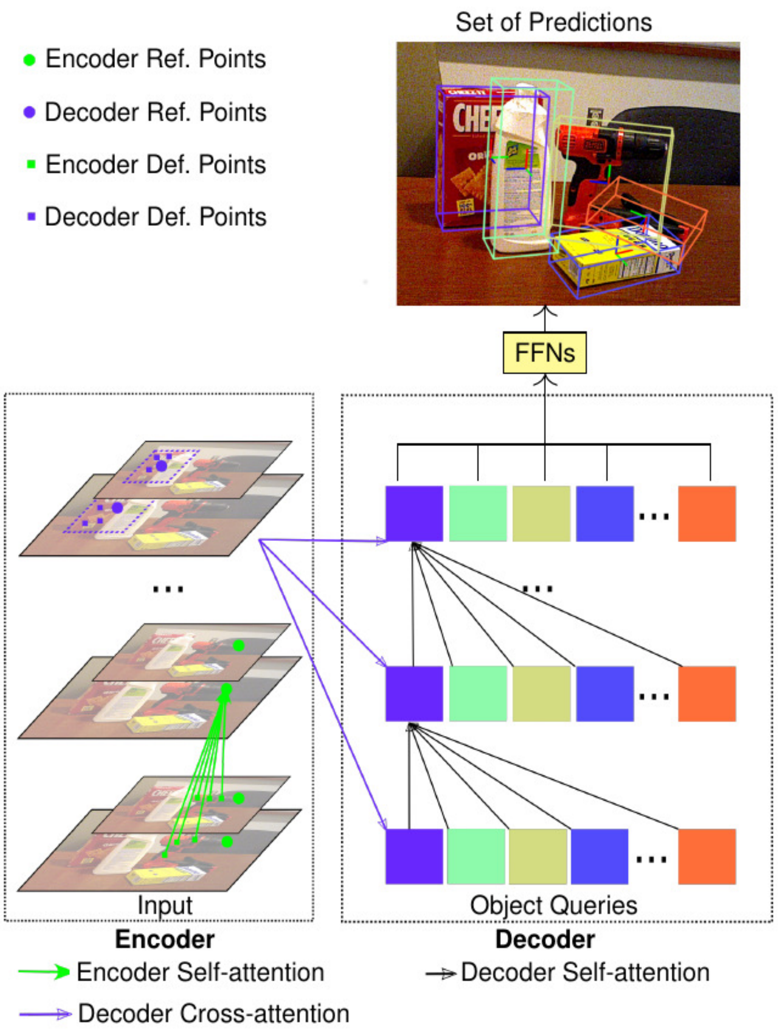


Figure 6.15: Multi-resolutional deformable attention model. Our architecture is based on an encoder-decoder design. In the encoder, image patch features of multiple resolutions are extracted using linear projection, and the self-attention mechanism is used to generate encoder embeddings. In the decoder, cross-attention is performed between the encoder embeddings and learned embeddings known as *object queries* to generate object embeddings. The object queries are initialized randomly at the beginning of training and learned jointly with the training objective. During inference, the object queries remain fixed. In contrast to the standard attention, in which attention is computed between all image patch features, in deformable attention, the attention operation is performed only between the deformed reference points. From the object embeddings, object predictions are generated using feed-forward neural networks (FFNs) in parallel. Object pose predictions generated by our model are visualized using 3D bounding boxes.

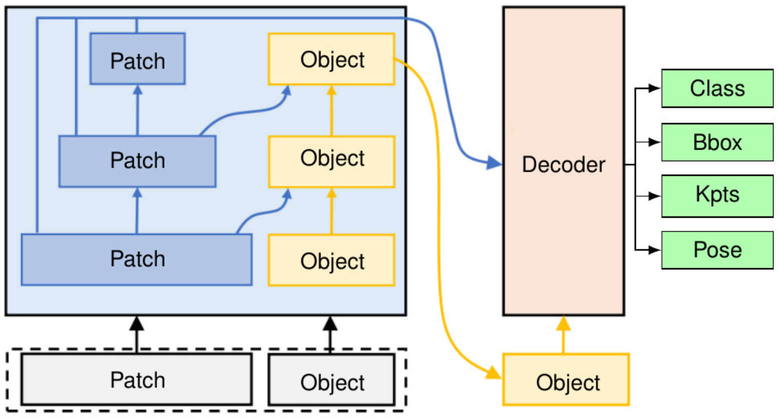


Figure 6.16: Early fusion of object queries. The patch embeddings and the object queries interact within the encoder module at all layers. Thus, not only the high-level features interact with the object queries but also the features of all resolutions.

are allowed to deform and the deformation is learned from the input tokens. Formally, MHA, introduced in Eq. (5.1), is extended as,

$$\mathrm{DMHA}(z_q, p_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} \cdot W_m^{'} x(p_q + \Delta p_{mqk})) \right], \quad (6.10)$$

where k represents the sampled key tokens and Δp_{mqk} denotes deformation offset. Bilinear interpolation enables fractional offsets. Furthermore, we incorporate multi-resolution feature processing (MR-DMHA) into the deformable attention:

$$\text{MR-DMHA}(z_q, p_q, \{x\}_{l=1}^{L}) = \sum_{m=1}^{M} W_m \left[\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mqk} \cdot W_m^{'} x^{l}(\phi(\hat{p}_q) + \Delta p_{mqk}) \right], \quad (6.11)$$

where l represents the feature level, and the function $\phi(\hat{p}_q)$ re-scales the pixel coordinates corresponding to the feature level. We refer to our multi-object pose estimation model based on MR-DMHA as YOLOPose-DMHA.

6.4.3 Early Fusion of Object Queries

While the MR-DMHA enables efficient attention with linear complexity, the encoder module interacts with the object queries only at the final encoder layer. This results in the encoder module learning generic features in the earlier layers and only the final layer learning features relevant to object prediction. Enabling encoder-object query interaction at the early layers helps the encoder to focus on features more relevant to the object predictions. Following Song et al. (2022), we introduce cross-attention between object queries and patch embeddings early in the encoder module. In each encoder layer, self-attention is performed between the patch embeddings and additionally, cross-attention is performed between object queries and patch embeddings of the previous layer, as shown

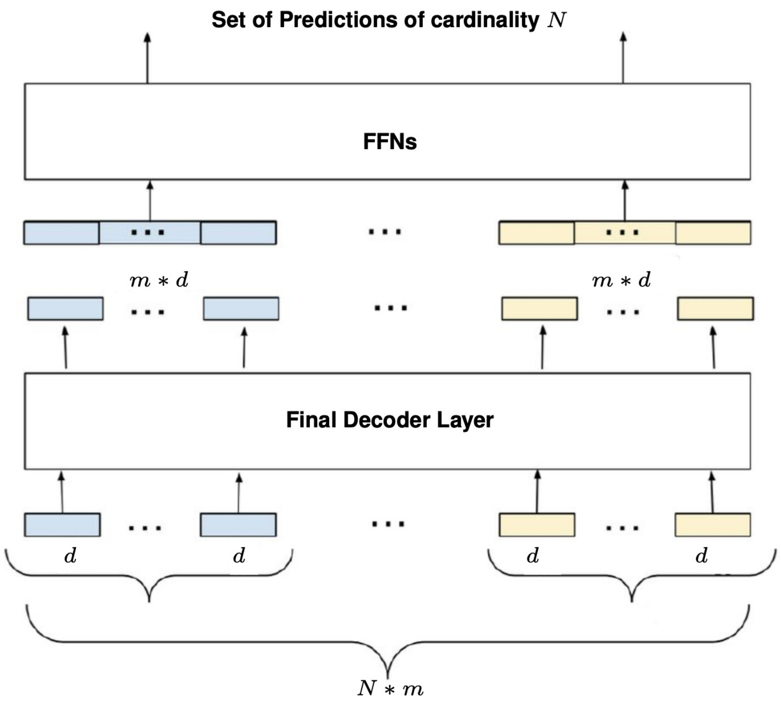


Figure 6.17: Query Aggregation. In contrast to the standard approach of using the same number of object queries as the cardinality of the set N, in the query aggregation approach, we set the number of object queries to $N \times m$. The query aggregation factor m is a hyperparameter. $N \times m$ decoder output embeddings of dimension d are concatenated to form N object embeddings of dimension $d \times m$ that are processed by the FFNs to generate N object predictions.

in Fig. 6.16. In the decoder module, cross-attention is performed between the aggregated patch embeddings from the encoder and the object query to generate object embeddings. The object embeddings are processed by the FFNs to generate object predictions. We call this variant YOLOPose-EarlyFusion.

6.4.4 Query Aggregation

The learned object queries play a crucial role in DETR-like architectures. Despite their importance, the object queries are not fully understood. One of the major reasons behind this lack of understanding is the fact that neither the architecture itself nor the loss function contains any mechanism to bind the object queries specifically to object classes or locations. Zhang et al. (2023) observed that using more object queries improves model accuracy. In our models, the number of object queries equals the cardinality of the set we predict. Thus, increasing the number of object queries also increases the computation cost of bipartite matching and thus, the overall training time. We propose a novel approach for increasing the number of object queries while keeping the computational cost low. We call this method *query aggregation* (shown in Fig. 6.17). To generate a set of predictions of cardinality N, the standard approach uses N object queries of dimension d. In contrast to the standard approach, in the query aggregation approach, we set the number of object queries to $N \times m$, where the aggregation factor m is a hyperparameter. After the last decoder layer, we concatenate each set of m embeddings to generate one object

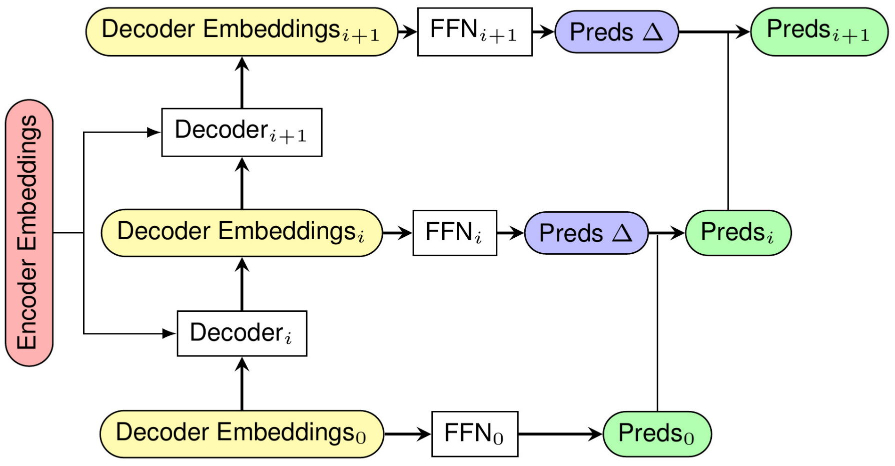


Figure 6.18: Prediction refinement. At each decoder layer, only a small Δ is predicted on top of the previous predictions enabling the refinement of predictions through the decoder layers.

embedding. Thus, from $N \times m$ output embeddings we generate N object embeddings. The FFNs process the N object embeddings to generate a set of object predictions. By decoupling the number of decoder output embeddings and the number of object embeddings, we enable a larger number of embeddings in the decoder layer without increasing the cardinality of the predicted set.

6.4.5 Refinement of Object Predictions

The decoder module in the standard architecture consists of six decoder layers. The output embeddings of each of the decoder layer serves an input for the subsequent decoder layer. The output embeddings of the final decoder layer are processed by the FFNs to generate object predictions. Instead of generating object predictions directly once, generating an initial prediction and refining it to generate the final object predictions allows the model to iteratively improve the initial predictions as well as the overall accuracy. The design of decoder layers naturally suits refinement. In Fig. 6.18, we show the refinement of the object predictions in the decoder module. Each decoder layer contains its own independent set of FFNs. The first decoder layer performs cross-attention between the encoder embeddings and the object queries to generate decoder embeddings that are processed by a corresponding set of FFNs to generate object predictions. The FFNs of subsequent decoder layers generate only a small Δ to refine the predictions made by the previous layer. This allows for the model to iteratively refine the initial predictions and generate more accurate final predictions.

6.4.6 Experimental Results

We implement our models using the PyTorch (Paszke et al. (2019)) library. Our models are trained using NVIDIA A100 GPUs with 40 GBs of memory. The size of the input image is 640×480. The models are trained for 200 epochs with a batch size of 32 using AdamW optimizer with an initial learning rate of 10^{-4}. During training, we supplement

Table 6.5: Results on YCB-Video.

Metric	AUC of ADD-S			AUC of ADD-(S)		
Method	YOLOPose	YOLOPose V2	YOLOPose DMHA	YOLOPose	YOLOPose V2	YOLOPose DMHA
master_chef_can	91.3	91.7	90.3	64.0	**71.3**	66.7
cracker_box	86.8	92.0	**92.3**	77.9	83.3	**86.0**
sugar_box	92.6	91.5	94.4	87.3	83.6	89.1
tomato_soup_can	90.5	87.8	89.2	77.8	72.9	76.3
mustard_bottle	93.6	96.7	96.5	87.9	93.4	93.3
tuna_fish_can	94.3	94.9	94.5	74.4	70.5	67.4
pudding_box	92.3	92.6	**95.5**	87.9	87.0	**91.9**
gelatin_box	90.1	92.2	**95.4**	83.4	85.7	91.8
potted_meat_can	85.8	85.0	**88.9**	**76.7**	71.4	76.4
banana	90.0	95.8	95.4	88.2	90.0	**91.0**
pitcher_base	93.6	95.2	94.9	88.5	90.8	89.9
bleach_cleanser	85.3	83.1	87.3	73.0	70.8	73.9
bowl*	92.3	**93.4**	91.9	**92.3**	**93.4**	91.9
mug	84.9	95.5	95.5	69.6	**90.0**	89.3
power_drill	92.6	92.5	**94.6**	86.1	85.2	**88.9**
wood_block*	84.3	**93.0**	**93.0**	84.3	**93.0**	**93.0**
scissors	**93.3**	80.9	89.5	**87.0**	71.2	76.2
large_marker	84.9	85.2	84.5	76.6	77.0	**77.4**
large_clamp*	92.0	**94.7**	**94.2**	92.0	**94.7**	94.2
extra_large_clamp*	**88.9**	80.7	79.2	**88.9**	80.7	79.2
foam_brick*	90.7	93.8	**95.0**	90.7	93.8	**95.0**
Mean	90.1	91.2	**92.0**	82.6	83.3	**84.7**

Symmetric objects are denoted by *. The best results are shown in bold.

Table 6.6: Results from Ablation Study.

Method	AUC of ADD-S	AUC of ADD-(S)	Params $\times 10^6$	fps
YOLOPose	90.1	82.6	**48.6**	**41.8**
YOLOPose V2	91.2	83.3	53.2	39.1
CosyPose (Labbe et al., 2020)	89.8	84.5	-	2.5
YOLOPose-LHSA	90.1	81.4	57	39.5
YOLOPose-DMHA 16 def pts	**92.0**	**84.7**	55.2	25.9
YOLOPose-DMHA 6 def pts	81.9	89.3	51.7	28.1
YOLOPose-DMHA 6 def pts + refine	90.2	83.0	87.1	25.6
YOLOPose-EarlyFusion	90.3	82.1	48.7	34.8

YOLOPose-LHSA: Local hierarchical attention-based model discussed in Section 6.4.1.
YOLOPose-DMHA: Model based on multi-resolution deformable multi-head attention discussed in Section 6.4.2.
YOLOPose-EarlyFusion: Model utilizing early fusion discussed in Section 6.4.3.
def pts: Number of deformable points.

the training images with the synthetic images provided with the dataset. The hyperparameters α, β, γ, and δ in Eqs. (6.3) and (6.5) are set to 2, 5, 10, and 1, respectively.

