DPOD: 6D Pose Object Detector and Refiner

Sergey Zakharov [*,‡,†], Ivan Shugurov [*,‡,†], and Slobodan Ilic[‡,†]

‡ Technical University of Munich   † Siemens Corporate Technology

sergey.zakharov@tum.de, ivan.shugurov@tum.de, slobodan.ilic@siemens.com

Abstract

In this paper we present a novel deep learning method for 3D object detection and 6D pose estimation from RGB images. Our method, named DPOD (Dense Pose Object Detector), estimates dense multi-class 2D-3D correspondence maps between an input image and available 3D models. Given the correspondences, a 6DoF pose is computed via PnP and RANSAC. An additional RGB pose refinement of the initial pose estimates is performed using a custom deep learning-based refinement scheme. Our results and comparison to a vast number of related works demonstrate that a large number of correspondences is beneficial for obtaining high-quality 6D poses both before and after refinement. Unlike other methods that mainly use real data for training and do not train on synthetic renderings, we perform evaluation on both synthetic and real training data demonstrating superior results before and after refinement when compared to all recent detectors. While being precise, the presented approach is still real-time capable.

1. Introduction

Object detection has always been an important problem in computer vision and a large body of research has been dedicated to it in the past. This problem, like many other vision problems, witnessed a complete renaissance with the advent of deep learning. Detectors like R-CNN [8], and its follow-ups Fast-RCNN [7], Faster-RCNN [28], Mask-RCNN [9], then YOLO [27] and SSD [20] marked this research field with excellent performance. All these works localize objects of interest in images in terms of tight bounding boxes around them. However, in many applications, e.g., augmented reality, robotics, machine vision, etc., this is not enough and a full 6D pose is necessary. While this problem is easier to solve in depth images, in RGB images it is still quite challenging due to perspective ambiguities
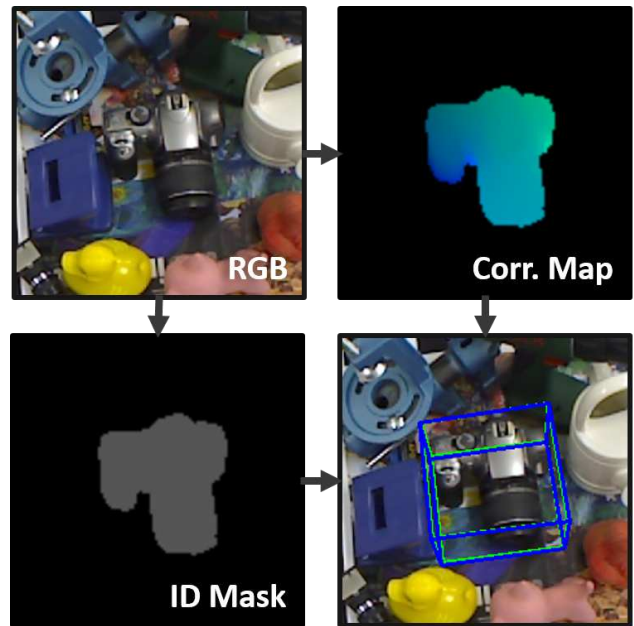
*These authors contributed equally to the work.

Figure 1: **Example output of the DPOD method:** Given a single RGB image, we regress its ID mask and its 2D-3D correspondences. PnP+RANSAC is then applied to estimate the final pose. The green bounding box shows the ground truth pose, while the blue one corresponds to the estimated pose. The almost perfect overlap of the bounding boxes indicates that estimations are very accurate.

and significant appearance changes of the object when seen from different viewpoints.

