EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Mathematisch-
Naturwissenschaftliche
Fakultät**

Lernbasierte Computer Vision

Masterarbeit

Multi-View RGB-D Fusion for 6D Pose Estimation

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Sebastian Koch, `se.koch@student.uni-tuebingen.de`

Bearbeitungszeitraum: 01.10.2021 – 31.03.2022

Betreuer:	Fabian Duffhauß, Bosch Center for Aritifical Intelligence
Gutachter:	Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter:	Prof. Dr. Gerhard Neumann, Karlsruhe Institute of Technology

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Sebastian Koch (Matrikelnummer 5559404), August 29, 2022

Abstract

Environment understanding is an essential task for a robot that is to interact with its environment. Robot grasping and object manipulation, for example, require that the robot exactly knows where objects are placed and how they are orientated in the 3D world. This is the problem that 6D Pose Estimation aims to solve. Utilizing different sensor modalities commonly used in robotics like images, depth maps and/or point clouds combined with modern deep learning methods, makes it possible to accurately estimate the 6D poses of different real-world objects. However, challenges like occlusions, symmetries and noisy sensor data make 6D Pose Estimation difficult. Nevertheless, current state-of-the-art methods already achieve almost perfect predictions required for robot grasping in simple scenarios. Most current approaches limited themselves to using only a single viewpoint of the observed scene. This is usually fine for scenes that are not too cluttered. But with a growing number of objects, it becomes likely that an object is partially or even completely occluded by another object. Here, it is advantageous to observe the scene from multiple viewpoints to decrease the amount of occluded area. Apart from occlusions, object symmetries offer a big challenge for current pose estimation methods. This is because current 6D Pose Estimation methods regress the object pose from predicted 3D keypoints, which might have symmetric counterparts. In supervised learning, this needs extra care, so the network does not learn a compromise between all symmetric keypoints.
In this thesis, we propose a novel multi-view and symmetry-aware extension of a state-of-the-art single-view approach for 6D Pose Estimation. This approach is trained and evaluated on multiple custom synthetic datasets, which are rendered in a photorealistic manner. Additionally, we adapt the public YCB-Video dataset to support multiple views and evaluate our model against the current best performing method on this dataset. We assume known camera poses, but also train and evaluate on imprecisely measured camera positions. Furthermore, we propose a new loss function for training in a symmetry-aware manner by explicitly supervising the keypoint predictions with the original keypoints and their symmetric counterparts. Our results show that our novel multi-view approach is robust to changes in the camera poses, and it outperforms its single-view equivalent in highly cluttered scenes by a large margin. We marginally improve over our closest multi-view competitor. Utilizing our symmetry-aware training further improves the accuracy of the estimated poses, outperforming all other methods on the pose estimation of symmetric objects across all evaluated datasets.

Acknowledgments

First of all, I would like to thank Prof. Dr. Andreas Geiger and Prof. Dr. Gerhard Neumann for agreeing to be the supervising professors of this thesis and providing guidance along the way. Furthermore, I would like to thank my advisor Fabian Duffhauß for his time, guidance and insights during my thesis. Finally, I would also like to thank Bosch for providing the computational resources needed to complete this thesis.

Contents

1. Introduction

Object pose estimation is essential for autonomous robots to be able to interact with their environment. It is an important task in many real-world applications, such as augmented reality, autonomous driving and robot grasping. Object pose estimation is the task of predicting the class of an object and simultaneously estimating the exact pose of the object in 3D space, consisting of the rotation and position of the object. 6D Pose Estimation has been proven to be a challenging problem due to sensor noise, varying lighting conditions and occlusions in the scene [35, 17, 18, 3, 26]. The recent advances in deep learning, motivate several works to tackle this problem using convolution neural networks (CNNs) on RGB images. Still, many challenges remain until 6D Pose Estimation can be used in tasks like robot grasping. A big challenge is the loss of geometry information caused by the perspective projection, limiting the performance of methods that solely rely on RGB images. In addition, the use of ever larger neural networks with ever rising complexity calls for ever larger datasets to train these large networks. But the generation of datasets is expensive and especially time-consuming to label. Inexpensive RGB-D cameras provide extra depth information to help with the problem of lacking geometry information in RGB images, and photorealistic synthetic datasets, which are inexpensive to generate, can be used for large-scale training.

Most current approaches estimate the poses of objects in the scene using only a single input RGB/RGB-D image. Yet, in practice, scenes are composed of many objects and multiple images of the scene are often available, obtained by a single moving camera or in a multi-camera setup. Using only a single image is fine in simple scenes with only a few objects. But if there are more objects, it becomes harder to estimate the poses of all objects present, because it is much more likely that an object is partially or completely occluded by another object. Utilizing multiple views can help to observe all objects in the scene with fewer occlusions. However, this approach is rarely examined in current papers.

In this thesis, we address this open research problem and develop an approach that combines information from multiple views and predicts 6D pose estimates. But even with multiple viewpoints, 6D Pose Estimation remains a challenging problem. For competitive results, we leverage a state-of-the-art single-view approach for 6D Pose Estimation and extend it to a novel multi-view method, which can handle extremely complex scenes with many severe occlusions.
We also examine the estimation of symmetric objects, which have long been considered more challenging to estimate because of their ambiguity in their pose. For this we first examine where the current state-of-the-art model struggles when predicting symmetric objects, to then introduce a symmetry-aware training which aims to solve the identified problems.

Because multi-view has not been tackled widely in literature, there do not exist suitable public multi-view datasets to train and evaluate a multi-view approach. Therefore, we utilize three synthetic datasets developed by Bosch to train and evaluate our method. We compare

our method against the state-of-the-art single-view network on each dataset individually. Additionally, we compare against an older multi-view method that also fuses information from multiple views. Finally, we compare against a multi-stage approach, which can be trained on single-view and evaluated on multi-view, on a public dataset that supports multi-view for limited sequences.
We evaluate our symmetry-aware training separately using a synthetic dataset and one public real dataset, comparing against the current state-of-the-art for 6D Pose Estimation.

Our contributions are as follows:

- We propose a novel multi-view architecture for 6D Pose Estimation based on an existing single-view 6D Pose Estimation method that outperforms the single-view method and all prior works.
- We introduce a symmetry-aware training method that improves the 6D Pose Estimation of symmetric objects without changing the inference architecture.
- We demonstrate the benefits of our multi-view architecture and symmetry-aware training using a extensive evaluation on multiple datasets.

The rest of this thesis is structured as follows. In chapter 2 we discuss the theoretical background for 6D Pose Estimation, including the metrics used in this thesis to evaluate a 6D Pose Estimation method. We also introduce advanced algorithms that are relevant for the related work and our network implementation. In chapter 3 we give an overview of the field of 6D Pose Estimation and highlight especially relevant works to this thesis. In chapter 4 we outline the datasets that we use in this thesis, introduce the FFB6D network which is the basis of this thesis and detail our multi-view and symmetry extensions. In chapter 5 we present our result for our multi-view and symmetry extension on the three synthetic datasets as well as on one real dataset that is publicly available. Finally, in chapter 6 we summarize this thesis and our results and outline future research directions that we find relevant.

2. Theoretical Background

2.1. 6D Pose Estimation

6D Pose Estimation or 6DoF (6 Degrees of Freedom) Pose Estimation describes the process of finding the exact posture or pose of one or more objects in 3D. The pose is composed of the object's location **t** and the object's orientation **R**. More specifically, the position is described by a translation vector consisting of three coordinates $x, y, z \in \mathbb{R}$, and the orientation is represented by the three rotational angles $\phi, \psi, \theta \in \mathbb{R}$. These angles are also commonly known as *roll*, *pitch* and *yaw*. In total, this makes six parameters or six *Degrees of Freedom*, to describe the pose of an object exactly, hence 6D Pose or 6DoF Pose Estimation. However, in 3D, rotations are not commutative, so other alternative representations are often used that use slightly more parameters.

The pose is always expressed relative to some reference frame, often either a global coordinate system or the coordinate system of the camera which is observing the object. In Fig. 2.1 the pose of the drill and the other objects are estimated relative to the observing camera. Complex

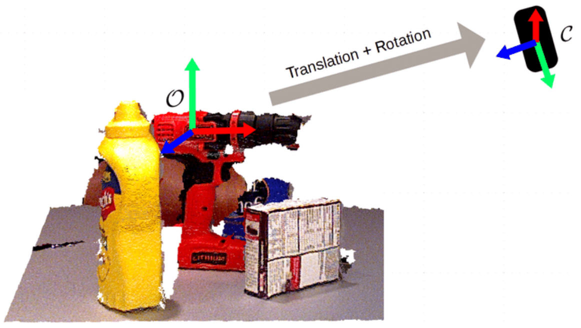


Figure 2.1.: The goal of 6D Pose Estimation is to find the translation **t** and rotation **R** from the object coordinate frame O to the camera coordinate frame C [10].

scenes where multiple objects are present are especially difficult for 6D Pose Estimation methods because larger objects can occlude smaller objects from the camera view. The pose of an object that is heavily occluded is more difficult to estimate because parts of the objects that specify the pose of the object cannot be observed.

2.2. 6D Pose Definition

The 6D Pose can be described in many different forms. Ideally, only using the three parameters for the translation and three parameters for the rotation of the object, as specified in Sec. 2.1. However, because rotations are not commutative in 3D, other representations for the rotation are chosen. The translation, however, can be easily described by three parameters in 3D. The most common form is a 3D vector to describe the position of the object relative to the camera or global coordinate system.

$$t = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \tag{2.1}$$

For the rotation, on the other hand, there exist multiple different ways that can be used to describe the same rotation in 3D. Some of the most commonly used representations are *Rotation Matrices*, *Axis-angle representations*, or *Quaternions*. In 6D Pose Estimation, the most common representations are either *Rotation Matrices* because of their intuitive representation or *Quaternions* because of their compact representation.

2.2.1. Quaternions

Quaternions use four parameters to express a 3D rotation. They are the most compact form to describe an unambiguous rotation in 3D. When predicting the 6D pose directly, they are often used because only four parameters have to be estimated. Another advantage of quaternions is that they do not require much storage space compared to other representations, as only four parameters have to be stored. A big disadvantage of quaternions is that they are not very intuitive. To express a 3D rotation as a quaternion, Euler's rotation theorem is used, which says that any rotation or sequence of rotations of an object or coordinate system about a fixed point is equivalent to a single rotation by an angle θ about a fixed axis a that runs through the fixed point. The four parameters can be used to express the corresponding rotation in a 3D position vector that represents the center of the rotation (see Eq. 2.2).

$$q = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \in \mathbb{R}^4 \tag{2.2}$$

Here the angle θ can be recovered by Eq. 2.3 and the rotation axis a can be recovered by Eq. 2.4.

$$\theta = 2\,\mathrm{atan2}\left(\sqrt{(q_x^2 + q_y^2 + q_z^2)}, q_w\right) \tag{2.3}$$

$$a = (a_x, a_y, a_z) = \frac{1}{\sqrt{(q_x^2 + q_y^2 + q_z^2)}} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \tag{2.4}$$

2.2.2. Rotation Matrix

A 3×3 rotation matrix is a more intuitive way to express a 3D rotation. However, it is not as compact as quaternions. To express the same 3D rotation, nine parameters are required when using a rotation matrix. However, the construction of such a rotation matrix is fairly simple when having access to three rotational angles ϕ, ψ, θ as can be seen in Eq. 2.5.

$$R = R_\theta R_\psi R_\phi = \overset{yaw}{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{pitch}{\begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix}} \overset{roll}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}}$$
$$= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi\sin\phi - \sin\theta\cos\phi & \cos\theta\sin\psi\cos\phi + \sin\theta\sin\phi \\ \sin\theta\cos\psi & \sin\theta\sin\psi\sin\phi + \cos\theta\cos\phi & \sin\theta\sin\psi\cos\phi - \cos\theta\sin\phi \\ -\sin\psi & \cos\psi\sin\phi & \cos\psi\cos\phi \end{bmatrix} \tag{2.5}$$

The inverse formulation of how to how to recover the rotational angles ϕ, ψ, θ from an arbitrary rotation matrix **R** is shown in Eq. 2.6.

$$R = \begin{bmatrix} r_{12} & r_{12} & r_{13} \\ r_{22} & r_{22} & r_{23} \\ r_{32} & r_{32} & r_{33} \end{bmatrix} \qquad \begin{aligned} \phi &= \text{atan2}(r_{32}, r_{33}) \\ \psi &= \text{atan2}\left(-r_{31}, \sqrt{(r_{32}^2 + r_{33}^2)}\right) \\ \theta &= \text{atan2}(r_{21}, r_{11}) \end{aligned} \tag{2.6}$$

2.2.3. Transformation Matrix

It is also possible to interpret the pose of the object, defined by a rotation and translation, as a coordinate transformation between the object coordinate system O and the camera/world coordinate system C (see Fig. 2.1). This coordinate transformation can be expressed by a transformation matrix $T = (\mathbf{R}|\mathrm{t})$ which combines the rotation matrix **R** and the translation vector **t** into a 3×4 matrix. To transform a point ${}^{O}p = (p_x, p_y, p_z, 1)$ on the object mesh defined in the object coordinate system O to the camera coordinate system C, the point can be transformed by the known pose expressed as the transformation matrix ${}^{C}T_O$.

$${}^{C}p = {}^{C}T_O \cdot {}^{O}p \tag{2.7}$$

These transformations can also be concatenated to not only transform a point ${}^{O}p$ defined in the object coordinate system O to the camera coordinate system C but also the world coordinate system W. For this, the transformation matrix has to be extended by a $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ row to keep the same vector dimensions.

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.8}$$

Using the known transformation between the world coordinate system and the camera coordinate system ${}^{W}T_C$ gives the point in world coordinates.

$${}^{W}p = {}^{W}T_C \cdot {}^{C}T_O \cdot {}^{O}p \tag{2.9}$$

2.3. Symmetry

A geometric object is called symmetric if, after a transformation, the object is indistinguishable from the object before the transformation. Symmetry is a big challenge in 6D Pose Estimation, because a symmetric object has multiple or sometimes infinite poses that describe the same object configuration because it has a symmetry. Symmetric objects therefore have to be considered with extra care in 6D Pose Estimation.

The most common symmetries in geometry and also the most relevant symmetries in 6D Pose Estimation are *reflectional symmetries* and *rotational symmetries*.

2.3.1. Reflectional Symmetry

Reflectional symmetry or mirror symmetry exists if there is a line in 2D or a plane in 3D that passes through the object, which divides the object into two pieces that are mirror images of each other. For instance, the big letter A has one reflectional symmetry, which goes through the center of the letter A in a vertical manner.

2.3.2. Rotational Symmetry

A rotational symmetry exists if an object can be rotated around a fixed point in 2D or a line in 3D without changing the overall shape. The symmetry can be either continuous/infinite or discrete to the n_{th} order. A circle, for example, is continuously symmetric around its center point, while a square is only discretely symmetric with order $n = 4$ around its center point when the rotation is a multiple of $360/4 = 90$ degrees.

2.4. Metrics

This section introduces various metrics which are commonly used to evaluate and compare 6D Pose Estimation methods. These metrics were first introduced by Hinterstoißer et al. [15] and later extended by PoseCNN [35] and DenseFusion [31].

2.4.1. Average Distance

The Average Distance (ADD) [15] is the most common metric to evaluate the precision of the 6D Pose Estimation methods. It measures how well the object mesh projected by the predicted pose is aligned with object mesh projected by the ground truth pose. To calculate the alignment, the average distance between the same vertex v projected by the predicted and ground truth pose is calculated. The formula for the metric is given in Eq. 2.10.

$$ADD = \frac{1}{|V|} \sum_{v \in V} ||(\mathbf{R}v + t) - (\mathbf{R}^{*}v + t^{*})||_2 \tag{2.10}$$

2.4.2. Average Closest Point Distance

The Average Closest Point Distance (ADD-S) [15] is a variation of the ADD metric. It extends the ADD metric to symmetric objects. Symmetric objects cannot be express by a single pose as discussed in Sec. 2.3. The pose is ambiguous along the symmetry axis/plane. Therefore, the ADD metric may indicate a poor pose estimate because the same vertex is projected by different poses. However, qualitative inspection shows that the meshes are well aligned.

Therefore, for a fair evaluation, instead of computing the distance between the same vertex projected by the reference pose and the predicted pose, the distance between the projected vertex and the closest vertex in the reference projection is computed.

$$ADD - S = \frac{1}{|V|} \sum_{v_1 \in V} \min_{v_2 \in V} \|(\mathbf{R}v_1 + t) - (\mathbf{R}^* v_2 + t^*)\|_2 \tag{2.11}$$

2.4.3. Average (Closest Point) Distance

The Average (Closest Point) Distance (ADD(S)) [15] is the combination of the two ADD and ADD-S metrics. The ADD-S metric is slightly more vague because it uses a minimum formulation. Therefore, to have the most precise formulation possible while still ensuring the correct evaluation on symmetric objects, the ADD(S) metric defines the ADD-S metric for pre-defined symmetric objects and the more strict ADD metric for non-symmetric objects.

$$ADD(S) = \begin{cases} ADD - S & \text{if symmetric object} \\ ADD & \text{otherwise} \end{cases} \tag{2.12}$$

2.4.4. Area under the Curve

Instead of reporting the ADD(S) metric directly, PoseCNN [35] has introduced the ADD(S) Area Under the Curve (AUC) metric to combine multiple measurements m of multiple predicted scenes M with a varying threshold accuracy for a single object class. For 6D Pose Estimation, the metric is defined on the integral of the threshold function between 0 meters and 0.1 meters.

$$AUC(M) = 10 \int_0^{0.1} \left(\frac{1}{|M|} \sum_{m \in M} \mathbb{1}_{m<x} \right) \tag{2.13}$$

2.4.5. Average Distance Precision

The Average Distance Precision (ADD(S)<2cm) metric [31] is another extension of the ADD metric. It is similar to the ADD(S)-AUC metric. Instead of having a sliding precision threshold, for ADD(S)<2cm metric the threshold is fixed at 2cm. This is considered the maximum tolerance for successful robot manipulation tasks. The Eq. 2.14 shows the formulation on how to calculate this metric.

$$ADD(S) < 2cm(M) = \frac{1}{|M|} \sum_{m \in M} \mathbb{1}_{m<0.02} \tag{2.14}$$

2.5. Advanced Network Functions

This section contains a short description of advanced algorithms that are used as pre-processing and post-processing steps in relevant related works and our pose estimation network.

2.5.1. Farthest Point Sampling

Most 6D Pose Estimation methods nowadays do not estimate the 6D pose directly but utilize tasks like 3D keypoint estimation as an intermediate task to regress the final 6D pose from the keypoints.

To estimate the 6D pose accurately from 3D keypoints it is important that the keypoints are well distributed across the object, something that will be further discussed in Sec. 4.2.3. For this, Peng et al. introduced Farthest Point Sampling (FPS) in their PVNet paper [26]. The goal of FPS is to obtain 3D keypoints from an object, where the keypoints are evenly spread across the object's surface. These farthest point sampled keypoints represent a subset S of all vertices V of the object's mesh. The first step is to calculate the center of the object c. In the next step, the vertex v_0 is calculated that is the farthest keypoint from the center c using Eq. 2.15.

$$v_0 = \arg\max_{v \in V} \|c - v\|_2 \tag{2.15}$$

The keypoint v_0 is the first keypoint in S. Additional keypoints v_i are added iteratively to S, by selecting keypoints that maximize the distance to all previously added keypoints in S.

$$v_i = \arg\max_{v \in V} \frac{1}{|S|} \sum_{s \in S} \|v - s\|_2 \tag{2.16}$$

This iterative process is repeated until a desirable amount of keypoints is reached.

2.5.2. Mean Shift Clustering

In combination with 3D keypoint estimation, Mean Shift Clustering is often used to cluster keypoint proposals. Mean Shift Clustering is an algorithm to find the maxima of a density function given discrete data X sampled from that function [7]. Here, the discrete data points are the keypoint proposals generated by the network. The algorithm works by iteratively moving an initial estimate or seed x towards the denser part of the cluster. The algorithm works for a single initial point $x \in X$ but can be easily parallelized to work with multiple initial points. To move the current seed point to the denser part of X an offset is computed by Eq. 2.17.