In Table 6.5, we report the quantitative comparison of the fully convolutional YOLOPose variants with the leading RGB model. For the sake of brevity, we refer the YOLOPose-A model presented in Table 6.2 as YOLOPose V2. The best-performing variant of our model is based on deformable multi-resolutional attention (YOLOPose-DMHA) discussed in Section 6.4.2 utilizing 16 deformable points. Our model achieves an impressive AUC of ADD-S score of 92.0 and AUC of ADD-(S) score of 84.7. Among the object categories, our model performs worse for `extra-large clamp` and `scissors`–both exhibit challenging geometry. `bowl`, for example, is often predicted upside down. Occlusion still remains a big challenge for our model. In Table 6.6, we compare the accuracy of the different fully vision transformer models

In comparison to YOLOPose-DMHA, YOLOPose-LHSA performs a little worse. This demonstrates that despite the careful design employing shifted window attention, the model suffers from inefficiencies in global dependency modeling. Moreover, Zhou et al. (2022) noted that the models based on local hierarchical shifting windows suffer from a lack of robustness. Although YOLOPose-EarlyFusion, based on the early fusion of object queries, performed better than YOLOPose-LHSA, it did not match the performance of YOLOPose-DMHA. However, in terms of the inference speed, YOLOPose-LHSA performs better than other models—except for the baseline YOLOPose model. Because of accessing random memory locations in deformable attention, YOLOPose-DMHA is the slowest.

6.4.7 Ablation Study

Query Aggregation

The cardinality N of the predicted set in the standard YOLOPose model is set to 20. This design choice is driven by the maximum number of objects present per image in the YCB-Video dataset. The query aggregation mechanism discussed in Section 6.4.4 enables for increasing the number of object queries without increasing the cardinality N of the predicted set. This allows the model to have more learnable object queries without adding any overhead to the costly bipartite matching step. We experimented with different query aggregation factor m. In Fig. 6.19, we present the results of training our model with different query aggregation factors. The performance of the model increases with the query aggregation factor and reaches the highest accuracy for factor 3 but drops for factor 4. However, compared the best performing YOLOPose variant with deformable multi-head attention, the model based on query aggregation achieves slightly lower accuracy in terms of all the pose estimation metrics.

Need for COCO Pre-training

Training the vision transformer models for set prediction is harder than training CNN models to perform single object pose prediction due to the usage of bipartite matching to find the matching pairs between the predicted and the ground-truth sets, which results in slower convergence. Moreover, training data requirements are also much larger for the set predictions task, compared to single object prediction. We hypothesize that in the initial phase of the training, the model learns to detect multiple objects in the image and only in

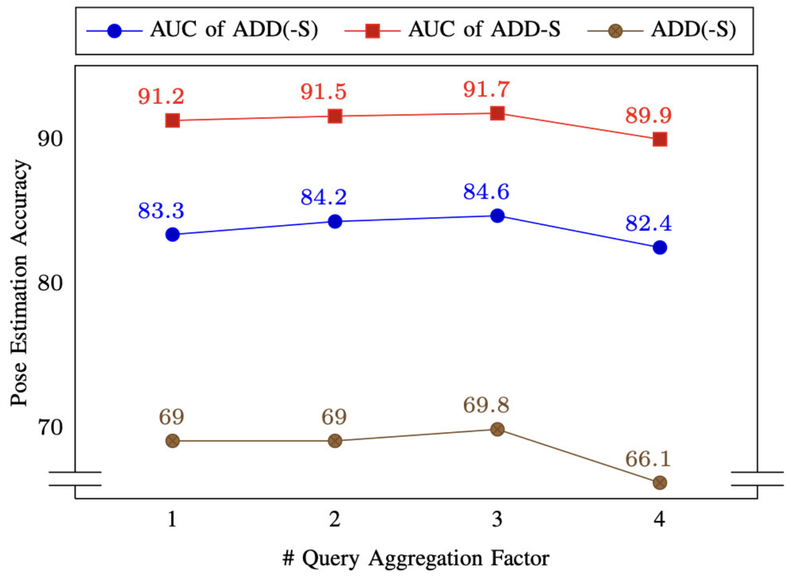


Figure 6.19: Results from query aggregation experiment.

Table 6.7: Effect of pretraining on the COCO dataset for Object Detection.

Method	AUC of ADD-S	AUC of ADD-(S))	ADD-(S)
Without pretraining	71.0	82.7	42.5
With pretraining	**81.4**	**90.1**	**61.1**

the later stages the model learns to predict keypoints and the pose parameters. Although the YCB-Video dataset (Xiang et al., 2018) is considerably larger than the other pose annotation datasets, it is not big enough to train vision transformer models for multi-object pose estimation. To overcome this limitation, we train our model initially on the COCO dataset for the task of multi-object detection (class probability and bounding box prediction) and then train the model for multi-object pose estimation on the YCB-Video dataset. In Table 6.7, we compare the pose estimation accuracy of models trained using only the YCB-Video dataset (Xiang et al., 2018) and using COCO dataset for pretraining. The model pretrained using the COCO dataset outperforms the model trained using only the YCB-Video dataset, highlighting the importance of large-scale pretraining for training vision transformers to learn the task of multi-object prediction.

Prediction Refinement

In Table 6.6, we present the results of the prediction refinement experiment. Refinement boosted the performance of YOLOPose-DMHA constructed with six deformable points by 0.9 and 1.1 accuracy points in terms of the AUC of ADD-(S) and AUC of ADD-(S) metrics, respectively. However, the improvements come at a cost of an increased number of parameters: $87.1{\times}10^6$ compared to $51.7{\times}10^6$. Interestingly, for YOLOPose-DMHA constructed using 16 deformable points that achieves an impressive accuracy of 92 AUC of ADD-S and 84.7 AUC ofADD-(S), the boost in performance is negligible.

6.5 Limitations

Both the T6D-Direct model described in previous Chapter (Chapter 5) and the YOLOPose model introduced in this Chapter follow the set prediction formulation. While formulating multi-object pose estimation as a set prediction problem facilitates the prediction of a varying number of objects in the given image, training the model needs complete set annotations for all objects in the given image. Most of the commonly used pose estimation datasets like Linemod-Occluded (Brachmann, 2020) and Linemod (Hinterstoisser et al., 2013) offer only partial annotations for training images. Thus, they are unsuitable for training our models. Acquiring complete pose annotations can be prohibitively expensive in real-world settings. Moreover, the pose loss function used to train our models needs 3D meshes of the objects. This limits the applicability of our models in the settings where the 3D meshes are not readily available.

6.6 Discussion & Conclusion

In this Chapter, we extend the multi-object pose estimation as set prediction formulation introduced in the previous Chapter by including keypoint regression and a learned rotation estimation module. The proposed YOLOPose model retains the simplicity and the fast inference capabilities of the T6D-direct model discussed in the previous Chapter while being significantly more accurate. We perform keypoint regression using the *intermediate bounding box* (IBB) representation. In our experiments, the (IBB) representation outperformed other standard keypoint representations. Moreover, we show empirically that the learned rotation estimation is considerably robust against noisy keypoint estimations. Additionally, we established the importance of training for object detection on the COCO dataset first before training for joint object detection and pose estimation on the YCB-Video dataset. This shows that training the YOLOPose model for set prediction needs more data and once the model is trained for this task, the complexity of adapting the model to joint object detection and pose estimation is significantly reduced. Moreover, by quantifying the dataset size-accuracy trade-off, we demonstrated that the model achieves a reasonable pose estimation accuracy using only 20% of the training set, but had a large number of object detection errors. When trained on the complete training set, the pose estimation accuracy improved slightly but the object detection errors dropped remarkably. Overall, YOLOPose and its variants performed competitively with the leading pose estimation methods. Furthermore, we analyzed the role of object queries in the YOLOPose model. Based on empirical evidence, we concluded that the correlation between object queries and the image patches are stronger than the correlation between object queries and the object classes. Thus, object queries encode the spatial positioning of objects. Despite these improvements, the YOLOPose architecture has a major drawback. The CNN backbone feature extractor module and the attention-based encoder-decoder modules were isolated from each other. Later, we investigated different variants of the fully vision transformer YOLOPose architecture. Among the variants experimented, the multi-resolutional deformable multi-head attend variant of YOLOPose produced the best accuracy while being slightly slower and containing more parameters than the YOLOPose model. In terms of future research directions, the keypoint regression pipeline proposed naturally extends to category-level object pose estimation (Lin et al., 2022; Di et al., 2022; Lin et al., 2021; Wang et al., 2019).

7 Multi-Object Pose Tracking

Object pose estimation methods we discussed in the previous chapters achieve impressive results by processing only a single image. However, single-view RGB models do not make use of the rich temporal information contained in the video sequence. In this Chapter, we augment the YOLOPose model introduced the previous Chapter with attention-based temporal fusion mechanism. Temporal fusion enables improved pose estimation accuracy as well as better object detection accuracy in highly cluttered dynamic environments.

Statement of Personal Contribution

Source material for this Chapter has been adapted from the following publications:

Arul Selvam Periyasamy, and Sven Behnke:

MOTPose: Multi-object 6D pose estimation for dynamic video sequences using attention-based temporal fusion

In: *IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024.

The author of this thesis substantially contributed to all aspects of the publication, including the literature survey, the conception, formalization, design, and implementation of the proposed methods, the preparation and conduct of experiments for the evaluation of the proposed approach, the analysis and interpretation of the experimental results, the preparation of the manuscript, as well as the revision and final editing of the version published.

7.1 Introduction

Object detection is the task of localizing instances of object categories in images—typically by predicting bounding box parameters. 6D pose estimation aims at predicting the position and orientation of objects in the sensor coordinate system. Both tasks are essential for many autonomous robots and a prerequisite for object manipulation.

Although single-view pose estimation models have made significant progress in recent years, they face difficulties in cluttered environments (Hodaň et al., 2020) hampered by occlusions, reflective surfaces, transparency, and other challenges. One way to address these challenges is to utilize a sequence of images of the scene instead of a single image. In a video sequence, image features and object attributes evolve smoothly over time. Models can benefit from imbuing image features and predictions from the previous frames while processing the current frame. Also, enforcing temporal consistency of the image features and pose predictions from consecutive frames can facilitate efficient learning and better accuracy. Despite the apparent advantages of temporal processing, the popularity of single-view pose estimation methods can be attributed to the complexity, computation, and memory overhead of video pose estimation methods. Furthermore, CNN-based models for video processing often utilize 3D convolutions, which need more parameters and are slow compared to their 2D counterparts.

Lately, the multi-head attention-based transformer architecture, which was initially proposed for natural language processing tasks, has shown tremendous capabilities in modeling long-term dependencies in many domains like audio, image, video, etc. (Han et al., 2022; Wen et al., 2023; Khan et al., 2022; Liu et al., 2023; Li et al., 2023). Vision transformer architectures also enable single-stage models that jointly perform object detection and pose estimation for all objects in the scene in one forward pass (Amini, Periyasamy, and Behnke, 2022; Amini, Periyasamy, and Behnke, 2021). This ability is handy when dealing with highly cluttered bin-picking scenarios (see Fig. 7.1). In this work, we propose a vision transformer model for multi-object 6D pose estimation from monocular video sequences. The core component of the proposed MOTPose method is a cross-attention-based temporal fusion mechanism that fuses features from multiple past frames while processing the current frame. We use the stacked object embeddings from the past time steps as key and value in the cross-attention computation while the object embeddings from the current time step serve as query. To counter the permutation-invariant nature of the attention mechanism in the temporal fusion modules, we utilize relative frame encoding (RFE).

7.2 Related Work

Monocular Pose Estimation

Object pose estimation from RGB images has been a long-standing problem in computer vision. In Chapters 5 and 6, we reviewed monocular pose estimation literature in detail. Here, we provide a brief overview of the literature. The traditional methods before the advent of deep learning include template-based methods (Hinterstoißer et al., 2012; Hinterstoisser et al., 2013) and feature-based methods (Rothganger et al., 2006; Pavlakos et al., 2017; Tulsiani and Malik, 2014). Modern deep-learning-based approaches include direct methods that regress the 6D pose parameters given the input RGB image (Xiang et al., 2018; Periyasamy, Schwarz, and Behnke, 2018; Wang et al., 2021; Di et al., 2021), keypoint-based methods that predict the pixel coordinates of 3D keypoints first and then use the *perspective-n-points* (P*n*P) algorithm to recover 6D pose (Rad and Lepetit, 2017; Tekin, Sinha, and Fua, 2018; Hu et al., 2019; Peng et al., 2019; Hu et al., 2020), and refinement-based methods. The latter iteratively refine an initial pose estimate using either the *render-and-compare* framework (Li et al., 2018b; Manhardt et al., 2018; Labbe et al., 2020; Periyasamy, Schwarz, and Behnke, 2019; Castro and Kim, 2023) or optical flow (Hai et al., 2023; Hu, Fua, and Salzmann, 2022). Most monocular pose estimation methods are multi-staged. In contrast, notable single-stage methods include (Capellen, Schwarz, and Behnke, 2019; Hu et al., 2019; Thalhammer et al., 2021). The proposed MOTPose method also incorporates single-stage design elements in its architecture.

Pose Estimation as Set Prediction

In recent years, vision transformer architectures, that formulate computer vision tasks like object detection, instance segmentation, and pose estimation as a set prediction problem, are achieving impressive results. Carion et al. (2020) introduced DETR, the pioneering work in this new class of methods. In Chapters 5 and 6, we presented the T6D-Direct and YOLOPose models, respectively, based on this formulation. Following these methods, the proposed MOTPose model also formulates multi-object pose estimation from video sequences as a set prediction problem.

6D Pose Tracking

Many of the early works for 6D pose tracking were based on particle filtering (Azad et al., 2011; Pauwels et al., 2013; Xiang et al., 2014), but the performance of particle filters heavily depends on the accuracy of the observation model. Deng et al. (2019) introduced PoseRBPF utilizing a CNN-based observation model in the particle filtering framework. Wen et al. (2020) introduced $se(3)$-TrackNet, which achieved state-of-the-art-results in object pose tracking from RGB-D images. In contrast to $se(3)$-TrackNet, MOTPose only needs RGB input and can estimate 6D pose for all objects in the input images in one stage.

Multi-Object Tracking

Multi-object tracking aims at tracking 2D bounding boxes of the target instances in a given video sequence. The task is often challenging, due to the presence of multiple instances of the same category. To address the problem of matching detections and tracked objects, sophisticated matching strategies were proposed (Bergmann, Meinhardt, and Leal-Taixé, 2019; Zhou, Koltun, and Krähenbühl, 2020; Zhou, Koltun, and Krähenbühl, 2020; Xu et al., 2021). In this work, we focus mainly on improving pose estimation accuracy by fusing information over multiple time steps. Thus, instead of focusing on the tracking metrics, we emphasize the standard pose estimation metrics—ADD-S and ADD(-S)—discussed in Section 7.4.2.

Tracking-by-Attention in DETR-like Models

Recently, Meinhardt et al. (2022) proposed TrackFormer, by introducing the *tracking-by-attention* framework in a DETR-like architecture. Their key idea is to use object embeddings from time step t as object queries in time step $t+1$. Propagating object embeddings over multiple time steps enables tracking the object over a long video sequence. State-of-the-art methods for multi-object tracking utilizing the tracking-by-attention framework include MOTR (Zeng et al., 2022b) and TransTrack (Sun et al., 2020). The main downside of such methods is that the number of object queries in a time step is dynamic, which makes efficient vectorized implementation harder and results in a slow training process. In contrast to the *tracking-by-attention* framework, in our model, we fuse a fixed set of object embeddings and object-specific outputs from multiple time steps using cross-attention-based modules.