Recent deep learning-based approaches, such as SSD6D [15], YOLO6D [33], AAE [31], PoseCNN [34] and PVNet [25], are the current top performers for this task in RGB images. Even though they all perform evaluation on LineMOD and OCCLUSION datasets, each of them focuses on different aspects of the 6D pose estimation pipeline. The majority is trained on real data [33, 34, 25, 14] while only SSD6D [15] and AAE [31] are trained on syn-

thetic renderings. Some are presented without refinement, like YOLO6D [33] and PoseCNN [34], while the others perform refinement. The most recent refiners are based on deep learning, e.g., DeepIM [18] that acts on poses from the PoseCNN detector and the refiner of Manhardt et al. [21] that uses SSD6D poses.

Inspired by the methods of Gueler et al. [1] and Taylor et al. [32], which estimate dense correspondences between the human body model and the humans in the image, we propose a novel 3D object detector and pose estimator that also estimates dense 2D-3D correspondences. Unlike DensePose for humans, which requires a sophisticated annotation tool and enormous annotation efforts, our method is annotation-free and only requires creation of arbitrary UV texture maps of the objects, that we do automatically— mainly by spherical projections. The two key elements of our approach are: the pixel-wise prediction of the multi-class object ID masks and classification of correspondence maps that directly provide a relation between image pixels and 3D model vertices. In this way, we end up with a large number of pixel-wise correspondences, which allow for a much better pose estimation than, for example, 9 regressed virtual points of the object's bounding box as in YOLO6D.

In addition to this, we introduce a deep learning-based pose refinement network that takes initial poses estimated with our DPOD detector and enhances them. The proposed refinement approach builds on the successes of [18, 22], but is shown to be faster, simpler to train, able to be trained both on synthetic and real data, and it outperforms the former solutions in terms of pose quality. We demonstrate that even our poses, which are already of high quality, can be further improved with our refiner.

We experimented by training our detector with only synthetic and only real images. In both cases, our unified method, named DPOD, composed of the dense pose detector and the refiner outperforms other related works. Dense correspondences not only allow for standard PnP and RANSAC to estimate accurate poses without refinement, but also pave the way for a successful pose refinement. For the models trained on real data, one iteration of refinement is enough to outperform all other reported results, even SSD6D with the depth-based ICP refinement.

In the remainder of the paper we first review related approaches, then introduce our approach, explaining data preparation, training, architectures and pose refinement. Finally, we present an exhaustive experimental validation and comparison with recent works, where we demonstrate the superiority of our approach.

2. Related Work

Detecting 3D objects and estimating their 6D pose has been addressed in many works in the past, but the majority of them used depth or RGB-D cameras [2, 5, 16, 17, 23,

30, 35]. Depth information disambiguates the object's scale that is the most critical in RGB images due to perspective projection. Therefore, using only RGB images for detection and 6D pose estimation is a quite challenging problem. Recent solutions are mainly based on deep learning and automatically learned features, while older ones use hand-crafted features or image information, e.g., gradients, or image pixel intensities directly.

Template matching approaches, e.g., [11, 12, 29], render synthetic image patches from different viewpoints distributed on a sphere around the 3D model of the object and store them as a database of templates. Then the input images are searched using this template database sequentially in a sliding window fashion. Efficient and robust template matching strategies have been presented for color, depth and RGB-D images. The most popular approach is arguably LineMOD [11], which also provided a first dataset with labeled poses. This dataset is still used as a benchmark for object detection and pose estimation. Another alternative to template matching approaches is the learning approaches that employ random forests [2, 3, 4].

Deep Learning 6D Pose Detectors. In the last two years deep learning approaches have shown that impressive results can be obtained for detection and pose estimation in RGB images. Here we review the following ones: SSD6D [15], YOLO6D [33], BB8 [26], iPose [14], AAE [31], PoseCNN [34] and PVNet [25].