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{2.17}$$

The offset is weighted by using a kernel function K which is usually a Gaussian Kernel (see Eq. 2.18 and summed over all points in X. The parameter b defines the bandwidth of the kernel. The smaller this bandwidth, the more focused the kernel, pulling the seed point towards the center of the cluster. The iterative process terminates when the seed point converges, or a maximum number of iterations is reached.

$$K(x_i - x) = \frac{1}{b\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\|x_i - x\|_2}{b}\right)^2\right) \tag{2.18}$$

3. Related Work

6D Pose Estimation is a widely explored topic in computer vision. In general, it can be divided into two categories. Instance-level pose estimation assumes that the target objects are known a priori, for example, a particular power drill or food item. Category-level pose estimation assumes the target objects are from specific known categories, but the exact object models are unknown [29, 32]. Instance-level pose estimation is the dominant problem in current research for topics such as robot grasping and represents the focus of this master thesis.

The BOP-Challenge [17] further divides the problem of instance-level pose estimation into the following sub-problems:
The simplest task is Single instance of a Single object (SiSo) pose estimation, where only the pose of a single object at the same time has to be estimated. In Single instance of Multiple objects (SiMo), multiple objects can be present in the scene, but the same object instance is present only once. Multiple instances of a Single object (MiSo) is the problem where multiple instances of the same objects are allowed. The most complex task is Multiple instances of Multiple objects (MiMo), where multiple objects from multiple instances can be present in the scene at the same time. A variation on MiMo is the Varying number of instances of a varying number of objects (ViVo) task where the number of objects are known, which simplifies the evaluation process drastically.

In this master thesis, we focus on the SiMo task as it is the task most commonly tackled task in current research in the context of robot grasping. In the following chapter, we present relevant works for instance-level pose estimation that solve the SiMo task.

3.1. Pose Estimation with RGB Data

Early pose estimation methods were dominated by feature-based methods and 3D template matching approaches. In template matching approaches, a ridged template of the object is constructed and is moved over the image with a sliding window. At each location, a similarity score is computed, and the best match is obtained by finding the highest similarity score [6]. In 6D Pose Estimation, a template is often obtained by rendering the corresponding 3D model of the object. Template-based methods are useful for detecting texture-less objects, however, they cannot handle occlusions very well.
Feature-based methods, extract local features in the image like Scale Invariant Feature Transform (SIFT) [24] features on the 3D models, to establish the 2D-3D correspondences to recover the poses in a Perspective-n-Point (PnP) manner. Feature-based methods are able to handle occlusions better, however, they require sufficient texture to find local features. The rise of deep learning has motivated many works to tackle the problem of 6D Pose Estimation with neural networks, specifically CNNs. One of the first deep learning approaches that

outperformed classical methods was PoseCNN [35] from Xiang et al. which is a two stage CNN approach. PoseCNN also introduced the YCB-Video dataset (see Sec. 4.1.1), which is a high quality 6D Pose Estimation dataset that exceeded previous 6D Pose Estimation datasets like LINEMOD [16] by multiple orders of magnitude in size. The architecture of PoseCNN

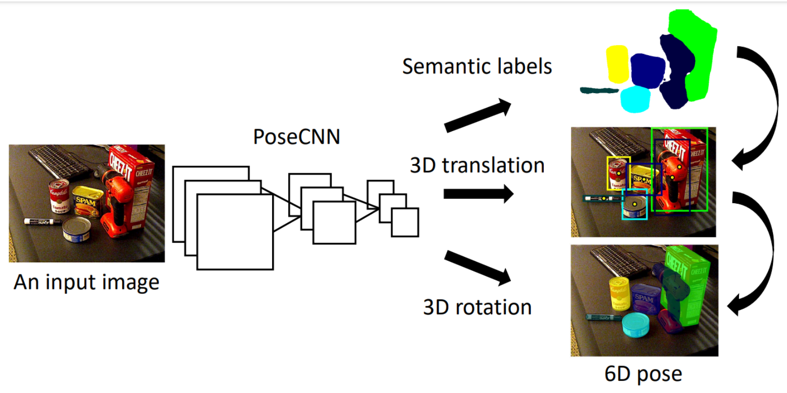


Figure 3.1.: PoseCNN architecture, designed to learn three tasks: Semantic labeling, 3D translation estimation and 3D rotation estimation [35].

can be seen in Fig. 3.1. The first stage of the network is a multi-layer convolutional neural network which extracts feature maps with different resolutions from the input image. This stage is the backbone of the network, the extracted features are shared across all downstream tasks. The second stage of the network embeds the high-dimensional features generated by the CNN into low-dimensional task-specific features. Then the network performs semantic segmentation, 3D translation estimation and 3D rotation regression, to output the 6D poses of all objects in the scene.

Many works followed PoseCNN that utilized deep learning for 6D Pose Estimation like DeepIM [22], however the loss of geometry information due to perspective projections limits the performance of these RGB only methods, which is why current research has moved beyond RGB only methods.

3.2. Pose Estimation with Depth Data

The development of depth sensors and point cloud representation learning techniques, such as PointNet [27] and its successor PointNet++ [28], motivate learning tasks such as object detection or segmentation tasks, only with only depth data, often represented as point clouds. Point cloud approaches can leverage direct geometric information about the scene, unlike approaches that use RGB images, where only limited geometry information can be extracted. However, sparsity and non-texture of point clouds limit the performance of these approaches. Also, objects with reflective surfaces cannot be captured by depth sensors. Therefore, the utilization of point cloud only methods for 6D Pose Estimation is rare. Gao et al. proposed one of the few methods that only utilize point clouds for 6D Pose Estimation. CloudPose [9] is considered to be the first deep learning approach that performs 6D Pose Estimation only from point clouds constructed from a depth image. Another work that only uses depth information for 6D Pose Estimation is the work from Bui et al. [4], who propose a multi-task framework to combine manifold learning and 3D orientation regression directly from depth images to learn view descriptors which can be also applied to either retrieve or regress the 6D pose.

3.3. Pose Estimation with RGB-D Data

Recovering the 6D pose from only images or only point clouds is hard. Both have their drawbacks. Images provide limited information about geometry in the scene, while point cloud only methods rely on very sparse data which does not contain any appearance information.
With the commercially available Kinect depth sensor in 2010, researchers began investigating 6D Pose Estimation using RGB-D images. With the additional depth channel, pose estimation could combine the advantages of both sensor modalities [14]. The most straightforward utilization of the two modalities is to first perform the initial pose estimation based on RGB images and then further refine with depth images, such as via Iterative Closest Point (ICP) refinement [15, 30].
PointFusion [36] was one of the earliest deep learning methods to fuse deep image features and point clouds features in the context of autonomous driving. They utilized a PointNet [27] to process the point cloud and a ResNet [11] to process the images. Based on PointFusion, Wang et al. developed DenseFusion [31] which was applied in the context of 6D Pose Estimation using RGB-D images.
Another big milestone model was PVNet [26] by Peng et al. and its successor PVN3D by He et al. [13]. These works deviated from previous works by not predicting the 6D pose directly but predicting 3D keypoints from which to regress the final 6D pose, similar to the PnP-methods using handcrafted features. They argue that methods that predict the 6D pose directly suffer from the non-linearity of the rotation space, which makes learning and generalizing with a data-driven deep learning model hard. They prove this by outperforming the previous works that predict the 6D pose directly by a large margin. The latest work in the line of models that predict the 6D pose via keypoints is FFB6D [12] by He et al. which stands for *Full Flow Bidirectional Fusion Network for 6D Pose Estimation*. It represents the state-of-the-art for 6D Pose Estimation at the time of writing this thesis. It outperforms previous works like PVN3D by utilizing an early multi-stage fusion approach between the separate CNN and PointNet branches, in contrast to the earlier fusion methods like PVN3D and DenseFusion which fuse depth and image feature later in the network in one layer. More details on the FFB6D model can be found in the Method chapter of this thesis.

3.4. Pose Estimation with Multi-View Data

So far, only a limited number of works exist that tackle 6D Pose Estimation from multiple viewpoints. Li et al. were one of the first to tackle multi-view 6D Pose Estimation with multiple views in *A Unified Framework for Multi-View Multi-Class Object Pose Estimation* [21].
CosyPose [20] from Labbe et al. is one of the more recent works for multi-view pose estimation, by utilizing a multi-stage approach. This novel multi-stage architecture (visualized in Fig. 3.2) makes training on single-view scenes possible, while being able to utilize multiple views during test time. This novelty is possible because the stages are not trained end-to-end. The first stage estimates 6D pose hypotheses from a single view. This step is learned with a neural

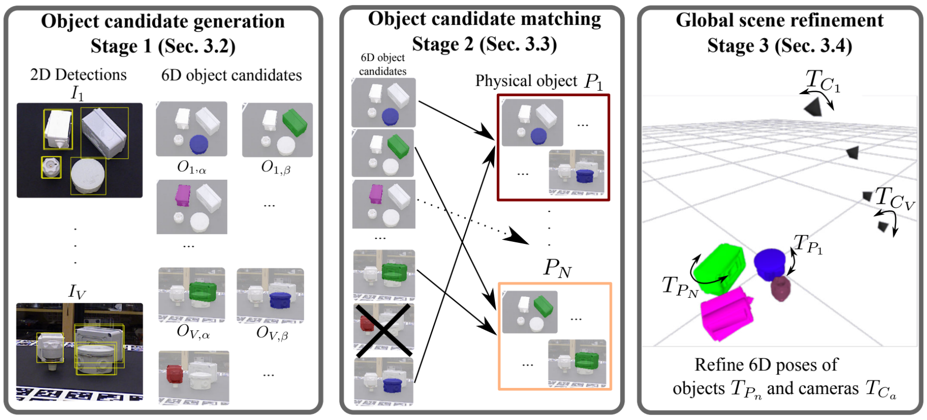


Figure 3.2.: Full multi-stage architecture of CosyPose [20]. In the first stage, the initial object candidates are detected in each view separately. In the second stage. the object candidates are matched across views. In the third stage, all object and camera poses are refined to minimize the reprojection error.

architecture. In the following step, the individual 6D object pose hypotheses are matched across different input images to get a globally consistent scene representation. In the final step, the camera viewpoints and 6D object poses are estimated jointly by minimizing the multi-view reprojection error with bundle adjustment. However, this approach utilizes only RGB information and assumes unknown camera positions, which makes this work different from the approach we are pursuing in this master thesis. Nevertheless, CosyPose represents, at the time of writing this thesis, the state-of-the-art for multi-view pose estimation on a publicly available dataset. In contrast to CosyPose, we are pursuing an approach utilizing RGB-D data and known camera positions, which should help increase the accuracy of such a multi-view 6D Pose Estimation method. A more similar approach to our proposal is a multi-view variation of PVN3D developed in [8] (see Fig. 3.3). This work improved upon the PVN3D work by He et al., by fusing the depth information of the multiple RGB-D cameras to a global point cloud and processing the RGB images in a batch-wise manner to finally fuse the features generated by the global point cloud and the batched images to finally predict the 6D pose proposal of objects in the scene. Unlike CosyPose, this approach assumes

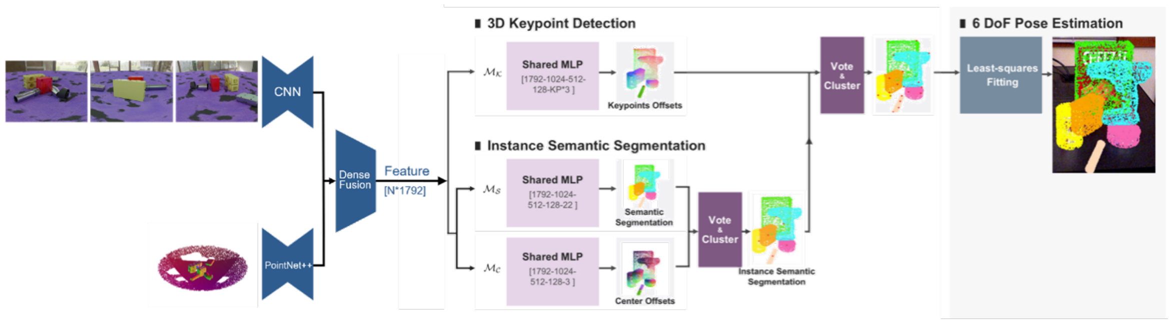


Figure 3.3.: Overview of the multi-view adaption of PVN3D [8]. The images from the multiple views are processed in the CNN branch. The point clouds are pre-transformed into a global point cloud to be processed by the PointNet++ branch. After the separate processing of both modalities, the features are fused densely to predict instance segmentations and 3D keypoints.

known camera viewpoints to fuse the depth information of the multiple RGB-D cameras. A drawback of this work is the late dense fusion of RGB and depth. Nevertheless, this approach shows promising results on an internal synthetic dataset. He et al. show in their follow-up work to PVN3D called FFB6D that an early fusion of appearance and depth is beneficial for the final 6D pose proposal. This early fusion is very challenging to adapted to multi-view, but promises further improvements over the multi-view PVN3D model, in line to improvements shown in the single-view case.

4. Method

Our goal is to make 6D Pose Estimation approaches more effective in complex scenes and more accurate for symmetric objects. For this we look into the pose estimation from multiple views while also explicitly handling object symmetries. This problem of estimating the 6D object poses from multiple views has not been widely explored yet. The most relevant work to address this problem is CosyPose [20] (see Sec. 3.4). CosyPose combines a deep learning object detection pipeline with classical multi-view reconstruction techniques to estimate the pose of the objects and cameras jointly in a multi-step approach. This approach has the advantage of being able to be trained on single-view scenes, but then at test time, be able to utilize further frames to make more accurate predictions. The advantage comes at the cost of having to solve a very hard problem of joint pose estimation of camera and object poses without deep learning.

In contrast to CosyPose, we want to incorporate the multi-view fusion into an end-to-end trainable model. We decide to exploit the fact that pose estimation for object manipulation is usually explored in a stationary and highly controlled setup. Therefore, it is reasonable to assume to have multiple cameras observing the scene from ideal observation points with known relative positions to each other. This offers the biggest gains of a multi-view setup compared to a single-view setup and makes the fusion of the multiple views easy because of the known relative camera positions. We decide to utilize three to five cameras, as this allows us to cover most of the viewing angles and observe the whole scene from 360 degrees. We choose to adapt the current state-of-the-art single-view 6D Pose Estimation model called FFB6D [12] to multiple views to assure improvements over the state-of-the-art in pose estimation.

As deep learning methods require large amounts of data to learn, it is common for 6D Pose Estimation methods to utilize a mixture of real and synthetic training data. For multi-view 6D Pose Estimation, however, there do not exist suitable datasets of real recordings that offer multiple views of the same scene that are distinct enough from one another. We therefore will introduce in the following sections the adaption of the YCB-Video dataset [35] to a real multi-view dataset, as first introduced in CosyPose. We will also introduce the synthetic only datasets SCAPE YCB, SCAPE 2 and SCAPE YCB2 which first appeared in [8]. Furthermore, we provide a detailed insight into the original FFB6D model and our novel extension of its multi-view variant. Further, we discuss our addition to make the predictions of FFB6D more robust towards symmetric objects by handling symmetries explicitly during training.

The multi-view extension and the symmetry extension are independent of each other and are therefore also discussed separately in this thesis.

4.1. Datasets

4.1.1. YCB-Video

The YCB-Video dataset is one of the most frequently used benchmark in 6D Pose Estimation. It was first introduced by Xiang et al. in the paper PoseCNN [35]. The dataset offers a wide variety of domains, containing 21 objects such as food items (food cans, fruits), kitchen items (bowl, mug) and tools (marker, power drill). The objects originate from the Yale-CMU-Berkeley (YCB) object and model set [5]. Some objects have rich texture, others are almost uncolored surfaces, and some have natural symmetries, which are challenging for 6D Pose Estimation and require additional processing during training and testing.

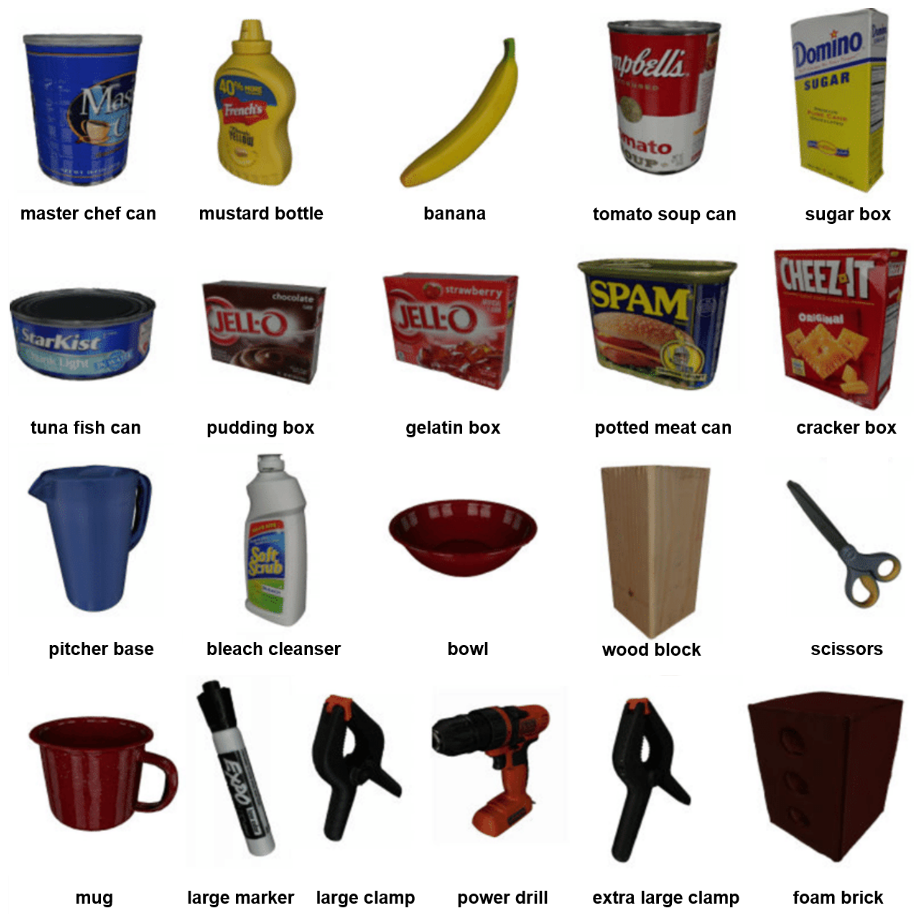
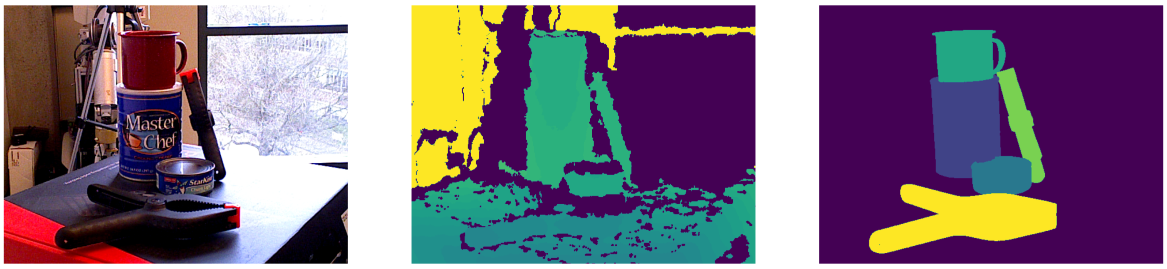


Figure 4.1.: The 21 objects of the YCB-Video dataset [5]. In the SCAPE YCB dataset we reuse the following non-symmetric objects: *master chef can, mustard bottle banana, tomato soup can, sugar box, tuna fish can, pudding box, gelatin box, potted meat can, cracker box and power drill*

The dataset contains 92 RGB-D video sequences. Each sequence shows a static scene with a subset of the 21 objects (see Fig. 4.1). In total there are 133,827 frames which consist of an RGB image, a depth map, as well as ground truth information like a segmentation map and the 6D pose of the objects in relation to the camera position (see Fig. 4.2). Additionally, the camera pose is provided in a global coordinate system. The official train/test split uses 80 video sequences for training. The testing is done on predefined keyframes chosen from the remaining 12 video sequences. In addition, to the real scenes, there are 80,000 synthetic non-sequential frames that can be used for training. For the synthetic frames, a subset of objects is randomly placed in the image in front of a uniform background.



