

# DPODv2: Dense Correspondence-Based 6 DoF Pose Estimation

Ivan Shugurov, Sergey Zakharov, Slobodan Ilic

**Abstract**—We propose a three-stage 6 DoF object detection method called DPODv2 (Dense Pose Object Detector) that relies on dense correspondences. We combine a 2D object detector with a dense correspondence estimation network and a multi-view pose refinement method to estimate a full 6 DoF pose. Unlike other deep learning methods that are typically restricted to monocular RGB images, we propose a unified deep learning network allowing different imaging modalities to be used (RGB or Depth). Moreover, we propose a novel pose refinement method, that is based on differentiable rendering. The main concept is to compare predicted and rendered correspondences in multiple views to obtain a pose which is consistent with predicted correspondences in all views. Our proposed method is evaluated rigorously on different data modalities and types of training data in a controlled setup. The main conclusion is that RGB excels in correspondence estimation, while depth contributes to the pose accuracy if good 3D-3D correspondences are available. Naturally, their combination achieves the overall best performance. We perform an extensive evaluation and an ablation study to analyze and validate the results on several challenging datasets. DPODv2 achieves excellent results on all of them while still remaining fast and scalable independent of the used data modality and the type of training data.

**Index Terms**—6 DoF pose estimation, dense correspondences, synthetic data.



## 1 INTRODUCTION

Object detection and 6 DoF pose estimation are not new fields in computer vision. In fact, they are among the major driving forces, being crucial for various application fields, such as augmented reality, robotics and autonomous driving. Therefore, there is a vast number of methods trying to tackle this problem.

In the pre-deep learning era, pose estimation was typically performed either completely using depth or using a combination of depth and RGB data. However, recent trends in 6 DoF pose estimation move towards disregarding depth data as a modality and using only RGB data. The reasons for that are manifold, but the most important are the omnipresent availability of RGB cameras in the modern devices and the success of deep learning methods. The progress in RGB deep learning allows the researchers to choose from a vast number of ready to use network architectures and focus their attention on extending them to predict the pose. The state of the art RGB solutions are solely based on convolutional neural networks (CNNs), demonstrating impressive results that could barely be imaginable a couple of years ago. Even though recent RGB methods, such as CosyPose [1] perform better than depth-based point pair feature approaches [2], they still suffer from perspective ambiguities and appearance changes. This can be remedied with the additional use of depth data or an extra step with multi-view refinement.

Inspired by the works of Taylor et al. [3] and Gueler et al. [4], Brachmann et al [5] and Jafari et al. [6], we developed and compared several variations of deep dense correspondence-based 6 DoF object detectors using either RGB or depth as input. We ran a thorough analysis and ablation studies to evaluate the strengths and weaknesses of the presented modalities and compared them to the state of the art. Moreover, each correspondence estimation networks is trained on two types of data: synthetic and real. While real labeled data are most commonly used to achieve the best possible results, the acquisition of such data is often infeasible in real applications due to the high costs and significant time efforts. Synthetic data, on the other hand, is free of these drawbacks and can be easily rendered in a variety of scenes and under an unlimited number of poses. Unfortunately, the methods trained on synthetic data are subject to the domain gap problem, due to the dissimilarity between real and synthetic images. This might causes them to perform worse than the methods trained on real data. However, as was shown in HomebrewedDB [7], networks trained only synthetic data are less prone to overfitting due to a larger data variability, which allows them to perform similarly to the methods trained on real data when train and test images do not come from the same image sequence. Therefore, detectors trained on synthetic data are desirable, because the required data is easier to obtain and because such detectors tend to generalize better, thus being of higher practical significance.

Even though there is a larger number of pose estimation methods for all kinds of data modalities, they are typically designed for a specific one and have completely different architectures from methods operating on other data modalities. This makes it impossible to measure which advances come from better methods and which come from a different data modality or from better training data. In this paper,

- I. Shugurov, and S. Ilic, are with the Department of Informatics, Technical University of Munich, Germany and with Siemens AG, Munich, Germany.  
E-mail: see <http://campar.in.tum.de>
- S. Zakharov was affiliated with the Department of Informatics, Technical University of Munich and Siemens AG at the time of the trial and is currently affiliated with Toyota Research Institute, Los Altos, USA  
E-mail: [sergey.zakharov@tri.global](mailto:sergey.zakharov@tri.global)

Manuscript received May 23, 2021.

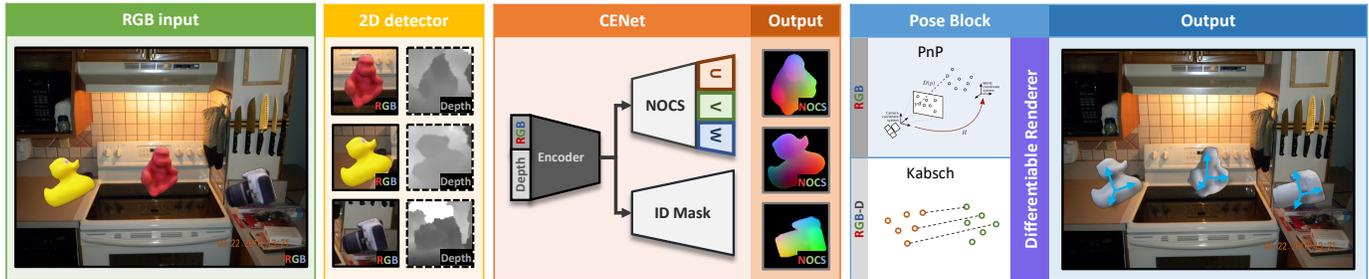


Fig. 1: **Synthetic toy example illustrating three-stage correspondence-based 6 DoF pose estimation:** A full RGB image is fed to a 2D object detector for bounding box estimation. Resulting bounding boxes are then used to generate crops on the available data, which are subsequently fed into the correspondence network. If only RGB images are provided, then the pose is estimated from correspondences using 2D-3D PnP. In case registered depth data is also in place, we project the estimated correspondences into 3D and use the 3D-3D Kabsch algorithm.

we propose a unified deep neural network architecture capable of predicting dense correspondences either from depth maps or from RGB images. The pose estimation part enables direct comparison of the data modalities on various datasets. Additionally, in contrast to the other dense correspondence methods, we report per-pixel correspondence error to deepen the understanding of where pose imprecision comes from. From the results obtained we conclude that RGB images excel in object localization and correspondence estimation, while 3D-3D correspondences between the depth images and the model result in more accurate pose estimates. Naturally, their combination brings the best performance. We also introduce a correspondence-based refinement method based on differentiable rendering. Moreover, if relative camera transformations between frames are known, the predicted poses can be additionally refined by aligning per-image dense correspondences between the object correspondences rendered in the predicted pose and the correspondences predicted by the network. The proposed loss function is implemented using a differentiable renderer, which allows for its minimization with the standard gradient-based methods.

The main contributions of this work can be summarized as follows:

- A unified framework for dense correspondence estimation, whose architecture is agnostic to the input modality type available (RGB or Depth);
- A thorough analysis and ablation study of the presented methods and estimation of their strengths and weaknesses;
- An analysis of the quality of predicted correspondences;
- A correspondence-based 6 DoF pose refinement extensible to multiple views.

## 2 RELATED WORK

The topic of 6 DoF pose estimation has been studied extensively of times in the past. In this section, we provide a quick overview over the most prominent works. We coarsely grouped the existing 6 DoF pose estimation methods into three categories: direct pose regression, template-based and correspondence-based methods. We additionally discuss multi-view methods for pose estimation and refinement methods.

### 2.1 Direct Pose Regression Methods

Given that neural networks are universal approximators, it feels natural to try to tackle the problem of pose estimation by direct prediction of the pose. One of the first works to ever do that was the PoseNet [8], where a CNN is utilized to regress the position and orientation of a camera given an RGB image. Subsequent works improved the localization results by combining a CNN with LSTM units [9] and studied the effect of different loss functions on the final performance [10]. PoseCNN [11] aims to disentangle the pose estimation into several sub-tasks, i.e. regressing object masks, estimating the translation of the object centroids, and regressing quaternions for rotation. Alternatively, the SSD6D detector [12] relies on a discrete viewpoint classification rather than direct regression of rotations. To make it possible, all poses are divided into a large number of discrete ones, and each of them is further divided into a smaller number of discrete in-plane rotations. As a result, this significantly reduces the solution space allowing for simpler training of the network. However, rotation discretization and approximation of translation lead to poor pose estimates, which unconditionally need further refinement. Despite their seeming simplicity, the direct pose regression methods report significantly lower accuracy when compared to correspondence-based solutions. Therefore, post-processing pose refinement, e.g. ICP, must be performed to overcome the issue. DenseFusion [13] was free of some of those problems by using RGBD input directly. Another notable exception is the CosyPose introduced in [1], which is the best performing method on several datasets from the BOP challenge [14]. The paper is a conceptual extension of PoseCNN and DeepIM [15], which achieves exceptionally good results by using better deep learning backbones and a better pose parameterization. Additionally, CosyPose [1] introduced a multi-view refinement method which boosted the results even further.

### 2.2 Template-Based Methods

In template-based methods, a template represents a global descriptor capturing the object's appearance under a certain view angle. A set of such templates is extracted to cover all possible object's views. At the test time, the entire template set is compared to the input image at all possible locations using a predefined matching score and the best candidates are pruned to get the final result.

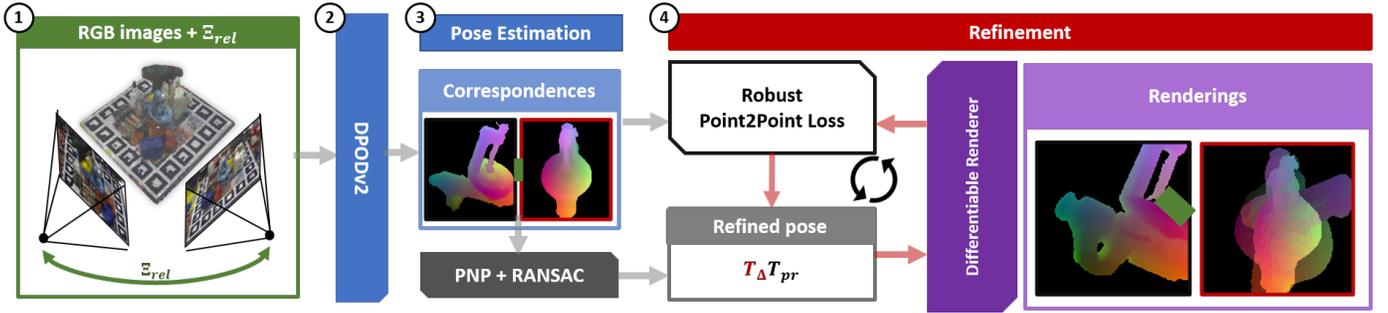


Fig. 2: **Multi-view pose optimization:** The algorithm takes the output of DPODv2 from several views with known relative camera transformations as input. An initial pose hypothesis is iteratively refined until it converges to a pose, which is consistent with predicted correspondences in all frames. The proposed loss function penalizes pixel-wise distance between a predicted correspondence and a correspondence corresponding to the current pose hypothesis. The loss function is implemented using a differentiable renderer.

Arguably the most popular template-based methods for 6 DoF pose estimation is LineMOD – a instance template method [16], [17] relying on a combination of the RGB gradients and surface normals computed from depth images. Thanks to a very efficient implementation, the method is capable of covering an impressive number of angle and scale combinations and still estimate the pose in real time. A follow-up work [18] demonstrated how to extract templates from synthetic 3D models and then apply them to real data. This was of great value to industrial applications since all possible views and scales for a given object can be generated at no cost. The further follow-up works aimed to improve the scalability issue related to the used number of templates allowing for significant speed-ups [19], [20], [21], [22].

With the advent of deep learning, more and more methods started to rely on CNNs to learn descriptors directly from data as opposed to using hand-crafted templates. The method introduced in [23] mapped image data directly to the similarity-preserving descriptor space. To learn the mapping, the authors utilized a triplet network and a cost function defined such that the squared Euclidean distance between similar objects was minimized and for dissimilar objects the hinge loss was applied forcing them to be pulled apart using a margin term. The matching is then done by applying efficient and scalable nearest neighbor search methods on the descriptor space to retrieve the closest neighbors. The follow-up works [24], [25] extend the method by introducing an additional rotational degree of freedom (making it 3 DoF), improving the descriptor space by introducing a modified loss function, and a multi-task extension combining manifold learning with direct pose regression leveraging the advantages of both. Another line of works [26], [27] strive to improve the domain generalization capabilities of the method making it possible to train it from synthetic data while performing on par and better than real methods by using the inverse domain adaptation technique. Building on the success of the above methods, the AAE [28] method proposed a full 6 DoF pose estimation pipeline based on learned descriptors by using already computed SSD detections as input.