SSD6D [15] extended the ideas of the 2D object detector [20] by 6D pose estimation based on a discrete viewpoint classification rather than direct regression of rotations. The method is rather slow and poses predicted this way are quite inaccurate since they are only a rough discrete approximation of the real poses. The refinement is a must in order to produce presentable results. BB8 [26] uses a three-stage approach. In the first two stages the 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 6D pose can be estimated with PnP. The main disadvantage of this pipeline is its multi-stage nature, resulting in very slow run times. Building on YOLO and BB8 ideas, YOLO6D [33] 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 reprojected bounding box corners in the image. The advantages of this parametrization are its relative compactness and that it does not introduce a pose ambiguity as opposed to a direct regression of the rotation. Moreover, in contrast to SSD6D, it does not suffer from pose discretization resulting in much more accurate pose estimates without refinement.

Among the methods that are specifically designed to be robust to occlusions we would like to highlight iPose [14],

PoseCNN [34], and PVNet [25]. iPose [14] operates in 3 separate stages: segmentation, 3D coordinate regression and pose estimation. By contrast, our approach unifies the first two stages into the end-to-end network. Moreover, we do not regress 3D coordinates, but rather UV maps that turned out to be a much easier task for the network, resulting in less erroneous correspondences. PoseCNN [34] also estimates object masks, but then separately estimates the translation of the object's centroid and regresses a quaternion for rotation. PVNet [25] takes a different approach and designs a network which for every pixel in the image regresses an offset to some predefined keypoints. Instead of bounding box points, they vote for the points located on the object itself. This allows them to handle occlusions very well. AAE (Augmented Autoencoders) [31] concentrates on pose estimation and training from synthetic models, while using already computed SSD detection bounding boxes as input.

Deep Learning 6D Pose Refiners. Deep learning-based 6D pose refinement has shown promising results in recent publications [22, 18]. Both refiners are conceptually very similar and are designed to output relative transformation between the real input image patch and the patch containing the object rendered with the predicted pose. Main differences are the used backbone architectures and loss functions. Both refinement algorithms rely on external object detection and pose estimation algorithms: for DeepIM [18] it is PoseCNN, for [22] it is SSD6D [15]. The former relies on real data, whereas the latter focuses on training on synthetic images. We propose a network architecture which takes the best of above architectures and is independent of the type of training data used.

Our work differs from the above approaches by being a complete end-to-end pipeline integrating a detector and pose estimator based on dense correspondences. We demonstrate that we can train either from real or synthetic data and in both cases we outperform all related approaches by a large margin on the LineMOD and OCCLUSION datasets.

3. Methodology

In this section we first discuss the training data preparation steps, followed by the neural network architecture and loss functions used, as well as the pose estimation step from dense correspondences. Finally, we describe our deep learning model-based pose refiner.

3.1. Data Preparation

Most recent RGB-based detectors can be divided in two groups based on the type of data they use for training: synthetic-based and real-based. The first group of methods, e.g., SSD6D [15] and AAE [31], makes use of textured 3D models, usually provided with the public 6D pose de-

tection datasets. The objects are rendered from different viewpoints, producing a synthetic training set. The methods of the second group on the other hand, e.g., BB8 [26], YOLO6D [33], PVNet [25], use the training split of the real dataset. They utilize ground truth poses provided with the dataset and compute object masks to crop the objects from real images producing a training set.

Both types of data generation have their pros and cons. When real images sufficiently covering the object are available, it is more advantageous to use them for training. The reason is that their close resemblance to the actual objects allows for faster convergence and better results. However, training on real images biases the detector to light conditions, poses, scales and occlusions present in the training set, which might lead to problems with generalization in new environments. When, however, no pose annotations are available, which can often be the case since acquiring pose annotations is an expensive process, we are left with 3D models of the objects. With synthetic renderings, one can produce a virtually infinite number of images from different viewpoints. Despite being advantageous in terms of the pose coverage, one has to deal with the domain gap problem severely hindering the performance if no additional data augmentation is applied. Potentially, one can benefit from the advantages of both data types by mixing real and synthetic data in the training set. Therefore, approaches which can be trained on both types of data are desirable. Since our pipeline is not data-specific, we show how to generate the training data for both scenarios.