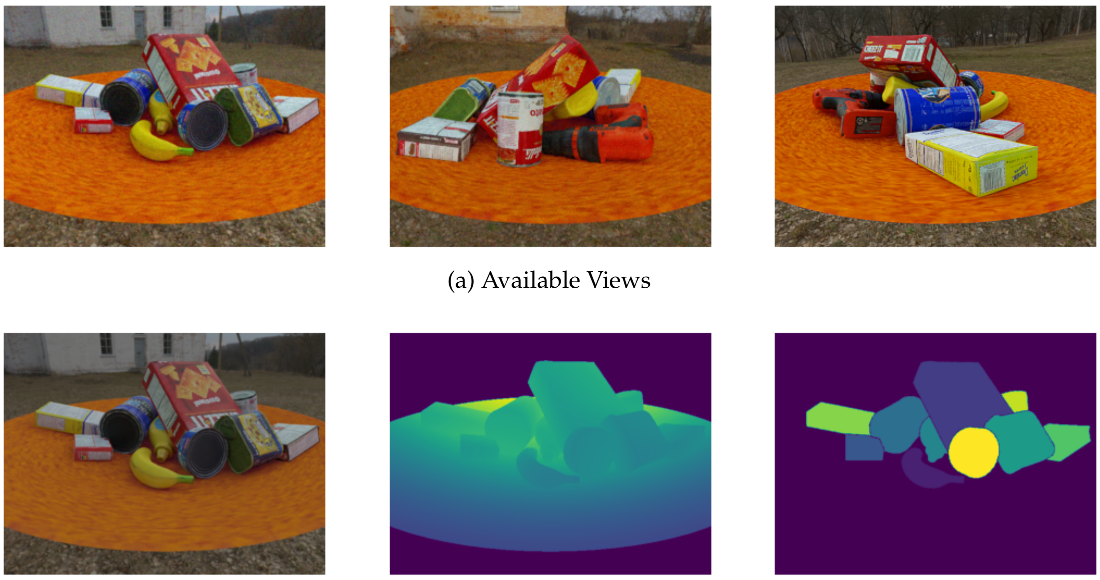
(a) RGB Image (b) Depth Map (c) Segmentation Map

Figure 4.2.: In the YCB-Video dataset, each training/test sample consists of an RGB image (a), a depth map (b) and a ground truth segmentation map (c). Additionally, the 6D poses of the objectss in the scene and the camera poses are provided.

For its multi-view testing, CosyPose utilizes a random group of the keyframes from each test sequence. We implement the same grouping algorithm with the same seed to achieve comparable results. We not only do this for the test keyframes, but also the frames in the training sequences to use them for training. Unfortunately, this is not possible for the non-sequential synthetic frames. This drastically reduces the training data, which is not a problem for CosyPose as it can be trained on single-view, but it is a challenge for our method because it is trained and tested with multi-view frames. In Sec. 5.4 we discuss how we can solve this problem.

4.1.2. SCAPE YCB

The Sequential Clustered Annotated Pose Estimation (SCAPE) dataset is a continuously improving synthetic dataset, rendered in a photorealistic manner, developed by the Bosch Group. The first variation by the name of SCAPE appeared in [2]. As the name suggests, this dataset was designed for sequential processing of frames for 6D Pose Estimation. But with the introduction of SCAPE YCB in [8] the focus changed to a static setup, with multiple independent cameras, which is the same use case as this thesis. The dataset includes 8333 scenes with three views each, resulting in 24,999 frames. Each scene consists of eleven objects from the YCB object set [5], hence the name SCAPE YCB. The cameras are pointing towards the center of the scene, they are equally distributed in a circle and share the same distance to the scene center. The objects are placed randomly in the scene by spawning them above a table surface in a random pose and then simulating their fall onto the surface. After the objects have reached a resting position, the scene is captured by the surrounding cameras. The table surface is randomly sampled from a collection of Blender materials and the background is generated using a randomly selected high dynamic range image. Lastly, the lighting is slightly randomized in intensity and colors, resulting in different realistic shadows and reflections. By choosing not a flat table surface but a slightly concave surface, the objects tend to gather in the center, which results in the objects occluding each other. This means that it is no longer possible to even see all objects from a single view (see Fig. 4.3). Additionally, a sub-variation of the SCAPE YCB dataset exists where the scenes are rendered from not perfectly placed cameras. This is highly relevant for this work, as one assumption of the method introduced in this thesis is that the relative camera positions are to be known to fuse the information of all cameras. It therefore can give valuable insight as to how well the introduced approach can generalize to incorrect camera placements.


(a) Available Views

(b) Available Representations per Image

Figure 4.3.: SCAPE YCB offers three views per scene with additional depth and segmentation maps. The scenes are rendered in a photorealistic manner, showing cluttered scenes with many occlusions.

4.1.3. SCAPE 2

The SCAPE 2 dataset is another variation on the original SCAPE dataset. It includes the same number of scenes and also offers three views for each scene. The cameras and objects are placed in a similar manner, but the number and types of objects are different. Three objects are chosen from the T-LESS dataset [18] and three additional objects are selected from the Bosch internal AMIRA project [33] (see Fig. 4.4). In contrast to the SCAPE YCB dataset where

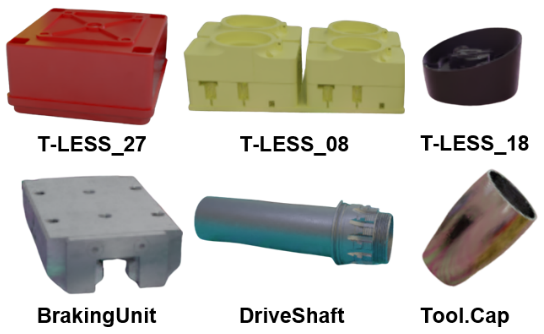


Figure 4.4.: The six objects contained in the SCAPE 2 dataset. The top three objects are taken from the T-LESS dataset [18], while the bottom three objects are selected from the Bosch internal AMIRA project [33].

none of the objects are symmetric, the selected objects in the SCAPE 2 dataset do all have at least one symmetry except for the T-LESS_18 object. However, because this dataset has fewer objects, the number of occlusions is lower as the SCAPE YCB variant (but still higher than in the YCB-Video sequences), which makes this dataset more suitable to test improvements made in the handling of object symmetries.

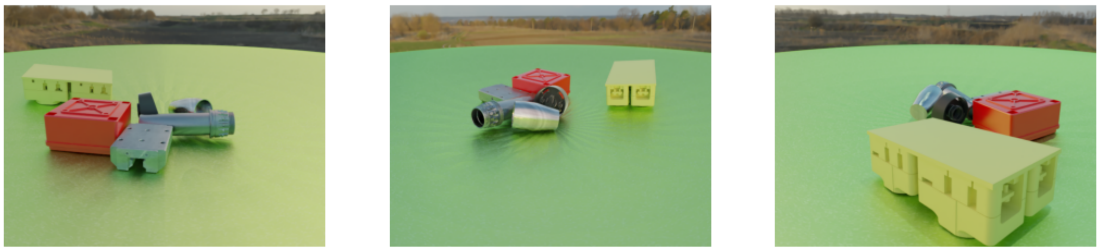


Figure 4.5.: The SCAPE 2 dataset offers three views per scene. The scenes are rendered in a photorealistic manner, showing many symmetric objects with only a few occlusions.

4.1.4. SCAPE YCB2

The SCAPE YCB2 dataset is a continuous development of the SCAPE YCB dataset. It consists of the same scenes, but in contrast to the SCAPE YCB dataset, the SCAPE YCB2 dataset offers up to four camera views from around the scene, with an additional top-down view. This makes the SCAPE YCB2 dataset ideal for evaluating the improvements of additional

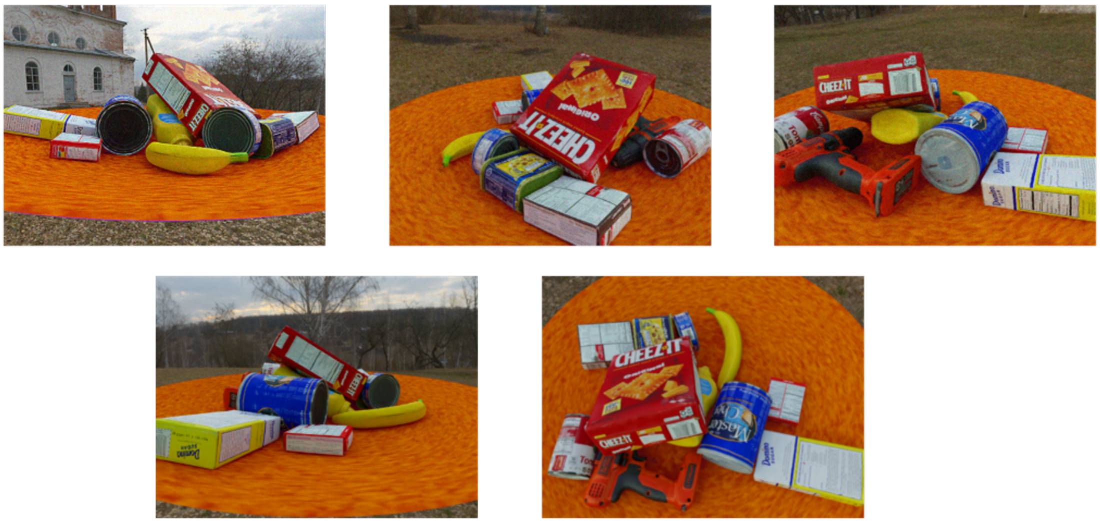


Figure 4.6.: The SCAPE YCB2 dataset offers four views per scene with an additional top-down view. The rendered scenes are identical to the SCAPE YCB dataset, showing photorealistic scenes with many occlusions.

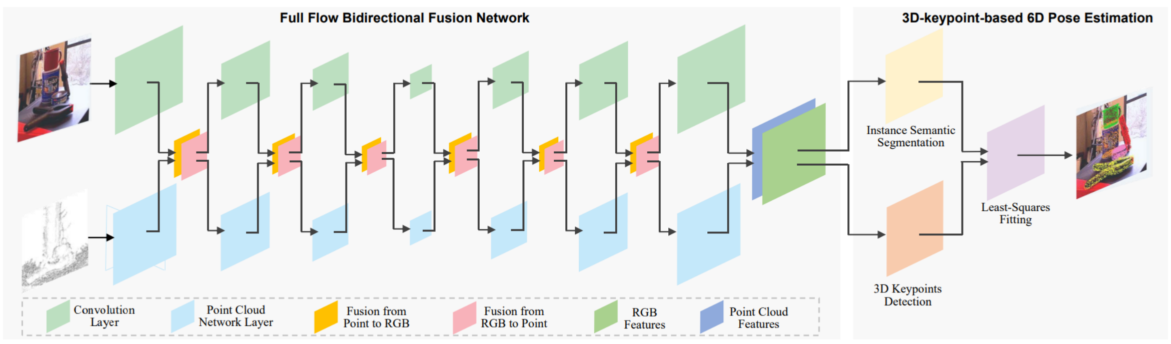
cameras and how many are needed for optimal results. Furthermore, the SCAPE YCB2 dataset is slightly harder than the SCAPE YCB dataset because the camera poses change in each scene. The camera poses are chosen randomly for each scene, with the constraint that each quadrant in the scene contains one camera pointing towards the scene center to ensure that the scene is observed by greatly varying viewing angles. This can give great insight whether the multi-view model overfits to specific camera poses or generalizes the fusions of multiple views beyond the preset camera poses.

4.2. Proposed Network

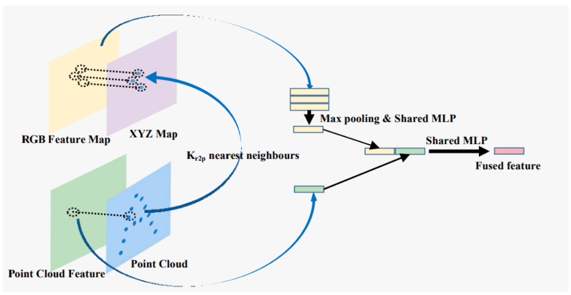
4.2.1. FFB6D

The FFB6D [12] network is an advancement on the PVN3D [13] network, both of which were developed by Yisheng He et al.. It represents the state-of-the-art approach for 6D Pose Estimation at the time of writing this thesis. It outperforms the previous methods by bidirectionally fusing image and depth information early in the network. Similarly to PVN3D and in contrast to methods such as PoseCNN [35] or CosyPose [20], it does not directly infer the 6D pose, but predicts multiple 3D keypoints in the scene that are then used in a post-processing step to infer the final 6D poses via a least-squares-fitting [1]. It is therefore the starting point for our multi-view extension and will be discussed in detail in the following paragraphs.

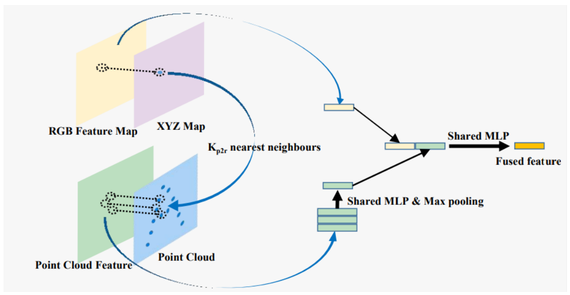
Like most other 6D Pose Estimation methods for robot grasping, FFB6D makes a couple of assumptions for pose estimation. One being that the object models of the objects in the scene must be known a priori and that maximally a single object of the same type must be present in the scene.



(a) FFB6D utilizes a CNN to extract high-dimensional features from the RGB image and a PCN to extract high-dimensional features from the depth information represented as a point cloud. The two separate branches are connected by bidirectional fusion modules to exchange high-dimensional geometry and appearance features. Finally, the features of both branches are concatenated and fed into an instance semantic segmentation and a 3D keypoint voting module. At test time, the 6D pose is recovered with a least-squares-fitting.



(b) Pixel-to-Point Fusion



(c) Point-to-Pixel Fusion

Figure 4.7.: Overview of FFB6D [12]

Input

The FFB6D network is designed to handle aligned RGB-D data. The RGB-D frame is split into the image and depth map, which are processed independently through the network. The image can be directly processed by the network, however the depth map is pre-processed before parsing it into the network. Using the camera intrinsics, it is possible to convert the depth map into a point cloud, which is then processed by the network. But to reduce the computational complexity, a fixed number of points is sampled before inputting the point cloud into the network. Additionally, points that are out of range for the sensor are also removed in this step. In addition to the 3D point location, each point is also encoded with the surface normals at these points, which can be also computed with the depth map and the camera intrinsics.

Labels

Each training sample contains the required labels to train the network. For the keypoint estimation, eight ground truth 3D keypoints per object are provided in camera coordinates. Experiments by He et al. have shown that eight keypoints work best for pose estimation, but more keypoints are also possible. The keypoints are selected using a farthest point sampling described in Sec. 2.5.1 of pre-computed SIFT keypoints on the known 3D object model. They are then projected into the camera coordinates by the ground truth 6D pose. Additionally, to the ground truth keypoints, a segmentation map of the scene is required. Lastly, the center position of each object is needed for training. The centers are computed by finding the mean of all corner points of the known 3D object models.

Architecture

The FFB6D architecture can be split into three parts. The first part is the feature extraction part. Here lies the novelty of FFB6D. The following 3D keypoint and semantic segmentation parts are known from PVN3D, as well as the final least-squares-fitting task to regress the 6D pose.
The feature extraction part can be further divided into two branches. The CNN branch to extract appearance features from the RGB image with a ResNet-34 [11] encoder and PSPNet [38] decoder architecture, and the point cloud branch to extract geometric features from the point cloud with a RandLA-Net [19]. The novelty lies in the fusion modules that connect both branches bidirectionally to exchange geometric and appearance information. In this way, the two branches can utilize the extra appearance/geometric information from the other branch to facilitate their own representation learning. The extracted features of both branches are then concatenated and fed into the instance semantic segmentation and the 3D keypoint module to obtain the per-object 3D keypoints. Finally, a least-squares-fitting with the estimated keypoints and the known object model points is applied to estimate the 6D poses.

Fusion

The novel fusion modules consist of two different fusion blocks. One fuses pixel information with point information from image features to point cloud features (see Fig. 4.7b). The other fuses the point information from the point cloud features with the image features of the CNN branch (see Fig. 4.7c).

To achieve this fusion, the fact that the RGB-D image is well-aligned is exploited. Rather than concatenating a global feature from the CNN branch to the Point-Cloud-Network (PCN) branch, which loses valuable information of the multiple objects in the scene, relevant features are fused densely with the other branch. For this, it is possible to use the 3D point cloud as a bridge to connect the pixel-wise and point-wise features.
To achieve this, the depth of each pixel is lifted with the intrinsic matrix to its corresponding 3D point to obtain an XYZ map aligned with the RGB map. Then for the pixel-to-point fusion, for each point feature with its 3D coordinate, the K_{r2p} nearest point can be found in the XYZ map to gather their corresponding features from the RGB feature map. Then a max-pooling followed by a shared MLP is applied to these features to squeeze the incoming feature to the same channel size as the point cloud features. A shared MLP is then utilized to generate the fused feature. This process is visualized in Fig. 4.7b and shown in Eq. 4.1 and Eq. 4.2. Here F_{r_i} is the i_{th} nearest pixel of the RGB feature and F_{r2p} the incoming appearance feature. The concatenation of the point feature F_{point} and the incoming appearance feature F_{r2p} is denoted by the $\oplus$ operator.

$$F_{r2p} = MLP(\max_{i=1}^{K_{r2p}} F_{r_i}) \tag{4.1}$$

$$F_{fused_p} = MLP(F_{point} \oplus F_{r2p}) \tag{4.2}$$

The point-to-pixel fusion follows a similar manner. The lifted XYZ map is again used to find, for each pixel with its XYZ coordinate, the K_{r2p} nearest neighboring points from the point cloud which are used to gather the corresponding point features. The point features are then squeezed by a shared MLP followed by a max-pooling operation to the same channel size as the RGB features to concatenate them to the RGB features. Finally, a shared MLP is used to generate the fused feature. The point-to-pixel fusion process is visualized in Fig. 4.7c) and shown in Eq. 4.1 and Eq. 4.2. Here F_{p_j} is the j_{th} nearest point of the point cloud feature and F_{p2r} the incoming geometry feature. The concatenation of the point feature F_{rgb} and the incoming geometry feature F_{p2r} is denoted by the $\oplus$ operator.

$$F_{p2r} = MLP(\max_{j=1}^{K_{p2r}} F_{p_j}) \tag{4.3}$$

$$F_{fused_r} = MLP(F_{rgb} \oplus F_{p2r}) \tag{4.4}$$

Semantic Instance Segmentation

Following the concatenation of the point and image features, one of the target tasks is semantic instance segmentation. The instance segmentation module is directly borrowed from the predecessor PVN3D. It consists of a semantic segmentation head and a center point voting module, where the first predicts per-point semantic labels and the second learns the per-point offset to the object centers Δx_i to distinguish different instances.

To optimize the segmentation task, a Focal Loss [23] is used. The Focal Loss is a variation on the classical cross-entropy loss. It adds a class balancing term α_i like in the balanced cross entropy loss, as well as a modulating factor $(1-q_i)^\gamma$, which scales the loss depending on the confidence of prediction. Meaning, easy examples with a high confidence will result in a smaller loss, compared to hard examples with low confidence. The hyperparameter γ scales this effect. The vector c_i represents the predicted confidence for the i_{th} point belonging to each class, and l_i is the one-hot representation of the ground truth class label.

$$L_{semantic} = -\alpha(1-q_i)^{\gamma}\log(q_i), \text{ where } q_i = c_i \cdot l_i \tag{4.5}$$

The center offsets Δx_i are optimized with a mean L1-norm of the difference between the predicted center offset Δx_i and the ground truth offset Δx_i^* if the point p_i of the offset Δx_i is part of the instance I using the indicator function $\mathbb{I}$. N represents the number of points in the point cloud.

$$L_{center} = \frac{1}{N}\sum_{i=1}^{N} ||\Delta x_i - \Delta x_i^*||\mathbb{I}(p_i \in I) \tag{4.6}$$

3D Keypoint Estimation

The 3D keypoint estimation module is also borrowed from the predecessor PVN3D. It is very similar to the center offset module but instead of learning the per-point center offset, a per-point offset of_i^j from the i_{th} point cloud point to the j_th keypoint is learned, where N is the number of points in the point cloud and M is the number of keypoints.

$$L_{keypoints} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} ||of_i^j - of_i^{j*}||\mathbb{I}(p_i \in I) \tag{4.7}$$

Loss

At training time, the whole network is simultaneously trained with backpropagation via a multi-task loss which is defined by the weighted sum of all previously discussed task specific losses. The weights $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters and must be tuned manually (in our implementation we use $\lambda_1 = 2, \lambda_2 = 1, \lambda_3 = 1$ similar to the public implementation of FFB6D).

$$L_{multi-task} = \lambda_1 L_{keypoint} + \lambda_2 L_{semantic} + \lambda_3 L_{center} \tag{4.8}$$

Pose Estimation

At test time, some post-processing steps are needed for the outputs of the semantic instance segmentation module and 3D keypoint estimation module. First, the offsets are added to the original point coordinates in order to obtain the actual keypoint proposals. This step is necessary because rather than learning the keypoints itself, the offsets to the keypoints are learned. This is because Peng et al. showed in their PVNet paper [26] that learning the keypoints via offsets combined with voting and clustering, rather than regressing the sparse keypoint locations directly, is more robust to occlusions and truncations. FFB6D follows this recommendation and clusters the keypoints via Mean Shift Clustering (see Sec. 2.5.2). The final 6D pose is estimated using a least-squares-fitting. Here the ground truth object keypoints p_i^* are available in the object coordinate system and the estimated keypoints p_i are predicted in the camera coordinate system in which the 6D pose should be estimated. The least-squares-fitting algorithm calculates the pose parameters $\mathbf{R}$ and $\mathbf{t}$ by minimizing the squared loss shown in Eq. 4.9.