7.3 MOTPose

Following YOLOPose (Periyasamy et al., 2023), we formulate multi-object pose estimation as a set prediction problem. YOLOPose exploits the permutation-invariant nature of the attention mechanism to generate a set of tuples—each consisting of class probabilities, 2D bounding box, 3D bounding box, position and orientation parameters. The 3D bounding box parameters are represented using the interpolated bounding box (IBB) keypoints (Li et al., 2021), which we discussed in Section 6.3.2 in detail. YOLOPose employs a ResNet backbone for feature extraction (CNN). Positional encoding compensates for the loss of spatial information in the permutation-invariant attention computation. Combined image features and positional encodings are provided to the encoder module, which uses the multi-head self-attention mechanism to generate encoder feature embeddings. In the decoder, the cross-attention mechanism is employed between the encoder feature embeddings and a set of N learned embeddings called object queries to generate N object embeddings, which are then processed by feed-forward prediction networks (FFPNs) to generate class probabilities, 2D bounding box, and IBB keypoints, in parallel. The IBB keypoints are then processed by a subsequent FFPN to estimate the translation and rotation parameters. Since the cardinality of the predicted set is fixed, the model is trained to predict Ø classes after detecting all the target objects present in the image. By associating the predictions and the ground truth using a bipartite matching algorithm (Kuhn, 1955), YOLOPose is trained end-to-end.

7.3.1 Model Architecture

The architecture of the proposed MOTPose model is shown in Fig. 7.2. We base the single-frame processing of MOTPose on the YOLOPose model. The transformer-based encoder-decoder modules generate object embeddings of cardinality N from CNN-computed image features that are augmented with positional encoding. FFPNs process the object embeddings to generate object-specific outputs. The object embeddings and the object-specific outputs from the past time steps provide rich temporal information that can be leveraged while processing the current frame. To this end, we fuse object embeddings and object-specific outputs from multiple past time steps using the Temporal Embedding Fusion Module (TEFM, Sec. 7.3.1) and the Temporal Object Fusion Module (TOFM, Sec. 7.3.1), respectively, before generating outputs for the current time step. To enable the fusion of embeddings and object parameters over multiple time steps using the permutation-invariant attention mechanism, we utilize *relative frame encoding* (RFE), which encodes the number of time steps relative to the current frame using 1D sinusoidal functions.

Different architectural designs exist for fusing temporal information into the neural network models. Here, we discuss these design choices in detail and introduce the design of the temporal fusion modules in MOTPose.

Early Fusion

A temporal stream of RGB images is represented using an array of dimension T×H×W×3, where T is the number of time steps, and H and W are height and width of the image, respectively. 4D convolutional networks (Choy, Gwak, and Savarese, 2019; Choy, Gwak, and Savarese, 2019; Karpathy et al., 2014; Zhang et al., 2020) and video vision transformer models (Lea et al., 2016; Arnab et al., 2021; Liu et al., 2022a; Yan et al., 2022; Herzig et al., 2022) are designed to process the 4D input arrays. This allows for fusing temporal at an early image input stage. The models can learn spatio-temporal features directly from the images. The major downside of such an approach is the larger memory and time requirement of the 4D convolution/transformer modules.

Late Fusion

On the end of the spectrum to the early fusion approach is the late fusion. The prediction made by the model at previous time step is temporally fused while processing the current time step. In general, the dimension of the output is much smaller compared to the inputs in the context of object detection and pose estimation. Thus late stage needs a much smaller memory and processing time compared to the early fusion approach. However, fusing the predictions directly leads to poor performance in practice (Zeng et al., 2022a; Coskun et al., 2017; Luo, Golestaneh, and Kitani, 2020; Véges and Lőrincz, 2020). This is mainly because the representation of the outputs generated are not suitable as inputs for subsequent processing. Thus, in the MOTPose model, we fuse intermediate object embeddings.

At each time step, object embedding are generated by the decoder module by performing cross attention between the object queries and encoder embeddings. In fact, the FFPNs that generate the object predictions receive only the object embeddings as input. Thus, object embeddings encode all the essential information contained in the images necessary for generating object predictions. Fusing object embeddings allows for efficient temporal information fusion and computationally efficient fusion mechanism.

Temporal Embedding Fusion Module (TEFM)

At each time step, the decoder generates object embeddings of shape $N \times 256$, where N is the cardinality of the object set to be predicted. TEFM, shown in Fig. 7.3, fuses object embeddings from multiple time steps to extract valuable temporal information. First, relative frame encoding is concatenated with the object embeddings, and then the resulting embeddings are projected back to 256 dimensions using linear layers. The stacked embeddings until $T-1$ time steps form *key* and *value* for the cross-attention operation in TEFM, whereas the embedding from the time step T is used as *query*. This allows the object embeddings from time step T to interact with object embeddings from all previous time steps. The key-query similarity is reflected in the resulting attention weights. These attention weights are used to weigh the *value* vectors, which in our case are the object embeddings from all previous time steps. After applying layer normalization, the output of TEFM is added element-wise to the object embeddings of time step T, representing a residual connection.

Temporal Object Fusion Module (TOFM)

In addition to fusing embeddings using TEFM, we employ two TOFM modules to fuse object-specific outputs. The design of TOFM is similar to that of TEFM, except for the usage of additional linear projection layers at the beginning and the end. The object embeddings are of shape $N \times 256$, whereas the shape of the predictions is $N \times P$, which depends on the prediction generated; three in the case of translation prediction, six in the case of rotation prediction, and 32 in the case of keypoints. We use a linear layer

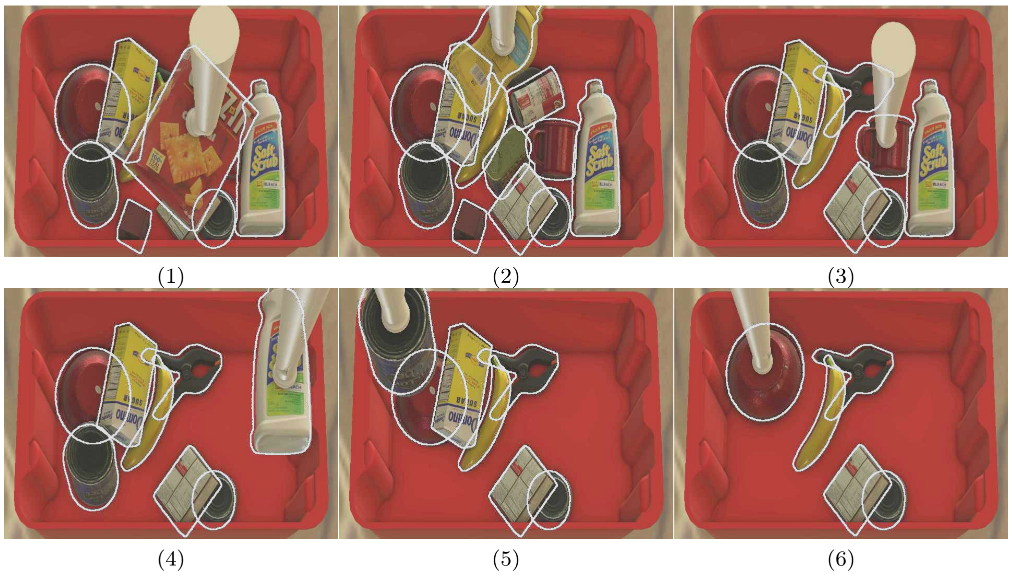


Figure 7.1: Multi-Object pose predictions for a cluttered bin-picking scene from the SynPick dataset (Untargeted-pick split, 38th video sequence). Our model jointly detects and estimates 6D pose for all objects in the scene in a single step using a vision-transformer model by fusing temporal information across multiple frames. The predicted objects' poses are visualized using contours.

to project the predictions to a 256-dimensional vector and supplement them with RFEs. After computing cross-attention, we project the resulting embeddings back to $N \times P$. TOFM_1 is used for fusing keypoints and TOFM_2 is used for fusing pose parameters.

7.3.2 Matching

We use the bipartite matching algorithm (Carion et al., 2020; Periyasamy et al., 2023; Kuhn, 1955) to associate predicted and ground-truth objects. At each time step, our model generates a set of object predictions $\hat{Y}$ of cardinality N. We perform bipartite matching between the predicted set and the $\emptyset$ object padded ground-truth set Y to find the permutation of elements between the two sets $\sigma \in \mathfrak{S}_N$ that minimizes the cost term:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(Y_i, \hat{Y}_{\sigma(i)}), \tag{7.1}$$

where $\mathcal{L}_{match}(Y_i, \hat{Y}_{\sigma(i)})$ is the pair-wise matching cost between the ground-truth tuple Y_i and the prediction at $\hat{Y}$ index $\sigma(i)$.

Despite jointly estimating 2D bounding box, class probabilities, key points, and pose parameters, similar to previous chapters Chapters 5 and 6, we use only the bounding box and the class probability components in the matching cost function.

$$\mathcal{L}_{match\_object}(Y_i, \hat{Y}_{\sigma(i)}) = -\mathbb{1}_{c_i \neq \emptyset} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{c_i \neq \emptyset} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}). \tag{7.2}$$

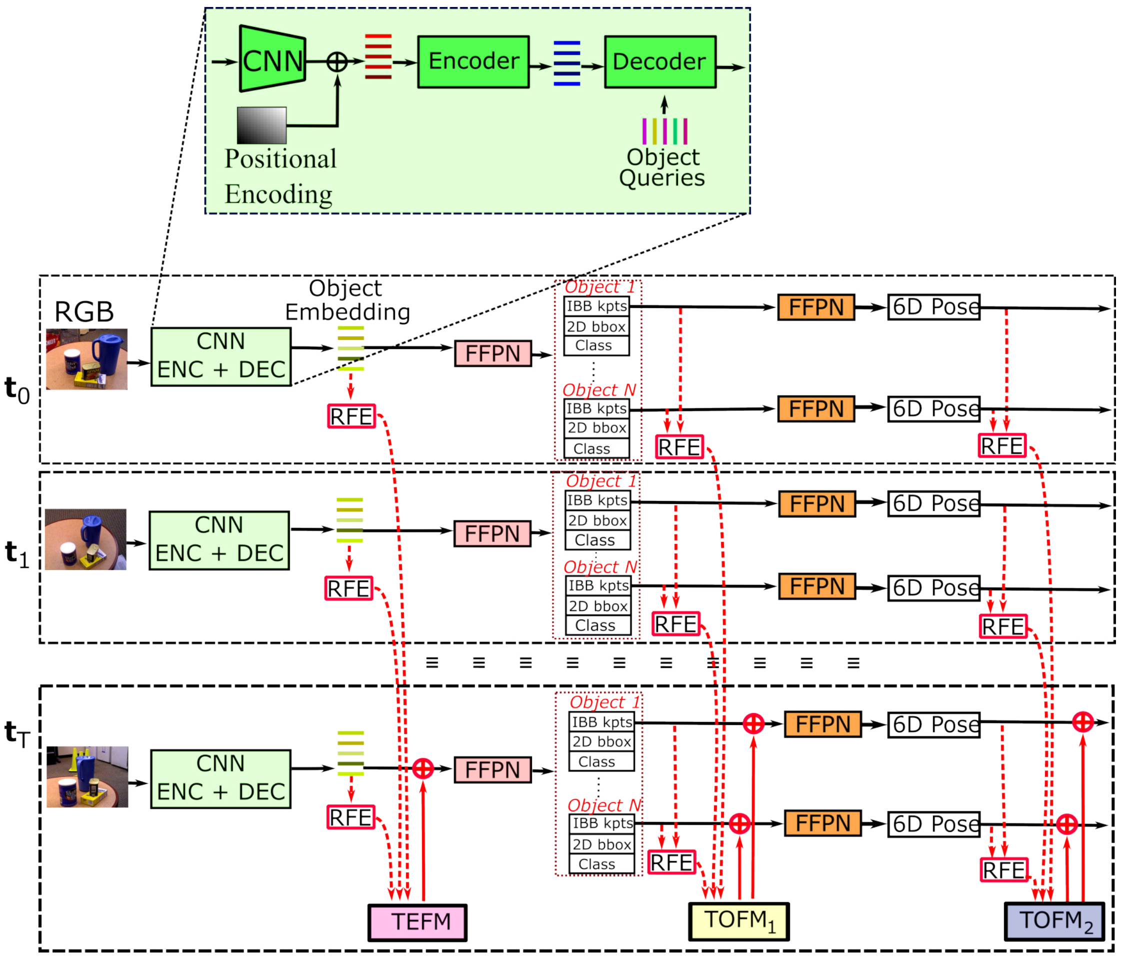


Figure 7.2: MOTPose architecture. Positional Encoding: pixel coordinates represented using sine and cosine functions of different frequencies. Object Queries: learned embeddings that are trained jointly with the model and remain fixed during inference (7.3). FFPN: Feed Forward Prediction Network. TEFM: Temporal Embedding Fusion Module (Section 7.3.1, Fig. 7.3). TOFM: Temporal Object Fusion Module (Section 7.3.1). ⊕: Element-wise addition. ⊕: Residual connection. The dashed red lines represent temporal connections. All modules that share a color also share weights. At each time step, object embeddings are generated using a CNN backbone and transformer-based encoder-decoder modules. The image features from the backbone are augmented with positional encoding. The object embeddings are processed in parallel using FFPNs to generate class probability, bounding box, and 6D pose parameters. At time step t_T, the object embeddings of different time steps are fused using TEFM. Similarly, object-specific predictions like the keypoints and the 6D pose parameters of different time steps are fused using TOFM. While fusing object embeddings and object-specific outputs from different time steps, Relative Frame Encoding (RFE) is added element-wise to uniquely identify the respective time step.

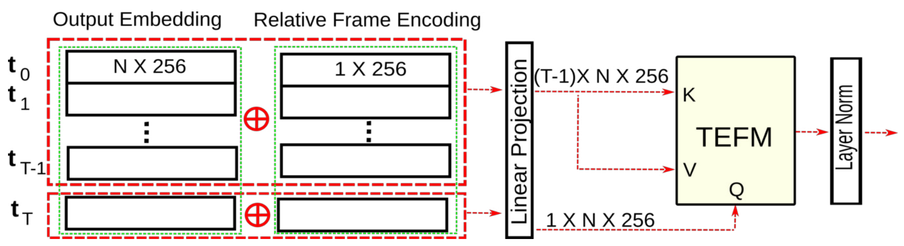


Figure 7.3: Temporal Embedding Fusion Module (TEFM). ⊕: Concatenation operation. The object embeddings at each time step of shape N×256 are added element-wise with relative frame encoding (RFE). The resulting vectors for time steps $t_0 - t_{T-1}$ are stacked to form the *key* as well as the *value* for the cross-attention operation in TEFM, whereas the embedding at time step T acts as the *query*.

This design choice is based on the empirical observation that a combination of the bounding box and the class probability components alone is enough to ensure an optimal match between the ground-truth and the predicted sets.

7.3.3 Loss Function

The *Hungarian loss* used to train MOTPose is a weighted combination of five components:

Class Probability Loss

We use the standard negative log-likelihood (NLL) loss to train the classification branch of the model.

$$\mathrm{NLL}(x) = -\log(p(x)), \tag{7.3}$$

where x is the target object. To deal with the class imbalance due to the ∅ class appearing disproportionately often, we weigh it down by a factor of 0.1.

Bounding Box Loss

$$\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \alpha\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \beta||b_i - \hat{b}_{\sigma(i)}||, \tag{7.4}$$

$$\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) = 1 - \left(\frac{|b_i \cap \hat{b}_{\sigma(i)}|}{|b_i \cup \hat{b}_{\sigma(i)}|} - \frac{|B(b_i, \hat{b}_{\sigma(i)}) \setminus b_i \cup \hat{b}_{\sigma(i)}|}{|B(b_i, \hat{b}_{\sigma(i)})|}\right), \tag{7.5}$$

where $B(b_i, \hat{b}_{\sigma(i)})$ defines the largest box enclosing both the ground truth b_i and the prediction $\hat{b}_{\sigma(i)}$, and α, β are hyperparameters. To train the bound box prediction branch of our model, we employ a linear combination of the generalized IoU (Rezatofighi et al., 2019) and the ℓ_1-loss.