Synthetic Training Data Generation. Given 3D models of the objects of interest, the first step is to render them from different poses sufficiently covering the object. The poses are sampled from the half-sphere above the object. Additionally, in-plane rotations of the camera around its viewing direction from -30 to 30 degrees are added. Then, for each of the camera poses, an object is rendered on a black background and both RGB and depth channels are stored.

Having the renderings at hand, we use a generated depth map as a mask to define a tight bounding box for each generated rendering. Cropping the image with this bounding box position, we store RGB patches, masks separating them from the background, and the camera poses. At this point, we have everything ready for the online augmentation stage, which is described in the later subsection. This step of data preparation is identical for the detector and for the refinement pipelines.

Real Training Data Generation. In this case, an available dataset with pose annotations is divided into non-overlapping train and test subsets. Here, we follow the protocol defined by BB8 [26] and YOLO6D [33] and use 15% of data for training and the rest 85% for evaluation. Poses are selected such that the relative orientation between them
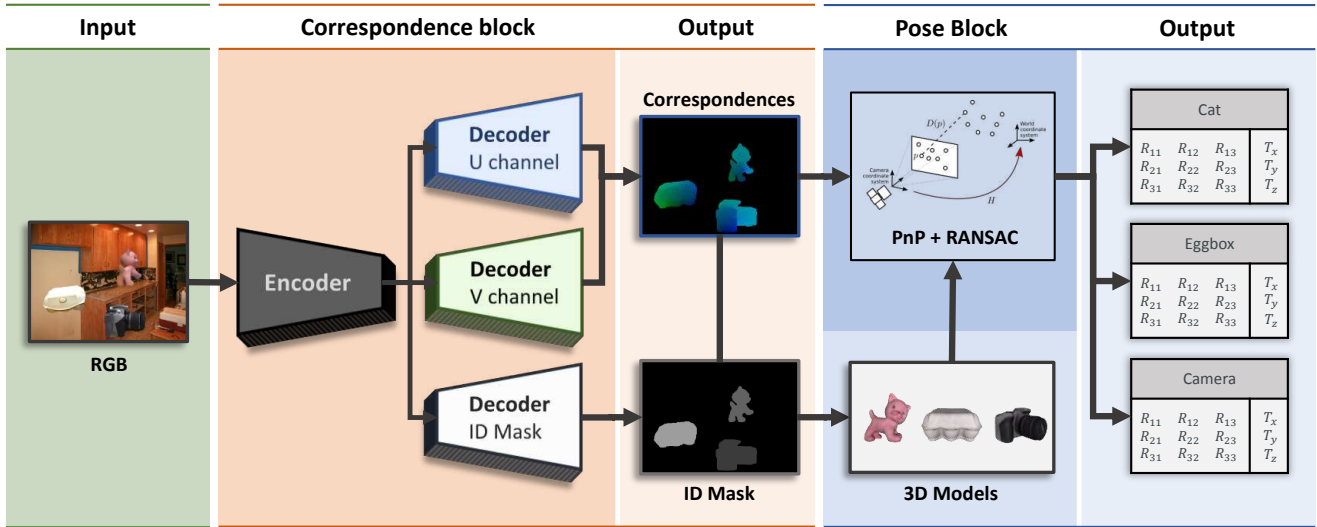
Figure 2: **Pipeline description:** Given an input RGB image, the correspondence block, featuring an encoder-decoder neural network, regresses the object ID mask and the correspondence map. The latter one provides us with explicit 2D-3D correspondences, whereas the ID mask estimates which correspondences should be taken for each detected object. The respective 6D poses are then efficiently computed by the pose block based on PnP+RANSAC.

is larger than a certain threshold. This approach guarantees that selected poses cover the object from all sides. For training the detector, objects are cut out from the original image using the provided mask and then stored as patches for the online augmentation stage. Additional in-plane rotations are added to artificially simulate new poses. For training the refinement, objects are left as they are.