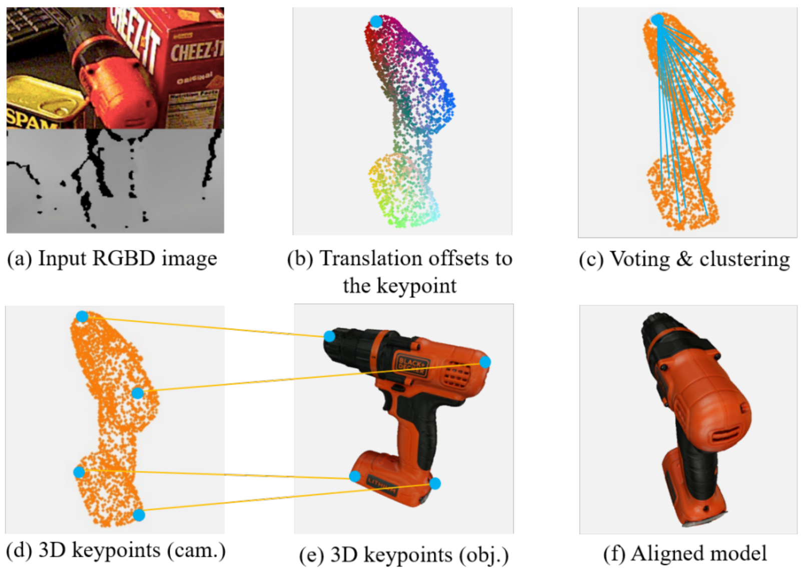


Figure 4.8.: Keypoint Voting and least-squares-fitting as introduced by PVN3D [13], using an RGB-D image (a). For each point in the point cloud, an offset to each keypoint is predicted (b). Using voting and mean shift clustering these proposals are refined into a final keypoint prediction (c). The final 3D keypoints (d) are matched to the keypoints of the known 3D model (e) to estimate the final 6D pose (f) via a least-squares-fitting.

$$L_{lsf} = \sum_{i=1}^{N} ||p_i^* - (\mathbf{R} \cdot p_i + \mathbf{t})||_2^2 \tag{4.9}$$

The entire pose estimation process is visualized in Fig. 4.8.

4.2.2. Multi-View FFB6D

To leverage the advances that FFB6D [12] has made, we decide to keep most of the architecture when extending it to multiple views. The primary difference is in the feature extraction part of the network, fully keeping the semantic instance segmentation module and 3D keypoint estimation module.
The first adaption we make is to add another preprocessing step from the depth map to the point cloud that is injected into the network. For a single view, there is only one point cloud generated from a single depth image. But in our multi-view setup, there are n depth maps coming from n RGB-D cameras. Each depth map is converted into a point cloud in its own camera coordinate system. Instead of passing the point clouds individually to the network, we utilize the fact of known relative camera positions and transform the point clouds into a shared global coordinate system.

To transform the point clouds of each camera into a shared global coordinate system, we have to know the camera poses in a shared reference frame or world coordinate system. To transform a point in the point cloud from one camera coordinate system to the other, we can use a similar transformation detailed in Sec. 2.2.3.
But instead of transforming all point clouds into the world coordinate system, we decided

to transform all point clouds into the first camera coordinate system. This way, the final predicted 6D object poses will be predicted in the first camera coordinate system instead of the world coordinate system, similar to the standard single-view pose estimation results.

To transform the point cloud of the i_{th} camera coordinate system into the first camera coordinate system, we first transform the points into the world coordinate system as described in equation 2.9. Then we use the transformation from world coordinates to the first camera coordinates to finally express the point cloud from the i_{th} camera in the coordinate system of the first camera.

$$^{C1}p = ^{C1}T_W \cdot ^{W}T_{C_i} \cdot ^{C_i}p = (^{W}T_{C1})^{-1} \cdot ^{W}T_{C_i} \cdot ^{C_i}p \tag{4.10}$$

We know the point clouds ^{C1}p and ^{C_i}p in their respective camera coordinate systems and we know the camera poses $^{C1}T_W$ and $^{C_i}T_W$ in world coordinates, but we do not know the transformation $^{C1}T_W$ from the first camera coordinate system to the world coordinate system. However, this is simply the inverse $(^{W}T_{C1})^{-1}$ of the camera pose of the first camera in world coordinates.

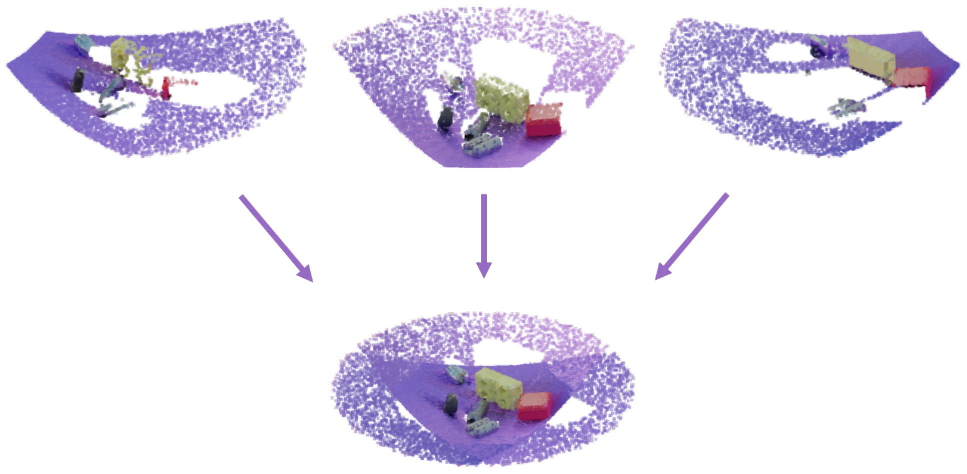
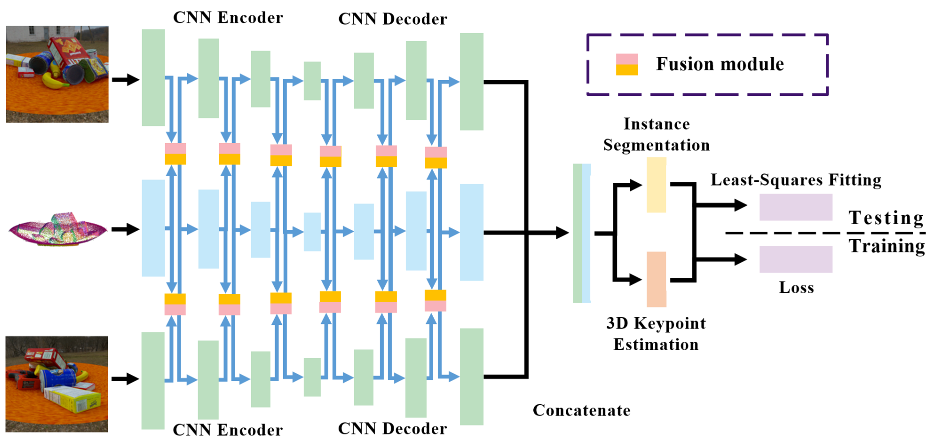


Figure 4.9.: Transformation process to fuse the separate point clouds of each camera into one global point cloud using the known relative camera positions.

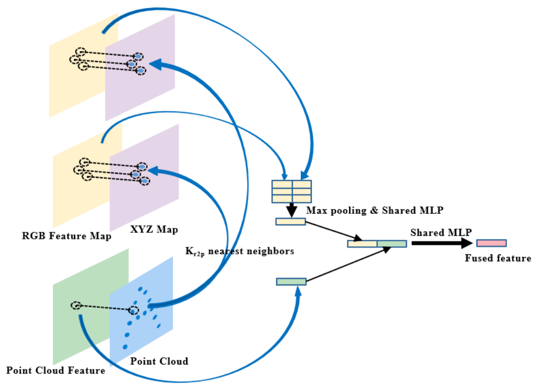
There are multiple advantages for this step. For once, after the fusion of the point clouds, the global point cloud can be passed into the network without any actual network changes to the RandLA-Net [19]. Another advantage is that this step perfectly fuses the geometry information together, without the network having to learn the geometry relation between the different views.

The second adaption we introduce is focused on the image branch of the network. Unlike with the point cloud, we decided against warping all images into one single image, as the images are recorded from drastically different viewing angles and positions. Because the fusion of the multiple viewpoints is already optimally performed by the point cloud branch, we decide to process the images individually by a separate ResNet-34 [11] encoder and PSPNet [38] decoder for each view. To do this efficiently and not grow the number of weights, the separate processing of the n input images is done in a batch-wise manner.

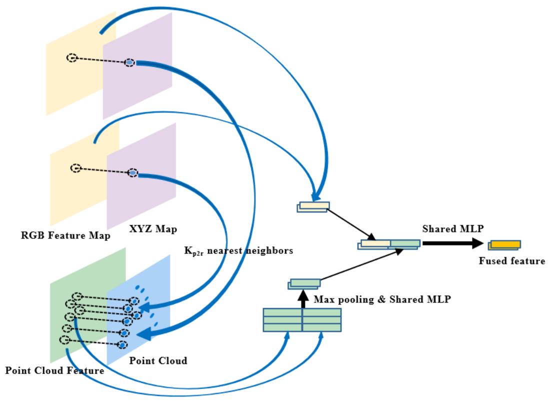
For the single-view case, the input of the ResNet is of shape [b, c, w, h] with b being the batch dimension, c being the number of channels (three for RGB) and w,h being the width and height of the image. For multi-view our input tensor grows to the shape [b, n, c, w, h], where n is the number of views. To now process the images independently in a batch-wise manner we flatten the input tensor to the shape of [b × n, c, w, h]. This way, the images are independently processed by the same CNN branch. This is similar to how the network processes multiple samples of the same batch independently of each other. The advantages of this design are that the number of weights for the CNN branch do not change, and the data is automatically processed in parallel when using a deep learning library such as PyTorch [25]. The output of the PSPNet decoder is consequently of the shape [b × n, f, w, h] where f is the feature dimension of the output. This is then can be easily reshaped to tensor of shape [b, n, f, w, h].



(a) Multi-View adaptation of FFB6D network for $n = 2$ views



(b) Multi-View Pixel-to-Point Fusion for $n = 2$ views



(c) Multi-View Point-to-Pixel Fusion for $n = 2$ views

Figure 4.10.: Overview of Multi-View FFB6D. The network differs from its single-view counterpart in three ways. The first is the fused point cloud of all views. The second is the batch-wise processing of images for each camera. The third change are the adapted fusion modules for Pixel-to-Point (b) and Point-to-Pixel (c) fusion, to support multiple views.

In addition to the changes in the CNN and PCN branches, the fusion between both branches has to be adapted as well to fuse the correct features of the correct views.
This task is not trivial anymore because not all point features and all image features belong to the same view. Nevertheless, we perform the same underlying process of fusing the features via the nearest-neighbor features. However, the K_{r2p} nearest-neighbors and K_{p2r}

nearest-neighbors have to be calculated independently for each view. This is challenging as we do not know which point feature belongs to which view but because of the batch-wise processing of the image features we can find point features that are nearest to the image features for which we know the view. Utilizing a few reshape and broadcasting operations, we implement the same batch-wise processing and concatenating for the fusion of incoming and outgoing image and point cloud features. The process is visualized in Fig. 4.10 for two views.

4.2.3. FFB6D Symmetry Extension

Problem investigation

Generally, 6D Pose Estimation methods perform worse for symmetric objects compared to other non-symmetric objects. This is true for FFB6D [12], but also many other works. To identify the problem, it is insightful to look at the keypoint predictions, which are an essential step in the pose estimation procedure. We have visualized the keypoint predictions of the single-view FFB6D model in Fig. 4.11 for an exemplary scene from the SCAPE 2 dataset, which has many symmetric objects. All the objects in the scene shown in Fig. 4.11

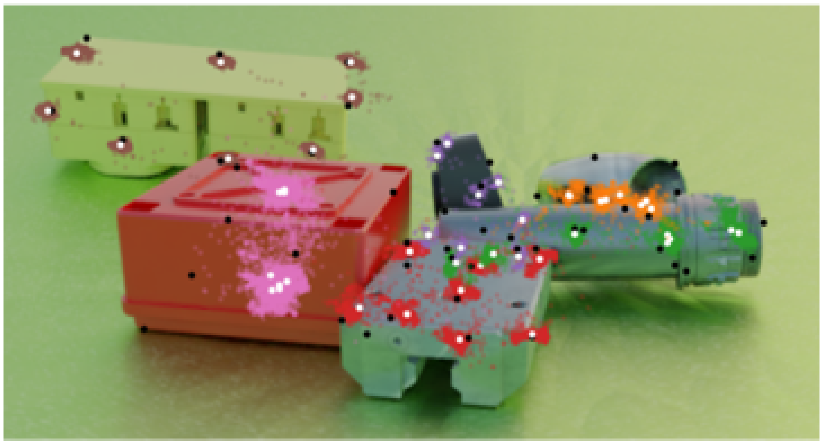


Figure 4.11.: Keypoint proposals and final predictions of the standard FFB6D network on a sample from the SCAPE 2 test set. Predicted keypoints are colored in white. The ground truth keypoints are colored in black. The keypoint proposals for each object are in a separate color.

are symmetric, except for the T-Less_18 object. The T-Less_08 object has one reflectional symmetry, and the keypoint predictions seem to align well with the ground truth keypoints. The keypoint predictions for the BrakingUnit also appear to be accurate, despite the fact that it also has two reflectional symmetries.
But for the T-Less_27, DriveShaft and Tool.Cap objects, which all have a rotational symmetry, the keypoint predictions are far away from their ground truth keypoints. It can be observed that the keypoint proposals and final keypoint predictions cluster on the symmetric axis of the object. This seems to be the reason for poor 6D pose estimates, as the least-squares-fitting will not succeed when the predicted keypoints and the object model keypoints are so different.

The reason for the incorrect keypoint prediction can be found in the keypoint loss function used in FFB6D (see Eq. 4.7) which does not consider symmetries at all. The L1-Loss works well for non-symmetric objects, but the loss function does not consider the ambiguity when

there is a keypoint that has a symmetric counterpart.
For the network, a keypoint that has a symmetric counterpart is indistinguishable from the symmetric keypoint, especially for a texture-less object. It is therefore possible that the network predicts the symmetric keypoint instead of the correct ground truth keypoint, resulting in a large L1-loss. An explanation for the final learned keypoints clustering at the symmetric axis could be, that the network is not able to predict the correct keypoints because of the symmetry ambiguity. But to minimize the loss, the network learns a compromise. Instead of learning the exact keypoint, which sometimes yields a large loss when the incorrect symmetric keypoint is predicted, the network learns keypoints on the symmetric axis, which is equidistant from the correct ground truth keypoints and their symmetric counterparts.

Only a limited number of works tackle the symmetry problem directly. However, none of these works use keypoints to predict the final 6D pose. CosyPose and PoseCNN are one of these works, both implement the object symmetry in the loss function between the ground truth pose and the predicted pose. Vote from the Center [34] is one keypoint-based approach that tries to elevate the symmetry problem by learning more advanced keypoints, however they do not tackle the symmetry problem directly. Another smaller work by Zhang et al. [37] that also learns 3D keypoints, learns the object symmetry and additional symmetric keypoints as an auxiliary task to improve the keypoint estimation. However, this approach also predicts the 6D pose directly and works under the assumption that only one object at the time is estimated, as well as that a ground truth mask of the scene is precomputed.
Although we want to keep the keypoint voting followed by a least-squares-fitting, which in the past has outperformed direct regression of the pose [26, 13, 12], we find this approach of learning symmetric keypoints the most suitable approach to adapt, to improve the results of FFB6D for symmetric objects. However, unlike [37], we utilize the predicted symmetric keypoints directly.

Symmetry-Aware Training

Our approach, keeps the least-squares-fitting and does not require a priori information of the object masks and works for multiple objects at the time. To improve the training, we input the ground truth symmetries of the objects present in the scene into the keypoint loss function seen in Fig. 4.12. We decided against learning these symmetries directly, as they will not be used at test time in our architecture. Also, learning these symmetries while simultaneously using them in a loss function for another task might make the training unstable. By only augmenting the loss function with the symmetries, which is the most critical part for this symmetry problem, we can keep the entire network architecture and keep the same inference procedure at test time.

For the final 6D pose it is irrelevant whether the network predicts the correct 6D pose or a correct symmetric variation because the 6D pose is only defined up to a certain symmetry. So it does not matter if the network predicts the correct ground truth keypoints or their symmetric counterparts. However, it is important that the network predicts either only the correct keypoints or only their symmetric counterparts, so that the least-squares-fitting can be performed correctly. To ensure this, the loss will be minimized over the symmetries of each offset for all object keypoints together, as shown in Eq. 4.11. Here S_o is the set of all symmetries of the object o, including the identity. Sof_i^{j*} is the offset transformed by the symmetry, pointing toward the symmetric keypoint.

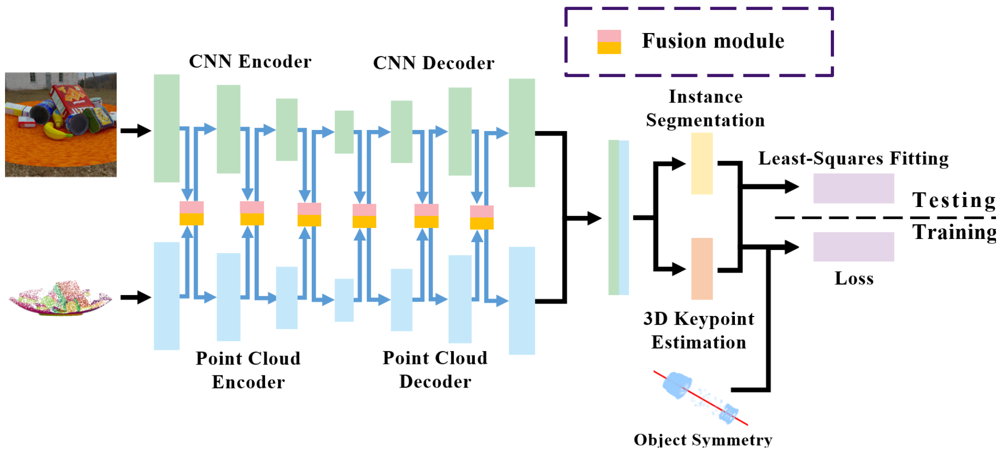


Figure 4.12.: The ground truth symmetry is used at training time to calculate the minimum loss over the keypoints and their symmetric counterparts. This addition prevents keypoints clustering on the symmetric axis because the network can learn either the correct keypoints or their symmetric counterparts directly.

$$L_{keypoints} = \frac{1}{N} \min_{\mathbf{S} \in \mathbf{S}_O} \sum_{i=1}^{N} \sum_{j=1}^{M} ||of_i^j - Sof_i^{j*}|| \mathbb{I}(p_i \in I) \tag{4.11}$$

The loss function also has to handle objects with infinite rotational symmetries, such as the DriveShaft object from the SCAPE 2 dataset. As we cannot minimize over infinite symmetries, we have to discretize the infinite rotational symmetries. In the experiments chapter, we do an ablation on how different discretizations affect the pose accuracy.

Ground Truth Symmetries

The adapted training regime requires ground truth symmetries for all objects in the dataset. This information is neither available in the YCB-Video dataset nor in the SCAPE 2 dataset. There are a couple of open-source algorithms for calculating the symmetry of an object. Often these algorithms are very complex or require input data that is not available in YCB-Video or the SCAPE 2 dataset. Therefore, we implemented a very small gradient optimization algorithm to solve for the symmetry of the object given the mesh of the object. We use the ADD-S metric from Eq. 2.11 as a loss function and optimize the center of the symmetry and the symmetry axis as parameters of a symmetry transformation function f_T. The symmetry transformation function f_T performs either a reflection or rotation along the symmetry axis/plane to generate the symmetric mesh. The goal is to have a minimal loss between the original mesh and the mesh that has been generated by the symmetry transformation, resulting in the correct symmetry axis and symmetry center. Selected symmetries resulting from this approach can be seen in Fig. 4.13 for all objects in the SCAPE 2 dataset. The algorithm for the symmetry generation is shown in Alg. 4.2.0.