Despite their popularity, the template based methods have a number of drawbacks. The first one is related to a limited discrete number of view templates. As a result, the

pose estimates are often very imprecise and additional post-processing refinement methods are required, e.g., Iterative Closest Point (ICP) [29]. Using a large number of templates can improve the accuracy, but in return leads to a slow matching during the test phase. Moreover, template-based methods are often very sensitive to occlusions and clutter since they rely on global descriptors. However, despite the drawbacks, these methods are still very popular due to their simplicity and proven efficiency.

### 2.3 Correspondence-Based Methods

A popular alternative to template-based methods are methods utilizing correspondences between the input image and the 3D object model. Their power lies in a reduced solution space and explicitly introduced geometric constraints. This results in more precise pose estimates and the increased robustness to occlusions and clutter.

The first family of methods rely on prediction of a pre-defined set of keypoints whose 3D locations on the object model are known, followed by the PnP algorithm to compute the full 6 DoF pose. Rad et al. presented a holistic three-stage approach called BB8 [30]. In the first two stages, a coarse-to-fine segmentation is performed, the result of which is then fed to the third network trained to output projections of the object’s bounding box points. Knowing 2D-3D correspondences, a 6 DoF pose can be estimated using a PnP (Perspective-n-Point) solver. The main disadvantage of this pipeline is its multi-stage nature, resulting in slow run times. Building on the popular 2D detector YOLO [31] and the BB8 ideas, YOLO6D [32] proposed a novel deep learning architecture capable of efficient and precise object detection and pose estimation without refinement. As is the case with BB8, the key feature here is to perform the regression of the reprojected bounding box corners in the image. Once correspondences are predicted, the pose can be estimated using a EPnP solver [33], similarly to BB8.

Another alternative is to use dense pixel-wise 2D-3D or 3D-3D correspondences between image pixels and the object model. The assumption is that a larger number of correspondences will mitigate the problem of their inaccuracies and will result in more precise poses. Moreover, it allows for a significantly better treatment of occlusions.

TABLE 1: Data modalities and types of train data used in the experiments.

Dataset/Modality	YOLO		CENet			
	Real RGB	Synt. RGB	Real RGB	Synt. RGB	Real depth	Synt. depth
Linemod	Yes	Yes	Yes	Yes	Yes	Yes
Occlusion	Yes	Yes	Yes	Yes	Yes	Yes
TLESS	Yes	Yes	Yes	Yes	No	Yes
HomebrewdDB	No	Yes	No	Yes	No	Yes

Arguably the first method to formulate pose estimation as dense 2D-3D correspondence prediction was [5]. The key idea was later revisited in several publications by switching from random forests to deep learning. Some of the most recent representatives include iPose [6], PVNet [34], DPOD [35], Pix2Pose [36], CDPN [37], and SDFlabel [38]. iPose [6] operates in 3 stages: segmentation, 3D coordinate regression and pose estimation. In contrast, DPOD [35] and Pix2Pose [36] unify the first two stages into the end-to-end network. Moreover, DPOD [35] classifies correspondences instead of regressing them that turned out to be a much easier task for the network, resulting in less erroneous correspondences. CDPN [37] disentangles 6 DoF pose estimation by using 3D correspondences for rotation estimation and regressing translation directly from the image. EPOS [39] proposed a different dense corresponding parameterization. Instead of a simple prediction of 3D locations of each 2D point, the network predicts two values. First, it classifies a pixel according to which object segment it belongs to. Then, a precise coordinate within the segment is regressed. This parameterization allows for easier handling of object symmetries. Combined with a robust and efficient variant of the PnP-RANSAC algorithm, the method achieves excellent results on the TLESS dataset. SDFlabel [38] uses 3D correspondences to estimate the pose and additionally estimates the shape of the object. NOCS [40] showed that dense correspondences prediction can generalize to objects not seeing during training if the network is trained and tested on the same class of objects. PVNet [34] takes a different approach and designs a network which for every pixel in the image regresses an offset to the predefined keypoints located on the object itself. The ideas on PVNet were further developed in several works. PVNet3D [41] takes both RGB and depth images as input. After independent feature extraction from each data modality, the features are merged and Hough voting is performed to estimate locations of the pre-defined keypoints in 3D space. Then, a least-square algorithm is used to estimate the pose. Alternatively, HybridPose [42] extends the initial PVNet approach by also predicting edges and axes of symmetries. They are used together with the predicted keypoints to estimate the pose with the modified PnP algorithm.

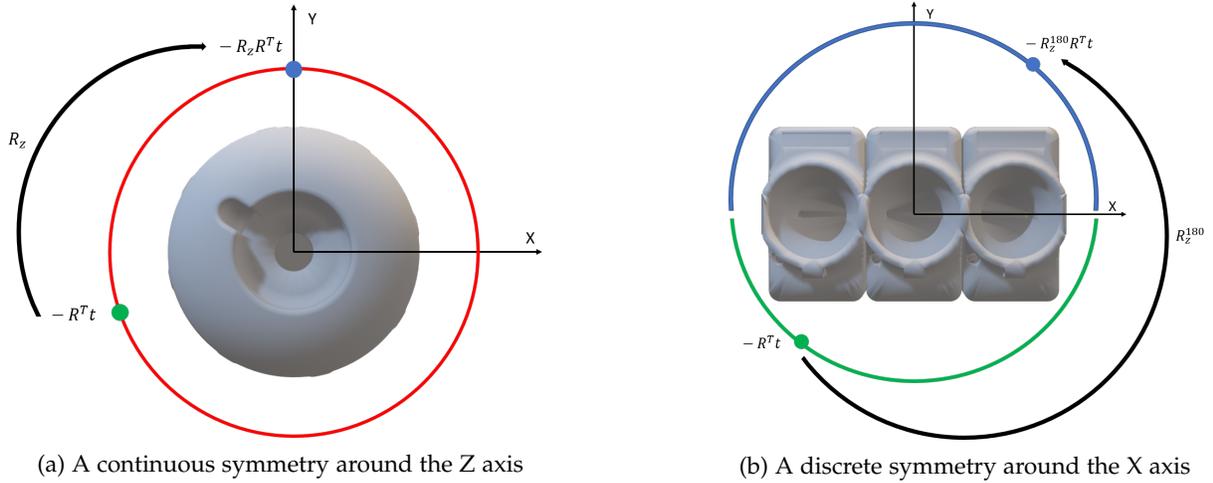
In contrast to the aforementioned dense correspondence methods, we show that DPODv2 can be used on data of various modalities, i.e. RGB and RGBD, and can be trained both on real and on synthetic data. Moreover we also introduce a generic correspondence-based pose refinement approach, which does not require any retraining. Finally, we provide a thorough evaluation of all the possible approaches to measure how those choices affect quality of dense correspondences and, subsequently, the quality of 6 DoF poses.

## 2.4 Monocular Pose Refinement

Numerous methods for monocular pose refinement have been introduced over the years. One of the most fundamental and widely used methods is Iterative Closest Point [29] (ICP). It works by iteratively aligning an observed point cloud with object’s vertices starting from an initial pose hypothesis. If only RGB image are available, the object pose can be refined using edge-based alignment, which in spirit resembles ICP. This approach is very sensitive to the image quality, clutter, occlusions and to the way in which the edge correspondences are computed. There have also been attempts to replace traditional optimization-based hand-crafted refiners with deep learning [15], [30], [35], [43]. They share a common principle of training a network to compare a detected object with a rendering of the object in the initial pose hypothesis. In BB8 [30], a network was trained to predict an offset for predicted keypoints, which will make the pose more precise. DPOD [35], DeepIM [15] and [15] design the network to directly predict the pose offset by re-parameterizing the pose disentangling rotation and translation. While excellent pose quality was reported in these papers, the methods are rather inconvenient because they require training for each particular model and the effectiveness of pose refinement depends heavily on the correct initial error prior used during training. Unlike these approaches our approach is an optimization approach and requires no lengthy retraining for each new object.

## 2.5 Pose Estimation From Multiple Views

There have been several attempts to take advantage of using several images to localize the object and estimate the pose. The most prominent examples are [1], [44], [45], [46], [47], [48]. The potential advantages of multi-view methods are better handling of occlusions, because the object is observed from different viewpoints, and better pose estimation facilitated by geometric multi-view constraints. However, to the best of our knowledge, the method of [47] is the only one which actually uses several views to detect the object jointly in each frame and estimate a single joint pose. This is achieved by first obtaining a sparse 3D reconstruction from several views and then object detection and pose estimation in the reconstructed point cloud. Methods in [44], [45], [46], on the other hand, use a monocular pose estimation network separately in each frame and use multiple frames with known relative camera transformations only for pose selection or pose refinement. CosyPose [1] follows the same paradigm as the aforementioned methods but does not require known relative camera poses. Instead, it uses a RANSAC scheme to match pose hypotheses from several frames and produce a unified object-level scene reconstruc-



(a) A continuous symmetry around the Z axis

(b) A discrete symmetry around the X axis

**Fig. 3: Pose recalculation for symmetric objects during training.** Poses are disambiguated during training to produce consistent NOCS maps. In case of a continuous symmetry around Z axis (Example a), a rotation around Z is added to the initial pose to ensure that the camera is always located on the same arch in the object coordinate system. In case of discrete symmetries (Example b), all symmetric poses are mapped to the one base pose by rotation around the symmetry axis.

tion and approximate relative camera poses. They are jointly optimized by minimizing the multi-view reprojection error.

### 3 METHODOLOGY

In this paper, we propose a three-stage object detection and pose estimation pipeline with a potential fourth refinement stage. It builds atop of the existing state of the art results with training on both real and synthetic data of potentially different modalities. The pipeline is visualized in Figure 1. The first stage is responsible for 2D object detection from RGB images, which is a standard and well-studied research area. In the second stage, a convolutional neural network, specifically designed for semantic segmentation, takes a detected patch with an object and regresses dense correspondences between input pixels and the 3D object model. Object poses is computed using the predicted correspondences in the third stage. The optional fourth stage, visualized in Figure 2, is responsible for pose refinement, as discussed later. Such a setup, allows for flexible mixing of data modalities and types of training data. Additionally, it simplifies the training procedure, as it breaks the task into smaller independent subtasks. In this section, we provide detailed explanations of each part. Then, we introduce and explain the multi-view refiner which operates atop of dense correspondences predicted by DPODv2.

#### 3.1 2D Object Detection

The first step of the pipeline consists of an off-the-shelf 2D object detector. As shown in previous work of [12], [49], [50], it is easier to achieve good 2D detection recall if it is done by conventional object detection approaches. SSD6D, which was trained purely on synthetic data, relied on fine-tuning a separate confidence threshold per each object class to achieve near-perfect recall. AAE [49], [50], on the other hand, trained their 2D detector completely on real data. We used YOLOv3 [51] in all our experiments, although the pipeline is agnostic to a particular choice of the detector. YOLO was trained either completely on real or completely on synthetic RGB data. Another crucial reason

for splitting object detection and correspondence estimation into two disjoint steps is scalability. Most of the approaches, mentioned in the related work, train a separate detector per each object to improve the results. While it is achievable on smaller datasets, such as Linemod [18], it is more challenging on more sophisticated datasets [7], [52] and close to impossible in real life.

#### 3.2 Pose Parameterization With Dense Correspondences

Instead of direct pose regression, which is still challenging and ill-posed for deep learning, we advocate the use of dense per-pixel correspondences between the image and the object model. With the correspondences at hand, the pose can be estimated with a number of well-studied methods depending on the data modality of choice [53]. While in the original DPOD paper, the authors relied on well-known 2-dimensional UV maps from computer graphics, we opted for the three-dimensional Normalized Object Coordinates Space (NOCS) [40] maps. The main reason is its simplicity and the lack of need for manual adjustment of the UV maps. Each dimension of NOCS corresponds to a dimension of the object scaled uniformly to fit in  $[0, 1]$  range. This parameterization allows for trivial conversion between the object coordinate system and the NOCS coordinate system, which is more suitable for regression with deep learning due to its constrained nature.