Keypoint Loss

We use a weighted combination of the ℓ_1-loss and the cross-ratio consistency loss (Li et al., 2021; Periyasamy et al., 2023) to train the keypoint estimation branch. The cross-ratio of any four collinear points remains constant under perspective projection, which is enforced using the cross-ratio consistency loss. Formally, given four collinear points (A, B, C, D), the cross-ratio CR is defined as:

$$\mathrm{CR} = \frac{||C - A||\;||D - B||}{||C - B||\;||D - A||}. \tag{7.6}$$

We enforce the cross-ratio consistency with their projected 2D coordinates (a, b, c, d) using the loss component $\mathcal{L}_{\mathcal{CR}}$:

$$\mathcal{L}_{\mathcal{CR}} = \left(\mathrm{CR}^2 - \frac{||c-a||^2||d-b||^2}{||c-b||^2||d-a||^2}\right). \tag{7.7}$$

Pose Loss

We decouple the pose loss into a translation and a rotation component. For translation, we employ the ℓ_2-loss. For rotation, we use the symmetry-aware ShapeMatch-loss proposed by Xiang et al. (2018).

$$\mathcal{L}_{pose}(R, y, \hat{R}, \hat{y}) = \mathcal{L}_{Rot}(R, \hat{R}) + ||y - \hat{y}||, \tag{7.8}$$

where,

$$\mathcal{L}_{Rot} = \begin{cases} \frac{1}{|\mathcal{M}|} \sum\limits_{\mathrm{x}_1 \in \mathcal{M}} \min\limits_{\mathrm{x}_2 \in \mathcal{M}} ||(R\mathrm{x}_1 - \hat{R}\mathrm{x}_2)|| \text{ if symmetric,} \\ \frac{1}{|\mathcal{M}|} \sum\limits_{\mathrm{x} \in \mathcal{M}} ||(R\mathrm{x} - \hat{R}\mathrm{x})|| \text{ otherwise.} \end{cases} \tag{7.9}$$

Temporal Consistency Loss

We enforce temporal consistency using the ℓ_2-loss between the object embeddings of consecutive time steps. Embeddings evolve smoothly over frames and any big changes are undesirable. Thus, the ℓ_2-loss, which penalizes bigger differences significantly more than smaller differences, is a natural choice.

7.4 Evaluation

7.4.1 Datasets

We evaluate MOTPose on two datasets: YCB-Video and SynPick. YCB-Video features static scenes captured using moving camera. SynPick, which we introduced in Chapter 2, features a dynamic bin-picking scenes.

YCB-Video

We use the challenging YCB-Video dataset (Xiang et al., 2018) to benchmark the performance of our model against other state-of-the-art methods. The dataset consists of 92 (80 training and 12 testing) moving-camera video sequences of static scenes with multiple objects. High-resolution 3D models of all 21 objects are provided with the dataset. Following Deng et al. (2019) and Li et al. (2018b), we use all the frames in the test split for evaluation. Additionally, we utilize the synthetic dataset provided by Xiang et al. (Xiang et al., 2018) to train our model.

SynPick

SynPick (Periyasamy, Schwarz, and Behnke, 2021) is a physically-realistic synthetic dataset of dynamic bin-picking scenes that contain a chaotic pile of the same 21 YCB-Video objects in a tote. It consists of simulations of three different bin-picking actions: move, targeted pick, and untargeted pick. For each action, SynPick provides 300 video sequences: 240 for training and 60 for testing. Moreover, the dataset generator is publicly available[1] making it easy to generate additional data, if needed. In contrast to the commonly used object pose estimation datasets (Brachmann, 2020; Hinterstoisser et al., 2013; Hodaň et al., 2020), which consist of static tabletop scenes with a relatively low degree of occlusion, SynPick is highly cluttered and the gripper movements generate complex object interactions. Moreover, the objects in the SynPick dataset appear in a wide range of pose configurations and multiple instances of the same object are present in the scenes. Thus, SynPick is an ideal dataset for evaluating the proposed MOTPose model.

7.4.2 Evaluation Metrics

We report the area under the curve (AUC) of the ADD and ADD-S metrics at an accuracy threshold of 0.1m for non-symmetric and symmetric objects, respectively (Xiang et al., 2018). The ADD metric is the average ℓ_2 distance between the subsampled mesh points $\mathcal{M}$ in the ground-truth and the predicted pose, whereas the symmetry-aware ADD-S metric is the average distance between the closest subsampled mesh points in the ground-truth and the predicted pose.

$$\text{ADD} = \frac{1}{|\mathcal{M}|} \sum_{x \in \mathcal{M}} \|(Rx + y) - (\hat{R}x + \hat{y})\|, \tag{7.10}$$

$$\text{ADD-S} = \frac{1}{|\mathcal{M}|} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \|(Rx_1 + y) - (\hat{R}x_2 + \hat{y})\|, \tag{7.11}$$

where R and y are orientation and translation components respectively.

The ADD-(S) metric combines both ADD and ADD-S into one metric by utilizing ADD for objects without symmetry and ADD-S for objects exhibiting symmetry.

7.4.3 Implementation Details

Following (Carion et al., 2020; Periyasamy et al., 2023), we choose the cardinality of the predicted set N proportional to the maximum number of objects in an image in the respective datasets: 30 for SynPick and 20 for YCB-Video. In Section 7.3.3, the bounding box components are weighted using factors 2 and 5, and the keypoint components

1 https://github.com/AIS-Bonn/synpick

Table 7.1: Quantitative results on the SynPick dataset.

	MOTPose without Temporal Fusion				MOTPose with Temporal Fusion			
Objects	AUC of ADD-S	AUC of ADD-(S)	AUC of ADD-S @0.1d	AUC of ADD-(S) @0.1d	AUC of ADD-S	AUC of ADD-(S)	AUC of ADD-S @0.1d	AUC of ADD-(S) @0.1d
master_chef_can	88.8	72.2	86.1	53.4	88.5	79.1	86.8	61.2
cracker_box	90.7	82.5	89.5	76.2	91.4	84.2	90.2	78.4
sugar_box	80.8	74.4	79.1	63.9	81.6	76.2	80.2	69.5
tomato_soup_can	72.5	64.1	70.1	43.9	73.5	68.0	71.2	45.1
mustard_bottle	80.3	72.2	78.8	62.3	80.2	74.8	78.9	67.9
tuna_fish_can	81.1	64.1	68.1	19.1	81.8	75.1	72.2	25.6
pudding_box	69.9	63.4	66.3	48.2	70.9	65.7	68.4	48.3
gelatin_box	65.8	60.3	60.6	40.9	67.4	62.1	63.3	32.0
potted_meat_can	84.3	76.1	80.6	56.4	85.1	78.9	82.5	56.5
banana	78.0	70.5	73.9	56.9	80.3	73.9	77.9	64.9
pitcher_base	92.8	84.7	92.2	79.4	93.1	85.8	92.4	81.8
bleach_cleanser	85.7	76.9	85.2	71.1	87.0	80.7	86.4	76.9
bowl*	89.0	89.0	83.2	83.2	89.5	89.5	85.9	85.9
mug	84.9	74.8	80.8	49.0	85.9	78.5	82.5	45.6
power_drill	90.5	83.7	89.9	75.2	92.9	87.2	92.3	83.0
wood_block*	90.0	90.0	88.8	88.8	90.0	90.0	88.9	88.9
scissors	72.0	65.0	65.1	49.7	75.9	69.5	71.1	55.2
large_marker	68.1	62.4	61.7	36.6	66.9	61.9	60.4	36.2
large_clamp*	76.0	76.0	73.6	73.6	79.0	79.0	77.5	77.5
extra_large_clamp*	80.5	80.5	75.7	75.7	83.6	83.6	81.8	81.8
foam_brick*	75.9	75.9	72.2	72.2	76.3	76.3	69.7	69.7
Mean	80.8	74.2	77.2	60.8	**82.0**	**77.1**	**79.1**	**63.4**

* Symmetric objects.

are weighted with factors 10 and 1. The pose component and the temporal consistency component are weighted down using factors 0.05 and 0.1, respectively. The encoder and decoder modules consist of six layers each. All the embeddings used in our model are of dimension 256. We train our model for 150 epochs using the AdamW optimizer with a learning rate of 1×10^{-4} and early stopping. We set the number of time steps T to eight in the temporal fusion modules and use a batch size of 32 (four groups of eight consecutive images).

7.4.4 Results on SynPick

Formulating multi-object pose estimation as a set prediction problem enables joint object detection and pose estimation of all objects in the scene. However, it compounds the size of the dataset required to train transformer models. Thus, to complement the existing 240 videos for training, we generate additional 300 video sequences for each action split. We call this extended version SynPick-Ext. We downsample the image resolution to 640×480. SynPick consists of objects piled up in a tote and in many cases, objects are completely occluded. To exclude heavily occluded objects, we use a minimum visibility threshold of 30% in our evaluation. In Fig. 7.4, we present pose estimates generated by our

Table 7.2: Cardinality Error on SynPick Splits [$\times 10^{-2}$].

Method	Move	Targeted pick	Untargeted pick	All
W/o temporal fusion	3.26	1.64	0.48	2.06
With temporal fusion	**0.62**	**0.52**	**0.44**	**0.53**

Table 7.3: False negative detections on SynPick Splits [$\times 10^{-2}$].

Method	Move	Targeted pick	Untargeted pick	All
W/o temporal fusion	2.79	1.39	1.36	1.79
With temporal fusion	**0.57**	**0.44**	**0.44**	**0.48**

model with and without temporal fusion. Both models generate predictions of admissible quality. However, the model without temporal fusion suffers from failed object detections (Fig. 7.4(a), (d)), and isolated highly erroneous pose predictions (Fig. 7.4(b), (c), (e)). Temporal fusion helps in alleviating these shortcomings.

In Table 7.1, we report quantitative results of our model. MOTPose achieves impressive AUC of ADD-S and AUC of ADD-(S) scores of 82.0 and 77.1, respectively, which is an improvement of 1.2 and 2.9 compared to the model without temporal fusion. Additionally, we also report the AUC metrics with a threshold of 10% of the object diameter (AUC@0.1d). This metric takes the object size into account better. In terms of AUC of ADD-S and ADD-(S)@0.1d, temporal fusion boosts the accuracy by 1.9 and 2.6 points, respectively.

Object Detection Accuracy

Furthermore, to understand the impact of temporal fusion on object detection, we analyze the cardinality error and the bounding box accuracy metrics. The cardinality error is the difference between elements in the ground-truth and predicted sets. Formally, given the ground-truth set $\mathcal{Y}$ and the predicted set $\hat{\mathcal{Y}}$, the cardinality error (CE) is defined as:

$$\mathrm{CE} = \frac{|(\mathcal{Y} - \hat{\mathcal{Y}}) \cup (\hat{\mathcal{Y}} - \mathcal{Y})|}{|\mathcal{Y}|}. \tag{7.12}$$

In Table 7.2, we report the cardinality error of our model on different splits of the SynPick dataset. Over the complete test set, the cardinality error of the model without temporal fusion is 0.021, whereas it is only 0.005 for the model with temporal fusion. The difference is more evident in the *Move* split, which is more challenging than the other two splits.

Although CE reflects the set prediction ability of a model, in real-world bin-picking systems, the identity of the objects present in the bin might be known a priori (Schwarz et al., 2018a; Schwarz et al., 2017). Thus, in this *informed detection* scenario, false positives can be easily mitigated, whereas false negatives (FN), i.e., $|(\mathcal{Y} - \hat{\mathcal{Y}})|/|\mathcal{Y}|$ are detrimental. In Table 7.3, we report the false negatives of object detection. Over the entire test set,

Table 7.4: Bounding Box Prediction Accuracy.

Method	AP†	AP@ [IoU=0.50]	AP@ [IoU=0.75]	AR†
W/o temporal fusion	0.756	0.872	0.853	0.789
With temporal fusion	**0.779**	**0.876**	**0.858**	**0.811**

† @[IoU=0.50:0.95]

the model without temporal fusion has a FN rate of 0.018; with temporal fusion, the FN rate drops to 0.005.

Bounding Box Detection Accuracy

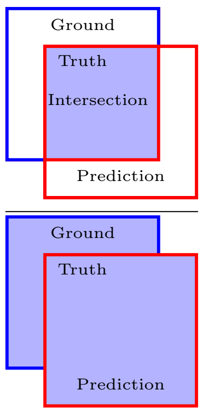
The standard metric for evaluating 2D bounding box detection accuracy of a model is the average precision (AP) and average recall (AR). AP is defined by the area under the precision-recall curve, whereas AR is corresponds to the recall-precision curve.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7.13}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{7.14}$$

The detections are classified as *true positives* or *false positives* based on the intersection over union (IoU) threshold.

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}} = \tag{7.15}$$



The COCO evaluation protocol[2] proposed AP using varying IoU thresholds from 0.5 to 0.95. In Table 7.4, we report the AP@[IoU=0.50:0.95], AP@[IoU=0.50], AP@[IoU=0.75], and AR@[IoU=0.50:0.95] metrics of the models with and without temporal fusion. The bounding box prediction accuracy of our model is comparable to the state-of-the-art object detection models (Hodan et al., 2024; Sundermeyer et al., 2023). Across all the reported metrics, temporal fusion yields consistent improvements.

2 https://cocodataset.org/#detection-eval

Table 7.5: Results on the YCB-Video dataset.

Method	AUC of ADD-S	AUC of ADD-(S)	*fps*
CRT-6D (Castro and Kim, 2023)	-	**87.5**	30
Periyasamy, Tsaturyan, and Behnke (2023)	**92.0**	84.7	26
DeepIM-Tracking (Li et al., 2018b)	91.0	85.9	13
MOTPose w/o temporal fusion	90.3	83.2	59
MOTPose with temporal fusion	91.2	84.5	30

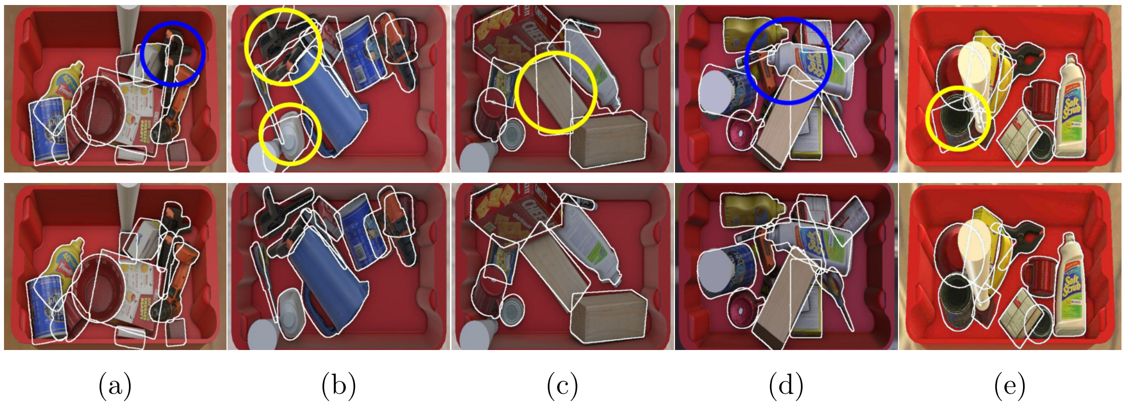


Figure 7.4: Qualitative results on SynPick. 6D pose predictions are visualized using object contours. Top: Predictions from the model without temporal fusion. Bottom: Predictions from the model with temporal fusion. Temporal fusion facilitates better pose prediction as well as object detection accuracies.