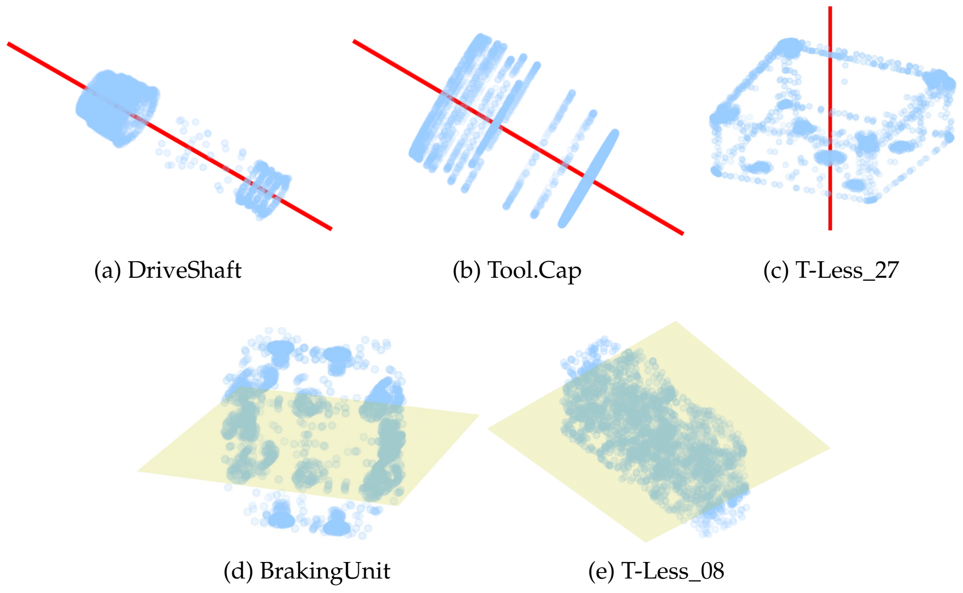


Figure 4.13.: Gradient optimized symmetries in the SCAPE 2 dataset. Shown are the meshes of the symmetric objects in the SCAPE 2 dataset with one selected symmetry each. A red pole indicates rotational symmetry around this axis. A yellow plane indicates reflectional symmetry across that plane.

Algorithm 4.2.0 Algorithm to estimate ground truth symmetries for objects in the SCAPE 2 dataset via gradient descent

```
Input: object_mesh $m$, initial symmetry center $c$, initial symmetry axis $s$, number of iterations $N$
Output: final symmetry axis $s^*$, symmetry center $c^*$
1.  $best\_loss \leftarrow +\infty$
2.  for $n = 0$ to $N-1$
3.      $s_m := f_T(m,s,c)$ // compute symmetric mesh
4.      loss := ADD-S($m,s_m$)
5.      if loss < best_loss
6.          loss := best_loss
7.          s* := s
8.          c* := c
9.      update s and c with SGD
10. return $s^*,c^*$
```

Symmetry Metric Extension

Besides using the ground truth symmetries in training, the ground truth symmetries can be also used in the evaluation. The ADD-S metric has been established as the main metric for evaluating the pose of a symmetric object. However, the comparison of the ADD and ADD-S metric on non-symmetric object shows that the ADD-S is a little more forgiving (because of the min operation) than the ADD metric, resulting in slightly higher ADD-S scores compared to ADD score. Therefore, it is now common to evaluate pose estimation methods on the ADD(S) metric, which combines the ADD and ADD-S metric, by only using the ADD-S metric for preselected symmetric objects.

Because we now have the ground truth symmetries available, we can formulate a more precise metric that works on symmetric and non-symmetric objects. We call this metric the

ADD-Sym metric (see Eq. 4.12). It is a similar extension to the symmetric keypoint loss function that we have introduced in Eq. 4.11. For non-symmetric objects this loss is identical to the ADD metric, but for symmetric objects, the metric is minimized only over the correct object symmetries, making this metric more precise than the ADD-S metric.

$$\text{ADD-Sym} = \frac{1}{|V|} \min_{\mathbf{S} \in \mathbf{S}_O} \sum_{v \in V} ||(\mathbf{R}v + \mathbf{t}) - (\mathbf{R}^*(\mathbf{S}v) + \mathbf{t}^*)||_2 \tag{4.12}$$

Similar to the extension of the loss function, it is crucial to handle the infinite rotational symmetries. We decide to set the maximum number of rotational symmetries to 64 to balance precision with memory consumption.

5. Experiments

In the previous chapter, we have introduced two independent extensions to the FFB6D architecture. Both the multi-view extension and the symmetry-aware training aim to increase the accuracy of 6D Pose Estimation in very complex scenes. To evaluate these changes, we utilize the datasets described in Sec. 4.1. The evaluation of our multi-view extension is done on the SCAPE YCB, SCAPE 2, SCAPE YCB2 and YCB-Video [35] dataset. The symmetry-aware training is also evaluated on the SCAPE 2 dataset and on the YCB-Video dataset, but not the SCAPE YCB(2) dataset, as those do not contain any symmetric objects.
The results of our novel architecture are compared against the standard single-view FFB6D model, as well as a multi-view variation of PVN3D [8] on the same input data. Additionally, on the YCB-Video dataset we evaluate against CosyPose [20]. Here the multi-view variations of PVN3D and FFB6D get RGB-D data as their input, while the CosyPose model only accepts RGB data.

Following PoseCNN [35], DenseFusion [31], PVN3D [13] and FFB6D [12] we report the ADD-S-AUC [35] and ADD(S)-AUC [35] score for all objects individually as well as an average over all objects. For a fairer comparison of the single-view and multi-view architecture, we evaluate the single-view approach on all views that the multi-view approach gets to see and rank them per scene by taking the average ADD(S) error for all objects in the scene. The view with the lowest average ADD(S) error is denoted the best single-view. Similarly, the view with the highest average ADD(S) error is denoted the worst single-view. For the final comparison, we compute the average over all n views per scene and denote this as average single-view.

5.1. SCAPE YCB

The SCAPE YCB dataset is the most similar dataset of our synthetic multi-view datasets to the public YCB-Video dataset. It consists of the same objects as found in the YCB-Video dataset. But we only use a subset of the objects that are non-symmetric. To evaluate our symmetry advances, we exclusively use the separate SCAPE 2 dataset. Although the number of different objects is lower in the SCAPE YCB dataset compared to the YCB-Video dataset, the number of objects per scene is generally higher than in the YCB-Video dataset. Also, the objects are way more cluttered in the center of the scene, which results in a higher number of occlusions compared to the YCB-Video dataset, which should highlight the advances of a multi-view approach better.

The ADD-S and ADD(S) results for the SCAPE YCB can be seen in Tab. 5.1. We evaluate our multi-view model against the single-view FFB6D [12] using the ranked views. Furthermore, we compare with the single-view and multi-view variation of PVN3D [13, 8].

Although FFB6D claims impressive performance on LINEMOD Occlusion [3], which is

	PVN3D				FFB6D							
	SV (Avg)		MV (Avg)		SV (Avg)		SV (Best)		SV (Worst)		MV (Avg)	
	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
banana	68.74	57.00	95.13	90.37	69.51	59.44	85.85	76.04	50.66	42.46	**95.72**	**91.48**
cracker box	95.99	94.87	96.87	96.49	95.96	94.93	95.88	94.85	96.07	95.10	**97.28**	**97.28**
gelatin box	62.82	55.37	94.11	89.67	65.91	58.54	82.58	74.01	47.59	41.77	**95.00**	**91.67**
master chef can	93.21	88.17	98.22	97.61	93.29	87.31	95.96	90.80	90.24	83.98	**98.43**	**97.87**
mustard bottle	86.38	79.50	97.59	96.52	87.16	81.37	93.56	88.40	80.19	73.10	**97.89**	**97.10**
potted meat can	81.53	75.30	97.34	95.82	84.36	77.42	91.53	85.47	75.31	66.87	**97.56**	**96.23**
power drill	90.92	87.09	96.94	95.79	91.36	88.25	94.60	91.82	87.16	83.06	**97.71**	**97.17**
pudding box	78.14	71.07	97.45	95.31	78.77	72.97	91.01	85.37	65.49	59.60	**97.68**	**96.22**
sugar box	88.72	83.66	97.97	97.15	89.15	85.42	94.61	91.41	82.95	79.49	**98.39**	**97.82**
tomato soup can	79.93	73.44	97.77	96.18	80.96	74.93	91.38	85.34	68.51	63.03	**97.97**	**96.72**
tuna fish can	68.18	58.22	96.68	91.59	69.12	58.93	85.40	74.04	50.60	43.06	**96.79**	**91.88**
ALL	81.32	74.88	96.91	94.77	82.32	76.32	91.12	85.23	72.25	66.50	**97.31**	**95.56**

Table 5.1.: Quantitative comparison between PVN3D single-view (SV), PVN3D multi-view (MV), FFB6D single-view and FFB6D multi-view on the SCAPE YCB dataset. The ADD-S and ADD(S)-AUC are reported. The best results for ADD-S-AUC and ADD(S)-AUC are highlighted in bold.

another single-view dataset that offers scenes with a high number of occlusion, it can be observed that the single-view version of FFB6D struggles on the SCAPE-YCB dataset. The average accuracy of the single-view FFB6D model for all scenes in the test set is drastically lower compared to our multi-view FFB6D model. The accuracy of our multi-view model on the ADD(S) metric is almost 20% better compared to the average single-view results and for the ADD-S metric, we achieve almost 15% better results for an AUC<0.1m. The separation into best and worst view also gives a great insight. For the worst view where objects are most occluded, the overall accuracy drops to ~66% for the ADD(S) metric. A clear outlier is the cracker box with ~95% accuracy on the ADD(S) metric. It is the largest object in the SCAPE YCB dataset and consequently it is rarely heavily occluded by other objects which explains the high accuracy. Furthermore, it can be observed that the single-view results of the best view fall short of the multi-view results, with ~85% accuracy for the single-view model and ~95% accuracy for our multi-view model on the ADD(S) metric. This clearly shows that our approach does more than focus on the view where most objects are best visible, but it actually fuses information from all views to achieve better results than just from a single view.

Tab. 5.1 also shows that the single-view FFB6D model is able to outperform the older and simpler network architecture of the single-view PVN3D model on the average single view. Consequently, we expect our multi-view model to be able to outperform the multi-view variation of PVN3D. Indeed, we are able to outperform the multi-view variation of PVN3D with our multi-view adaptation of FFB6D. However, the accuracy difference between the multi-view PVN3D and the multi-view FFB6D model is smaller than the accuracy difference between the single-view PVN3D and single-view FFB6D model. While single-view FFB6D improves about 1.5% over PVN3D on the average single-view for the overall ADD(S) metric, the multi-view FFB6D model is only able to outperform multi-view PVN3D by about 0.8% on the ADD(S) metric. We argue that this improvement is still a big improvement because the pose accuracy for the multi-view case is almost saturated for some objects, which makes larger improvements harder.

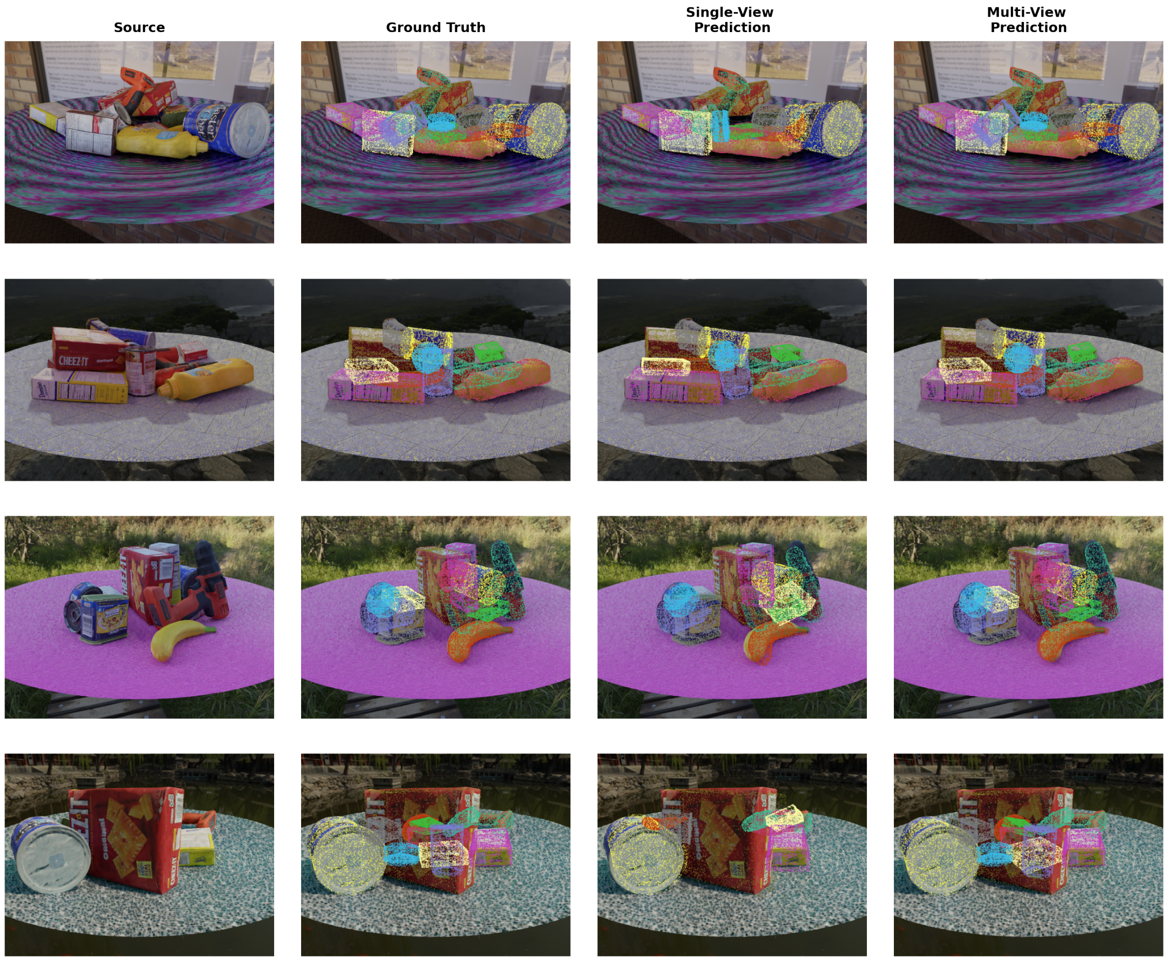


Figure 5.1.: Qualitative comparison between single-view FFB6D and multi-view FFB6D. Shown are four randomly selected samples from SCAPE YCB test set. The object meshes are projected into the image space using the camera intrinsics and ground truth poses/predicted single-view poses/predicted multi-view poses. A good alignment between the projected meshes and the shown objects in the image indicates an accurate 6D pose.

The qualitative results confirm the quantitative results from Tab. 5.1. Fig. 5.1 shows a random selection from the SCAPE YCB test set. The meshes are projected into the image plane using the camera intrinsics and the predicted/ground truth poses. Qualitatively, the single-view predictions are already very good. Here, we will highlight objects and scenes where the multi-view model outperforms the single-view model.
In the first row, the pose of the power drill (green mesh) is predicted much more accurately for the multi-view model. Also, the tomato soup can (blue mesh) is not detected at all using the single-view model, while the multi-view model predicts the pose almost perfectly.
In the second row, the multi-view and single-view predictions are very similar. A big difference can be seen for the pudding box (light yellow mesh) for which only a tiny corner is visible.
The predictions of the multi-view model for the scene shown in the third row are much better compared to the single-view predictions. Not only are the predicted poses for the banana

(red mesh), power drill (green mesh) and sugar box (purple mesh) much more accurate, but the multi-view model also predicts the pose of the pudding box (light yellow mesh) correctly, which the single-view model completely displaces. Additionally, the multi-view model is able to predict the mustard bottle (pink mesh) correctly, which is not visible from this view. The fourth row highlights this advantage of the multi-view model even further. The multi-view model is able to correctly predict the pose of the banana (red mesh), gelatin box (light yellow mesh), pudding box (green mesh), tuna fish can (light blue mesh) and tomato soup can (blue mesh), all of which are fully occluded by the large cracker box in the foreground of the view.

Precision Metric

To evaluate how close our multi-view approach is to predicting the object pose below the error threshold for successful robot manipulation, we evaluate our multi-view FFB6D model and also the single-view variation on the ADD(S)<2cm metric. We compare against the predecessor PVN3D as well.

	PVN3D				FFB6D							
	SV (Avg)		MV (Avg)		SV (Avg)		SV (Best)		SV (Worst)		MV (Avg)	
	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
banana	64.47	51.80	96.88	91.21	67.15	56.63	84.05	73.98	47.72	39.33	**97.20**	**92.57**
cracker box	98.64	97.92	**98.92**	**98.80**	98.72	98.32	98.68	98.44	98.68	98.44	**98.92**	98.76
gelatin box	63.95	51.48	96.16	89.37	66.87	53.92	83.93	68.71	47.96	37.77	**96.52**	**92.09**
master chef can	95.16	88.49	**99.76**	**99.60**	95.36	87.49	98.20	91.01	92.09	84.17	**99.76**	99.28
mustard bottle	87.01	78.70	**99.52**	98.68	88.33	81.14	94.96	88.97	80.82	71.82	99.40	**98.76**
potted meat can	83.41	73.46	**99.24**	97.92	86.37	75.62	93.65	84.77	76.98	63.55	99.08	**97.76**
power drill	91.97	88.85	99.16	98.80	92.81	90.13	96.28	93.76	87.65	84.41	**99.24**	**99.04**
pudding box	78.90	69.58	**99.40**	97.52	80.34	72.10	93.05	84.65	66.91	58.15	99.28	**97.80**
sugar box	89.53	83.77	**100.00**	**99.76**	90.37	86.13	96.16	92.81	83.81	80.10	99.96	99.68
tomato soup can	81.18	71.10	**99.56**	**98.68**	82.37	72.14	93.05	82.61	69.78	59.95	99.48	98.48
tuna fish can	69.66	48.08	**98.72**	**90.97**	70.66	49.24	87.65	62.83	51.44	36.45	98.64	90.01
ALL	82.17	73.02	98.85	96.48	83.58	74.81	92.70	83.87	73.08	64.92	**98.86**	**96.75**

Table 5.2.: Quantitative comparison between PVN3D single-view (SV), PVN3D multi-view (MV), FFB6D single-view and FFB6D multi-view on the SCAPE YCB dataset for the precision metric. The ADD-S<2cm and ADD(S)<2cm metrics are reported. The best results for the ADD-S<2cm and ADD(S)<2cm metric are highlighted in bold.

In Tab. 5.2 we compare our multi-view model against the multi-view variation of PVN3D and the average, best and worst view from the FFB6D single-view model. Similarly to the full ADD(S)-AUC and ADD-S-AUC results the multi-view FFB6D model outperforms the single-view model on the average and worst view and even the best view. However, the comparison between multi-view PVN3D and multi-view FFB6D is much closer. While the single-view variation of FFB6D is able to outperform the single-view variation of PVN3D by about 1.8% on the ADD(S)<2cm metric, the multi-view results for FFB6D are only approximately 0.25% better than the multi-view results on the ADD(S)<2cm metric. For many objects like the cracker box, master chef can, sugar box, tomato soup can and tuna fish can the multi-view variation of PVN3D is able to outperform the FFB6D results on the ADD(S)<2cm metric by a small amount. Similarly to the standard ADD(S)-AUC results, we argue that the accuracy is already very saturated, which makes improvements very hard.

To investigate this issue in more detail, Fig 5.2 shows the AUC of multi-view FFB6D and multi-view PVN3D for an ADD(S) error from 0m to 0.05m for all objects combined, with the 2cm threshold marked. In Appendix A, we list the AUCs for all objects separately. Fig.