3.1.1 Correspondence Mapping

To be able to learn dense 2D-3D correspondences, each model of the dataset is textured with a correspondence map (see Figure 3). A correspondence map is a 2-channel image with values ranging from 0 to 255. Objects are textured using either simple spherical or cylindrical projections. Once textured, we get a bijective mapping between the model's
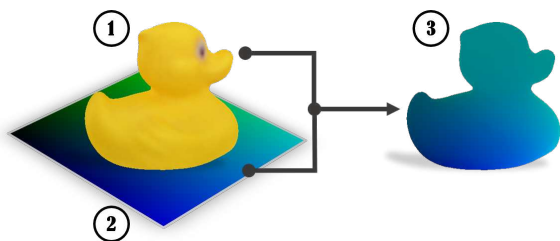

Figure 3: **Correspondence model:** Given a 3D model of interest (1), we apply a 2 channel correspondence texture (2) to it. The resulting correspondence model (3) is then used to generate GT maps and estimate poses.

vertices and pixels on the correspondence map. This provides us with easy-to-read 2D-3D correspondences since given the pixel color, we can instantaneously estimate its position on the model surface by selecting the vertex with the same color value. For convenience, we call the copies of the original models textured with the correspondence map *correspondence models*. Given the predicted correspondence map, we estimate the object pose with respect to the camera using the pose estimation block, which is described later. Similar to the synthetic or real data generation steps, we render correspondence models under the same poses as for training data and store correspondence patches for each RGB patch.

3.1.2 Online Data Generation and Augmentation

Detection and Pose Estimation. The final stage of data preparation is the online data generation pipeline, which is responsible for providing full-sized RGB images ready for training. Generated patches (real or synthetic) are rendered on top of images from MS COCO dataset [19] producing training images containing multiple objects. It is an important step, which ensures that the detector generalizes to different backgrounds and prevents it from overfitting to backgrounds seen during training. Moreover, it forces the network to learn the model's features needed for pose estimation rather than to learn contextual features which might not be present in images when the scene changes. This step is performed no matter whether the training is being done with synthetic or real patches. We additionally augment

the RGB image by random changes in brightness, saturation, and contrast, and by adding Gaussian noise. Moreover, object ID masks and correspondence patches are also rendered on top of the black background in order to generate ground truth correspondence maps. An object ID mask is constructed by assigning a class ID number to each pixel that belongs to the object.

Pose Refinement. In the case of pose refinement, pairs of images containing the object in the current (searched) pose and in the predicted pose are provided to the network. The final stage of data preparation differs considerably depending on the type of data used. In case of synthetic data, images are generated by in-painting objects on random backgrounds in a current pose. A crucial part of the augmentation is to add random light sources for every image. If real images are used for training, no in-painting is performed. In any case, produced images are further augmented as discussed above. Then a random pose is sampled around the current pose simulating the predicted pose from the detector, which will be used as an original guess of the poses to be refined. It is crucial to choose the proper prior distribution from which distorted poses are sampled.

4. Dense Object Detection Pipeline

Our inference pipeline is divided into two blocks: the correspondence block and the pose block (see Figure 2). In this section, we provide their detailed description.

Correspondence Block. The correspondence block consists of an encoder-decoder convolutional neural network with three decoder heads which regress the ID mask and dense 2D-3D correspondence map from an RGB image of size $320 \times 240 \times 3$. The encoder part is based on a 12-layer ResNet-like [10] architecture featuring residual layers that allow for faster convergence. The decoders upsample the feature up to its original size using a stack of bilinear interpolations followed by convolutional layers. However, in principle the proposed method is agnostic to a particular choice of encoder-decoder architecture. Any other backbone architectures can be used without any need to change the conceptual principles of the method. For the ID mask head the output is a $H \times W \times O$ tensor, where H and W are the height and width of the original input image and O is the number of objects in the dataset plus one additional class for background. Similar to the ID mask head, the two correspondence heads regress tensors with the following dimensions $H \times W \times C$, where C stands for the number of unique colors of the correspondence map, i.e., 256. Each channel of the output tensors stores the probability values for the class corresponding to the channel number. Once tensors are regressed, we store them as single channel images where each pixel stores the class with the maximal estimated probability, forming the ID mask, U and V channels

of the correspondence image.