7.4.5 Results on YCB-Video

In Table 7.5, we report the quantitative comparison of our MOTPose model against state-of-the-art methods on the YCB-Video dataset. In our experiments, we fuse seven previous frames (T=8) in MOTPose. Since our model does not produce outputs for the initial $T-1$ frames in a video sequence, we report the accuracy scores excluding the initial frames. Temporal fusion enables considerable improvement in the MOTPose model: 0.9 and 1.3 accuracy points in terms of the AUC of ADD-S and AUC of ADD-(S) metric, respectively. Compared to DeepIM-Tracking (Li et al., 2018b), our method achieves a comparable AUC of ADD-Sscore and a slightly worse AUC of ADD-(S) score. DeepIM-Tracking formulates 6D pose tracking as pose refinement, i.e., pose prediction from the previous frame is used to initialize the render-and-compare pose refinement for the current step. To initialize the first frame, the authors used the ground-truth pose. While Castro and Kim (2023) achieve a significantly better accuracy than MOTPose, they perform only pose refinement. In contrast, our method performs multi-object detection and pose estimation jointly. Moreover, MOTPose accuracy is comparable to the state-of-the-art multi-object pose estimation method of Periyasamy, Tsaturyan, and Behnke (2023) in terms of the AUC of ADD-(S) metric and only slightly worse in terms of the AUC of ADD-Smetric. Note that the frame rates reported in Table 7.5 are observed on GPUs of different generations and the values are provided only for a relative comparison.

Table 7.6: Ablation study results on the SynPick dataset.

Method	AUC of ADD-S	AUC of ADD-(S)
MOTPose	**82.0**	**77.1**
MOTPose without temporal fusion	80.8	74.2
MOTPose without TEFM	81.1	74.9
MOTPose without TOFM	81.4	75.3
MOTPose without SynPick-Ext	76.4	69.2
MOTPose [T=4]	80.9	76.4
MOTPose [T=8]	82.0	**77.1**
MOTPose [T=12]	**82.2**	76.7

7.4.6 Ablation Study

To understand the contribution of the individual components to the overall performance of MOTPose, we investigated removing different components of the model and varying the number of time steps used in the fusion modules. In Table 7.6, we report the results of the ablation experiment on SynPick. Removing the TEFM module resulted in a big drop in the overall accuracy of the MOTPose model. In terms of the AUC of ADD-(S) metric, the MOTPose model without the TEFM module achieves a score of 74.9, compared to 77.1, while the AUC of ADD-S metric score drops by 0.6. Similarly, removing the TOFM module results in a drop of 0.9 AUC of ADD-(S) and 1.8 AUC of ADD-S accuracy scores. Moreover, in terms of the number of time steps used in the fusion modules, eight time steps resulted in the best performance overall.

7.5 Limitations

Similar to the methods presented in Chapters 5 and 6, our formulation of multi-object pose estimation as a set prediction task limits the datasets available for training our model. Compared to 2D annotations, 6D pose annotations are significantly harder to obtain. Thus, many of the standard datasets for evaluating object pose estimation like Linemod-Occluded (Brachmann, 2020) and Linemod (Hinterstoisser et al., 2013) provide pose annotations only for a partial number of objects per scene in the training dataset. While this is not a limitation for multi-stage methods that process the cropped version of the images for estimating the pose of target objects, our method needs 6D pose annotation for all objects in the scene, which can be prohibitively expensive to acquire in some scenarios.

7.6 Discussion & Conclusion

In this Chapter, we presented MOTPose, a multi-object pose estimation model for RGB video sequences. We process individual frames using the YOLOPose model discussed in Chapter 6. For each frame, attention-based encoder-decode module produces object embeddings of cardinality N, which are processed independently using the feed-forward MLPs to generate object-specific outputs. Employing the cross-attention-based TEFM and TOFM modules, we fuse object embeddings and object-specific outputs over multiple time steps, respectively. We evaluate our model on the SynPick dataset presented in Chapter 2, which features dynamic bin-picking scenarios, and the YCB-Video dataset (Xiang et al., 2018), which features static scenes captured using a moving camera. Aided by the temporal information, our model performs significantly better than the single-frame RGB model while being lighter and significantly faster than other pose tracking methods. In addition to the improved pose estimation accuracy, temporal fusion enables fewer object detection failures and better bounding box prediction accuracy. As discussed in Section 6.6, the keypoint regression mechanism can be easily adapted for category-level pose estimation. Similarly, the temporal fusion modules introduced in this Chapter can also be adapted for fusing category-level temporal information. Therefore, in the future, MOT-Pose can be effortlessly adapted for category-level pose tracking. Moreover, bridging the synthetic-to-real domain gap is a necessary future research direction to make large-scale real-world deployment of the temporal pose estimation pipeline discussed in this Chapter.

8 Conclusion

In this thesis, we presented methods for multi-object pose estimation using single-view RGB input as well as RGB video sequences, refining pose and shape parameters using abstract render and compare, and introduced an automated ground-truth generation scheme to train implicit pose distribution for modeling object symmetries without explicit symmetry labels. To facilitate research in dynamic scene understanding, we developed SynPick, a photo- and physically-realistic dataset featuring commonly occurring bin-picking scenarios.

The pose estimation models formulate multi-object pose estimation as a set prediction task. Given an input image, our models jointly detect objects and estimate their 6D pose parameters. In contrast to the standard multi-stage methods that decouple object detection and object pose estimation, our model detects and estimates pose for all objects in the scene in a single forward pass. The architecture consists of a CNN backbone to extract image features, an encoder-decoder module based on multi-head attention mechanism and feed-forward MLPs that estimate the class probabilities, 2D bounding boxes and 6D pose parameters. The encoder modules perform self-attention between image features supplemented with positional encoding, which are used to uniquely identify the spatial position of the pixels. On the decoder side, we perform cross-attention between encoder embeddings and learned object queries. Object queries are special embeddings that are randomly initialized during training. They are trained along with the rest of the model parameters but remain fixed during inference. We investigated the role of object queries and concluded that they highly correlate with image spatial locations. Our models generate a set of predictions with a fixed cardinality. Thus, after predicting all the objects in the given input, our models are trained to predict $\varnothing$ class, which is similar to the background class in the semantic segmentation task. Employing bipartite matching, our models are trained end-to-end without the need for any specialized layers like NMS, RoI, or anchor boxes. We evaluated our models on the challenging YCB-Video dataset. Overall, our model based on intermediate keypoint regression performed better than direct regression. Since the predicted keypoints can be easily visualized, the keypoint-based methods also improve the interpretability of the models compared to direct regression. Moreover, analyzing the attention masks in the encoder and decoder modules showed that the attention mechanism learns to focus on the pixels relevant for object prediction automatically without explicit training. To facilitate an end-to-end differentiable pipeline for keypoint-based pose estimation, we introduced the learnable RotEst module as an alternative to the non-differentiable analytical PnP module. Our experiments demonstrated the robustness of the RotEst module against noisy keypoint predictions.

Although the single-view RGB pipelines discussed in Chapters 5 and 6 achieved impressive pose estimation accuracy, they do not benefit from the rich temporal information contained in the RGB video sequences. We address this limitation by introducing temporal fusion modules in the pose estimation model. The temporal embeddings fusion module and the temporal object fusion module fuse object embeddings and object predictions, respectively, over multiple past timesteps while processing the current timestep.

We evaluated the temporal model on two datasets: YCB-Video and SynPick. YCB-Video features static scenes with a moving camera, whereas SynPick features dynamic scenes. In our experiments, the temporal fusion resulted in improved pose estimation as well as improved object detection accuracy.

The SynPick generator is built by extending the publicly available Stillleben renderer, which is designed for easy GPU access and NVIDIA PhysX physics engine integration for physically-realistic simulations. We make the SynPick dataset and the generator code public. Robustness is a highly desirable property of any perception system. To improve the robustness of the pose estimation models, we introduce pose refinement models based on abstract render-and-compare. We render the scene according to the current parameter estimate and compare the rendered image with the observed image. The pose and shape parameters are iteratively improved to minimize the rendered and the observed image comparison error. The iterative nature of this optimization scheme necessitates an efficient differentiable renderer. We develop StilllebenDR for this purpose by extending the Stillleben library. Since the pixel-wise image comparison is error-prone in the RGB space, we propose to use the learned feature space.

The standard pose estimation models fall under the category of single pose predictors. I.e., these models predict a single pose that best fits the observation. However, for symmetric objects, an observation can correspond to multiple poses, which are formally known as the *set of proper symmetries*. Predicting a single pose for such objects does not reveal any information about the nature of symmetry. In the literature, several parametric distributions like Bingham distribution, von Mises-Fisher distribution, etc., have been proposed to model object pose symmetries. However, in the context of machine learning, these parametric distributions are hard to train and inference using these models is time-consuming. To this end, we investigate implicit pose distribution networks for modeling object symmetries. In particular, we propose an automatic pose labeling scheme to train implicit pose distribution networks. Unlike the parametric distributions for modeling symmetries, our approach does not need explicit symmetry labels and offers time-efficient inference mechanism based multi-resolutional equivolumetric sampling of the $\mathbf{SO}(3)$ manifold to recover the complete set of proper symmetries.

Outlook and Future Work

The methods we introduce in this thesis open up a number of future research direction. In this section, we outline some of them. Additionally, we discuss the limitations of the methods we presented and suggest potential improvements.

Formulating multi-object pose estimation as set prediction enables single-stage architectures we discussed in Chapters 5, 6 and 7. Since we fix the cardinality of the prediction set, the models are trained to estimate $\varnothing$ class after predicting all the objects in the given input. This design choice necessitates annotation for all the objects in an image in the training dataset. Many of the standard datasets for pose estimation do not feature a complete set of annotations (Brachmann, 2020; Hinterstoisser et al., 2013). In many real-world scenarios, obtaining complete annotations can be prohibitively expensive. Investigating mechanisms to train multi-object pose estimation using partial annotations will make our models appropriate for a wide range of real-world settings.

For a large-scale deployment covering a wide range of objects, instance-level pose estimation is not scalable. However, real-world objects can be organized into categories and performing pose estimation at the category-level offers a scalable alternative to instance-level pose estimation. Lately, category-level pose estimation models are gaining significant traction (Lin et al., 2022; Wang et al., 2019; Rodriguez, Huber, and Behnke, 2020). The multi-object pose estimation models introduced in this thesis can be extended for category-level pose estimation naturally; the model architecture, as well as the loss functions, do not need any modification.

A different approach to make instance-level pose estimation scalable is the zero-shot methods (Wen et al., 2024; Ausserlechner et al., 2024; Örnek et al., 2025; Shugurov et al., 2022). They aim to estimate pose of objects unseen during training by leveraging the capability of the *vision foundational models* (Dosovitskiy et al., 2021; Radford et al., 2021; Jia et al., 2021; Kirillov et al., 2023) or the *latent diffusion models* (Rombach et al., 2022; Xu et al., 2024; Chen et al., 2024) to generalize to unseen visual domains.

In terms of dynamic scene understanding, the SynPick dataset we introduced in this thesis opens up many new research directions. The most significant among them is understanding physical interactions between objects. Imbuing models with knowledge about how the scene evolves due to object interactions will improve the dynamic scene understanding systems. The physically realistic simulation made possible by the SynPick generator and the wide range of object interactions featured in the dataset enable research in understanding object interactions. The models capable of accurately simulating object interactions can be used for predicting the success rate of a pick action before executing the action in the real world. Moreover, temporal models trained using dynamic video sequences can be employed to reason about the object that are currently occluded but were visible in the past. Such models will be handy in the real-world dynamic settings with a high degree of occlusion.

In general, models trained using real-world images perform better than those trained with synthetic images. However, manual annotation of highly cluttered dynamic scenes is not feasible. Thus, addressing this *sim-to-real gap* (James et al., 2019; Chebotar et al., 2019; Golemo et al., 2018; Anderson et al., 2021) is a necessary step towards dynamic scene understanding in real-world settings.

The pose and shape refinement methods discussed in Chapter 4 demonstrate the potential of the abstract render-and-compare approach. Nevertheless, iterative refinement using analytical gradients from the differentiable renderer suffers from the complexities of rendered and observed image comparison. Li et al. (2018b) and Labbe et al. (2020) showed that learning-based render-and-compare methods produce better results than analytical render-and-compare. Moreover, Yen-Chen et al. (2021) and Bortolon et al. (2024) demonstrated that the recent neural volume rendering approaches (Mildenhall et al., 2021) can also be inverted to perform pose refinement. Both the learning-based render-and-compare and the neural volume rendering approaches do not suffer from the complexities of image comparison.

The implicit pose distribution model discussed in Chapter 3 offers an efficient alternative to parametric models and opens up possibilities for symmetry-aware manipulation planning. For example, while planning a grasp, the knowledge of object symmetry can be used to find the grasp that needs the least end effector movement.

In an orthogonal direction to the perception pipelines based on pose estimation discussed in this thesis, in the context of bin picking, some approaches circumvent pose estimation by learning generalizable grasp/manipulation planning (Yang et al., 2023; Xiao et al., 2023; Mosbach and Behnke, 2024; Vuong et al., 2023). Recent advancements in reinforcement learning and multi-modality foundational models have made such approaches feasible. However, such approaches will not make research in object pose estimation superfluous. Object pose estimation is a necessary prerequisite in use cases beyond bin-picking like virtual and augmented reality, medical imaging, autonomous navigation, manufacturing, and quality control, etc. Thus, object pose estimation will remain a crucial research problem in the future as well.

To conclude, in this thesis, we presented methods for multi-object pose estimation from single-view RGB as well as video sequences, pipelines for pose and shape parameter refinement based on abstract render-and-compare, an automatic ground-truth generation scheme for training implicit probability density functions for modeling object symmetries. We also introduced a photo- and physically- realistic dataset for dynamic scene understanding along with the generator. We evaluated our methods on the standard object pose estimating datasets as well as the newly developed SynPick dataset. Furthermore, we discussed the limitations of our approaches and the future research directions.

List of Figures

List of Tables

Acronyms

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
5D	five-dimensional
6D	six-dimensional
AAE	Augmented AutoEncoder
ADD	Average distance of the corresponding 3D model points
ADD-S	Average distance of the closest 3D model points
ADD-(S)	combination of ADD & ADD-S
AP	Average Precision
APC	Amazon Picking Challenge
ARC	Amazon Robotics Challenge
AR	Average Recall
AUC	Area Under Curve
BB	Bounding Box
BRISK	Binary Robust Invariant Scalable Keypoints

CNN	Convolutional Neural Network
CPD	Coherent Point Drift
DAE	Denoising AutoAncoder
EMA	Exponentially Moving Average
EPnP	Efficient PnP
FFN	Feed-Forward Network
FFPN	Feed-Forward Prediction Network
FOV	Field Of View
FPFH	Fast Point Feature Histogram
FPS	Farthest Point Sampling
GIoU	Generalized IoU
GPU	Graphics Processing Unit
GPGPU	General Purpose GPU
HOG	Histogram of Oriented Gradients
IBB	Interpolated BB
IBL	Image-Based Lighting
ICP	Iterative Closest Point
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ImplicitPDF	Implicit Probability Distribution Function
IoU	Intersection over Union
LLH	Log-Likelihood
LPIPS	Learned Perceptual Image Patch Similarity Metric
MAAD	Mean Absolute Angular Error
MHA	Multi-Head Attention
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
NLL	Negative Log-Likelihood
NMS	Non Maximum Suppression
ORB	Oriented FAST and Rotated BRIEF
PCA	Principal Component Analysis
PnP	Perspective n Points
RANSAC	Random Sample Consensus
RFE	Relative Frame Encoding
RGB	Red Green Blue
RGB-D	Red Green Blue & Depth
RGB-(D)	Red Green Blue & optional Depth

RoI	Region of Interest
SDF	Signed Distance Function
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
TEFM	Temporal Embedding Fusion Module
TOFM	Temporal Object Fusion Module
ViT	Vision Transformer
fps	frames per second

Bibliography

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. (2016). *TensorFlow: A system for large-scale machine learning*. In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283.