Let us formally define a model as a set of its vertices:  $\mathcal{M} := \{v \in \mathbb{R}^3\}$ . Operators that compute minimal and maximal coordinate along the vertex dimension  $i$  of all  $v \in \mathcal{M}$  are defined as

$$\min_i(\mathcal{M}) := \min_{v \in \mathcal{M}} v_i, \quad \max_i(\mathcal{M}) := \max_{v \in \mathcal{M}} v_i \quad (1)$$

Then, for any model's vertex  $v$ , the NOCS projection operator is defined w.r.t. the model as

$$\pi_{\mathcal{M}}(v) := \left\{ \frac{v_d - \min_d(\mathcal{M})}{\max_d(\mathcal{M}) - \min_d(\mathcal{M})} \right\}_{d \in \{x, y, z\}} \quad (2)$$

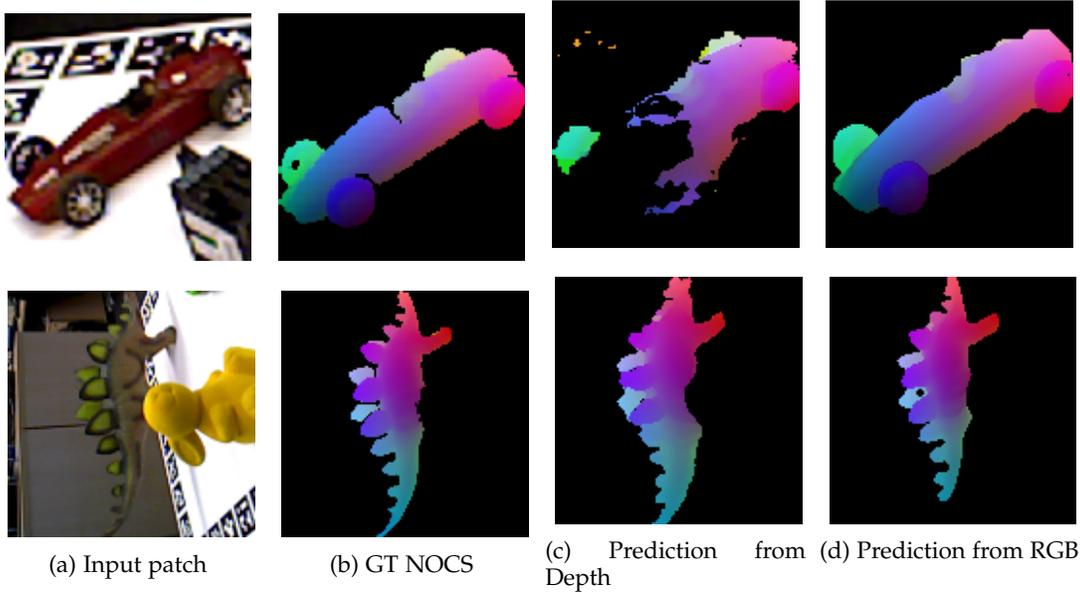


Fig. 4: Visual comparison of predicted segmentation maps of the visible object parts and color-coded NOCS for the same patch depending on whether RGB or Depth CENet is used.

The inverse of the transformation is denoted by  $\pi_{\mathcal{M}}^{-1}$ . We use the standard  $SE(3)$  definition of 6 DoF object pose [53]. A model with the given parameterization can easily be rendered to produce pixel-wise ground truth 2D-3D correspondences. Thus, it allows for instant establishment of 2D-3D and 3D-3D correspondence if RGB or depth data is used respectively.

### 3.3 CENet: Correspondence Estimation Network

The architecture of the Correspondence Estimation Network builds on the architecture of DPOD [35]. The network is an encoder-decoder convolutional neural network with skip connections resembling UNet [54]. The network accepts the input of size  $\mathcal{I} \in \mathbb{R}^{128 \times 128 \times D}$ , where  $D$  stands for the number of input channels and depends on the data modality. The network has a common encoder and four separate decoder heads. One head is responsible for predicting binary per-pixel segmentation mask  $\tilde{S} \in \mathbb{R}^{128 \times 128 \times 2}$ , while the other three output tensors, each of which  $\tilde{C}_d \in \mathbb{R}^{128 \times 128 \times 256}$  after softmax corresponds to pixel-wise probabilities of discretized NOCS coordinates. The encoder is based on the 34-layer ResNet [55] architecture, which proved to be both sufficiently effective and fast. The decoders upsample the computed features up to its original size using a stack of bilinear interpolations followed by convolutional layers. We used bilinear upsampling instead of upconvolutions to make inference more rapid and memory-efficient. However, in principle any other architecture for semantic segmentation could be used.

Formulating NOCS coordinate regression problem as discrete classification problem proved to be useful for much faster convergence and for the superior quality of correspondences as was also confirmed in previous work [35], [40]. Initial experiments on direct coordinate regression showed very poor results in terms of correspondence quality. The main reason for the problem was the infinite continuous solution space, i.e.,  $[0, 1]^3$ , where 3 is the number of dimensions and  $[0, 1]$  is the normalized coordinate range of a 3D

model. Classification of the discretized 2D correspondences allowed for a large boost of the output quality by dramatically decreasing the output space, which is now  $256^3$  with 256 being the size of a discretized NOCS dimension.

As a result, the network is trained by optimizing two classification losses. Segmentation loss is defined as a per-pixel binary Dice loss [56]  $\mathcal{L}_{DICE}$ , which helps overcome the imbalanced number of foreground and background pixels and NOCS classification cross entropy loss defined only for foreground pixels (pixels occupied with non-occluded objects of interest) separately per each NOCS dimension  $d$ :

$$\mathcal{L}_{NOCS}^d = \sum_{i=1}^{128} \sum_{j=1}^{128} \mathbb{1}[\mathcal{I}_{i,j} \text{ is foreground}] \mathcal{L}_{cls}(\tilde{C}_{d,i,j}, C_{d,i,j}^{gt}) \quad (3)$$

Which results in the total per-image loss:

$$\mathcal{L} = \alpha \mathcal{L}_{DICE} + \sum_{d=1}^3 \mathcal{L}_{NOCS}^d \quad (4)$$

where  $\alpha$  is a weight factors set to 5 in all our experiments.

### 3.4 Inference with The Correspondence Estimation Network

Given YOLO predictions, we explored three different ways to first get the dense correspondences and then utilize them for pose estimation. The first approach follows the conventional DPOD paradigm of running an RGB correspondence network followed by the PnP with RANSAC. Alternatively, if depth maps are available, it is possible to project the predicted correspondences from 2D into 3D and then use the Kabsch algorithm with RANSAC to predict the pose from 3D-3D correspondences. Depth maps can also directly be used instead of RGB for inferring segmentation masks and 3D-3D correspondences. In this case, Kabsch+RANSAC is again used for pose estimation.

### 3.5 Multi-View Refinement With Differentiable Renderer

The overall pipeline of the proposed refiner is provided in Figure 2. The key idea of the refiner is to predict NOCS correspondences and poses separately for each frame with DPODv2 as described above. Then, if several views with known relative camera positions are available, the predicted poses can be refined to produce a pose which is plausible given correspondences in all frames and the geometric constraints imposed by the relative camera poses. This is achieved by defining a loss function which penalizes pixel-wise difference between predicted NOCS coordinates with CENet, and NOCS coordinates corresponding to the object rendered in the predicted camera poses. The loss function is implemented using the differentiable renderer and minimized iteratively by computing a pose update  $\mathbf{T}_\Delta \in SE(3)$  using gradient descent at each step.

We rely on the assumption of the availability of relative camera transformations. There are multiple ways how they can be estimated. Camera transformations can be obtained by placing the object on the markerboard and either using an actual multi-camera system or using a single camera but moving the markerboard. In the scenario of robotic grasping, a camera can be mounted on the robotic arm to enable observation of the object from several viewpoints. Alternatively, SLAM methods could be used to estimate the transformations. While relative transformations are assumed to be known, the object poses still remain unknown and cannot be inferred directly from the relative camera transformations.

The differentiable renderer is used for a differentiable definition of the following functions: binary foreground/background rendering  $S : \mathcal{M} \times SE(3) \rightarrow \{0, 1\}^{W \times H}$  and NOCS rendering  $C : \mathcal{M} \times SE(3) \rightarrow [0, 1]^{W \times H \times 3}$ . We will omit their dependence on the CAD model  $\mathcal{M}$  to keep the notation concise. For each set of images used for multi-view refinement, one of the images is taken as the reference frame, and then for each  $f$ -th image, its relative pose  $\mathbf{\Xi}_{ref \rightarrow f} \in SE(3)$  is recomputed w.r.t. to the reference frame. Initial pose hypotheses  $\mathcal{T} := \{\mathbf{T}_1, \dots, \mathbf{T}_N\}$  are estimated separately for each frame with PnP+RANSAC. The pose of the object in the coordinate system of the reference frame is later denoted by  $\mathbf{T}_{pr}$ .  $\mathbf{T}_s \in SE(3)$  is a symmetry transformation associated with each object, which must be adjusted to produce consistent NOCS maps for symmetric objects. Computation of  $\mathbf{T}_s$ , described in the data preparation section, is closed-form and quick but has to be performed for each object in each frame after each iteration.

For a pixel  $p$  in the  $f$ -th frame, the loss function is defined as follows:

$$\mathcal{L}_{f,p} := \rho \left( \pi^{-1} \left( \tilde{\mathbf{C}}_{f,p} \right), \pi^{-1} \left( C \left( \mathbf{\Xi}_{ref \rightarrow f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \cdot \mathbf{T}_s \right)_p \right) \right) \quad (5)$$

where  $\rho$  is an arbitrary distance function. In a nutshell,  $\tilde{\mathbf{C}}_{f,p}$  is a predicted NOCS correspondence, while  $C \left( \mathbf{\Xi}_{ref \rightarrow f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \cdot \mathbf{T}_s \right)_p$  projects the model in the pose  $\mathbf{T}_\Delta \cdot \mathbf{T}_{pr}$  onto the corresponding frame and computes a NOCS coordinate for that pose. The inverse transformation allows us to relate the pixel-wise discrepancy to the actual error in the 3D coordinate system of the model.

On the level of the full set of images used for refinement, the objective of the refinement is defined as:

$$\mathbf{T}_\Delta^* = \underset{\mathbf{T}_\Delta \in SE(3)}{\operatorname{argmin}} \sum_{f=1}^N \sum_{p \in \mathcal{I}} \left[ \tilde{\mathbf{S}}_{f,p} \cdot S \left( \mathbf{\Xi}_{ref \rightarrow f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \cdot \mathbf{T}_s \right)_p \right] \cdot \mathcal{L}_{f,p} \quad (6)$$

Here,  $\left[ \tilde{\mathbf{S}}_{f,p} \cdot S \left( \mathbf{\Xi}_{ref \rightarrow f} \cdot \mathbf{T}_\Delta \cdot \mathbf{T}_{pr} \right)_p \right]$  serves as an indicator function which shows whether or not the pixel is foreground in both the rendered and the predicted NOCS maps.

Choice of the reference frame might have a significant effect on the effectiveness of pose refinement, especially in case of occlusions which cause bad initial poses. We aim at choosing a reference frame in such a way that its pose has the lowest error when re-projected to other frames. To deal with degenerate cases when there is no overlap or very small overlap between the projected model and the predicted foreground segmentations, the loss is scaled by their intersection over union as follows:

$$\underset{ref \in [1, \dots, N]}{\operatorname{argmin}} \frac{1}{K} \sum_{f=1}^N \frac{\sum_{p \in \mathcal{I}} \left[ \tilde{\mathbf{S}}_{f,p} \cdot S \left( \mathbf{\Xi}_{ref \rightarrow f} \cdot \mathbf{T}_{ref} \cdot \mathbf{T}_s \right)_p \right] \cdot \mathcal{L}_{f,p}}{IOU \left( \tilde{\mathbf{S}}_f, S \left( \mathbf{\Xi}_{ref \rightarrow f} \cdot \mathbf{T}_{ref} \cdot \mathbf{T}_s \right) \right)} \quad (7)$$

where  $K$  stands for the number of frames with non-zero loss. Additionally,  $\operatorname{argmin}$  ignores zero values, as they correspond to degenerate poses that are not re-projected correctly onto other frames.

### 3.6 Implementation Details

Our pipeline is implemented using the Pytorch [57] deep learning framework. All the experiments were conducted on an Intel Core i9-9900K CPU 3.60GHz with NVIDIA Geforce RTX 2080 TI GPU. YOLO was trained with the ADAM optimizer with a constant learning rate of  $1 \times 10^{-3}$  and weight decay of  $3 \times 10^{-5}$  for 100 epochs. Correspondence estimation network was trained for 240 epochs with the learning rate of  $5 \times 10^{-5}$  which was decreased after 40th, 120th and 200th epochs. We used EPnP [33]+RANSAC implementation from OpenCV [58] and point-to-plane ICP from Open3D [59]. For Kabsch+RANSAC we used our own minimalistic implementation.

The correspondence inference network uses  $128 \times 128$  images for training and for testing, while YOLO is trained to predict tight bounding boxes around the objects. Therefore, the shortest side of each bounding box predicted by YOLO is padded to match the longest side, after which the corresponding image patch is resized preserving the aspect ratio. During training, small random imprecisions are added to each side of the ground truth bounding box to ensure that CENet is robust to noisy YOLO predictions. The ablation studies show the effectiveness of this approach, as the quality of predicted correspondences does not deteriorate much when the predicted bounding boxes are used instead of the ground truth boxes.