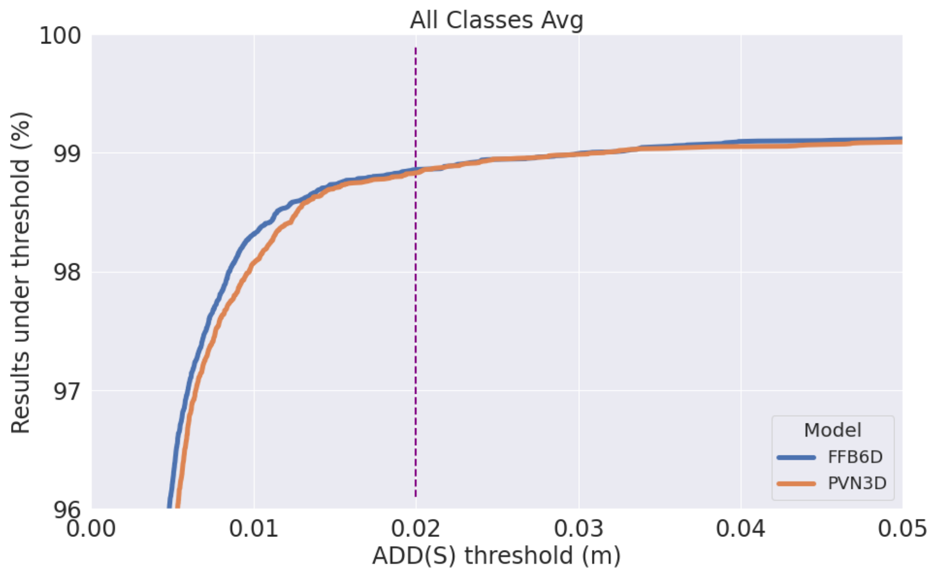


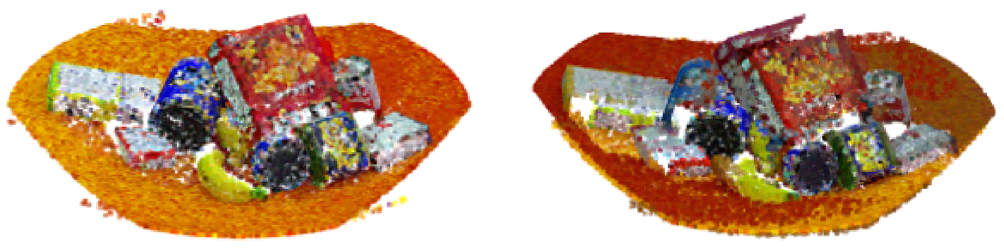
Figure 5.2.: Comparison of the full ADD(S)-AUC for multi-view FFB6D and multi-view PVN3D. The 2cm threshold is highlighted separately.

5.2 shows that the multi-view FFB6D model generally predicts more objects with higher precision below the error threshold of 1.5cm compared to PVN3D. However, at the 2cm precision threshold (which represents the ADD(S)<2cm metric) multi-view PVN3D and multi-view FFB6D converge to approximately perform equally well. With a growing precision threshold above 4cm multi-view FFB6D begins to outperform PVN3D again, which explains the improved performance on the full ADD(S)-AUC metric.

Noisy Fusion

Since our multi-view approach relies heavily on the known relative camera positions for the fusion of the depth information of each camera to a global point cloud, we wanted to evaluate how well our approach could handle situations where the relative camera position are not perfectly known. This is especially relevant for a real-world setup, where it is difficult to perfectly place the cameras at the assumed positions. For this, a variation of the SCAPE YCB dataset exists where some noise of up to 2.5cm is added to the camera positions. The resulting fused point cloud is not properly aligned, as the assumed camera positions used to project the point cloud into a global coordinate system do not reflect the actual positions. The effects of these noisy camera positions can be seen in Fig. 5.3 for the first scene in the SCAPE YCB dataset.

Fig. 5.3 shows that even a small offset between the assumed and actual camera positions can result in objects overlapping or being split apart. Nevertheless, the objects are still recognizable but the estimation of the exact pose is much more difficult because of the imprecise representation of the objects in the point cloud.



(a) Accurate Camera Positions (b) Noisy Camera Positions

Figure 5.3.: Comparison between a point cloud generated using the accurate camera positions and a point cloud generated using noisy camera positions from the SCAPE YCB dataset. The effects of the noisy camera positions can be best identified by the large cracker box object in the scene center.

In Tab. 5.3 we compare the results of single-view and multi-view FFB6D against the single-view and multi-view results of PVN3D on noisy SCAPE YCB. Although the noisy camera positions only affect the fusion directly, we also train the single-view models on the noisy dataset because the single-view viewing angles are also affected by the random shift of the cameras.

	PVN3D				FFB6D							
	SV (Avg)		MV (Avg)		SV (Avg)		SV (Best)		SV (Worst)		MV (Avg)	
	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
banana	67.60	55.21	94.28	88.11	69.11	58.17	84.87	73.72	50.66	42.34	**94.90**	**89.05**
cracker box	94.81	92.56	95.64	94.29	95.49	93.88	95.63	94.24	95.56	94.01	**95.92**	**94.67**
gelatin box	62.39	54.52	93.80	87.70	63.72	56.17	80.37	72.09	45.09	38.39	**94.28**	**88.99**
master chef can	92.39	85.92	97.29	**96.10**	92.91	86.75	95.36	89.90	90.41	83.54	**97.31**	95.93
mustard bottle	85.28	77.80	96.83	**94.83**	86.05	79.28	92.04	85.99	79.09	71.61	**96.95**	94.73
potted meat can	80.15	72.95	**96.60**	**94.31**	82.05	74.86	91.23	84.13	70.06	62.03	96.27	94.30
power drill	89.98	84.90	96.27	94.41	90.75	86.32	94.11	90.44	86.41	81.22	**96.64**	**95.17**
pudding box	76.83	69.14	96.77	93.59	78.12	70.85	89.07	81.94	65.06	58.39	**96.96**	**94.38**
sugar box	87.43	81.49	97.14	95.34	88.55	83.70	93.63	89.10	83.19	78.01	**97.35**	**95.82**
tomato soup can	79.35	71.75	96.98	94.45	80.56	73.27	92.04	84.63	68.14	61.29	**97.08**	**94.74**
tuna fish can	67.49	56.31	96.20	**89.86**	68.98	58.01	85.34	72.69	50.23	41.90	**96.26**	89.14
ALL	80.34	72.96	96.16	93.00	81.48	74.66	90.33	83.53	71.26	64.80	**96.36**	**93.35**

Table 5.3.: Quantitative comparison between PVN3D single-view (SV), PVN3D multi-view (MV), FFB6D single-view and FFB6D multi-view on the SCAPE YCB (Noisy) dataset. The ADD-S and ADD(S)-AUC are reported. The best results for ADD-S-AUC and ADD(S)-AUC are highlighted in bold.

Similar to the results using accurate poses, single-view FFB6D clearly outperforms single-view PVN3D. But although the point cloud fusion is highly affected by the noisy camera positions, the multi-view FFB6D model is able to outperform the average, best and worst view for all objects. The multi-view FFB6D model is also able to outperform the multi-view PVN3D model. However, for some objects, the PVN3D model is better than the FFB6D model.

Fig. 5.4 shows a quantitative comparison of FFB6D with FFB6D on noisy data. For the multi-view models each object has one data point belonging to the ADD(S)-AUC accuracy for that object. Each single-view model also has one data point belonging to the ADD(S)-AUC accuracy on the average view. The error bars for the single-view results indicate the ADD(S)-AUC accuracy for the best and worst view respectively. The multi-view model evaluated on the accurate data outperforms the model evaluated on noisy data for all objects. However, the difference is marginal. The noisy multi-view FFB6D model is still able to outperform the single-view model on accurate data for all objects except for the cracker box where the best single-view model achieves an ADD(S)-AUC accuracy of 94.85% while the noisy multi-view model achieves only 94.67%.

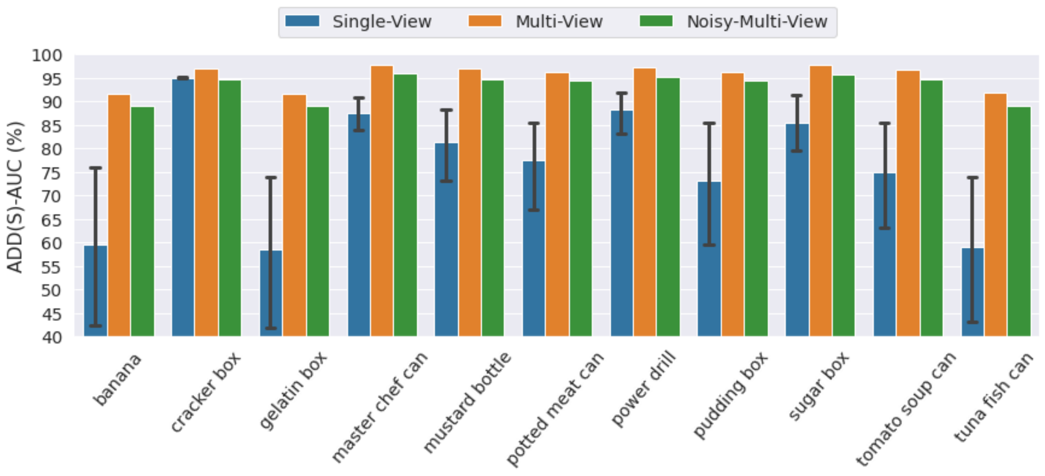


Figure 5.4.: Quantitative comparison of single-view FFB6D on accurate camera poses and multi-view FFB6D on accurate and noisy camera poses respectively. Results taken from Tab. 5.1 and Tab. 5.3.

Fig. 5.5 confirms the small difference between the multi-view pose estimates on data with accurate camera positions and the multi-view pose estimates on data with noisy camera positions. The first row shows an example where the pose difference between the model using accurate camera positions and the model using noisy camera positions is large, especially for the power drill (green mesh) and the tuna fish can (light blue mesh) . The second row shows a similar result. While the model utilizing the accurately fused point cloud predicts the tuna fish can (light blue mesh) and mustard bottle (pink mesh) perfectly, the model only having access to the imperfectly fused point cloud predicts poses that are much worse. However, the predictions shown in the third and fourth rows are almost identical for the multi-view model using accurate camera positions and the multi-view model using noisy camera poses. Overall, the qualitative results confirm that our multi-view approach is robust towards imperfectly placed cameras.

Object Size

In our standard, precision and noisy experiments we observe that the pose accuracy of larger objects, especially the cracker box, do not benefit as much from the multi-view setup as smaller objects like the gelatin box. This intuitively makes sense. Larger objects will be better visible from different views, while it is very likely that a smaller object gets occluded by a bigger object. This also correlates well with the evaluation of the best and worst single

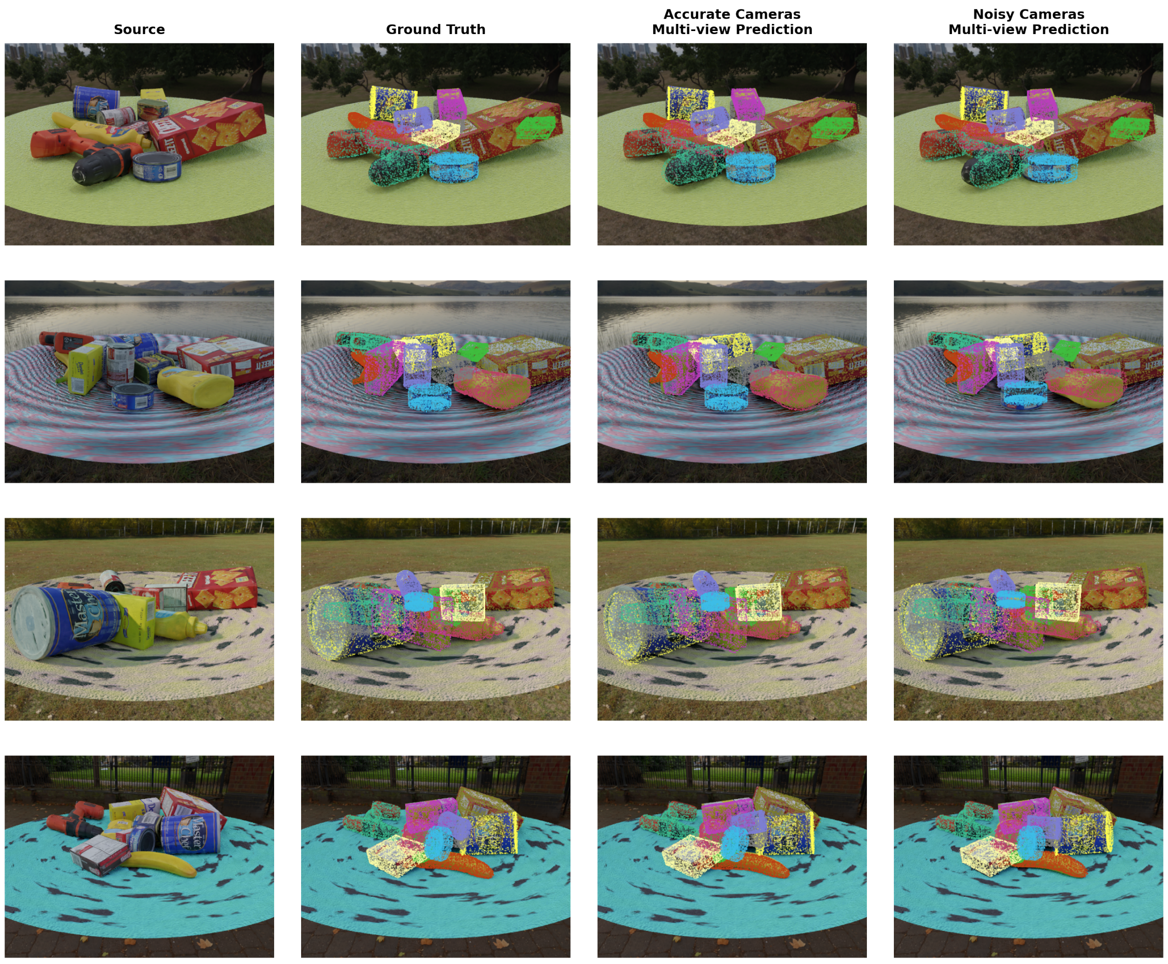


Figure 5.5.: Qualitative comparison between multi-view FFB6D using accurate camera positions and multi-view FFB6D using noisy camera positions. Shown are four randomly selected samples from SCAPE YCB test set. The object meshes are projected into the image space using the camera intrinsics and 6D poses. A good alignment between the projected meshes and the shown objects in the image indicates an accurate 6D pose.

view. While large objects like the cracker box, master chef can and sugar box have a smaller variation across views, smaller objects like the banana, gelatin box, or tuna fish can have very large variation across views.

5.2. SCAPE 2

The SCAPE 2 dataset is more simple than the SCAPE YCB dataset, as it only contains at most six objects, which results in far fewer occlusions. However, all of the objects in this dataset are texture-less, which makes it harder for the network to learn visual features on them. But the biggest challenge in this dataset are the symmetries of the objects because all objects in the SCAPE 2 dataset except for the T-Less_18 object have at least one symmetry. Tab. 5.4 shows the quantitative results on the SCAPE 2 dataset. Similarly to the SCAPE YCB dataset we compare our multi-view architecture against the best, worst, and average single-view of the single-view FFB6D [12] model. Additionally, we compare against single-view PVN3D

[13] and its multi-view variation [8].

	PVN3D				FFB6D							
	SV (Avg)		MV (Avg)		SV (Avg)		SV (Best)		SV (Worst)		MV (Avg)	
	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
Tool.Cap	90.49	90.49	98.57	98.57	90.27	90.27	97.66	97.66	79.77	79.77	**98.62**	**98.62**
DriveShaft	92.71	92.71	98.63	98.63	92.69	92.69	97.80	97.80	85.86	85.86	**98.71**	**98.71**
BrakingUnit	91.44	91.44	97.94	97.94	91.74	91.74	97.50	97.50	84.21	84.21	**98.19**	**98.19**
T-Less_18	90.30	86.85	98.41	**97.58**	89.72	86.55	97.83	96.10	77.64	73.01	**98.45**	97.56
T-Less_08	98.43	98.43	98.71	98.71	98.13	98.13	98.20	98.20	98.12	98.12	**99.09**	**99.09**
T-Less_27	**94.46**	**94.46**	94.17	94.17	92.91	92.91	94.06	94.06	91.71	91.71	93.55	93.55
ALL	92.93	92.40	97.74	97.60	92.58	92.05	97.17	96.89	86.22	85.45	**97.77**	**97.62**

Table 5.4.: Quantitative comparison between PVN3D single-view (SV), PVN3D multi-view (MV), FFB6D single-view and FFB6D multi-view on the SCAPE 2 dataset. The ADD-S and ADD(S)-AUC are reported. Symmetric objects are highlighted in bold. The best results for ADD-S-AUC and ADD(S)-AUC are also highlighted in bold.

The performance difference between the FFB6D multi-view architecture and the FFB6D single-view architecture is smaller compared to the SCAPE YCB dataset. In the SCAPE YCB dataset, the multi-view architecture outperforms the single-view architecture on all objects for the ADD-S and ADD(S) metric by a large margin. On the SCAPE 2 dataset, however, the performances of both methods are closer. Especially for the larger objects like T-Less_08 and T-Less_27 where the average ADD-S and ADD(S) score between the single-view and the multi-view model are very close. For the T-Less_27, the single-view architecture is even able to outperform the multi-view architecture for the best view. A reason for this might be that the multi-view architecture gets features of the whole object while the single-view architecture only gets features from one side of the object, this might make the symmetry problem even worse. For the smaller objects, however, the FFB6D multi-view architecture is able to outperform the single-view method. This supports the results from the SCAPE YCB dataset, that the multi-view method is especially effective for smaller objects.

The results between FFB6D and PVN3D are very close. Overall, our multi-view FFB6D model outperforms the multi-view PVN3D architecture. However, the difference is very small. A significant factor for this is the rotational symmetric T-Less_27 object. The single-view PVN3D model achieves the best accuracy on this object, followed by the multi-view PVN3D model, our multi-view FFB6D model and the original single-view model perform much worse on this object. For all other objects, FFB6D and our multi-view variation outperforms the PVN3D models by a small margin. It appears that the FFB6D model and consequently our multi-view extension is more susceptible to the symmetry problem than the PVN3D model.