Amini, A., A. S. Periyasamy, and S. Behnke (2021). *T6D-Direct: Transformers for multi-object 6D pose direct regression*. In: *DAGM German Conference on Pattern Recognition (GCPR)*. Springer, pp. 530–544.

– (2022). *YOLOPose: Transformer-based multi-object 6D pose estimation using keypoint regression*. In: *International Conference on Intelligent Autonomous Systems (IAS)*. Springer, pp. 392–406.

Anderson, P., A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, D. Batra, and S. Lee (2021). *Sim-to-real transfer for vision-and-language navigation*. In: *Conference on Robot Learning (CoRL)*. PMLR, pp. 671–681.

Arnab, A., M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid (2021). *ViViT: A video vision transformer*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6836–6846.

Ausserlechner, P., D. Haberger, S. Thalhammer, J.-B. Weibel, and M. Vincze (2024). *ZS6D: Zero-shot 6D object pose estimation using vision transformers*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 463–469.

Azad, P., D. Münch, T. Asfour, and R. Dillmann (2011). *6-DoF model-based tracking of arbitrarily shaped 3D objects*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5204–5209.

Barrow, H., J Tenenbaum, A Hanson, and E Riseman (1978). *Recovering intrinsic scene characteristics*. In: *The Journal of Computer and System Sciences* 2.3-26, p. 2.

Bay, H., T. Tuytelaars, and L. V. Gool (2006). *SURF: Speeded up robust features*. In: *European Conference on Computer Vision (ECCV)*.

Behnke, S. (2003a). *Face localization in the neural abstraction pyramid*. In: *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES)*. Springer, pp. 139–146.

– (2003b). *Hierarchical neural networks for image interpretation*. Vol. 2766. Lecture Notes in Computer Science (LNCS), Springer.

– (2005). *Face localization and tracking in the neural abstraction pyramid*. In: *Neural Computing & Applications* 14, pp. 97–103.

Bergmann, P., T. Meinhardt, and L. Leal-Taixé (2019). *Tracking without bells and whistles*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 941–951.

Bortolon, M., T. Tsesmelis, S. James, F. Poiesi, and A. D. Bue (2024). *IFFNeRF: Initialisation free and fast 6DoF pose estimation from a single image and a NeRF model*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1985–1991.

Brachmann, E. (2020). *6D object pose estimation using 3D object coordinates [Data]*. Version V1. DOI: 10.11588/data/V4MUMX. URL: https://doi.org/10.11588/data/V4MUMX.

Brachmann, E., A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother (2014). *Learning 6D object pose estimation using 3D object coordinates*. In: *European Conference on Computer Vision (ECCV)*, pp. 536–551.

Brégier, R., F. Devernay, L. Leyrit, and J. L. Crowley (2018). *Defining the pose of any 3D rigid object and an associated distance*. In: *International Journal of Computer Vision (IJCV)* 126, pp. 571–596.

Calli, B., A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar (2015). *Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set*. In: *IEEE Robotics & Automation Magazine* 22.3, pp. 36–52.

Calonder, M., V. Lepetit, and P. Fua (2008). *Keypoint signatures for fast learning and recognition*. In: *European Conference on Computer Vision (ECCV),* Springer, pp. 58–71.

Cao, Y.-H., H. Yu, and J. Wu (2022). *Training vision transformers with only 2040 images*. In: *European Conference on Computer Vision (ECCV)*, pp. 220–237.

Cao, Z., Y. Sheikh, and N. K. Banerjee (2016). *Real-time scalable 6DOF pose estimation for textureless objects*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2441–2448.

Capellen, C., M. Schwarz, and S. Behnke (2019). *ConvPoseCNN: Dense convolutional 6D object pose estimation*. In: *International Conference on Computer Vision Theory and Applications (VISAPP)*, pp. 162–172.

Carion, N., F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko (2020). *End-to-end object detection with transformers*. In: *European Conference on Computer Vision (ECCV)*, pp. 213–229.

Castro, P. and T.-K. Kim (2023). *CRT-6D: Fast 6D object pose estimation with cascaded refinement transformers*. In: *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 5746–5755.

Chang, A. X. et al. (2015). *ShapeNet: An information-rich 3D model repository*. In: *ArXiv e-prints* arXiv:1512.03012.

Chebotar, Y., A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox (2019). *Closing the sim-to-real loop: Adapting simulation randomization with real world experience*. In: *International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979.

Chen, W., H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler (2019). *Learning to predict 3D objects with an interpolation-based differentiable renderer*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 9609–9619.

Chen, Y., Y. Di, G. Zhai, F. Manhardt, C. Zhang, R. Zhang, F. Tombari, N. Navab, and B. Busam (2024). *SecondPose: SE(3)-consistent dual-stream feature fusion for category-level pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9959–9969.

Cho, M. and K. M. Lee (2009). *Bilateral symmetry detection via symmetry-growing.* In: *British Machine Vision Conference (BMVC)*, pp. 1–11.

Choy, C. B., D. Xu, J. Gwak, K. Chen, and S. Savarese (2016). *3D-R2N2: A unified approach for single and multi-view 6D object reconstruction*. In: *European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands*, pp. 628–644.

Choy, C., J. Gwak, and S. Savarese (2019). *4D spatio-temporal convnets: Minkowski convolutional neural networks*. In: *IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3075–3084.

Chum, O. and J. Matas (2005). *Matching with PROSAC-progressive sample consensus*. In: *IEEE Computer society conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1, pp. 220–226.

Cireşan, D. C., U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber (2011). *High-performance neural networks for visual object classification*. In: *ArXiv e-prints* arXiv:1102.0183.

Cireşan, D. C., U. Meier, and J. Schmidhuber (2012). *Multi-column deep neural networks for image classification*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649.

Cohen, N. and A. Shashua (2017). *Inductive bias of deep convolutional networks through pooling geometry*. In: *International Conference on Learning Representations (ICLR)*.

Cordonnier, J.-B., A. Loukas, and M. Jaggi (2020). *On the relationship between aelf-attention and convolutional layers*. In: *International Conference on Learning Representations (ICLR)*.

Corona, E., K. Kundu, and S. Fidler (2018). *Pose estimation for objects with rotational symmetry*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7215–7222.

Correll, N., K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman (2016). *Analysis and observations from the first amazon picking challenge*. In: *Transactions on Automation Science and Engineering (T-ASE)* 15.1, pp. 172–188.

Coskun, H., F. Achilles, R. DiPietro, N. Navab, and F. Tombari (2017). *Long short-term memory kalman filters: Recurrent neural estimators for pose regularization*. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 5524–5532.

Dalal, N. and B. Triggs (2005). *Histograms of oriented gradients for human detection*. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE, pp. 886–893.

Deng, C., C. Jiang, C. R. Qi, X. Yan, Y. Zhou, L. Guibas, D. Anguelov, et al. (2023). *NeRDi: Single-view NeRF synthesis with language-guided diffusion as general image priors*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20637–20647.

Deng, H., M. Bui, N. Navab, L. Guibas, S. Ilic, and T. Birdal (2022). *Deep Bingham networks: Dealing with uncertainty and ambiguity in pose estimation*. In: *International Journal of Computer Vision (IJCV)*, pp. 1–28.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). *ImageNet: A large-scale hierarchical image database*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255.

Deng, L. and Y. Liu (2018). *Deep learning in natural language processing*. Springer.

Deng, X., A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox (2019). *PoseRBPF: A Rao–Blackwellized particle filter for 6D object pose tracking*. In: *IEEE Transactions on Robotics (TRO)* 37, pp. 1328–1342.

Deng, X., Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox (2020). *Self-supervised 6D object pose estimation for robot manipulation*. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Di, Y., F. Manhardt, G. Wang, X. Ji, N. Navab, and F. Tombari (2021). *SO-Pose: Exploiting self-occlusion for direct 6D pose estimation*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12396–12405.

Di, Y., R. Zhang, Z. Lou, F. Manhardt, X. Ji, N. Navab, and F. Tombari (2022). *GPV-Pose: Category-level object pose estimation via geometry-guided point-wise voting*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6781–6791.

Dosovitskiy, A. et al. (2021). *An image is worth 16 ×16 words: Transformers for image recognition at scale*. In: *International Conference on Learning Representations (ICLR)*.

Endres, F., J. Hess, J. Sturm, D. Cremers, and W. Burgard (2013). *3-D mapping with an RGB-D camera*. In: *IEEE Transactions on Robotics (TRO)* 30.1, pp. 177–187.

Finlayson, G. D., M. S. Drew, and C. Lu (2004). *Intrinsic images by entropy minimization*. In: *European conference on computer vision (ECCV)*. Springer, pp. 582–595.

Flynn, P. J. (1994). *3-D object recognition with symmetric models: symmetry extraction and encoding*. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 16.8, pp. 814–818.

Fu, K., J. Peng, Q. He, and H. Zhang (2021). *Single image 3D object reconstruction based on deep learning: A review*. In: *Multimedia Tools and Applications* 80.1, pp. 463–498.

Gani, H., M. Naseer, and M. Yaqub (2022). *How to train vision transformer on small-scale datasets?* In: *British Machine Vision Conference (BMVC)*. BMVA Press.

Garcia-Castellanos, D. and U. Lombardo (2007). *Poles of inaccessibility: A calculation algorithm for the remotest places on earth*. In: *Scottish Geographical Journal* 123.3, pp. 227–233.

Geiger, A., P. Lenz, and R. Urtasun (2012). *Are we ready for autonomous driving? the kitti vision benchmark suite*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361.

Gilitschenski, I., R. Sahoo, W. Schwarting, A. Amini, S. Karaman, and D. Rus (2019). *Deep orientation uncertainty learning based on Bingham loss*. In: *International Conference on Learning Representations (ICLR)*.

Golemo, F., A. A. Taiga, A. Courville, and P.-Y. Oudeyer (2018). *Sim-to-real transfer with neural-augmented robot simulation*. In: *Conference on Robot Learning (CoRL)*. PMLR, pp. 817–828.

Goodfellow, I. J., J. Shlens, and C. Szegedy (2015). *Explaining and harnessing adversarial examples*. In: *International Conference on Learning Representations (ICLR)*.

Gorski, K. M., E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann (2005). *HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere*. In: *The Astrophysical Journal* 622.2, p. 759.

Grenander, U. (1976). *Lectures in pattern theory-volume I: Pattern synthesis*. Springer-Verlag.

– (1978). *Lectures in pattern theory-volume II: Pattern analysis*. Springer-Verlag.

Gupta, S., R. Girshick, P. Arbeláez, and J. Malik (2014). *Learning rich features from RGB-D images for object detection and segmentation*. In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 345–360.

Hai, Y., R. Song, J. Li, and Y. Hu (2023). *Shape-constraint recurrent flow for 6D object pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4831–4840.

Halevy, A., P. Norvig, and F. Pereira (2009). *The unreasonable effectiveness of data*. In: *Intelligent Systems* 24.2, pp. 8–12.

Han, K., Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, et al. (2022). *A survey on vision transformer*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 45.1, pp. 87–110.

Han, X.-F., H. Laga, and M. Bennamoun (2019). *Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era*. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.5, pp. 1578–1604.

Hao, Q., R. Cai, Z. Li, L. Zhang, Y. Pang, F. Wu, and Y. Rui (2013). *Efficient 2D-to-3D correspondence filtering for scalable 3D object recognition*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 899–906.

Hara, K. and R. Chellappa (2014). *Growing regression forests by classification: Applications to object pose estimation*. In: *European Conference on Computer Visionm (ECCV)*, pp. 552–567.

Hartley, R. and A. Zisserman (2004). *Multiple view geometry in computer vision*. 2nd ed. Cambridge University Press.

He, K., X. Zhang, S. Ren, and J. Sun (2016). *Deep residual learning for image recognition*. In: *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

He, Y., H. Huang, H. Fan, Q. Chen, and J. Sun (2021). *FFB6D: A full flow bidirectional fusion network for 6D pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3003–3013.

He, Y., W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun (2020). *PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Herzig, R., E. Ben-Avraham, K. Mangalam, A. Bar, G. Chechik, A. Rohrbach, T. Darrell, and A. Globerson (2022). *Object-region video transformers*. In: *IEE/CVF conference on computer vision and pattern recognition*, pp. 3148–3159.

Hinterstoißer, S., C. Cagniart, S. Ilic, P. F. Sturm, N. Navab, P. V. Fua, and V. Lepetit (2012). *Gradient response maps for real-time detection of textureless objects*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34, pp. 876–888.

Hinterstoisser, S., V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab (2013). *Model-based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes*. In: *Asian conference on computer vision (ACCV)*. Springer, pp. 548–562.

Hinton, G., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. In: *ArXiv e-prints* arXiv:1207.0580.

Hodan, T., F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, et al. (2018). *BOP: Benchmark for 6D object pose estimation*. In: *European Conference on Computer Vision (ECCV)*, pp. 19–34.

Hodaň, T., M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas (2020). *BOP challenge 2020 on 6D object localization*. In: *European Conference on Computer Vision (ECCV)*, pp. 577–594.

Hodan, T., M. Sundermeyer, Y. Labbe, V. N. Nguyen, G. Wang, E. Brachmann, B. Drost, V. Lepetit, C. Rother, and J. Matas (2024). *BOP challenge 2023 on detection segmentation and pose estimation of seen and unseen rigid objects*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5610–5619.

Hu, Y., P. Fua, and M. Salzmann (2022). *Perspective flow aggregation for data-limited 6D object pose estimation*. In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 89–106.

Hu, Y., P. Fua, W. Wang, and M. Salzmann (2020). *Single-stage 6D object pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2930–2939.

Hu, Y., J. Hugonot, P. Fua, and M. Salzmann (2019). *Segmentation-driven 6D object pose estimation*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3385–3394.

Huang, J., Y. Zhou, and L. Guibas (2020). *ManifoldPlus: A robust and scalable watertight manifold surface generation method for triangle soups*. In: *ArXiv e-prints* arXiv:2005.11621.

James, S., P. Wohlhart, M. Kalakrishnan, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis (2019). *Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks*. In: *IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12627–12637.

Jarrett, K., K. Kavukcuoglu, M. Ranzato, and Y. LeCun (2009). *What is the best multi-stage architecture for object recognition?* In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2146–2153.

Jia, C., Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig (2021). *Scaling up visual and vision-language representation learning with noisy text supervision*. In: *International Conference on Machine Learning (ICML)*, pp. 4904–4916.

Jing, C., J. Potgieter, F. Noble, and R. Wang (2017). *A comparison and analysis of RGB-D cameras' depth performance for robotics application*. In: *International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 1–6.

Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei (2014). *Large-scale video classification with convolutional neural networks*. In: *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1725–1732.

Kaskman, R., S. Zakharov, I. Shugurov, and S. Ilic (2019). *HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects*. In: *IEEE International Conference on Computer Vision (ICCV) Workshops.*

Kato, H., Y. Ushiku, and T. Harada (2018). *Neural 3D mesh renderer*. In: *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 3907–3916.

Kehl, W., F. Manhardt, F. Tombari, S. Ilic, and N. Navab (2017). *SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1521–1529.

Kerbl, B., G. Kopanas, T. Leimkühler, and G. Drettakis (2023). *3D Gaussian splatting for real-time radiance field rendering*. In: *ACM Transactions on Graphics* 42.4, pp. 139–1.

Khan, S., M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah (2022). *Transformers in vision: A survey*. In: *ACM Computing Survey* 54.10s, pp. 1–41.