Formulating color regression problem as discrete color class classification problem proved to be useful for much faster convergence and for the superior quality of 2D-3D matches. 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., $[-1; 1]^3$, where 3 is the number of dimensions and $[-1, 1]$ is the normalized coordinate range of a 3D model. Classification of the discretized 2D correspondences allowed for a huge boost of the output quality by dramatically decreasing the output space (now 256^2, where 256 is the size of a single UV map dimension). Moreover, this parametrization also ensures that 3D points of the predicted correspondences always lie on the object surface.

The network parameters are optimized subject to the composite loss function:

$$\mathcal{L} = \alpha \mathcal{L}_m + \beta \mathcal{L}_u + \gamma \mathcal{L}_v, \tag{1}$$

where $\mathcal{L}_m$ is the mask loss, and $\mathcal{L}_u$ and $\mathcal{L}_v$ are the losses responsible for the quality of the U and V channels of the correspondence image. α, β, and γ are weight factors set to 1 in our case. Both $\mathcal{L}_u$ and $\mathcal{L}_v$ losses are defined as multi-class cross-entropy functions, whereas $\mathcal{L}_m$ uses the weighted version of it.

Pose Block. The pose block is responsible for the pose prediction. Given the estimated ID mask, we can observe which objects were detected in the image and their 2D locations, whereas the correspondence map maps each 2D point to a coordinate on an actual 3D model. The 6D pose is then estimated using the Perspective-n-Point (PnP) [36] pose estimation method that estimates the camera pose given correspondences and intrinsic parameters of the camera. Since we get a large set of correspondences for each model, RANSAC is used in conjunction with PnP to make camera pose prediction more robust to possible outliers. For the results presented in the evaluation section, for each pose we run 150 RANSAC iterations with the reprojection error threshold set to 1.

5. Deep model-based pose refinement

The proposed pose refiner is a natural extension of refiners presented in [22, 18] and relies on the strengths of both approaches. Similar to [22, 15, 13] we exploit an idea of using a network already pre-trained on ImageNet as a backbone architecture. Analogous to the detector, we used a ResNet-based architecture. Similar to [18], our loss function for pose estimation is the ADD measure with a more robust L_1 norm:

$$m = \underset{\mathbf{x} \in \mathcal{M}_s}{\mathrm{avg}} \left\| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \right\|_1, \tag{2}$$

representing the vertex to vertex distance between the object in a ground truth pose and predicted pose. $\mathbf{R}$, $\mathbf{t}$ denote the ground truth pose rotation and translation, whereas $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ denote the predicted transformation; $\mathcal{M}_s$ is a set of points sampled from the CAD model. Points are resampled at every iteration. The number of sampled points was limited to ten thousand in order to ensure the efficiency of training iterations and reasonable memory consumption.

In Figure 4 we show a schematic representation of the refiner. In order to be able to benefit from the network weights pretrained on ImageNet, the network has two parallel input branches, each composed of the first five ResNet layers. These layers are initialized from the pre-trained network. One branch receives an input image patch (E_{11}), while the other (E_{12}) one extracts features from the rendering of the object in the predicted pose. Then features $\mathbf{f}_r$ and $\mathbf{f}_s$ from these two networks are subtracted and fed into the next ResNet block (E_2) producing the feature vector $\mathbf{f}$. If the refinement is trained on synthetic data, it is essential to keep the first five layers unchanged and use them as the feature extractor as was shown in [20, 13, 22]. Freezing the branch that extracts features from object renderings is unnecessary as it always operates on synthetic data. The network ends with three separate output heads: one for regressing the rotation, one for regressing the translation in X and Y directions, and one for regressing the translation in Z direction. We opted for three separate heads as the scale of their outputs is different. Each head is implemented as two fully connected layers.