When training on synthetic data, the problem of domain adaptation becomes one of the main challenges. Training the network without any prior parameter initialization makes generalization to the real data difficult if possible at all. A simple solution to this problem was proposed in several works, including [43], [60], where they freeze the first layers of the network trained on a large dataset of real images, e.g., ImageNet [61] or MS COCO [62], for the object classification

task. The common observation of the authors is that these layers, learning low-level features, very quickly overfit to the perfect object renderings. We found out that this setup is only partially needed for our networks. We froze first 20 layers when we trained YOLO. The weights were initialized from the YOLO trained on the MS COCO dataset [62]. In the correspondence estimation network, the weights were initialized from the Resnet pretrained on ImageNet [61] and then were all modified during training. When the correspondence network was trained on depth maps, the first layer was instead initialized randomly and then all layers were modified during training. We have found out that replacing BatchNorm [63] with InstanceNorm [64] helps overcome overfitting and allows for reliable and stable training in this case.

We used the Soft Rasterizer renderer [65] in all experiments with the multi-view refiner. The general robust function [66] was used as a distance measure  $\rho$  in the loss function to effectively deal with outliers. Continuous rotation parameterization from [67] was used to achieve faster and more stable convergence during the optimization procedure in comparison to quaternions and Euler angles.

## 4 DATA PREPARATION

When it comes to data preparation for the tasks of object detection and pose estimation, two distinctive paradigms are observed. The most popular and conventional one follows the standard way of machine learning of collecting a large database of labeled data. Unfortunately, pose estimation requires more involved labels, which cannot be produced manually, in contrast to the standard classification or segmentation labels. Pose labeling requires sophisticated multi-step procedures, such as in [7], [52]. In spite of those challenges, real label train data remain the predominant choice of data especially because it allows to achieve excellent performance on academic benchmarks. In the deep learning on point clouds, real train data are also predominant.

On the other hand, it is possible to leverage advances in computer graphics and to rely solely on synthetically rendered train data if object CAD models are available. The advantages of this include virtually unlimited number of images and poses with no labeling effort. However, the domain gap between the real and the rendered images comes into play to hinder the detection rate and pose accuracy. In the past, the predominant way to train a deep neural network on synthetic data was to render the object on random backgrounds, as was done in DPOD [35], SSD6D [68] and in [60]. On the other hand, complete photo-realistic scene rendering is becoming more popular [69].

### 4.1 RGB Data Preparation

**Real Data Preparation.** Preparation of real train data is rather straightforward. Given an image, CAD models, and associated ground truth poses, it is enough to render present objects in ground truth poses. Binary segmentation masks and per-pixel NOCS coordinates are obtained by rendering the models in the ground truth poses. Following the paradigm of the previous works [32], [35] object patches corresponding to the visible object parts were extracted and inpainted in random backgrounds sampled

from MSCOCO [62]. This ensures that the network learns object-specific features rather than overfits to backgrounds, leading to better generalization to new unseen scenes. For training the detector, a random number of objects were added to each background. For the correspondence estimation network, only one object is in-painted. The object patch is resized randomly to simulate different distances from camera. Additionally, a random in-plane rotation is performed. Then, we use standard color augmentations used by the backbone networks. When training the correspondence network, random occlusions also have to be added to make the network more robust. Realistically looking occlusions are approximated by sampling a random patch, corresponding to a different object, which is then in-painted randomly atop of the object of interest. Occlusions are added at random, but it is made sure of that the target object is still visible in the patch.

**Synthetic Data Preparation.** We relied on the PBR images provided by the BOP challenge organizers for the preparation of synthetic training data. Those images were produced by simulation using BlenderProc [70]. No additional transformations were performed on those images apart from random photometric augmentations, such as brightness augmentation. Usage of the provided PBR images also ensured more fairer comparison of our methods to the other methods on the BOP challenge datasets by making sure that the competing methods were trained on exactly the same data.

### 4.2 Depth Data Preparation

In general, training deep learning networks on depth is harder than on RGB images due to holes in the depth and a much larger range of values. To get rid of empty depth values, we fill them in using nearest neighbor interpolation. To alleviate the range problem, depth maps are parameterized similar to PointNet++ [71] by replacing a depth value in each pixel with a distance  $t_i$  to the mean depth in the neighborhood of size  $T$ :

$$\mathcal{I}_{i,j} = \mathcal{I}_{i,j}^{raw} - \frac{1}{T^2} \sum_{x=-T}^T \sum_{y=-T}^T \mathcal{I}_{i-x,j-y}^{raw} \quad (8)$$

for  $i, j \in \mathbb{R}^{128}$  and  $\mathcal{I}^{raw}$  being the original depth image with already interpolated missing values. This depth parameterization ensured simpler and more stable training. We set the neighborhood radius  $T$  to 5 in our experiments.

In train time, depth maps undergo several random augmentations. First, random holes of random size are added to the depth map and then interpolated accordingly. Then, random Perlin noise [72] is added to the background, that results in smooth random backgrounds, which is especially important for synthetically generated data. Foreground depth values are augmented with random Gaussian noise. Synthetic train data was again taken from the synthetic PBR images.

### 4.3 Handling Object Symmetries

While there is a large body of theoretical work on dealing with ambiguities induced by object symmetries, for instance [73], [74], [75], we followed a more conventional

and hands-on approach. Similarly to the DPOD and SSD6D papers, we relied on training on consistent and unambiguous ground truth NOCS, rather than inferring multiple pose hypotheses as in [73], [75]. While DPOD and SSD6D were trained only on images of objects taken from non-ambiguous poses, we used all the images and then disambiguated the poses in order to render consistent NOCS coordinates as visualized in Figure 3. There are two main symmetry types in the used datasets: a continuous rotation symmetry around an axis, when every rotation around the axis results in an identically looking object, and discrete symmetry around an axis.

For objects, which are fully symmetric around an axis, NOCS maps are rendered from a consistent camera location in the object coordinate system as visualized in Figure 3a. Typically, the object is symmetric around its Z axis. Given an object pose  $R \in SO(3), t \in \mathbb{R}^3$ , which transforms from the model coordinate system into the camera coordinate system, the camera location in the model coordinate system is given by  $-R^\top t$ . To produce consistent NOCS maps, we rotate the object around Z axis in such a way, that a camera location always lies in the YZ plane of the model system, i.e.  $R_z \in SO(3)$  such that the X coordinate of  $-R_z R^\top t$  is 0. This makes the object always visible from the same position.  $R_z$  can easily be computed in a closed form by computing an angle between vectors  $\left( (-R^\top t)_x, (-R^\top t)_y, 0 \right)$  and  $(0, 1, 0)^\top$ . The final disambiguated object pose is defined by  $RR_z^\top$  and  $t$ .

Handling objects with discrete symmetries in respect to the plane is rather straightforward. Assuming for the illustration purposes that rotating the object around Z axis by 180 degrees (denoted by  $R_z^{180}$ ) produces an identically looking object as in Figure 3b. One of the symmetry sides is taken as the base region of the symmetry, denoted by blue in the figure. Then, a camera location is computed and if the camera lies on the opposite side of the object, the rotation is updated as  $RR_z^{180}$ . Basically, it ensures that the object is always visible from the same side. Even though [73] noted that this transformation might be unfavourable for neural networks due to its non-continuity, it works reliably in practice.

## 5 EXPERIMENTS

### 5.1 Datasets

Over the past decade, numerous datasets for object detection and 6 DoF pose estimation have been proposed [5], [7], [18], [52], [76], [77], [78], [79], [80]. However, in this work we focus on the following datasets: Linemod [18], Occlusion [5] (LMO), TLESS [52] and Homebrewed [7] (HBD).

Linemod [18] is a classical dataset for object pose estimation. It is still an actively used benchmark even though it dates back to the pre-deep learning era. The dataset provides researchers with 13 3D object models. Each object comes with approximately 1200 annotated images. The images exhibit strong background clutter but almost no occlusions. In experiments with real training data, we used exactly the same training/testing split as in BB8 [30]. In the experiments with synthetic data, we used the entire datasets as test set, analogously to SSD6D [12] and DPOD [35]. We

report the standard ADD recall [18] in all experiments, as it is the most widely reported metric for this dataset.

Occlusion [5] is an extension of the Linemod dataset, which consists of one Linemod sequence with 8 annotated objects. This dataset is a step forward to more complex datasets with occlusions of various degrees and a varying number of objects. We use the BOP subsequence of the dataset and report the Average Recall (AR) score of the BOP challenge. This allows us to fairly compare our method to all other methods participating in the challenge.

The authors of the Homebrewed [7] advocated the importance of training on synthetic data. Therefore, the dataset provides no train images, but has labeled validation set and a test set with hidden labels, as in other major machine learning datasets. Here, we also report the AR score from the BOP challenge. All ablation studies were conducted on the publicly available validation set.

The TLESS dataset [52] provides researchers with 30 objects and 20 scenes comprising of various number of objects. The dataset includes two kind of object models: precise untextured CAD models and reconstructions of relatively low quality. There is a clear separation between train and test images, with train images not coming from the test domain. The dataset exhibits three main challenges: 1) texture-less objects, which are also similar to each other, 2) heavy occlusions and 3) various object symmetries.

As the proposed approach can be trained both on real and synthetic data and on data of different modalities, we experimented with each option given the availability of this data type. Table 1 summarizes which data types were used and whether synthetic or real data was used in each experiment. YOLO was trained only on RGB data as it was proven that in previous works that object detection networks can be relatively easy trained both on real and synthetic data thanks to transfer learning [12], [60]. The correspondence estimation network was trained either on RGB images or on depth maps. In our experiments with the Homebrewed dataset, we completely relied on synthetically simulated train data as the dataset does not provide real train data. Real train depth data from the TLESS dataset was also not used as it is too simplistic and has no background depth to train the correspondence estimation network. We also experimented with ICP refinement to see how it compares to the proposed multi-view refiner. Exactly the same parameters of ICP were used on all datasets.

In the remaining of the section a following naming convention for our approach is used:

- *RGB* - both YOLO and CENet operate on RGB images, the final pose is computed using PnP+RANSAC;
- *RGBD* - YOLO operates on RGB images, while CENet takes a depth map as its input. The final pose is computed with Kabsch+RANSAC;
- *RGB + D-Kabsch* - YOLO and CENet use RGB as input, but then the predicted correspondences are projected into 3D, and the pose is computed with Kabsch+RANSAC.

TABLE 2: **Pose estimation performance on Linemod on RGB images of methods trained on synthetic data:** The table reports the percentages of correctly estimated poses w.r.t. the ADD score. Our approach sets the new state of the art both among the methods trained on real data and the methods trained on synthetic data. Our refiner outperforms other RGB refiners. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data Method	Synthetic				Synthetic			
	SSD6D [12]	AAE [50]	DPOD [35]	Ours	SSD6D [12]	DPOD [35]	Ours	Ours
<b>Refinement</b>	-				DL [43]	DL [35]	2 calib. views	4 calib. views
Ape	2.60	3.96	37.22	<b>62.14</b>	-	55.23	98.54	<b>100.00</b>
Bvise	15.10	20.92	66.76	<b>88.39</b>	-	72.69	<b>100.00</b>	<b>100.00</b>
Cam	6.10	30.47	24.22	<b>92.51</b>	-	34.76	99.67	<b>100.00</b>
Can	27.30	35.87	52.57	<b>96.66</b>	-	83.59	<b>100.00</b>	<b>100.00</b>
Cat	9.30	17.90	32.36	<b>86.17</b>	-	65.1	99.66	<b>100.00</b>
Driller	12.00	23.99	66.60	<b>90.15</b>	-	73.32	99.83	<b>100.00</b>
Duck	1.30	4.86	26.12	<b>54.86</b>	-	50.04	99.68	<b>100.00</b>
Eggbox	2.80	81.01	73.35	<b>98.64</b>	-	89.05	97.70	<b>99.04</b>
Glue	3.40	45.49	74.96	<b>95.41</b>	-	84.37	97.70	<b>98.03</b>
Holep.	3.10	17.60	24.50	<b>27.08</b>	-	35.35	94.01	<b>99.03</b>
Iron	14.60	32.03	85.02	<b>98.26</b>	-	98.78	<b>100.00</b>	<b>100.00</b>
Lamp	11.40	60.47	57.26	<b>91.04</b>	-	74.27	99.51	<b>100.00</b>
Phone	9.70	33.79	29.08	<b>74.34</b>	-	46.98	<b>100.00</b>	<b>100.00</b>
<b>Mean</b>	9.10	28.65	50.00	<b>81.20</b>	34.1	66.42	98.95	<b>99.70</b>
<b>Time (ms)</b>	-	24	-	32	-	-	202.00	132.00

## 5.2 Results

**Linemod.** We start this section by discussing the quantitative results of single object pose estimation on the Linemod dataset. Table 2 and Table 3 present the accuracy of pose estimation on monocular RGB images of methods trained on synthetic and real data respectively. We report run times of each algorithm if the corresponding papers clearly state them. Run times are taken from the original papers, meaning that they are obtained using non-identical hardware. The table reports the percentages of correctly estimated poses w.r.t. the ADD score. The tables compare separately methods trained on synthetic data and methods trained on real data and methods with or without refinement. On synthetic data, the original DPOD reached the state of the art results beating the second best method almost by a factor of 2. The proposed approach further improved the results, surpassing DPOD by over 30%, making the ADD score comparable even to ADD scores of some of the methods trained on real data. If real training data is used, the proposed approach reaches the

best results among other methods trained on real data. The proposed multi-view refiner clearly outperforms all other RGB-based refiners on the dataset even if only 2 views are used for refinement. It also proves that the refinement method handles symmetric object, such as Glue and Eggbox, well.