The qualitative results seen in Fig. 5.6 support the quantitative results of Tab. 5.4. Since the scenes in the SCAPE 2 dataset are less cluttered than the scenes in the SCAPE YCB dataset, it barely happens that an object is not visible from one single view. The first row of Fig. 5.6 shows pretty similar results between the single-view and multi-view results. However, the pose of the T-Less_27 object (gray mesh) is much worse for the multi-view predictions. Row two shows one of the few scenes where an object is fully occluded by another object. The multi-view architecture is able to estimate the pose of this object, while the single-view

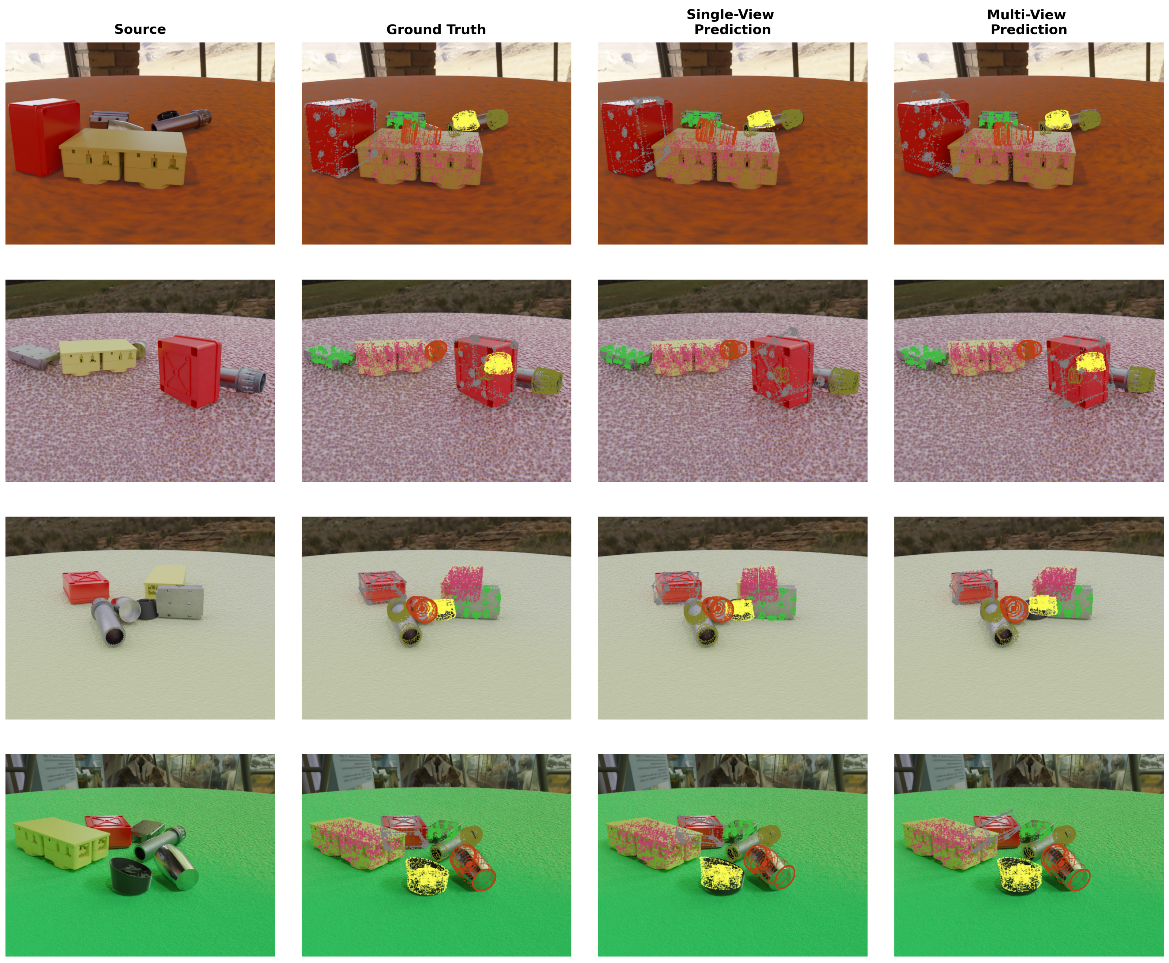


Figure 5.6.: Qualitative comparison between single-view FFB6D and multi-view FFB6D. Shown are four randomly selected samples from SCAPE 2 test set. The object meshes are projected into the image space using the camera intrinsics and ground truth poses/predicted single-view poses/predicted multi-view poses. A good alignment between the projected meshes and the shown objects in the image indicates an accurate 6D pose.

model is not able to do this. All other objects are roughly equally well estimated for both the single-view and multi-view model. The T-Less_27 object (gray mesh) is again estimated poorly with the multi-view model but the single-view model also did not estimate the pose well, In the third row all objects are estimated pretty well including the T-Less_27 object (gray mesh). The T-Less_08 (pink mesh) object, which is very far away from the camera recording this view, is estimated a little more accurately with the multi-view model because it has access to another view where the object is closer to the camera. The fourth row also shows a simple scene where all objects are visible. The single-view model again predicts the pose of the T-Less_27 object (gray mesh) much more accurately. However, the multi-view model estimates the pose of the non-symmetric T-Less_18 (yellow mesh) object much better.

Precision Metric

Additionally, to the ADD(S)-AUC and ADD-S-AUC metric we evaluate our multi-view model on the precision ADD(S)<2cm and ADD-S<2cm metric. We compare our multi-view

FFB6D model against its single-view variation as well as PVN3D. Tab. 5.5 shows the results.

	PVN3D				FFB6D							
	SV (Avg)		MV (Avg)		SV (Avg)		SV (Best)		SV (Worst)		MV (Avg)	
	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
Tool.Cap	91.93	91.93	**100.00**	**100.00**	91.89	91.89	99.52	99.52	81.06	81.06	**100.00**	**100.00**
DriveShaft	93.61	93.61	**100.00**	**100.00**	93.80	93.80	99.40	99.40	86.33	86.33	99.92	99.92
BrakingUnit	92.61	92.61	**99.64**	**99.64**	93.09	93.09	99.04	99.04	85.25	85.25	99.52	99.52
T-Less_18	90.73	85.73	**99.40**	**99.28**	90.45	86.13	98.92	98.68	77.58	68.82	**99.40**	99.16
T-Less_08	99.92	99.92	**100.00**	**100.00**	99.92	99.92	**100.00**	**100.00**	99.88	99.88	**100.00**	**100.00**
T-Less_27	98.72	98.72	**99.84**	**99.84**	98.88	98.88	99.64	99.64	97.72	97.72	99.76	99.76
ALL	94.58	93.75	**99.81**	**99.79**	94.67	93.95	99.42	99.38	87.97	86.51	99.77	99.73

Table 5.5.: Quantitative comparison between PVN3D single-view (SV), PVN3D multi-view (MV), FFB6D single-view and FFB6D multi-view on the SCAPE 2 dataset for the precision metric. The ADD-S<2cm and ADD(S)<2cm metrics are reported. Symmetric objects are highlighted in bold. The best results for the ADD-S<2cm and ADD(S)<2cm metric are also highlighted in bold.

Similar to the non-precision metric, multi-view FFB6D outperforms the average single-view of FFB6D. The results for the multi-view model are very good. The multi-view model even reaches 100% accuracy for the Tool.Cap and T-Less_08 object. This means that the multi-view FFB6D model is always able to predict those objects with an accuracy that is satisfactory for object manipulation. But also for other objects, the precision is in the high 99% resulting in an overall ADD(S)<2cm score of 99.73%. The best view of the single-view model can match these extraordinary results of 100% accuracy for the T-Less_08 object. But on the average single-view the single-view FFB6D model does not reach 100% for any object, but it still reaches a very respectable ADD(S)<2cm score of 93.95%. Similarly to the SCAPE YCB dataset, the multi-view FFB6D model struggles to improve over the multi-view PVN3D results. In fact, PVN3D outperforms FFB6D averaged over all objects for the ADD(S)<2cm metric. However, the accuracy for the Tool.Cap and T-Less_08 objects is fully saturated, which makes a meaningful quantitative comparison very challenging.

Symmetry Extension

The single-view and multi-view results with FFB6D clearly show that the network drastically underperforms for symmetric objects, especially for the T-Less_27 object. A symmetry-aware training is therefore highly relevant.

Using the symmetry-aware training introduced in Sec. 4.2.3, we first evaluate the effects qualitatively in Fig. 5.7 by first analyzing how our symmetry-aware training benefits the final keypoint predictions.

In Fig. 5.7, we compare the keypoint predictions of a single-view FFB6D model using our new symmetry-aware training and a single-view FFB6D model using the standard training procedure. Fig. 5.7b demonstrates that the symmetry-aware training succeeds in producing better keypoint proposals. Rather than having keypoints cluster on the symmetric axis, the new predicted keypoints are well distributed across the object. Additionally, Fig. 5.7 gives an insight into the improved loss function. The ground truth keypoints colored in black in Fig. 5.7b are different from the ground truth keypoints in Fig. 5.7a. This is because only those ground truth keypoints are visualized that have the smallest distance to the predicted keypoints, which are the only keypoints that contribute to the loss from Eq. 4.11. For the

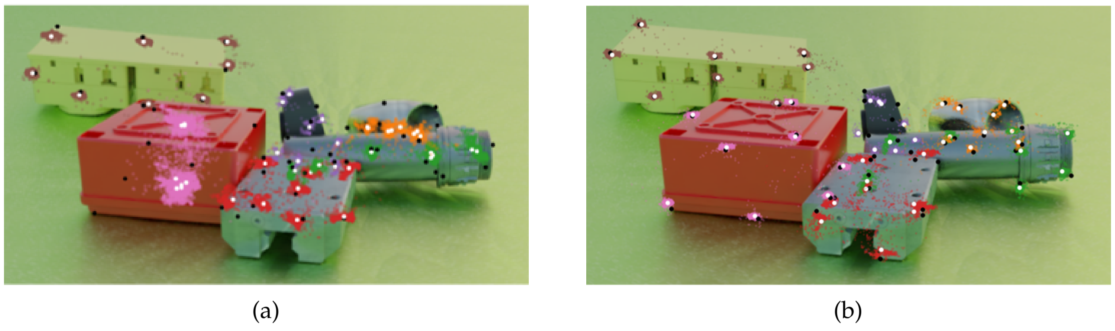


Figure 5.7.: Qualitative results of the keypoint estimation on SCAPE 2. Predicted keypoints are colored in white. The ground truth keypoints are colored in black. The pre-clustered keypoint proposals for each object are highlighted in a separate color for each object. The keypoints that were trained in a symmetry-aware manner, are well distributed across the whole object (b), while the keypoints that were trained with the standard loss function, cluster around the symmetry axis.

infinite rotational objects, we discretize the symmetries into four discrete symmetries to utilize their symmetries in training.

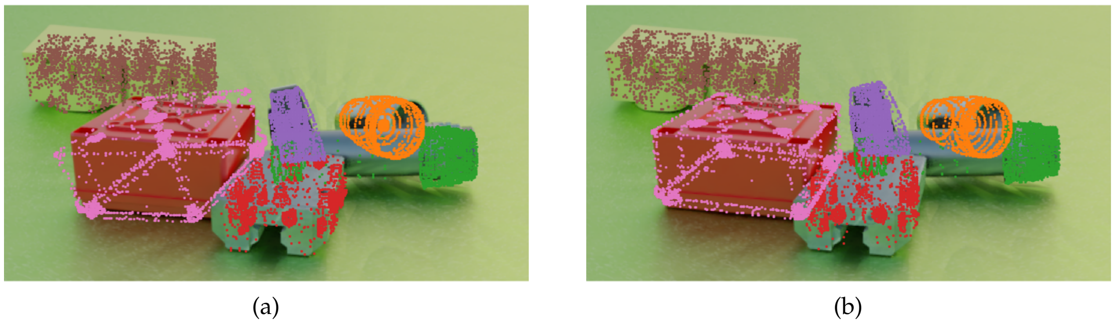


Figure 5.8.: Qualitative results on SCAPE 2. The object meshes are projected into the image space with the estimated poses and the camera intrinsics. A good object mesh alignment indicates a good pose estimate. Symmetry-aware training (b) shows large improvements over the standard training (a), especially for the red T-Less_27 object.

But the improved results are visible not only in the keypoint predictions but also in the final pose predictions. Fig. 5.8 shows the mesh projections into the image space based upon the predicted 6D pose with and without symmetry-aware training. Especially the pose for the T-Less_27 object is much better. But also the rotational symmetric Tool.Cap object and DriveShaft object align better with the projected mesh. The reflectional symmetric BrakingUnit and the T-Less_08 object do not benefit as much from the symmetry-aware training as the rotational symmetric objects, but their pose predictions improve slightly too.

Besides showing great qualitative improvements over the standard training, the quantitative results confirm the improvement with symmetry-aware training. In Tab. 5.6 we compare the improved pose estimation performance for single-view and multi-view on the ADD-S, ADD(S) metric and our in Eq. 4.12 introduced ADD-Sym metric.

	Standard						Symmetry-Aware					
	Single-View (Avg)			Multi-View (Avg)			Single-View (Avg)			Multi-View (Avg)		
	ADD-S	ADD(S)	ADD Sym	ADD-S	ADD(S)	ADD Sym	ADD-S	ADD(S)	ADD Sym	ADD-S	ADD(S)	ADD Sym
Tool.Cap	90.57	90.57	88.43	98.62	98.62	97.74	90.80	90.80	90.15	**99.05**	**99.05**	**98.59**
DriveShaft	92.73	92.73	90.47	98.71	98.71	98.06	93.68	93.68	92.00	**99.00**	**99.00**	**98.69**
BrakingUnit	91.63	91.63	87.09	98.19	98.19	96.17	92.00	92.00	89.93	**98.65**	**98.65**	**98.02**
T-Less_18	89.34	86.33	86.33	98.45	97.56	97.56	89.42	87.16	87.16	**98.46**	**97.65**	**97.65**
T-Less_08	98.37	98.37	97.49	**99.09**	**99.09**	98.93	98.51	98.51	98.42	98.98	98.98	**99.06**
T-Less_27	92.47	92.47	82.09	93.55	93.55	83.82	97.70	97.70	96.65	**98.94**	**98.94**	**98.41**
ALL	92.52	92.02	88.64	97.77	97.62	95.37	93.80	93.43	92.39	**98.85**	**98.71**	**98.40**

Table 5.6.: Quantitative comparison between PVN3D single-view (SV), PVN3D multi-view (MV), FFB6D single-view and FFB6D multi-view on the SCAPE 2 dataset. The ADD-S, ADD(S) and ADD-Sym-AUC metrics are reported. Symmetric objects are highlighted in bold. The best results for ADD-S-AUC and ADD(S)-AUC are also highlighted in bold.

Firstly, it becomes clear that the introduction of our ADD-Sym metric works very well. The ADD-Sym metric is almost always a little lower compared to the ADD-S metric, which was expected since the ADD-Sym metric is, like the ADD metric, a little more precise than the ADD-S metric. This should make the accuracy on symmetric and non-symmetric objects a little more comparable.
Secondly, we observe that the single-view FFB6D model as well as the multi-view FFB6D model greatly benefit from the symmetry-aware training. Using the symmetry-aware training, the single-view results improve averaged over all objects by about 1.4% for the ADD(S)-AUC metric and for the multi-view model the results improve also by a respectable 1.3% on the ADD(S)-AUC metric to achieve a final score of 98.94%. The improvements per object vary. The only object that does not improve for both the single-view and multi-view symmetry-aware models is the T-Less_08 object, which already had a very high ADD(S) score. But all other objects improve by a large margin for both the single-view as well as the multi-view model. The greatest improvement can be seen for the T-Less_27 object which improves by over 5% for both the single-view and multi-view models on the ADD(S) metric. For our new symmetry metric, this improvement becomes even more clear. For the single-view model, we achieve an overall improvement of about 7% and for the multi-view model the improvement is about 6%. Also, the T-Less_08 object, which is the only object that does not improve on the ADD-S metric, improves slightly on the ADD-Sym metric. As our ADD-Sym metric is stricter than the ADD-S metric, we argue that the ADD-Sym metric is more reliable to indicate the improvements made by our symmetry-aware training.

Overall, the symmetry-aware training shows great promise. A big challenge remains as to how to model the infinite rotational symmetries. The results in Tab. 5.7 have been generated with four rotational symmetries during training. However, it is valuable to investigate whether additional rotational symmetries improve results. In Tab. 5.7 we show an ablation with a varying number of rotational symmetries for the two infinite rotational symmetric objects.

n Symmetries	0	4	8	16	32	64
Tool.Cap	90.57	90.80	91.19	90.96	91.34	**91.82**
DriveShaft	92.73	93.52	93.70	**94.28**	93.76	93.94

Table 5.7.: Ablation study for the modeling of the infinite rotational objects using varying discretization steps.

Tab. 5.7 shows that a finer discretization of the infinite rotational symmetry improves the final pose predictions. For the Tool.Cap we see a continuous improvement for finer/more discrete rotational symmetries. In our experiments, training with more rotational symmetries does not noticeably increase training time, nor GPU memory consumption, however, it requires more CPU memory in the data loading process. As CPU memory is often not a limiting factor, it is advisable to train with a very fine discretization of the infinite rotational symmetries. For the DriveShaft however, we achieve the best ADD-S score for 16 rotational symmetries. We believe this is because the DriveShaft object seen in Fig. 4.4 is not fully infinite rotational symmetric. The infinite rotational symmetry seems to be broken by the groves at the head of the DriveShaft. Consequently, the results from Tab. 5.7 indicate that the DriveShaft has at most 16 discrete rotational symmetries.

5.3. SCAPE YCB2

The SCAPE YCB2 dataset is slightly more difficult for the network to learn than the SCAPE YCB dataset because the camera poses are different across scenes. Here, the network has to generalize the fusion of different camera poses. Additionally, because the SCAPE YCB2 datasets offers four views around the scene with an extra top-down view, it is possible to evaluate how beneficial additional cameras are.

In Tab. 5.8 we have evaluated the effect of different amounts of cameras to observe the scene. For the single-view model we only report the average view as we want to focus in this comparison on the different amounts of views and the relation between best, average and worst view is very similar to the SCAPE YCB dataset. For the multi-view model we have trained the model on three, four, and four cameras with an additional top-down view. Similar to the SCAPE YCB results, the multi-view models outperform the averaged single-view FFB6D [12] model over all views of the scene. The multi-view networks are able to generalize multi-view fusion even if the camera setup changes each training sample. However, there is no clear number of views that outperforms the others. The models utilizing three, four and five views, respectively, achieve very similar results when averaged over all objects. Especially, four views and five views (four views plus top-down view) perform almost identical overall. Four views with the additional top-down view performs the best overall on the most relevant ADD(S) metric. For large objects like the cracker box, mustard bottle, sugar box and power drill, the model trained and evaluated on three views outperforms or matches the performance of the models with additional views. We hypothesize that for these larger objects, three views are enough to estimate the pose precisely and that further views

	Single-View		Multi-View					
	Single-View (avg)		Multi-View (3)		Multi-View (4)		Multi-View (5)	
	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)	ADD-S	ADD(S)
banana	78.16	70.59	95.85	91.76	96.96	94.41	**97.05**	**95.07**
cracker box	95.87	94.58	96.89	96.16	96.91	96.37	**97.00**	**96.50**
gelatin box	75.62	70.11	95.95	93.05	**97.02**	94.95	96.97	**94.72**
master chef can	93.20	88.34	97.66	96.84	**97.72**	**96.94**	97.32	96.29
mustard bottle	89.53	84.83	97.51	96.33	**97.54**	**96.69**	97.36	96.50
potted meat can	86.84	82.08	97.69	96.23	**97.84**	**96.72**	97.67	96.62
power drill	93.13	90.48	96.85	96.11	**96.95**	**96.32**	96.62	95.89
pudding box	84.07	79.77	97.20	95.76	**97.74**	**96.58**	97.53	96.32
sugar box	90.91	87.78	**97.47**	**96.74**	97.39	96.67	97.28	96.50
tomato soup can	85.54	80.48	97.99	96.38	98.17	96.99	97.85	96.61
tuna fish can	76.31	67.45	96.06	91.51	97.44	94.00	**97.49**	**94.55**
ALL	86.29	81.50	97.01	95.17	**97.43**	**96.06**	97.30	95.96

Table 5.8.: Quantitative comparison between single-view FFB6D and multi-view FFB6D using three, four or five views on the SCAPE YCB2 dataset. The ADD-S and ADD(S)-AUC are reported. The best results for ADD-S-AUC and ADD(S)-AUC are highlighted in bold.

do not add any beneficial information about the object, but increase the multi-view fusion complexity for the fusion modules described in Sec. 4.2.2, which may lead to more unstable training and slightly worse results.
The additional top-down view, included in the five views model, does not appear to be highly beneficial compared to the four views model. For some objects five views do improve the ADD(S) score and for some objects even the ADD-S score improves compared to only four views. It is hard to argue why exactly these objects benefit from a top-down perspective.

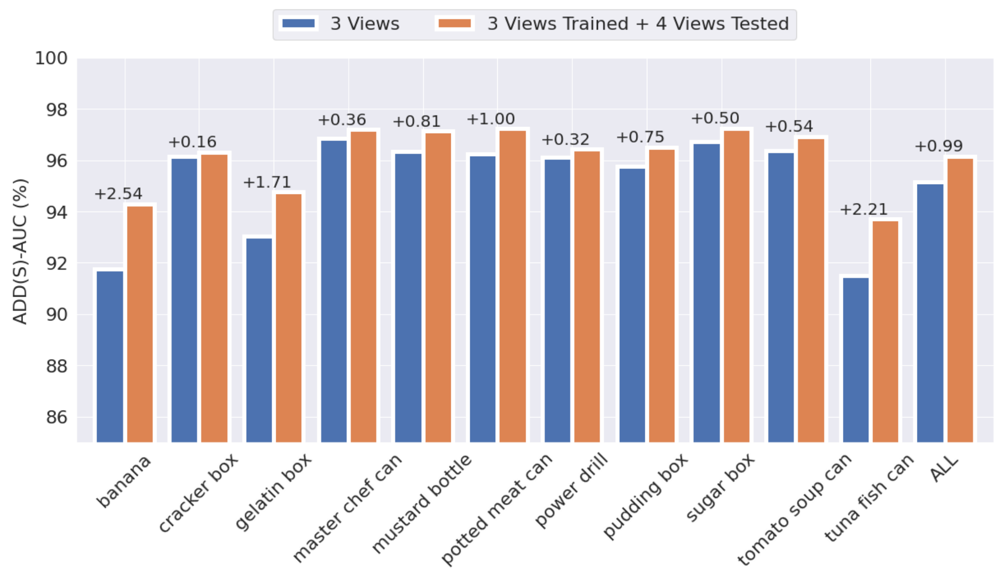
An explanation could be that those objects that benefit from the additional top-down view appear symmetric from some angles which deteriorates the pose estimation performance of the model and that only the additional top-down view detects features that break the symmetry.