Rotation is always represented in the object coordinate system, which ensures that identically looking objects have the same rotation and that the network does not have to learn a more complicated transformation which arises if the world coordinate system is used. The first layer of the rotation re-
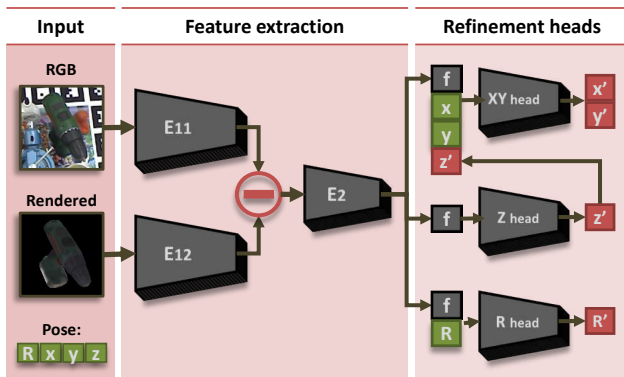

Figure 4: **Refinement architecture:** The network predicts a refined pose given an initial pose proposal. Crops of the real image and the rendering are fed into two parallel branches. The difference of the computed feature tensors is used to estimate the refined pose.

gression head takes the feature vector $\mathbf{f}$ produced by ResNet and adds four values, which are the quaternion representing an initial rotation. The second layer takes the output of the previous one, stacks with the initial quaternion and outputs the final rotation.

The head responsible for the regression of X and Y translations operates in the coordinate system of the image rather than in the full 3D space, which significantly restricts the space of possible solutions. Similar to the rotation head, the XY regression head takes the initial 2D location of the object as input and refines it. Additionally, it takes the refined prediction of Z translation.

Weights of the fully connected layers are initialized in such a way that for the 0th iteration the network just outputs the input pose, and then during training learns how to refine those values. That significantly increases stability and speed of the training procedure as the network produces meaningful results from the very start.

6. Training Details

Our pipeline is implemented using the Pytorch deep learning framework. All the experiments were conducted on an Intel Core i7-6900K CPU 3.20GHz with NVIDIA TITAN X (Pascal) GPU. To train our method, we used the ADAM solver with a constant learning rate of 3×10^{-4} and weight decay of 3×10^{-5}.

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 it impossible to generalize to the real data. The easy solution to this problem was proposed in several works, including [13, 22], where they freeze the first layers of the network trained on a large dataset of real images, e.g., ImageNet [6] or MS COCO [19], for the object classification task. The common observation that the authors conclude is that these layers, learning low-level features, very quickly overfit to the perfect object renderings. We follow this setup, and freeze the first five layers of our encoder initialized with the weights of the same network pretrained on ImageNet. Last but not least, we found it crucial for the performance of the detector to use various light sources during the rendering of synthetic views to account for changing light conditions and shadows in the real data.

7. Evaluation

In this section we evaluate our algorithm in terms of its pose and detection performance, as well as its runtime, and compare it with the state of the art RGB detector solutions.

7.1. Datasets

All experiments were conducted on LineMOD [12] and OCCLUSION [2] datasets, as they are the standard datasets

Table 1: **Pose estimation performance:** Comparison of our approach to the other RGB detectors on the LineMOD dataset. The table reports the percentages of correctly estimated poses w.r.t. the ADD score. Among the methods trained on synthetic data, our method shows the best results significantly surpassing the former state-of-the-art. The variant of our method trained on real data again demonstrates outstanding performance outperforming most of the competitors. Moreover, our new refinement pipeline improves the estimated poses even further and shows the best overall results.