Khronos (2018). *Vulkan: A specification (version 1.0)*. `https://www.khronos.org/news/press/khronos-releases-vulkan-1-0-specification`. [Online; accessed 20-December-2024].

Kirillov, A., E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. (2023). *Segment anything*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026.

Klamt, T., D. Rodriguez, M. Schwarz, C. Lenz, D. Pavlichenko, D. Droeschel, and S. Behnke (2018). *Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1–8.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). *ImageNet classification with deep convolutional neural networks*. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 25. Curran Associates, Inc.

Krull, A., E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother (2015). *Learning analysis-by-synthesis for 6D pose estimation in RGB-D images*. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 954–962.

Kuhn, H. W. (1955). *The Hungarian method for the assignment problem*. In: *Naval Research Logistics Quarterly* 2.1-2, pp. 83–97.

Kundu, A., Y. Li, and J. M. Rehg (2018). *3D-RCNN: Instance-level 3D object reconstruction via render-and-compare*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3559–3568.

Kuznetsova, A., H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, et al. (2020). *The open images dataset V4: Unified image classification, object detection, and visual relationship detection at scale*. In: *International Journal of Computer Vision (IJCV)* 128.7, pp. 1956–1981.

Labbe, Y., J. Carpentier, M. Aubry, and J. Sivic (2020). *CosyPose: Consistent multi-view multi-object 6D pose estimation*. In: *European Conference on Computer Vision (ECCV)*, pp. 574–591.

Labonte, F., Y. Shapira, and P. Cohen (1993). *A perceptually plausible model for global symmetry detection*. In: *International Conference on Computer Vision (ICCV)*. IEEE, pp. 258–263.

Lai, K., L. Bo, X. Ren, and D. Fox (2011). *A scalable tree-based approach for joint object and pose recognition*. In: *AAAI Conference on Artificial Intelligence*. Vol. 25. 1, pp. 1474–1480.

Lassner, C. and M. Zollhofer (2021). *Pulsar: Efficient sphere-based neural rendering*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1440–1449.

LeCun, Y., Y. Bengio, and G. Hinton (2015). *Deep learning*. In: *Nature* 521.7553, pp. 436–444.

LeCun, Y., Y. Bengio, et al. (1995). *Convolutional networks for images, speech, and time series*. In: *The handbook of brain theory and neural networks*, 255–258.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). *Gradient-based learning applied to document recognition*. In: *IEEE* 86.11, pp. 2278–2324.

LeCun, Y., F. J. Huang, and L. Bottou (2004). *Learning methods for generic object recognition with invariance to pose and lighting*. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2, pp. II–104.

LeCun, Y., K. Kavukcuoglu, and C. Farabet (2010). *Convolutional networks and applications in vision*. In: *Proceedings of IEEE international symposium on circuits and systems*, pp. 253–256.

Lea, C., R. Vidal, A. Reiter, and G. D. Hager (2016). *Temporal convolutional networks: A unified approach to action segmentation*. In: *Proceeding of European Conference on Computer Vision (ECCV) Workshops, Part III 14*. Springer, pp. 47–54.

Leon, S. J., Å. Björck, and W. Gander (2013). *Gram-Schmidt orthogonalization: 100 years and more*. In: *Numerical Linear Algebra with Applications* 20.3, pp. 492–532.

Leutenegger, S., M. Chli, and R. Y. Siegwart (2011). *BRISK: Binary robust invariant scalable keypoints*. In: *International Conference on Computer Vision (ICCV)*. IEEE, pp. 2548–2555.

Li, F., H. Zhang, H. Xu, S. Liu, L. Zhang, L. M. Ni, and H. Shum (2023). *Mask DINO: Towards a unified transformer-based framework for object detection and segmentation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3041–3050.

Li, J. and N. M. Allinson (2008). *A comprehensive review of current local features for computer vision*. In: *Neurocomputing* 71.10-12, pp. 1771–1787.

Li, S., Z. Yan, H. Li, and K.-T. Cheng (2021). *Exploring intermediate representation for monocular vehicle pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1873–1883.

Li, T.-M., M. Aittala, F. Durand, and J. Lehtinen (2018a). *Differentiable monte carlo ray tracing through edge sampling*. In: *SIGGRAPH Asia 2018 Technical Papers*. ACM, p. 222.

Li, Y., G. Wang, X. Ji, Y. Xiang, and D. Fox (2018b). *DeepIM: Deep iterative matching for 6D pose estimation*. In: *European Conference on Computer Vision (ECCV)*, pp. 683–698.

Li, Z., G. Wang, and X. Ji (2019). *CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation*. In: *International Conference on Computer VisionICCV*, pp. 7678–7687.

Lin, G., A. Milan, C. Shen, and I. Reid (2017). *RefineNet: Multi-path refinement networks for high-resolution semantic segmentation*. In: *IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 1925–1934.

Lin, J., Z. Wei, Z. Li, S. Xu, K. Jia, and Y. Li (2021). *DualPoseNet: Category-level 6D object pose and size estimation using dual pose network with refined learning of pose consistency*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3560–3569.

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014). *Microsoft COCO: Common objects in context*. In: *European Conference on Computer Vision (ECCV)*, pp. 740–755.

Lin, Y., J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield (2022). *Single-stage keypoint-based category-level object pose estimation from an RGB Image*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1547–1553.

Liu, S., W. Chen, T. Li, and H. Li (2019). *Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7708–7717.

Liu, X., H. Peng, N. Zheng, Y. Yang, H. Hu, and Y. Yuan (2023). *EfficientViT: Memory efficient vision transformer with cascaded group attention*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14420–14430.

Liu, Z., Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo (2021). *Swin transformer: Hierarchical vision transformer using shifted windows*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022.

Liu, Z., J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu (2022a). *Video swin transformer*. In: *IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3202–3211.

Liu, Z., H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie (2022b). *A ConvNet for the 2020s*. In: *IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 11976–11986.

Long, J., E. Shelhamer, and T. Darrell (2015). *Fully convolutional networks for semantic segmentation*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440.

Loper, M. M. and M. J. Black (2014). *OpenDR: An approximate differentiable renderer*. In: *European Conference on Computer Vision (ECCV)*, pp. 154–169.

Loshchilov, I. and F. Hutter (2017). *Decoupled weight decay regularization*. In: *International Conference on Learning Representations (ICLR)*.

Lowe, D. (2004). *Distinctive image features from scale-invariant keypoints*. In: *International Journal of Computer Vision (IJCV)* 60, pp. 91–110.

Luck, S. J. and M. A. Ford (1998). *On the role of selective attention in visual perception*. In: *National Academy of Sciences* 95.3, pp. 825–830.

Luo, Z., S. A. Golestaneh, and K. M. Kitani (2020). *3D human motion estimation via motion compression and refinement*. In: *Asian Conference on Computer Vision (ACCV)*.

Lyons, D. W. (2003). *An elementary introduction to the Hopf fibration*. In: *Mathematics Magazine* 76.2, pp. 87–98.

Makoviychuk, V. et al. (2021). *Isaac Gym: High performance GPU based physics simulation for robot learning*. In: *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS)*.

Mamou, K., E Lengyel, and A Peters (2016). *Volumetric hierarchical approximate convex decomposition*. In: *Game engine gems* 3, pp. 141–158.

Manhardt, F., D. M. Arroyo, C. Rupprecht, B. Busam, T. Birdal, N. Navab, and F. Tombari (2019). *Explaining the ambiguity of object detection and 6D pose from visual data*. In: *IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 6841–6850.

Manhardt, F., W. Kehl, N. Navab, and F. Tombari (2018). *Deep model-based 6D pose refinement in RGB*. In: *European Conference on Computer Vision (ECCV)*, pp. 800–815.

Manhardt, F., M. Nickel, S. Meier, L. Minciullo, and N. Navab (2020). *CPS: Class-level 6D pose and shape estimation from monocular images*. In: *ArXiv e-prints* arXiv:2003.05848.

Mehrish, A., N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria (2023). *A review of deep learning techniques for speech processing*. In: *Information Fusion*, p. 101869.

Meinhardt, T., A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer (2022). *TrackFormer: Multi-object tracking with transformers*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8844–8854.

Mescheder, L., M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger (2019). *Occupancy networks: Learning 3D reconstruction in function space*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4460–4470.

Microsoft (2019). *DirectX-Specs*. `https://microsoft.github.io/DirectX-Specs/`. [Online; accessed 20-December-2024].

Mildenhall, B., P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng (2021). *NeRF: Representing scenes as neural radiance fields for view synthesis*. In: *Communications of the ACM* 65.1, pp. 99–106.

Moreno-Barea, F. J., F. Strazzera, J. M. Jerez, D. Urda, and L. Franco (2018). *Forward noise adjustment scheme for data augmentation*. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 728–734.

Moreno, P., C. K. Williams, C. Nash, and P. Kohli (2016). *Overcoming occlusion with inverse graphics*. In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 170–185.

Mosbach, M. and S. Behnke (2024). *Grasp anything: Combining teacher-augmented policy gradient learning with instance segmentation to grasp arbitrary objects*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7515–7521.

Murphy, K. A., C. Esteves, V. Jampani, S. Ramalingam, and A. Makadia (2021). *Implicit-PDF: Non-parametric representation of probability distributions on the rotation manifold*. In: *International Conference on Machine Learning (ICML)*. PMLR, pp. 7882–7893.

Myronenko, A. and X. Song (2010). *Point set registration: Coherent point drift*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 32.12, pp. 2262–2275.

Nair, V. and G. E. Hinton (2010). *Rectified linear units improve restricted Boltzmann machines*. In: *International conference on machine learning (ICML)*, pp. 807–814.

Nair, V., J. Susskind, and G. E. Hinton (2008). *Analysis-by-synthesis by learning to invert generative black boxes*. In: *International Conference Artificial Neural Networks (ICANN)*. Springer, pp. 971–981.

Oberweger, M., M. Rad, and V. Lepetit (2018). *Making deep heatmaps robust to partial occlusions for 3D object pose estimation*. In: *European Conference on Computer Vision (ECCV)*, pp. 119–134.

Örnek, E. P., Y. Labbé, B. Tekin, L. Ma, C. Keskin, C. Forster, and T. Hodan (2025). *Foundpose: Unseen object pose estimation with foundation features*. In: *European Conference on Computer Vision (ECCV)*, pp. 163–182.

Palacio, S., J. Folz, J. Hees, F. Raue, D. Borth, and A. Dengel (2018). *What do deep networks like to see?* In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3108–3117.

Paszke, A. et al. (2019). *PyTorch: An imperative style, high-performance deep learning library*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035.

Pauwels, K., L. Rubio, J. Díaz, and E. Ros (2013). *Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2347–2354.

Pavlakos, G., X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis (2017). *6-DoF object pose from semantic keypoints*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2011–2018.

Pavlichenko, D., D. Rodriguez, M. Schwarz, C. Lenz, A. S. Periyasamy, and S. Behnke (2018). *Autonomous dual-arm manipulation of familiar objects*. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 1–9.

Peñate Sanchez, A., E. Serradell, J. Andrade-Cetto, and F. Moreno-Noguer (2013). *Simultaneous pose, focal length and 2D-to-3D correspondences from noisy observations*. In: *British Machine Vision Conference (BMVC)*, pp. 82–1.

Peng, S., Y. Liu, Q. Huang, X. Zhou, and H. Bao (2019). *PVNet: Pixel-wise voting network for 6DOF pose estimation*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4561–4570.

Pérez-García, F., R. Sparks, and S. Ourselin (2021). *TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning*. In: *Computer Methods and Programs in Biomedicine* 208, p. 106236.

Periyasamy, A. S., A. Amini, V. Tsaturyan, and S. Behnke (2023). *YOLOPose V2: Understanding and improving transformer-based 6D pose estimation*. In: *Robotics and Autonomous Systems* 168, p. 104490.

Periyasamy, A. S., C. Capellen, M. Schwarz, and S. Behnke (2020). *ConvPoseCNN2: Prediction and refinement of dense 6D object pose*. In: *Imaging and Computer Graphics Theory and Applications (VISIGRAPP). Communications in Computer and Information Science (CCIS)* 1474, pp. 353,371.

Periyasamy, A. S., M. Schwarz, and S. Behnke (2018). *Robust 6D object pose estimation in cluttered scenes using semantic segmentation and pose regression networks*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain*, pp. 6660–6666.

– (2019). *Refining 6D object pose predictions using abstract render-and-compare*. In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, Canada*, pp. 739–746.

– (2021). *SynPick: A dataset for dynamic bin picking scene understanding*. In: *17th International Conference on Automation Science and Engineering (CASE)*, pp. 488–493.

Periyasamy, A. S., V. Tsaturyan, and S. Behnke (2023). *Efficient multi-object pose estimation using multi-resolution deformable attention and query aggregation*. In: *7th IEEE International Conference on Robotic Computing (IRC)*, pp. 247–254.

Pitteri, G., M. Ramamonjisoa, S. Ilic, and V. Lepetit (2019). *On object symmetries and 6D pose estimation from images*. In: *IEEE International Conference on 3D Vision (3DV)*, pp. 614–622.

Popel, M., M. Tomkova, J. Tomek, Ł. Kaiser, J. Uszkoreit, O. Bojar, and Z. Žabokrtskỳ (2020). *Transforming machine translation: A deep learning system reaches news translation quality comparable to human professionals*. In: *Nature communications* 11.1, p. 4381.

Prokudin, S., P. Gehler, and S. Nowozin (2018). *Deep directional statistics: Pose estimation with uncertainty quantification*. In: *European Conference on Computer Vision (ECCV)*, pp. 534–551.

Rad, M. and V. Lepetit (2017). *BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth*. In: *International Conference on Computer Vision (ICCV)*, pp. 3828–3836.

Radford, A., J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. (2021). *Learning transferable visual models from natural language supervision*. In: *International Conference on Machine Learning (ICML)*, pp. 8748–8763.

Raghu, M., T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy (2021). *Do vision transformers see like convolutional neural networks?* In: *Advances in Neural Information Processing Systems (NeurIPS)* 34, pp. 12116–12128.

Ravi, N., J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari (2020). *Accelerating 3D deep learning with PyTorch3D*. In: *European Conference on Computer Vision (ECCV)*.

Rennie, C., R. Shome, K. E. Bekris, and A. F. De Souza (2016). *A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place*. In: *Robotics and Automation Letters (RA-L)*, pp. 1179–1185.

Rezatofighi, H., N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese (2019). *Generalized intersection over union: A metric and a loss for bounding box regression*.

In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 658–666.

Rezende, D. J., S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess (2016). *Unsupervised learning of 3D structure from images*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4996–5004.

Robinson, M. and R. I. Contributors (2020). *Comprehensive list of 3D sensors commonly leveraged in ROS development*. `https://rosindustrial.org/3d-camera-survey`. Accessed: 2024-09-30.

Rodriguez, D., C. Cogswell, S. Koo, and S. Behnke (2018). *Transferring grasping skills to novel instances by latent space non-rigid registration*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8.

Rodriguez, D., F. Huber, and S. Behnke (2020). *Category-level 3D non-rigid registration from single-view RGB images*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10617–10624.

Rombach, R., A. Blattmann, D. Lorenz, P. Esser, and B. Ommer (2022). *High-resolution image synthesis with latent diffusion models*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695.

Ronneberger, O., P. Fischer, and T. Brox (2015). *U-Net: Convolutional networks for biomedical image segmentation*. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, pp. 234–241.

Rothganger, F., S. Lazebnik, C. Schmid, and J. Ponce (2006). *3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints*. In: *International Journal of Computer Vision (IJCV)* 66, pp. 231–259.

Rublee, E., V. Rabaud, K. Konolige, and G. Bradski (2011). *ORB: An efficient alternative to SIFT or SURF*. In: *International Conference on Computer Vision (ICCV)*. IEEE, pp. 2564–2571.