Table 4 and Table 5 present quantitative results on Linemod datasets in case depth data is available for methods trained on synthetic and real data respectively. Only PPF, PointFusion, DenseFusion and our method use depth as input, all other methods use it only for pose refinement. Our detector easily performs on par or outperforms other methods even if no refinement is applied. It also beats the PPF-based solutions. Another important conclusion from these tables is relative independence of pose accuracy on whether synthetic or real data or which data modality was used for training. Due to the simplicity of the dataset and to the lack of occlusions, correspondences predicted from depth maps prove to be good enough for excellent pose

TABLE 3: **Pose estimation performance on Linemod on RGB images of methods trained on real data:** The table reports the percentages of correctly estimated poses w.r.t. the ADD score. Our approach sets the new state of the art both among the methods trained on real data and the methods trained on synthetic data. Our refiner outperforms other RGB refiners. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data Method	Real					Real					
	Pix2Pose [36]	DPOD [35]	PVNet [34]	CDPN [37]	HybridPose [42]	Ours	BB8 [30]	PoseCNN [76]	DPOD [35]	Ours	Ours
<b>Refinement</b>	-						DL [30]	DeepIM [15]	DL [35]	2 calib. views	4 calib. views
Ape	58.10	53.28	43.62	64.38	63.1	80.09	40.40	76.95	87.73	98.66	<b>100.00</b>
Bvise	91.00	95.34	<b>99.90</b>	97.77	<b>99.9</b>	99.71	91.80	97.48	98.45	<b>100.00</b>	<b>100.00</b>
Cam	60.90	90.36	86.86	91.67	90.4	<b>99.21</b>	55.70	93.53	96.07	<b>100.00</b>	<b>100.00</b>
Can	84.40	94.10	95.47	95.87	98.5	<b>99.60</b>	64.10	96.46	99.71	<b>100.00</b>	<b>100.00</b>
Cat	65.00	60.38	79.34	83.83	89.4	<b>95.11</b>	62.60	82.14	94.71	<b>100.00</b>	<b>100.00</b>
Driller	76.30	97.72	96.43	96.23	98.5	<b>98.91</b>	74.40	94.95	98.80	<b>100.00</b>	<b>100.00</b>
Duck	43.80	66.01	52.58	66.76	65.0	<b>79.54</b>	44.30	77.65	86.29	98.12	<b>100.00</b>
Eggbox	96.80	<b>99.72</b>	99.15	<b>99.72</b>	<b>100.0</b>	99.63	57.80	97.09	99.91	99.68	<b>100.00</b>
Glue	79.40	93.83	95.66	99.61	98.8	<b>99.81</b>	41.20	<b>99.42</b>	96.82	97.46	98.84
Holep.	74.80	65.83	81.92	85.82	<b>89.7</b>	72.30	67.20	52.81	86.87	99.81	<b>100.00</b>
Iron	83.40	<b>99.80</b>	98.88	97.85	<b>100.0</b>	99.49	84.70	98.26	<b>100.00</b>	<b>100.00</b>	<b>100.00</b>
Lamp	82.00	88.11	99.33	97.89	<b>99.5</b>	96.35	76.50	97.5	96.84	<b>100.00</b>	<b>100.00</b>
Phone	45.00	74.24	92.41	90.75	94.9	<b>96.88</b>	54.00	87.72	94.69	<b>100.00</b>	<b>100.00</b>
<b>Mean</b>	72.40	82.98	86.27	89.86	91.3	<b>93.59</b>	62.70	88.61	95.15	99.52	<b>99.91</b>
<b>Time (ms)</b>	100-167	36	40	30	1000	31	330	-	-	201.00	131.00

TABLE 4: **Pose estimation performance on Linemod on depth and RGBD images:** The table reports the percentages of correctly estimated poses w.r.t. the ADD score. The proposed detector shows state of the art results both if only real or only synthetic train data is used. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data Modality	Real					Synthetic		RGB + D-Kabsch	
	RGBD	Depth	Depth	RGBD	RGBD	RGBD	RGBD	RGB + D-Kabsch	RGB + D-Kabsch
Method	AAE [50]	PPF [2]	PPF++ [81]	Ours	Ours	SSD6D [12]	Brachmann et. al. [5]	Ours	Ours
Refinement	ICP	ICP	ICP	-	ICP	ICP	iterative	ICP	-
Ape	20.55	86.50	98.50	87.54	91.91	-	-	98.38	<b>98.79</b>
Bvise	64.25	70.70	99.80	<b>99.92</b>	<b>99.92</b>	-	-	<b>99.92</b>	<b>99.92</b>
Cam	63.20	78.60	99.30	96.42	<b>97.84</b>	-	-	99.75	99.75
Can	76.09	80.20	98.70	98.16	98.66	-	-	<b>99.75</b>	99.58
Cat	72.01	85.40	99.90	99.41	99.83	-	-	<b>100.00</b>	<b>100.00</b>
Driller	41.58	87.30	93.40	97.55	97.64	-	-	<b>98.82</b>	<b>98.82</b>
Duck	32.38	46.00	98.20	90.82	94.26	-	-	98.41	<b>98.96</b>
Eggbox	98.64	97.00	98.80	98.48	98.80	-	-	<b>99.52</b>	<b>99.52</b>
Glue	96.39	57.20	75.40	99.26	99.34	-	-	<b>99.84</b>	99.75
Holep.	49.88	77.40	98.10	93.45	96.12	-	-	<b>97.81</b>	97.17
Iron	63.11	84.90	98.30	48.35	49.21	-	-	<b>100.00</b>	99.91
Lamp	91.69	93.30	96.00	50.12	50.20	-	-	99.02	<b>99.59</b>
Phone	70.96	80.70	<b>98.60</b>	96.70	97.02	-	-	98.15	98.07
<b>Mean</b>	64.67	78.86	96.38	88.94	90.06	90.90	98.3	99.18	<b>99.22</b>
<b>Time (ms)</b>	224	-	-	49	58	100	545	55	49

accuracy if the Kabsch algorithm is used. In comparison to deep learning methods which utilize real train data [13], [82], we did not explicitly rely on sensor fusion and used purely RGB or purely depth for correspondence estimation. This experiment also demonstrates the effectiveness of our proposed refiner, which easily reaches the performance scores of the best RGBD methods.

**Occlusion.** Moving to a more difficult scenario, where a varying number of objects is present in each image, we start evaluation with experiments on the Occlusion dataset. In this case, the ability to predict accurate segmentation masks and precise correspondences becomes of critical importance. The results for all types of detectors and training data types are summarized in Table 6. The table clearly shows that our method outperforms all other dense correspondence methods, e.g. CDPN, EPOS and Pix2Pose, while being only worse than CosyPose if PBR training data and no refinement were used. With the multi-view refinement our methods easily outperform all other RGBD methods and shows similar or slightly worse results than the top performing RGBD

methods with ICP refinement. It also always improves the poses and the final score in contrast to ICP which worsened the AR score for some of the experiments.

Adding depth information and predicting segmentations and correspondences from depth rather than from RGB actually hurts the performance, as can be seen in the *RGBD* block of the table. It indicates inferior quality of the predicted segmentations and correspondences, that are not sufficient for reliable and precise pose estimation. Additionally, due to imprecise segmentations, Kabsch and ICP can potentially align the object to nearby objects. On the other hand, if depth is only used for the Kabsch algorithm with the correspondences predicted by the RGB correspondence network, a larger performance boost is observed. In this configuration, the proposed method sets the state-of-the-art results on the HBD dataset even without the ICP refinement. ICP boosts the results even further. This experiment and later the experiments on TLESS show that correspondence and segmentation prediction from pure depth maps works well only on very simple datasets such as Linemod, where

TABLE 5: **Pose estimation performance on Linemod on depth and RGBD images:** The table reports the percentages of correctly estimated poses w.r.t. the ADD score. The proposed detector shows state of the art results both if only real or only synthetic train data is used. Run times are provided as they are reported in the original papers using non-identical hardware.

Train Data Modality Method	Real									
	RGBD PointFusion [82]	RGBD DF [13]	RGBD DF [13]	RGBD Brachmann et. al. [5]	RGBD Ours	RGBD Brachmann et. al. [83]	RGBD Ours	RGBD PVN3D [41]	RGB + D-Kabsch Ours	RGB + D-Kabsch Ours
Refinement	-	-	DL [13]	-	-	iterative	ICP	-	ICP	-
Ape	70.40	79.50	92.30	-	97.34	-	97.62	97.30	97.72	<b>98.56</b>
Bvise	80.70	84.20	93.20	-	99.90	-	<b>100.00</b>	99.70	<b>100.00</b>	99.90
Cam	60.80	76.50	94.40	-	99.41	-	99.70	99.60	<b>99.90</b>	<b>99.90</b>
Can	61.10	86.60	93.10	-	99.41	-	99.51	99.50	<b>100.00</b>	99.80
Cat	79.10	88.80	96.50	-	99.90	-	99.90	99.80	<b>100.00</b>	<b>100.00</b>
Driller	47.30	77.70	87.00	-	97.52	-	98.02	99.30	<b>99.90</b>	<b>99.90</b>
Duck	63.00	76.30	92.30	-	94.74	-	97.09	98.20	98.31	<b>99.25</b>
Eggbox	99.90	99.90	99.80	-	99.63	-	99.72	<b>99.80</b>	99.72	99.72
Glue	99.30	99.40	100.00	-	99.71	-	99.71	<b>100.00</b>	<b>100.00</b>	<b>100.00</b>
Holep.	71.80	79.00	92.10	-	99.53	-	99.71	99.90	99.71	99.61
Iron	83.20	92.10	97.00	-	98.88	-	99.18	99.70	<b>100.00</b>	99.90
Lamp	62.30	92.30	95.30	-	99.14	-	99.04	99.80	<b>100.00</b>	99.90
Phone	78.80	88.00	92.80	-	99.43	-	99.43	99.50	<b>100.00</b>	<b>100.00</b>
<b>Mean</b>	73.70	86.20	94.30	98.10	98.81	99.00	99.13	99.40	99.64	<b>99.73</b>
<b>Time (ms)</b>	-	-	-	545	49	-	57	-	55	49

TABLE 6: **Pose estimation performance comparison on the Occlusion dataset:** Results are reported in terms of the Average Recall score. The results prove the effectiveness of the proposed approach on all used data modalities. Run times are provided as they are reported in the BOP challenge [14] using non-identical hardware. PPF-based methods, labeled with \* in the Time column, use only CPU.