Changing View Count

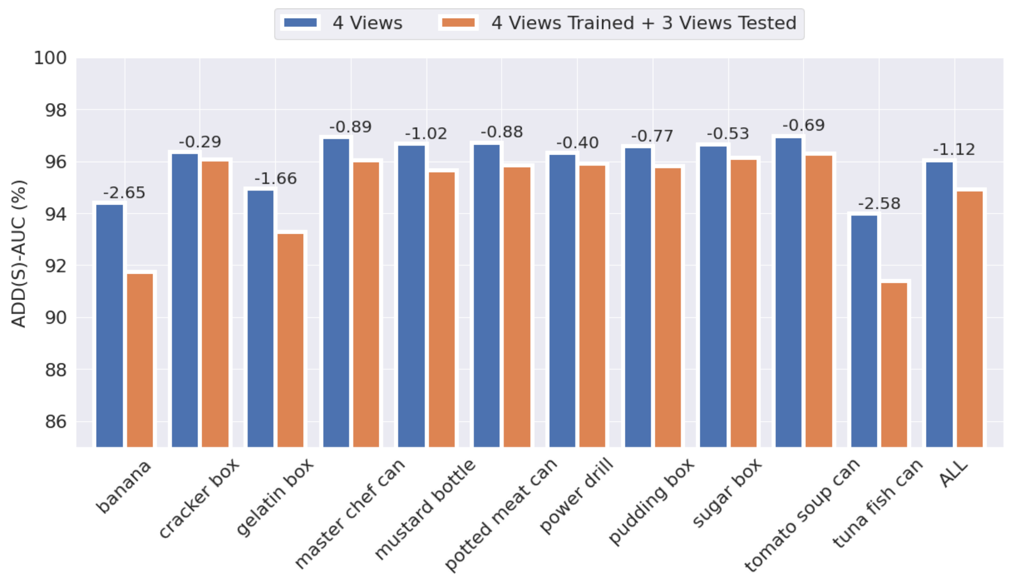
Tab. 5.8 shows the results of multi-view models that have been trained and evaluated on the same number of views. In Fig. 5.9 we evaluate whether a model trained on three views is able to utilize four views at test time and vice-versa. This gives great insight into whether the models learn a fixed fusion of fixed views or whether a trained model is robust towards changing cameras.

In Fig. 5.9a, it is shown that a model trained on three views is able to leverage four views at test time without any changes to the network architecture or normalization. The network is able to use the additional view, which gives extra information of the scene, to improve the pose estimation accuracy for all objects.

Similarly, Fig. 5.9b shows that it is possible to apply a model, that has been trained with four views, to only three views. Although the pose estimation accuracy decreases when testing with one view less, multi-view FFB6D still demonstrates that it is robust towards changing number of cameras.



(a) Comparison of multi-view FFB6D trained and tested on three views vs multi-view FFB6D trained on three views and tested on four views.



(b) Comparison of multi-view FFB6D trained and tested on four views vs multi-view FFB6D trained on four views and tested on three views. views.

Figure 5.9.: Quantitative comparison of changing view count at test time.

5.4. YCB-Video

Multi-View

Evaluating our approach on YCB-Video [35] is challenging, because we cannot use the synthetic frames provided by the YCB-Video datasets to train our multi-view FFB6D [12] network. However, evaluating on YCB-Video is necessary, as it is the only public real dataset where we can compare against another multi-view approach.

We solve the issue of the limited training data, by pretraining the network for the single-view task and then fine-tuning the parameters for the multi-view task. This is possible because our multi-view approach has the same architecture as the single-view network and the handling of the multiple input views is done through reshaping, batching and concatenating of the data. This is a major advantage of our approach. Nevertheless, fine-tuning on the limited real frames of the YCB-Video dataset is still difficult, and further improvements might be made if a larger dataset would be available. CosyPose [20] does not suffer from this drawback, as CosyPose can be trained on single views and tested on multiple views with the same network configuration.

		3 views	5 views
CosyPose[20]	ADD-S	92.29	93.26
	ADD(S)	87.66	88.80
PVN3D-MV[8]	ADD-S	92.98	91.06
	ADD(S)	87.72	84.03
FFB6D-MV[Ours]	ADD-S	**95.16**	**95.29**
	ADD(S)	**91.37**	**91.58**

Table 5.9.: Quantitative comparison between CosyPose, multi-view PVN3D and multi-view FFB6D on the YCB-Video dataset. The multi-view frames are randomly selected from the same sequence. The ADD-S and ADD(S)-AUC are reported. The best results for ADD-S-AUC and ADD(S)-AUC are highlighted in bold.

Our novel multi-view approach outperforms the current state-of-the-art for multi-view 6D Pose Estimation on the YCB-Video dataset by a large margin (see Tab. 5.9). However, using five instead of three views does not improve results very much. This is because the sequences in YCB-Video are very short and do not offer many different viewing angles. For the multi-view PVN3D model, for which we employ the same training strategy as for multi-view FFB6D, the accuracy deteriorates for five views compared to three views. We assume this effect has two causes. On the one hand, PVN3D has more parameters and therefore might need more training samples to find the optimal weights. On the other hand, using five views reduces the number of training samples drastically because more frames are grouped into a single multi-view training sample.

Additionally, the comparison with CosyPose is not quite fair as CosyPose only operates on RGB data while FFB6D and PVN3D use RGB-D data. Our multi-view adaption of FFB6D also assumes known camera poses, something that CosyPose does not assume. This means that CosyPose has to solve a much harder task than our approach.

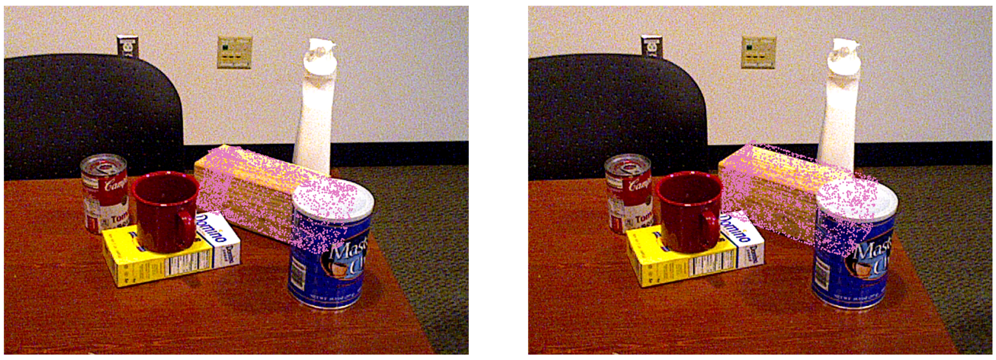
Symmetry

FFB6D already achieves impressive results on YCB-Video even on symmetric objects. One reason for this might be the novel SIFT farthest point sampling keypoint selection process. Still, the performance on symmetric objects is lower compared to non-symmetric objects in the YCB-Video dataset. To make a fair comparison with the original FFB6D model, we trained our symmetry-aware model with the same hyperparameters and the same training procedure. We are able to outperform the standard FFB6D on all objects that are defined as

	FFB6D[12]	Symmetry-Aware FFB6D
024 Bowl	95.04*	**96.39**
036 Wood block	92.61	**95.17**
051 large clamp	96.70	**96.86**
052 extra large clamp	94.28*	**95.30**
061 foam brick	97.31	**97.60**

Table 5.10.: Quantitative comparison between standard FFB6D and FFB6D trained using the symmetry-aware loss function. The evaluation is limited to the symmetric classes in the YCB-Video dataset.The ADD-S and ADD(S)-AUC are reported. (*paper results deviate from published checkpoints)

symmetric and achieve similar results for all non-symmetric objects. For the large clamp and foam brick, we observe marginal improvements over the standard network. However, for the bowl, wood block and extra large clamp we see big improvements. Especially for the wood block, we see the largest improvements. The wood block is very similar to the T-Less_27 object from the SCAPE 2 dataset. Both have a discrete rotational symmetry of order $n = 4$. It appears that these squarish objects benefit most from our symmetry-aware training because a wrong estimation of the rotation misaligns the meshes drastically.



(a) Standard training (b) Symmetry-aware training

Figure 5.10.: Qualitative comparison for standard (a) and symmetry-aware (b) FFB6D on the 036 Wood block object. The sample is selected from the YCB-Video test set. The mesh of the wood block object is projected into the image space using the camera intrinsics and predicted poses.

In Fig. 5.10 we have evaluated this qualitatively for the wood block object. The pose estimation definitely improves with the symmetry-aware training. However, the pose prediction is not perfect using the symmetry-aware training. A reason for this might be that the wood block in the scene is partially occluded, which makes the pose prediction harder.

6. Conclusion

The goal of this thesis was to utilize multiple views for the 6D Pose Estimation to make 6D Pose Estimation more accurate in complex scenes with multiple objects and many occlusions. Additionally, we aimed to tackle one of the biggest challenges in 6D Pose Estimation, which is symmetric objects, by introducing a novel symmetry-aware training that explicitly handles object symmetries.

For this, we propose a novel multi-view architecture that supports multiple RGB-D frames, based on the current state-of-the-art network for single-view pose estimation called FFB6D. To show the effectiveness of our approach, we evaluated both the single-view and multi-view network on three custom synthetic datasets, designed to highlight the advantages of a multi-view architecture. The experiments show that although FFB6D claims to handle severe occlusions [12], our multi-view approach is able to outperform the single-view results by a large margin in highly cluttered scenes with many occlusions. On the SCAPE 2 dataset, the single-view model is able to achieve similar results to our multi-view architecture with the ideal view. However on the SCAPE YCB dataset, even on the best view, the single-view network is outperformed by the multi-view network by a large margin. Additionally, we show that our proposed architecture is robust towards a noisy fusion when the camera poses are not perfectly known. Furthermore, our novel multi-view architecture is robust towards a changing camera set-up at test time, including a changing amount of cameras which we show on the SCAPE YCB2 dataset.

We also compare against a multi-view variant that uses a similar approach to our multi-view fusion method on an older 6D Pose Estimation architecture. We are able to outperform the network on the standard ADD-S-AUC and ADD(S)-AUC metric on the SCAPE YCB dataset. But we achieve only slightly better results on the ADD(S)<2cm metric, as well as on the SCAPE 2 dataset in general.

Further, we adapted the public YCB-Video dataset to support multiple views. However, the YCB-Video dataset does not offer diverse views and many occlusions, nevertheless our approach does outperform CosyPose which is the current state-of-the-art network for multi-view pose estimation.

In addition to our multi-view extension, we designed a symmetry-aware loss function used for the training of the 3D keypoint estimation task within the FFB6D network. The improved loss function prevents the network from learning keypoints that lie exclusively on the symmetric axis. We outperform our single-view and multi-view results on SCAPE 2 with our symmetry-aware training, especially for the rotational symmetric objects. On the YCB-Video dataset, we are able to confirm our improvements for real world data.

Future Work

Although we have achieved great results with this master thesis, there are still improvements to be made to bring 6D Pose Estimation methods to the standard required for robot grasping and object manipulation.

We already conducted small experiments on the YCB-Video dataset with our novel multi-view approach. However, we believe that the YCB-Video dataset is not ideal to show our advances with our multi-view architecture. The low number of occlusions in each scene combined with only a limited range of different viewing angles limit the advantages of a multi-view approach. We believe that a larger non-synthetic dataset that is designed for multi-view pose estimation is needed to demonstrate the advances that we have made for 6D Pose Estimation.

Further, we would like to explore a fairer comparison with CosyPose. For this, we would like to extend CosyPose to RGB-D data. There are a couple of ways of how to approach this. The most common approach to lift an RGB-only method to RGB-D data is to apply ICP to the 6D Pose Estimates. However, a more interesting approach would be to exchange the first stage of CosyPose with the single-view FFB6D network to generate pose candidates for each view. For a completely fair comparison, it would be necessary to fix the camera poses in the joint object-camera-pose optimization step in the last stage of CosyPose.

We would also like to solve some other limitations that FFB6D and by extension our multi-view variation of FFB6D has. There is for instance the limitation of only a single instance of each object being allowed in each scene, limiting the network to only solve the SiMo task. A simple way to resolve this limitation is to adapt the center prediction module. Instead of allowing only one center prediction per object class within the mean shift clustering, it is possible to handle multiple clusters to distinguish multiple instance of the same object.

Another topic that would be interesting to explore would be to replace the whole least-squares-fitting with a trainable network. The idea is to model the uncertainty of the keypoints better to finally predict pose estimates better. Here, more than eight keypoints that are optimal for the least-squares-fitting could be valuable as well because the network could attend to the uncertainty of the estimated keypoints and focus on keypoints with high certainty, but also take the low certainty keypoints into account.

A. SCAPE YCB ADD(S)-AUC

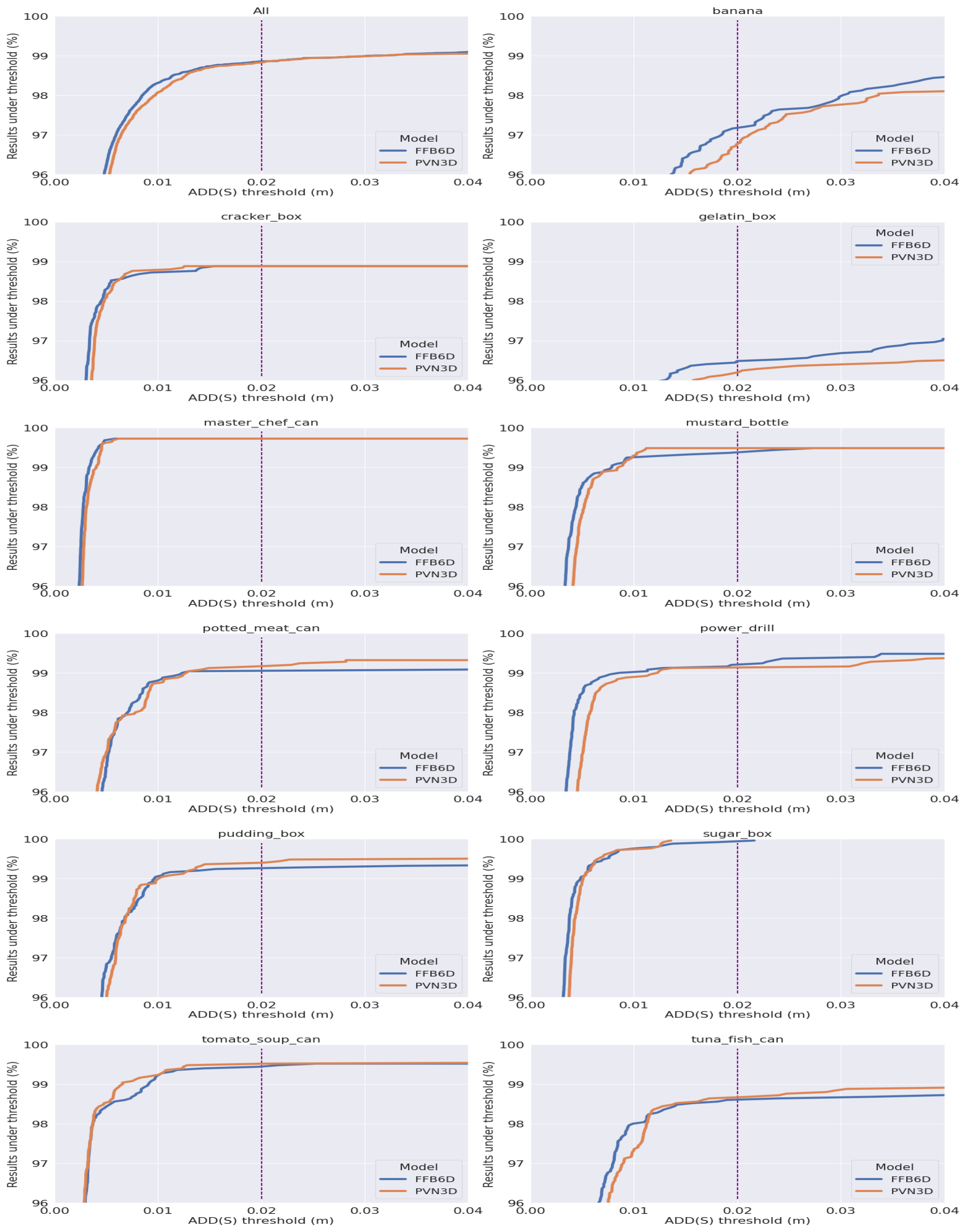


Figure A.1.: Individual AUC comparison of multi-view FFB6D and multi-view PVN3D for all objects from the SCAPE YCB dataset.

Bibliography

[1] K. S. Arun et al. “Least-Squares Fitting of Two 3-D Point Sets”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), pp. 698–700.

[2] Lennard Bodden. “Point cloud based object recognition in RGB-D video streams for robot manipulation”. MA thesis. Eberhard’s Karl University of Tübingen, Nov. 2020.

[3] Eric Brachmann et al. “Learning 6D Object Pose Estimation Using 3D Object Coordinates”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 536–551.

[4] Mai Bui et al. “When Regression Meets Manifold Learning for Object Recognition and Pose Estimation”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 6140–6146.

[5] Berk Calli et al. “The YCB object and Model set: Towards common benchmarks for manipulation research”. In: *2015 International Conference on Advanced Robotics (ICAR)*. 2015, pp. 510–517.

[6] Zhe Cao et al. “Real-time scalable 6DOF pose estimation for textureless objects”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2441–2448.

[7] Yizong Cheng. “Mean shift, mode seeking, and clustering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 790–799.

[8] Tobias Demmler. “Multi-view 6D Pose Estimation on RGB-D Frames Using a Deep Point-wise Voting Network”. MA thesis. Albert-Ludwigs-University Freiburg, Sept. 2021.

[9] Ge Gao et al. “6D Object Pose Regression via Supervised Learning on Point Clouds”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3643–3649.

[10] Ge Gao et al. “Occlusion resistant object rotation regression from point cloud segments”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[11] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[12] Yisheng He et al. “FFB6D: A Full Flow Bidirectional Fusion Network for 6D Pose Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 3003–3013.

[13] Yisheng He et al. “PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[14] Stefan Hinterstoisser et al. “Gradient Response Maps for Real-Time Detection of Textureless Objects”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (2012), pp. 876–888.

[15] Stefan Hinterstoisser et al. "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes". In: *Computer Vision – ACCV 2012*. Ed. by Kyoung Mu Lee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562. ISBN: 978-3-642-37331-2.

[16] Stefan Hinterstoisser et al. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". In: *2011 International Conference on Computer Vision*. 2011, pp. 858–865.

[17] Tomáš Hodaň et al. "BOP Challenge 2020 on 6D Object Localization". In: *European Conference on Computer Vision Workshops (ECCVW)* (2020).

[18] Tomáš Hodaň et al. "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017).

[19] Qingyong Hu et al. "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[20] Yann Labbé et al. "Cosypose: Consistent multi-view multi-object 6d pose estimation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 574–591.

[21] Chi Li et al. "A Unified Framework for Multi-View Multi-Class Object Pose Estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[22] Yi Li et al. "DeepIM: Deep Iterative Matching for 6D Pose Estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[23] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[24] G Lowe. "Sift-the scale invariant feature transform". In: *Int. J* 2.91-110 (2004), p. 2.

[25] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[26] Sida Peng et al. "PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[27] Charles R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[28] Charles Ruizhongtai Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[29] Caner Sahin et al. "Instance-and category-level 6d object pose estimation". In: *RGB-D Image Analysis and Processing*. Springer, 2019, pp. 243–265.

[30] Martin Sundermeyer et al. "Multi-Path Learning for Object Pose Estimation Across Domains". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[31] Chen Wang et al. "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[32] Jiaze Wang et al. "Category-Level 6D Object Pose Estimation via Cascaded Relation and Recurrent Reconstruction Networks". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 4807–4814.

[33] Nicolai Waniek et al. *AMIRA Blender Rendering: a tool for photorealistic rendering with Blender*. `https://github.com/boschresearch/amira_blender_rendering`. Nov. 2020.

[34] Yangzheng Wu et al. "Vote from the Center: 6 DoF Pose Estimation in RGB-D Images by Radial Keypoint Voting". In: *CoRR* abs/2104.02527 (2021). arXiv: `2104.02527`.

[35] Yu Xiang et al. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". In: *Robotics: Science and Systems (RSS)*. 2018.

[36] Danfei Xu et al. "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[37] Hongjia Zhang et al. "Symmetry-Aware 6D Object Pose Estimation via Multitask Learning". In: *Complexity* 2020 (2020).

[38] Hengshuang Zhao et al. "Pyramid Scene Parsing Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.