Rusu, R. B., N. Blodow, and M. Beetz (2009). *Fast point feature histograms (FPFH) for 3D registration*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3212–3217.

Sattler, T., B. Leibe, and L. Kobbelt (2011). *Fast image-based localization using direct 2D-to-3D matching*. In: *International Conference on Computer Vision (ICCV)*, pp. 667–674.

Saxena, A., J. Driemeyer, and A. Y. Ng (2009). *Learning 3D object orientation from images*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 794–800.

Schmidt, T., R. Newcombe, and D. Fox (2017). *Self-supervised visual descriptor learning for dense correspondence*. In: *IEEE Robotics and Automation Letters (RA-L)* 2.2, pp. 420–427.

Schulter, S., C. Leistner, P. Wohlhart, P. M. Roth, and H. Bischof (2013). *Alternating regression forests for object detection and pose estimation*. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 417–424.

Schulz, H. and S. Behnke (2012). *Deep learning: Layer-wise learning of feature hierarchies*. In: *KI-Künstliche Intelligenz* 26, pp. 357–363.

Schwarz, M. and S. Behnke (2020). *Stillleben: Realistic scene synthesis for deep learning in robotics*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10502–10508.

Schwarz, M., C. Lenz, G. M. García, S. Koo, A. S. Periyasamy, M. Schreiber, and S. Behnke (2018a). *Fast object learning and dual-arm coordination for cluttered stowing,* *picking, and packing*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3347–3354.

Schwarz, M., A. Milan, C. Lenz, A. Munoz, A. S. Periyasamy, M. Schreiber, S. Schüller, and S. Behnke (2017). *NimbRo picking: Versatile part handling for warehouse automation*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3032–3039.

Schwarz, M., A. Milan, A. S. Periyasamy, and S. Behnke (2018b). *RGB-D object detection and semantic segmentation for autonomous manipulation in clutter*. In: *International Journal of Robotics Research (IJRR)* 37.4-5, pp. 437–451.

Segal, M. and K. Akeley (1999). *The OpenGL graphics system: A specification (version 1.1)*. `https://registry.khronos.org/OpenGL/specs/gl/glspec11.pdf`. [Online; accessed 20-December-2024].

Serences, J. T. and S. Yantis (2006). *Selective visual attention and perceptual coherence*. In: *Trends in cognitive sciences* 10.1, pp. 38–45.

Shao, J., Y. Jiang, G. Wang, Z. Li, and X. Ji (2020). *PFRL: Pose-free reinforcement learning for 6D pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11454–11463.

Shorten, C. and T. M. Khoshgoftaar (2019). *A survey on image data augmentation for deep learning*. In: *Journal of big data* 6.1, pp. 1–48.

Shugurov, I., F. Li, B. Busam, and S. Ilic (2022). *OSOP: A multi-stage one shot object pose estimation framework*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6835–6844.

Song, H., D. Sun, S. Chun, V. Jampani, D. Han, B. Heo, W. Kim, and M. Yang (2022). *ViDT: An efficient and effective fully transformer-based object detector*. In: *International Conference on Learning Representations (ICLR)*.

Stewart, R., M. Andriluka, and A. Y. Ng (2016). *End-to-end people detection in crowded scenes*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2325–2333.

Sun, C., A. Shrivastava, S. Singh, and A. Gupta (2017). *Revisiting unreasonable effectiveness of data in deep learning era*. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 843–852.

Sun, P., Y. Jiang, R. Zhang, E. Xie, J. Cao, X. Hu, T. Kong, Z. Yuan, C. Wang, and P. Luo (2020). *Transtrack: Multiple-object tracking with transformer*. In: *ArXiv e-Prints* arXiv:2012.15460.

Sundermeyer, M., T. Hodan, Y. Labbe, G. Wang, E. Brachmann, B. Drost, C. Rother, and J. Matas (2023). *BOP challenge 2022 on detection, segmentation and pose estimation of specific rigid objects*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Sundermeyer, M., Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel (2018). *Implicit 3D orientation learning for 6D object detection from RGB images*. In: *European Conference on Computer Vision (ECCV)*, pp. 699–715.

Sundermeyer, M., Z.-C. Marton, M. Durner, and R. Triebel (2020). *Augmented autoencoders: Implicit 3D orientation learning for 6D object detection*. In: *International Journal of Computer Vision (IJCV)* 128.3, pp. 714–729.

Takahashi, R., T. Matsubara, and K. Uehara (2019). *Data augmentation using random image cropping and patching for deep CNNs*. In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.9, pp. 2917–2931.

Tappen, M. F., W. T. Freeman, and E. H. Adelson (2003). *Recovering intrinsic images from a single image*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1367–1374.

Tejani, A., D. Tang, R. Kouskouridas, and T.-K. Kim (2014). *Latent-class Hough forests for 3D object detection and pose estimation*. In: *European Conference on Computer Vision (ECCV)*, pp. 462–477.

Tekin, B., S. N. Sinha, and P. Fua (2018). *Real-time seamless single shot 6D object pose prediction*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 292–301.

Thalhammer, S., M. Leitner, T. Patten, and M. Vincze (2021). *PyraPose: Feature pyramids for fast and accurate object pose estimation under domain shift*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13909–13915.

Thrun, S. and B. Wegbreit (2005). *Shape from symmetry*. In: *International Conference on Computer Vision (ICCV)*. Vol. 2. IEEE, pp. 1824–1831.

Tremblay, J., T. To, and S. Birchfield (2018). *Falling things: A synthetic dataset for 3D object detection and pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2038–2041.

Tulsiani, S. and J. Malik (2014). *Viewpoints and keypoints*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1510–1519.

Ungerleider, S. K. and L. G (2000). *Mechanisms of visual attention in the human cortex*. In: *Annual review of neuroscience* 23.1, pp. 315–341.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017). *Attention is all you need*. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30, pp. 5998–6008.

Véges, M. and A Lőrincz (2020). *Temporal smoothing for 3D human pose estimation and localization for occluded people*. In: *International Conference Neural Information Processing (ICONIP)*. Springer, pp. 557–568.

Vetter, T, T Poggio, and H. Bülthoff (1994). *The importance of symmetry and virtual views in three-dimensional object recognition*. In: *Current Biology* 4.1, pp. 18–23.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol (2010). *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion*. In: *Journal of Machine Learning Research (JMLR)* 11, pp. 3371–3408.

Vuong, A. D., M. N. Vu, H. Le, B. Huang, B. Huynh, T. Vo, A. Kugi, and A. Nguyen (2023). *Grasp-anything: Large-scale grasp dataset from foundation models*. In: *ArXiv e-prints* arXiv:2309.09818.

Wang, G., F. Manhardt, F. Tombari, and X. Ji (2021). *GDR-Net: Geometry-guided direct regression network for monocular 6D object pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16611–16621.

Wang, H., S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas (2019). *Normalized object coordinate space for category-level 6D object pose and size estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642–2651.

Wang, H. and H. Huang (2017). *Group representation of global intrinsic symmetries*. In: *Computer Graphics Forum*. Vol. 36. 7. Wiley Online Library, pp. 51–61.

Wang, H., Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen (2020). *Axial-DeepLab: Stand-alone axial-attention for panoptic segmentation*. In: *European Conference on Computer Vision (ECCV)*, pp. 108–126.

Wang, W., J. Zhang, Y. Cao, Y. Shen, and D. Tao (2022). *Towards data-efficient detection transformers*. In: *European Conference on Computer Vision (ECCV)*, pp. 88–105.

Wen, B., C. Mitash, B. Ren, and K. E. Bekris (2020). *se(3)-TrackNet: Data-driven 6D pose tracking by calibrating image residuals in synthetic domains*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Wen, B., W. Yang, J. Kautz, and S. Birchfield (2024). *Foundationpose: Unified 6D pose estimation and tracking of novel objects*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17868–17879.

Wen, Q., T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun (2023). *Transformers in time series: A survey*. In: *International Joint Conference on Artificial Intelligence (IJCAI)*.

Williams, R. J. (1992). *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. In: *Machine learning* 8.3-4, pp. 229–256.

Wohlhart, P. and V. Lepetit (2015). *Learning descriptors for object recognition and 3D pose estimation*. In: *Conference on computer vision and pattern recognition (CVPR)*. IEEE, pp. 3109–3118.

Wong, S. C., A. Gatt, V. Stamatescu, and M. D. McDonnell (2016). *Understanding data augmentation for classification: when to warp?* In: *IEEE International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–6.

Xiang, Y., T. Schmidt, V. Narayanan, and D. Fox (2018). *PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes*. In: *Robotics: Science and Systems (RSS)*.

Xiang, Y., C. Song, R. Mottaghi, and S. Savarese (2014). *Monocular multiview object tracking with 3D aspect parts*. In: *European Conference on Computer Vision (ECCV)*, pp. 220–235.

Xiao, X., J. Liu, Z. Wang, Y. Zhou, Y. Qi, Q. Cheng, B. He, and S. Jiang (2023). *Robot learning in the era of foundation models: A survey*. In: *ArXiv e-prints* arXiv:2311.14379.

Xie, Y., T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar (2022). *Neural fields in visual computing and beyond*. In: *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library, pp. 641–676.

Xu, L., H. Qu, Y. Cai, and J. Liu (2024). *6D-Diff: A keypoint diffusion framework for 6D object pose estimation*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9676–9686.

Xu, Y., K.-Y. Lin, G. Zhang, X. Wang, and H. Li (2022). *RNNPose: Recurrent 6-DoF object pose refinement with robust correspondence field estimation and pose optimization*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14880–14890.

Xu, Y., Y. Ban, G. Delorme, C. Gan, D. Rus, and X. Alameda-Pineda (2021). *TransCenter: Transformers with dense representations for multiple-object tracking*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pp. 7820–7835.

Xu, Z.-B. and F.-L. Cao (2005). *Simultaneous LP-approximation order for neural networks*. In: *Neural Networks* 18.7, pp. 914–923.

Xu, Z. and F. Cao (2004). *The essential order of approximation for neural networks*. In: *Science in China Series F: Information Sciences* 47, pp. 97–112.

Yan, S., X. Xiong, A. Arnab, Z. Lu, M. Zhang, C. Sun, and C. Schmid (2022). *Multiview transformers for video recognition*. In: *IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3333–3343.

Yang, J., W. Tan, C. Jin, B. Liu, J. Fu, R. Song, and L. Wang (2023). *Pave the way to grasp anything: Transferring foundation models for universal pick-place robots*. In: *ArXiv e-prints* arXiv:2306.05716.

Yen-Chen, L., P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin (2021). *iNeRF: Inverting neural radiance fields for pose estimation*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1323–1330.

Zagoruyko, S. and N. Komodakis (2015). *Learning to compare image patches via convolutional neural networks*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4353–4361.

Zakharov, S., I. Shugurov, and S. Ilic (2019). *DPOD: 6D pose object detector and refiner*. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1941–1950.

Zeng, A., L. Yang, X. Ju, J. Li, J. Wang, and Q. Xu (2022a). *Smoothnet: A plug-and-play network for refining human poses in videos*. In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 625–642.

Zeng, F., B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei (2022b). *MOTR: End-to-end multiple-object tracking with transformer*. In: *European Conference on Computer Vision (ECCV)*.

Zhang, R., P. Isola, A. A. Efros, E. Shechtman, and O. Wang (2018). *The unreasonable effectiveness of deep features as a perceptual metric*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595.

Zhang, S., X. Wang, J. Wang, J. Pang, C. Lyu, W. Zhang, P. Luo, and K. Chen (2023). *Dense distinct query for end-to-end object detection*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7329–7338.

Zhang, S., S. Guo, W. Huang, M. R. Scott, and L. Wang (2020). *V4D: 4D convolutional neural networks for video-level representation learning*. In: *IEEE/CVF International Conference on Learning Representations (ICLR)*.

Zhang, Z. (2012). *Microsoft kinect sensor and its effect*. In: *MultiMedia* 19.2, pp. 4–10.

Zhou, D., Z. Yu, E. Xie, C. Xiao, A. Anandkumar, J. Feng, and J. M. Alvarez (2022). *Understanding the robustness in vision transformers*. In: *International Conference on Machine Learning (ICML)*, pp. 27378–27394.

Zhou, Q.-Y., J. Park, and V. Koltun (2016). *Fast global registration*. In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 766–782.

Zhou, X., V. Koltun, and P. Krähenbühl (2020). *Tracking objects as points*. In: *European Conference on Computer Vision (ECCV)*, pp. 474–490.

Zhou, Y., C. Barnes, J. Lu, J. Yang, and H. Li (2019). *On the continuity of rotation representations in neural networks*. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5745–5753.

Zhu, X., W. Su, L. Lu, B. Li, X. Wang, and J. Dai (2021). *Deformable DETR: Deformable transformers for end-to-end object detection*. In: *International Conference on Learning Representations (ICLR)*.

Zienkiewicz, J., A. Davison, and S. Leutenegger (2016). *Real-time height map fusion using differentiable rendering*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4280–4287.

Zollhöfer, M., P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb (2018). *State of the art on 3D reconstruction with RGB-D cameras*. In: *Computer graphics forum*. Vol. 37. 2. Wiley Online Library, pp. 625–652.

APPENDIX

A Incorporated Publications

The following publications have been removed for the online publication of the doctoral thesis and can be accessed via the DOI links.

1. Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke: Refining 6D object pose predictions using abstract render-and-compare In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, Toronto, Canada, 2019.
   DOI:10.48550/arXiv.1910.03412

2. Arul Selvam Periyasamy*, Max Schwarz*, and Sven Behnke: SynPick: A dataset for dynamic bin picking scene understanding In: *17th International Conference on Automation Science and Engineering (CASE)*, Lyon, France, 2021.
   DOI:10.48550/arXiv.2107.04852

3. Arash Amini, Arul Selvam Periyasamy, and Sven Behnke T6D-Direct: Transformers for multi-object 6D pose direct regression In: *43rd DAGM German Conference on Pattern Recognition (GCPR)*, Bonn, Germany, 2022.
   DOI:10.48550/arXiv.2109.10948

4. Arul Selvam Periyasamy, Max Schwarz, and Sven Behnke: Iterative 3D deformable registration from single-view RGB images using differentiable rendering In: *17th International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal, 2022.
   DOI:10.5220/0010817100003124

5. Arash Amini*, Arul Selvam Periyasamy*, and Sven Behnke: YOLOPose: Transformer-based multi-object 6D pose estimation using keypoint regression In: *17th International Conference on Intelligent Autonomous Systems (IAS)*, Zagreb, Croatia, 2022.
   DOI:10.48550/arXiv.2205.02536

6. Arul Selvam Periyasamy*, Luis Denninger*, and Sven Behnke: Learning implicit probability distribution functions for symmetric orientation estimation from RGB images without pose labels In: *6th IEEE International Conference on Robotic Computing (IRC)*, Naples, Italy, 2022.
   DOI:10.48550/arXiv.2211.11394

7. Arul Selvam Periyasamy, Arash Amini, Vladimir Tsaturyan, and Sven Behnke: YOLOPose V2: Understanding and improving transformer-based 6D pose estimation In: *Robotics and Autonomous Systems (RAS)*, Volume 168, pp 104490, 2023.
   DOI:10.48550/arXiv.2307.11550

8. Arul Selvam Periyasamy*, Vladimir Tsaturyan*, and Sven Behnke: Efficient multi-object pose estimation using multi-resolution deformable attention and query aggregation In: *7th IEEE International Conference on Robotic Computing (IRC)*, Laguna Hills, USA, 2023.
   DOI:10.48550/arXiv.2312.08268

9. Arul Selvam Periyasamy, and Sven Behnke: MOTPose: Multi-object 6D pose estimation for dynamic video sequences using attention-based temporal fusion In: *IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, 2024.
DOI:10.48550/arXiv.2403.09309