Train data	Data modality	Method	Refinement	VSD	MSSD	MSPD	AR	Time (s)
PBR	RGBD	CosyPose [1]	ICP	0.567	0.748	0.826	0.714	8.289
		Ours + Kabsch	-	0.565	0.748	0.788	0.700	0.334
		Ours + Kabsch	ICP	0.557	0.749	0.787	0.698	0.387
		PVNet [34]	ICP	0.502	0.683	0.73	0.638	-
		CDPN [37]	ICP	0.469	0.689	0.731	0.630	0.506
		Pix2Pose [36]	ICP	0.473	0.631	0.659	0.588	5.191
		Ours	ICP	0.472	0.621	0.654	0.582	0.398
		Ours	-	0.422	0.58	0.621	0.541	0.325
	RGB	Ours	4 calibrated views	0.572	0.735	0.777	0.695	2.479
		Ours	2 calibrated views	0.520	0.690	0.771	0.660	1.353
		CosyPose [1]	-	0.480	0.606	0.812	0.633	0.550
		Ours	-	0.432	0.560	0.761	0.584	0.274
		PVNet [34]	-	0.428	0.543	0.754	0.575	-
		CDPN [37]	-	0.393	0.537	0.779	0.569	0.279
		EPOS [39]	-	0.389	0.501	0.750	0.547	0.468
		Pix2Pose [36]	-	0.233	0.307	0.550	0.363	1.310
synt	RGB	EPOS [39]	-	0.29	0.38	0.659	0.443	0.487
		DPOD [35]	-	0.101	0.126	0.278	0.169	0.172
mix	RGBD	AAE [49]	ICP	0.208	0.218	0.285	0.237	1.197
	RGB	Multi-Path AAE	-	0.15	0.153	0.346	0.217	0.200
-	depth	AAE [49]	-	0.09	0.095	0.254	0.146	0.201
		Drost, PPF [2]	ICP	0.437	0.563	0.581	0.527	15.947*
real	RGBD	Drost, PPF [2]	ICP, 3D edges	0.409	0.525	0.542	0.492	3.389*
		Ours + Kabsch	ICP	0.557	0.739	0.773	0.69	0.393
		Ours + Kabsch	-	0.557	0.721	0.759	0.679	0.329
		Ours	ICP	0.478	0.62	0.66	0.586	0.453
	RGB	Ours	-	0.403	0.553	0.595	0.517	0.343
		Ours	4 calibrated views	0.598	0.751	0.793	0.714	1.326
		Ours	2 calibrated views	0.528	0.692	0.764	0.661	2.201
		Ours	-	0.427	0.55	0.728	0.568	0.278

the object is not occluded and clearly stands out from the scene. With the more challenging datasets, RGB-based correspondence estimation starts to dominate due to richer RGB information available for the network.

**Homebrewed.** This dataset is the next step towards more complicated pose estimation benchmarks, as it consists of a larger number of more diverse objects and a varying degree of occlusions. The comparison is provided in Table 7. Pose accuracy is reported in terms of Average Recall as defined in [14]. As indicated before in Table 1, we used exclusively synthetic PBR train data data. Our method easily outperforms all other RGB methods. With the multi-view refinement, it performs on par with the best RGBD methods.

**TLESS.** Table 8 compares Average Recall of various configurations of the proposed detector to other participants of the BOP challenge 2020. Since TLESS dataset comes with a separate train set, we were able to test various combinations of YOLO and CENet. Our approach consistently outperforms all competing methods apart from CosyPose. It is interesting that usage of depth does improve the average recall as much as on other datasets. The lack of performance boost can be partially explained by the fact that approximately 16% of images are taken by a camera looking almost parallel to the table with the objects. In this case, Primesense produces mostly empty depth maps for flat surfaces and empty on edges of objects, making pose prediction from occluded objects close to impossible. This also explains why in some cases AR score is lower

if ICP refinement is applied. Again, the proposed multi-view refinement consistently improves the poses even in the presence of large occlusions and imperfect initial poses.

Another important insight from Table 8 and from Table 12 is that the correspondence estimation network trained on synthetic data actually outperforms the one trained on real data provided they are tested on exactly the same patches. There are several explanations to that. The real train data is very limited in the number of images and requires a lot of data augmentation to work on the real test set, while the synthetic data has considerably more images. Additionally, simulated synthetic train data naturally exhibits realistically looking occlusions, which helps generalize to the test images. It is also confirmed by the plot in Figure 5, where CENet trained on synthetic data consistently outperforms the one trained on real data. If ground truth patches are used, the synthetic network achieves AR of 0.729, whereas the real network only 0.535. On the other hand, YOLO trained on real data outperforms YOLO train on synthetic data, which can be seen in much lower AR drop if the full real pipeline is used. As a result, we achieve the best result on TLESS if we combine a real YOLO and a synthetic correspondence estimation network, analogously to how it is done in AAE [49].

### 5.3 Ablation Studies

In the ablation studies, we measure what influences pose estimation the most, how imprecise YOLO detections affect

TABLE 7: **Pose estimation performance comparison on the Homebrewed dataset:** Results are reported in terms of the Average Recall score [14]. The results prove the effectiveness of the proposed approach on all used data modalities. Run times are provided as they are reported in the BOP challenge [14] using non-identical hardware. PPF-based methods, labeled with \* in the Time column, use only CPU.

Train data	Data modality	Method	Refinement	VSD	MSSD	MSPD	AR	Time (s)
PBR	RGBD	Ours + Kabsch	ICP	0.784	0.872	0.879	0.845	0.329
		Ours + Kabsch	-	0.796	0.856	0.867	0.84	0.238
		CosyPose [1]	ICP	0.679	0.719	0.737	0.712	5.326
		CDPNv2 [37]	ICP	0.629	0.757	0.749	0.712	0.713
		Pix2Pose [36]	ICP	0.64	0.721	0.724	0.695	3.248
		Ours	ICP	0.462	0.468	0.473	0.468	0.47
		Ours	-	0.344	0.398	0.403	0.382	0.245
	RGB	Ours	4 calibrated views	0.79	0.857	0.86	0.835	0.584
		Ours	2 calibrated views	0.754	0.84	0.858	0.818	0.926
		CosyPose [1]	8 uncalibrated views	0.691	0.744	0.804	0.746	0.427
		Ours	-	0.642	0.704	0.828	0.725	0.163
		CDPNv2 [37]	-	0.614	0.708	0.845	0.722	0.273
		CosyPose [1]	4 uncalibrated views	0.646	0.685	0.756	0.696	0.445
		CosyPose [1]	-	0.613	0.634	0.721	0.656	0.417
EPOS [39]	-	0.484	0.527	0.729	0.58	0.657		
synt	RGBD	Sundermeyer-IJCV19 [49]	ICP	0.479	0.506	0.533	0.506	1.352
	RGB	Sundermeyer-IJCV19 [50]	-	0.273	0.306	0.461	0.346	0.19
		DPOD [35]	-	0.218	0.262	0.379	0.286	0.18
-	Depth	Vidal-Sensors18 [84]	ICP	0.707	0.704	0.708	0.706	2.608*
		Drost-CVPR10-3D-Edges [2]	ICP, 3D edges	0.618	0.624	0.626	0.623	127.372*
		Drost-CVPR10-3D-Only [2]	ICP	0.593	0.627	0.627	0.615	16.136*
	RGBD	Drost-CVPR10-Edges [2]	ICP,3D edges, images	0.677	0.665	0.67	0.671	144.029*

the quality of correspondences and which data modality is better. Tables 9, 10, 11 and 12 report how pose estimation performance drops if YOLO detections are used instead of ground truth bounding boxes on Linemod and Occlusion, Homebrewed and TLESS datasets respectively. Those tables essentially report the upper bound of what the proposed approach can achieve, assuming perfect 2D detections, and how far from them the results actually are. They also additionally report quality of correspondences. In each object in the image, a correspondence quality is computed as median per-correspondence L2 error between a predicted correspondence and its correct ground truth location. Then, it is averaged across all objects. The metric takes into account object symmetries by choosing an equivalent symmetric pose.

**RGB vs Depth vs RGB + D-Kabsch.** With the advent of deep learning, much of pose estimation research has moved to monocular RGB [1], [12], [35], [42], [49]. Even if depth information was used, it was used in a separate post-

processing step for pose refinement and not for inference. In this paper, we introduced a pose estimation network capable of pose estimation from either modality, which leads to the question of which one is better. On the Linemod dataset (Table 9), RGB-only detectors still lag behind the depth-based counterparts especially if only synthetic data is used for their training. It is also visible from the table that correspondences predicted from RGB tend to be better than those predicted from depth even for such a simplistic dataset. Table 9 clearly shows the superior quality of correspondences estimated from RGB, Nevertheless, depth-based approaches still better than purely RGB approaches because of the simplicity of the dataset and the lack of occlusions.

On the other hand, the situation is different on more complicated datasets with occlusions and background clutter. On the Occlusion dataset, the RGB version of CENet works better than the depth version. Depth brings improvement only if the correspondences are computed from RGB images and then projected onto the point cloud for the Kab-

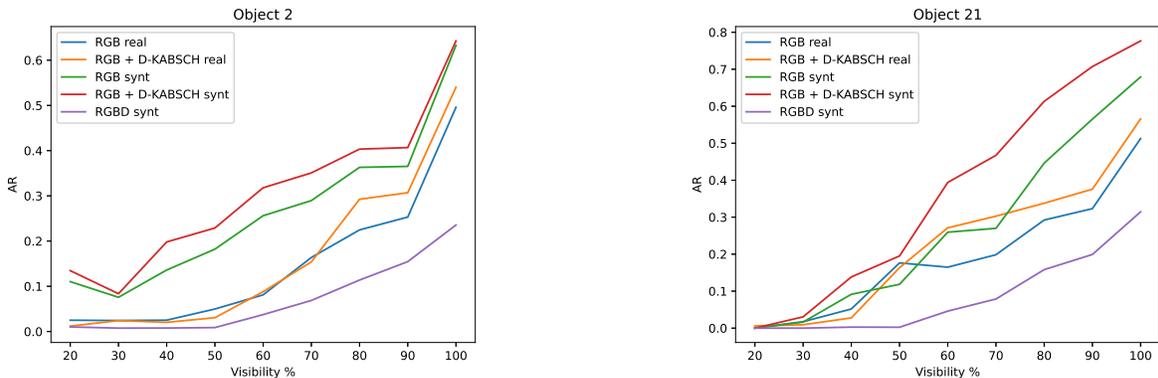


Fig. 5: **Dependence of quality of pose estimation on object's visibility.** Pose quality is reported in terms of the Average Recall [14] on objects 2 and 21 from the TLESS dataset. The plots show that correspondence prediction works from RGB works considerably more reliably than from depth in case of large occlusions. It also shows that synthetic training data allows for more reliable pose estimation of occluded objects.

TABLE 8: **Pose estimation performance comparison on the BOP images of the TLESS dataset:** Results are reported in terms of the Average Recall score [14]. The results prove the effectiveness of the proposed approach on all used data modalities. Run times are provided as they are reported in the BOP challenge [14] using non-identical hardware. PPF-based methods, labeled with \* in the Time column, use only CPU.

Train data	Data modality	Method	Refinement	VSD	MSSD	MSPD	AR	Time (s)
PBR	RGBD	Ours + Kabsch	-	0.646	0.716	0.736	<b>0.699</b>	0.320
		Ours + Kabsch	ICP	0.485	0.632	0.652	0.590	0.523
		CDPNv2 [37]	ICP	0.368	0.449	0.488	0.435	2.486
		Ours	ICP	0.277	0.320	0.339	0.312	0.688
		Ours	-	0.251	0.303	0.312	0.289	0.330
	RGB	Ours	2 calibrated views	0.634	0.707	0.728	0.690	1.293
		Ours	4 calibrated views	0.645	0.710	0.712	0.689	0.886
		CosyPose [1]	-	0.571	0.589	0.761	0.64	0.493
		Ours	-	0.561	0.602	0.744	0.636	0.328
		EPOS [39]	-	0.38	0.403	0.619	0.467	1.992
		CDPNv2 [37]	-	0.303	0.338	0.579	0.407	1.849
synt	RGB	EPOS [39]	-	0.369	0.423	0.635	0.476	1.177
		DPOD [35]	-	0.048	0.055	0.139	0.081	0.206
-	Depth	Vidal-Sensors18 [84]	ICP	0.464	0.575	0.574	0.538	7.063*
		Drost-CVPR10-3D-Only [2]	ICP	0.375	0.478	0.480	0.444	9.204*
		Drost-CVPR10-3D-Edges [2]	ICP, 3D edges	0.370	0.422	0.420	0.404	62.507*
real	RGBD	Drost-CVPR10-Edges [2]	ICP, 3D edges, images	0.469	0.512	0.518	0.500	70.914*
		Ours + Kabsch	-	0.537	0.557	0.586	<b>0.56</b>	0.315
		Pix2Pose [36]	ICP	0.438	0.548	0.549	0.512	4.180
		Ours + Kabsch	ICP	0.416	0.516	0.537	0.490	0.578
real	RGB	Ours	4 calibrated views	0.467	0.541	0.563	0.524	0.727
		Ours	2 calibrated views	0.461	0.530	0.578	0.523	0.973
		Ours	-	0.469	0.491	0.595	0.518	0.367
		Pix2Pose [36]	-	0.261	0.296	0.476	0.344	1.084
		Pix2Pose-Original-ICCV19 [36]	-	0.214	0.238	0.432	0.295	1.522
mix	RGB	CosyPose [1]	8 uncalibrated views	0.773	0.836	0.907	0.839	0.969
		CosyPose [1]	4 uncalibrated views	0.742	0.795	0.864	0.801	0.792
		CosyPose [1]	-	0.669	0.695	0.821	0.728	0.451
		Ours	4 calibrated views	0.679	0.742	0.740	0.72	0.909
		Ours	2 calibrated views	0.665	0.739	0.753	0.719	1.306
		Ours	-	0.579	0.621	0.764	0.655	0.306
		CDPN [37]	-	0.377	0.418	0.674	0.49	0.708
		CDPNv2 [37]	-	0.386	0.426	0.62	0.478	1.852
		Sundermeyer-IJCV19 [50]	-	0.196	0.211	0.504	0.304	0.194
mix	RGBD	Ours + Kabsch	-	0.665	0.738	0.756	0.720	0.7367
		CosyPose [1]	ICP	0.587	0.749	0.767	0.701	0.274
		Ours + Kabsch	ICP	0.503	0.656	0.672	0.610	0.455
		Sundermeyer-IJCV19 [50]	ICP	0.459	0.489	0.514	0.487	0.531
		CDPNv2 [37]	ICP	0.385	0.489	0.516	0.464	2.645

sch algorithm. This is also confirmed by the correspondence error analysis in Table 10, where correspondences computed from RGB are almost two times more precise than the once computed from depth maps. This trend is also noticeable on the more complicated Occlusion dataset (Table 10). It is clear from the table that the correspondence error is much larger due to occlusions. As seen from Table 7, on the Homebrewed dataset the proposed approach trained on RGB only significantly outperforms its counterpart trained on depth, while the RGB variant with the Kabsch algorithm shows outstanding AR. That is explained by less accurate predictions of segmentation masks and correspondences for occluded objects when only depth information is used for that. Table 11 clearly shows that correspondences predicted from RGB are almost five times better than their depth-based counterparts, which explains the difference in pose estimation performance. On TLESS dataset, as seen from Table 8 and Table 12, using depth again does not help at all as the quality of correspondences deteriorate. Depth CENet also produced visually worse segmentations. This phenomenon can be explained by larger degrees of occlusion and camera angles resulting in missing or very noisy depth values. Figure 5, additionally proves strong dependence of pose estimation quality from depth maps on the visibility of objects. It show that RGB-based CENet is

much better capable of handling occlusions than the depth-based CENet.

To sum up, semantic segmentation and correspondence estimation is considerably more precise when done completely from RGB. If depth information is present, it is more beneficial to still predict them from RGB and then project into the 3D space and than use the Kabsch algorithm. Such an approach potentially also allows for alternating between PnP and Kabsch based on the quality of depth maps.

**Real vs Synthetic Data.** As an experiment from [7] showed, training pose estimation networks on real RGB data tends to outperformed the one on synthetic if images and poses come from exactly the same domain. That is still clearly visible on Linemod dataset in Table 2. Detectors trained on synthetic data are still behind their real counterparts even in spite of the progress in synthetic data generation. It is also visible from the worse quality of correspondences as seen in Table 9. The same holds for CENet train on synthetic depth data. On the other hand, as can be seen from TLESS Tables 8 and 12 YOLO benefits a lot from training on real data even if the train data is out of test set domain. At the same time, CENet trained on synthetic data clearly outperforms the CENet trained on real data both in terms of AR and the quality of correspondences due to better pose space coverage.

To sum up, real train data is better for training 2D

TABLE 9: **Pose estimation performance on Linemod on different data modalities and types of train data:** The table reports the percentages of correctly estimated poses w.r.t. the ADD score. An ADD score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median  $L_2$  correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	ADD	Corr. Err.
Linemod	RGB	real	GT	-	93.99	2.48
				2 views	97.42	
				4 views	99.79	
		YOLO	-	93.59	2.52	
			2 views	97.68		
			4 views	99.91		
	synt	GT	-	81.20	4.61	
			2 views	96.47		
			4 views	99.63		
		YOLO	-	80.39	4.67	
			2 views	96.33		
			4 views	99.70		
	RGBD	real	GT	-	99.85	5.31
				ICP	99.85	
				YOLO	98.81	
		YOLO	-	98.13	5.38	
			ICP	99.13		
			ICP	99.13		
RGB + D-Kabsch	real	GT	-	97.99	15.81	
			ICP	98.19		
			YOLO	98.19		
	YOLO	-	88.94	17.32		
		ICP	90.06			
		ICP	90.06			
RGB + D-Kabsch	real	GT	-	99.99	2.48	
			ICP	99.99		
			YOLO	99.73		
	YOLO	-	99.64	2.52		
		ICP	99.64			
		ICP	99.64			
synt	GT	-	99.94	4.61		
		ICP	99.97			
		YOLO	99.97			
YOLO	-	99.22	4.67			
	ICP	99.18				
	ICP	99.18				

detectors. For training correspondence estimation network, synthetic data is beneficial unless train and test data come from the same domain and contain objects in very similar poses.

**Choice of PnP and RANSAC.** In all the reported RGB results we relied on the standard implementation of EPnP [33] and of RANSAC from OpenCV, although uncountable number of improved versions of PnP and RANSAC have been published in the past [85], [86], [87]. Additionally, EPOS [39] reported, that a large improvement can be achieved by simple switching from OpenCV PnP+RANSAC to Graph Cut RANSAC [86] with DLS-PnP [88]. We have tried this combination on scene 3 of TLESS. It resulted in increase of AR from 21.45 to, at most, 24.02. However, we faced two complications. First, due to a large number of correspondences, RANSAC takes over a second per each object, making the entire pipeline slow and not applicable in practice. Secondly, the algorithm seemed to be very sensitive to hyper-parameters, and optimal hyper-parameters are not shared among the objects. As a result, OpenCV implementation proved to be more rapid and robust, even though potentially resulting in sub-optimal AR scores.

**Limitations of CENet.** There are two main limiting factors of the CENet. The first comes from the fact that the reliable pose estimation with correspondences requires their good quality and the right number of them to deal with outliers. Therefore, large occlusions might be problematic, as indicated in Figure 5 and Figure 6. However, pose estimation of occluded object is a common problem for all

TABLE 10: **Pose estimation performance on the Occlusion dataset on different data modalities and type of train data:** The table reports in terms of the AR score [14]. An AR score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median  $L_2$  correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	AR	Corr. Err.
LMO	RGB	real	GT	-	0.601	13.830
				2 views	0.690	
				4 views	0.751	
		YOLO	-	0.568	16.150	
			2 views	0.661		
			4 views	0.714		
	synt	GT	-	0.649	12.275	
			2 views	0.741		
			4 views	0.773		
		YOLO	-	0.584	12.987	
			2 views	0.660		
			4 views	0.695		
	RGBD	real	GT	-	0.534	26.482
				ICP	0.609	
				YOLO	0.517	
		YOLO	-	0.517	26.611	
			ICP	0.586		
			ICP	0.586		
synt	GT	-	0.606	21.267		
		ICP	0.657			
		YOLO	0.582			
	YOLO	-	0.582	22.356		
		ICP	0.541			
		ICP	0.541			
RGB + D-Kabsch	real	GT	-	0.738	13.830	
			ICP	0.757		
			YOLO	0.679		
	YOLO	-	0.679	16.150		
		ICP	0.690			
		ICP	0.690			
synt	GT	-	0.796	12.987		
		ICP	0.793			
		YOLO	0.582			
YOLO	-	0.582	22.356			
	ICP	0.698				
	ICP	0.698				

methods. Another issue, illustrated in Figure 5, arises when CENet receives a patch containing similarly looking objects or object of the same class.

**Multi-View Refinement** We have experimented with the multi-view refinement on all datasets, and on all datasets it proved to be effective, as opposed to ICP which did not always improve the poses. It can be explained by its independence from the quality of depth data and potentially better handling of objects which are occluded in only some of the frames used for refinement. We split each sequence into non-overlapping sets of 2 or 4 images used for refinement. The pose is optimized jointly and then transformed into the coordinate system of each frame. Even though the number of frames is lower than in CosyPose [1], the proposed refiner anyway produces better or similar relative improvement. This is also explained by the fact that CosyPose optimized camera and object poses jointly while we assume the availability of relative camera poses. The CosyPose refiner performs noticeably better on TLESS, but this can be explained by a much better initial poses produced by CosyPose. For each set, we compute relative camera poses from individual ground truth camera poses. In these experiments, the images were split into batches absolutely at random. The analysis of how a choice of the camera sampling strategy affects the final score is provided in the supplementary.

## 6 CONCLUSIONS

In this paper, we extend the Dense Pose Object Detector (DPOD) method, by splitting its inference into three stages

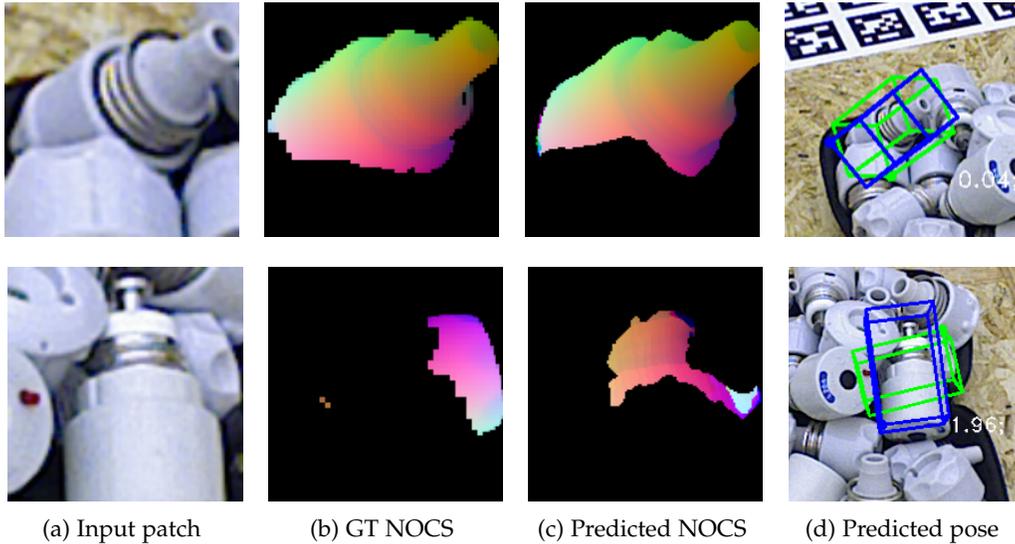


Fig. 6: **Success cases and failure cases.** Top row provides an example of successfully estimated segmentation mask and NOCS correspondences even in case of occlusions and similar objects present in the image patch. The bottom row illustrates a failure case, where the CENet is confused by occlusions and similar objects belonging to other object classes, which leads to an incorrect estimated pose. The green cuboid represents the ground truth pose (up to a symmetry transformation), whereas the blue cuboid represents the estimated pose.

with a optional fourth refinement stage. In the first stage, YOLO outputs 2D bounding boxes of the objects of interest. In the second stage, our unified CENet network predicts foreground object masks and dense 2D-3D correspondences between image pixels and corresponding 3D models either from RGB or depth input modalities with the same architecture. Object’s 6 DoF is then reliably estimated with PnP+RANSAC or with Kabsch+RANSAC depending on the available data modality in the third stage. In the optional fourth stage, predicted poses are refined either with ICP or with the proposed novel multi-view refiner. The second stage network can be trained both on RGB images and on depth masks. We additionally proposed a new multi-view pose refiner based on differentiable rendering, which is used to produce a pose which is globally consistent with dense NOCS correspondences predicted in all frames. The proposed approaches have been tested on four popular datasets, each of which is challenging in its own way: Linemod, Occlusion, Homebrewed and TLESS. The detector shows excellent results on all the datasets and data modalities while still staying fast and scalable. The refiner improves the poses even further while still being reasonably fast. We provide an extensive evaluation and measured effects of the choice of data modality, and of the choice of the training data: synthetic or real. The overall results indicated that RGB is good for correspondence prediction, while depth is good for pose prediction. The more occlusions we have in the data the more benefits we get from the dense correspondences estimated from RGB. In that case, noisy depth still improves the pose with respect to the pose estimated from RGB only.

## 7 SUPPLEMENTARY MATERIAL

**Frame selection strategy** In this experiment, we tested how selection of the frames for joint refinement affects the overall performance of the refiner. Tables 13,14,15,16 summarize these experiments. We experimented with three different

TABLE 11: **Pose estimation performance on the Homebrewed on different data modalities and type of train data:** The table reports the percentages of correctly estimated poses w.r.t. the Average Recall score [14]. An AR score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median  $L_2$  correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	AR	Corr. Err.
HBD	RGB	synt	GT	- 2 views 4 views	0.723 0.808 0.807	9.52
			YOLO	- 2 views 4 views	0.668 0.766 0.783	
	RGBD	synt	GT	- ICP	0.375 0.453	52.69
			YOLO	- ICP	0.362 0.438	
	RGB + D-Kabsch	synt	GT	- ICP	0.843 0.853	9.52
			YOLO	- ICP	0.793 0.801	

strategies: closest view sampling, random sampling and furthest view sampling. Closest view sampling corresponds to the worst case scenario because it does not provide strong enough geometric constraints. In other words, an error in pose estimation from the reference frame might not be necessarily be visible in the other frames used for refinement. Occlusions will cause further problems.

As expected, the closest view sampling performs the worst among these strategies, but poses still improve in most of the cases. Random and furthest sampling tend to perform similarly to each other. This is explained that furthest views do not necessarily correspond to the best possible configuration, as the error might not be visible in them. Additionally, furthest view sampling might result in selecting views which are too far from the reference frame,

TABLE 12: **Pose estimation performance on the TLESS on different data modalities and type of train data:** The table reports the percentages of correctly estimated poses w.r.t. the Average Recall score [14]. An AR score for each data modality and train data type is provided separately on ground truth 2D detections and on YOLO detections. Symmetry-aware median  $L_2$  correspondence error demonstrates the quality of predicted correspondences.

Dataset	Data modality	Data type	Bboxes	Refinement	AR	Corr. Err.
TLESS	RGB	real	GT	-	0.535	26.946
				2 views	0.542	
				4 views	0.548	
		YOLO	-	0.518	27.516	
			2 views	0.523		
			4 views	0.524		
	synt	GT	-	0.729	10.400	
			2 views	0.798		
			4 views	0.795		
	YOLO	-	0.636	20.616		
		2 views	0.689			
		4 views	0.690			
	mix	YOLO	-	0.655	18.373	
			2 views	0.719		
4 views			0.720			
RGBD	synt	GT	-	0.322	34.396	
			ICP	0.354		
	YOLO	-	0.289	41.087		
		ICP	0.312			
real	GT	-	0.583	26.946		
		ICP	0.513			
	YOLO	-	0.560	27.516		
		ICP	0.490			
RGB + D-Kabsch	synt	GT	-	0.812	10.400	
			ICP	0.678		
			YOLO	0.699		
	YOLO	-	0.699	20.616		
		ICP	0.590			
		YOLO	0.720			
mix	YOLO	-	0.720	18.373		
		ICP	0.610			
		ICP	0.610			

introducing new occlusions.

## REFERENCES

- [1] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *ECCV*, 2020.
- [2] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*, 2010.

TABLE 13: Different view sampling strategies for the multi-view refiner on the Linemod [18] dataset.

Dataset	Data type	N views	Views sampling	ADD
Linemod	real	-	-	93.59
		2 views	Closest	86.43
			Random	99.52
			Furthest	99.71
		4 views	Closest	96.53
			Random	99.91
	Furthest		99.97	
	synt	-	-	81.20
		2 views	Closest	87.45
			Random	98.95
Furthest			99.02	
4 views	Closest	94.65		
	Random	99.70		
	Furthest	99.80		

TABLE 14: Different view sampling strategies for the multi-view refiner on the Occlusion [83] dataset.

Dataset	Data type	N views	Views sampling	AR
LMO	real	-	-	0.568
		2 views	Closest	0.627
			Random	0.661
			Furthest	0.648
		4 views	Closest	0.693
			Random	0.714
	Furthest		0.706	
	synt	-	-	0.584
		2 views	Closest	0.627
			Random	0.660
Furthest			0.664	
4 views	Closest	0.693		
	Random	0.695		
	Furthest	0.687		

TABLE 15: Different view sampling strategies for the multi-view refiner on the Homebrewed [7] dataset.

Dataset	Data type	N views	Views sampling	AR
HBD	synt	-	-	0.668
		2 views	Closest	0.693
			Random	0.808
			Furthest	0.762
		4 views	Closest	0.724
			Random	0.783
Furthest	0.775			

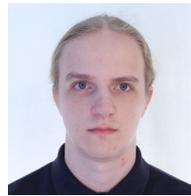
- [3] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon, "The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation," in *CVPR*, 2012.
- [4] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [5] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *ECCV*, 2014.
- [6] O. H. Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother, "ipose: instance-aware 6d pose estimation of partly occluded objects," in *ACCV*, 2018.
- [7] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects," in *ICCV Workshops*, 2019.
- [8] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2015.
- [9] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using lstms for structured feature correlation," in *ICCV*, 2017.
- [10] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *CVPR*, 2017.
- [11] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.
- [12] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *ICCV*, 2017.
- [13] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *CVPR*, 2019.
- [14] T. Hodan and A. Melenovsky. (2019) Bop: Benchmark for 6d object pose estimation: <https://bop.felk.cvut.cz/home/>.
- [15] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," in *ECCV*, 2018.
- [16] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *ICCV*, 2011.

TABLE 16: Different view sampling strategies for the multi-view refiner on the TLESS [52] dataset.

Dataset	Data type	N views	Views sampling	AR
TLESS	real	-	-	0.518
		2 views	Closest	0.524
			Random	0.523
			Furthest	0.534
		4 views	Closest	0.544
			Random	0.524
	Furthest		0.540	
	synt	-	-	0.636
		2 views	Closest	0.693
			Random	0.690
			Furthest	0.694
		4 views	Closest	0.712
Random			0.689	
Furthest	0.707			
mix	-	-	0.655	
	2 views	Closest	0.715	
		Random	0.719	
		Furthest	0.725	
	4 views	Closest	0.738	
		Random	0.720	
Furthest		0.742		

- [17] S. Hinterstoisser, C. Cagniard, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *TPAMI*, 2012.
- [18] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV*, 2012.
- [19] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto, "Fast 6d pose estimation from a monocular image using hierarchical pose trees," in *ECCV*, 2016.
- [20] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3d object detection: A real time scalable approach," in *ICCV*, 2013.
- [21] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit, "Hashmod: A hashing method for scalable 3d object detection," in *BMVC*, 2015.
- [22] T. Hodaň, X. Zabulis, M. Lourakis, Š. Obdržálek, and J. Matas, "Detection and fine 3d pose estimation of texture-less objects in rgb-d images," in *IROS*, 2015.
- [23] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR*, 2015.
- [24] S. Zakharov, W. Kehl, B. Planche, A. Hutter, and S. Ilic, "3d object instance recognition & pose estimation using triplet loss with dynamic margin," in *IROS*, 2017.
- [25] M. Bui, S. Zakharov, S. Albarqouni, S. Ilic, and N. Navab, "When regression meets manifold learning for object recognition and pose estimation," in *ICRA*. IEEE, 2018.
- [26] S. Zakharov, B. Planche, Z. Wu, A. Hutter, H. Kosch, and S. Ilic, "Keep it unreal: Bridging the realism gap for 2.5 d recognition with geometry priors only," in *3DV*, 2018.
- [27] B. Planche, S. Zakharov, Z. Wu, A. Hutter, H. Kosch, and S. Ilic, "Seeing beyond appearance-mapping real images into geometrical domains for unsupervised cad-based recognition," in *IROS*, 2019.
- [28] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, 2018.
- [29] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, 1992.
- [30] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *ICCV*, 2017.
- [31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.
- [32] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *CVPR*, 2018.
- [33] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnnp: An accurate o (n) solution to the pnp problem," *IJCV*, 2009.
- [34] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, 2019.
- [35] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *ICCV*, 2019.
- [36] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *ICCV*, 2019.
- [37] Z. Li, G. Wang, and X. Ji, "Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation," in *ICCV*, 2019.
- [38] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon, "Autolabeling 3d objects with differentiable rendering of sdf shape priors," in *CVPR*, 2020.
- [39] T. Hodan, D. Barath, and J. Matas, "Epos: Estimating 6d pose of objects with symmetries," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [40] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *CVPR*, 2019.
- [41] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *CVPR*, 2020.
- [42] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *CVPR*, 2020.
- [43] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6d pose refinement in rgb," in *ECCV*, 2018.
- [44] J. Sock, S. Hamidreza Kasaei, L. Seabra Lopes, and T.-K. Kim, "Multi-view 6d object pose estimation and camera motion planning using rgbd images," in *ICCV*, 2017.
- [45] Ö. Erkent, D. Shukla, and J. Piater, "Integration of probabilistic pose estimates from multiple views," in *ECCV*, 2016.
- [46] C. Li, J. Bai, and G. D. Hager, "A unified framework for multi-view multi-class object pose estimation," in *ECCV*, 2018.
- [47] R. Kaskman, I. Shugurov, S. Zakharov, and S. Ilic, "6 dof pose estimation of textureless objects from multiple rgb frames," in *ECCV Workshop*, 2020.
- [48] I. Shugurov, I. Pavlov, S. Zakharov, and S. Ilic, "Multi-view object pose refinement with differentiable renderer," *IEEE Robotics and Automation Letters*, 2021.
- [49] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, 2018.
- [50] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented autoencoders: Implicit 3d orientation learning for 6d object detection," *IJCV*, 2020.
- [51] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
- [52] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *WACV*, 2017.
- [53] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.
- [55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [56] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *3DV*, 2016.
- [57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
- [58] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [59] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.
- [60] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *ECCV*, 2018.
- [61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
- [62] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.

- [63] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
- [64] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.
- [65] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," in *ICCV*, 2019.
- [66] J. T. Barron, "A general and adaptive robust loss function," in *CVPR*, 2019.
- [67] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *CVPR*, 2019.
- [68] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *ECCV*, 2016.
- [69] M. Denninger, M. Sundermeyer, D. Winkelbauer, D. Olefir, T. Hodan, Y. Zidan, M. Elbadrawy, M. Knauer, H. Katam, and A. Lodhi, "Blenderproc: Reducing the reality gap with photorealistic rendering,"
- [70] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "Blenderproc," *arXiv preprint arXiv:1911.01911*, 2019.
- [71] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, 2017.
- [72] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, 1985.
- [73] G. Pitteri, M. Ramamonjisoa, S. Ilic, and V. Lepetit, "On object symmetries and 6d pose estimation from images," in *3DV*, 2019.
- [74] M. Bui, T. Birdal, H. Deng, S. Albarqouni, L. Guibas, S. Ilic, and N. Navab, "6d camera relocalization in ambiguous scenes via continuous multimodal inference," 2020.
- [75] F. Manhardt, D. M. Arroyo, C. Rupprecht, B. Busam, T. Birdal, N. Navab, and F. Tombari, "Explaining the ambiguity of object detection and 6d pose from visual data," in *ICCV*, 2019.
- [76] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.
- [77] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, "Introducing mvtec itodd-a dataset for 3d object recognition in industry," in *ICCV Workshops*, 2017.
- [78] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved rgb-d based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, 2016.
- [79] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *CVPR*, 2016.
- [80] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *ECCV*, 2014.
- [81] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *ECCV*, 2016.
- [82] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *CVPR*, 2018.
- [83] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *CVPR*, 2016.
- [84] J. Vidal, C.-Y. Lin, X. Lladó, and R. Martí, "A method for 6d pose estimation of free-form rigid objects using point pair features on range data," *Sensors*, 2018.
- [85] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm, "Usac: a universal framework for random sample consensus," *TPAMI*, 2012.
- [86] D. Barath and J. Matas, "Graph-cut ransac," in *CVPR*, 2018.
- [87] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Joint Pattern Recognition Symposium*, 2003.
- [88] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (dls) method for prnp," in *2011 International Conference on Computer Vision*, 2011.



**Ivan Shugurov** is a PhD student at the Chair for Computer Aided Medical Procedures / TU Munich and at Siemens Corporate Technology under supervision of PD. Dr. Slobodan Ilic. He completed his Bachelor's degree in Software Engineering at Higher School of Economics. Subsequently, he studied Informatics at TU Munich. His research focuses on multimodal 2D/3D object detection and pose estimation.



**Sergey Zakharov** is a research scientist at Toyota Research Institute (TRI) in Los Altos, CA. He obtained his MSc degree in Computational Science and Engineering and a MSc degree in Biomedical Computing from the Technical University of Munich (TUM) in 2015 and 2016 respectively. Subsequently, he completed his PhD degree at the Chair for Computer Aided Medical Procedures, TUM and Siemens Corporate Technology. His research focuses on scene understanding and covers such topics as scalable 3D

object detection, neural implicit representations, differentiable rendering, and domain adaptation. He owns a number of industry-relevant solutions published and presented at first-tier computer vision and robotics conferences including CVPR, ICCV, ICRA, and IROS.



**Slobodan Ilic** is currently senior key expert research scientist at Siemens Corporate Technology in Munich, Perlach. He is also a visiting researcher and lecturer at Computer Science Department of TUM and closely works with the CAMP Chair. From 2009 until end of 2013 he was leading the Computer Vision Group of CAMP at TUM, and before that he was a senior researcher at Deutsche Telekom Laboratories in Berlin. In 2005 he obtained his PhD at EPFL in Switzerland under supervision of Pascal Fua.

His research interests include: 3D reconstruction, deformable surface modelling and tracking, real-time object detection and tracking, human pose estimation and semantic segmentation.