Train data	Synthetic			+ Refinement		Real				+ Refinement	
Object	SSD6D [15]	AAE [31]	Ours	SSD6D [22]	Ours	YOLO6D [33]	PoseCNN [34]	PVNet [25]	Ours	DeepIM [18]	Ours
Ape	2.6	3.96	**37.22**	-	55.23	21.62	-	43.62	**53.28**	77.0	**87.73**
Benchvise	15.1	20.92	**66.76**	-	72.69	81.80	-	**99.90**	95.34	97.5	**98.45**
Cam	6.1	**30.47**	24.22	-	34.76	36.57	-	86.86	**90.36**	93.5	**96.07**
Can	27.3	35.87	**52.57**	-	83.59	68.80	-	**95.47**	94.10	96.5	**99.71**
Cat	9.3	17.90	**32.36**	-	65.10	41.82	-	**79.34**	60.38	82.1	**94.71**
Driller	12.0	23.99	**66.60**	-	73.32	63.51	-	96.43	**97.72**	95.0	**98.80**
Duck	1.3	4.86	**26.12**	-	50.04	27.23	-	52.58	**66.01**	77.7	**86.29**
Eggbox	2.8	**81.01**	73.35	-	89.05	69.58	-	99.15	**99.72**	97.1	**99.91**
Glue	3.4	45.49	**74.96**	-	84.37	80.02	-	**95.66**	93.83	**99.4**	96.82
Holepuncher	3.1	17.60	**24.50**	-	35.35	42.63	-	**81.92**	65.83	52.8	**86.87**
Iron	14.6	32.03	**85.02**	-	98.78	74.97	-	98.88	**99.80**	98.3	**100**
Lamp	11.4	**60.47**	57.26	-	74.27	71.11	-	**99.33**	88.11	97.5	96.84
Phone	9.7	**33.79**	29.08	-	46.98	47.74	-	**92.41**	74.24	87.7	**94.69**
Mean	9.1	28.65	**50**	34.1	**66.43**	55.95	62.7	**86.27**	82.98	88.6	**95.15**

for evaluation of object detection and pose estimation methods. The LineMOD dataset consists of 13 sequences, each containing ground truth poses for a single object of interest in a cluttered environment. CAD models for all the objects are provided as well. The OCCLUSION dataset is an extension of LineMOD, suitable for testing how well detectors can deal with occlusions. Although it comprises only one sequence, all visible objects from the LineMOD dataset are supplied with their poses.

7.2. Evaluation Metrics

We evaluate the quality of 6DoF pose estimation following the procedure suggested at SSD6D [15] also used in other papers. Analogously to other related papers [33, 15, 25, 34], we measure the accuracy of pose estimation using the *ADD score* [12]. ADD is defined as an average Euclidean distance between model vertices transformed with the predicted and the ground truth pose. More formally it is defined as follows:

$$m = \operatorname*{avg}_{\mathbf{x} \in \mathcal{M}} \left\| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \right\|_2, \qquad (3)$$

where $\mathcal{M}$ is a set of vertices of a particular model, $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation of a ground truth transformation whereas $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ correspond to those of an estimated transformation. The ADD metric can be extended in order to handle symmetric objects as in [12]:

$$m = \operatorname*{avg}_{\mathbf{x}_2 \in \mathcal{M}} \min_{\mathbf{x}_1 \in \mathcal{M}} \left\| (\mathbf{R}\mathbf{x}_1 + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{t}}) \right\|_2 \quad (4)$$

Instead of measuring distance from a predicted location of each particular model's vertex to its ground truth location, it suggests to take the closest vertex of the model transformed with the ground truth transformation.

Conventionally, a pose is considered correct if ADD is smaller than the 10% of the model's diameter. The accuracy of pose estimation is reported as the percentage of correctly estimated poses.

7.3. Single Object Pose Estimation

Results of the pose estimation experiments on the LineMOD dataset are reported in Table 1. We separately compared our method trained either on real data or on synthetic data. The table provides the comparison of deep learning-based refinement pipelines as well. The left-hand side of the table reports the accuracy of pose estimation as percentages of poses which are correct according to the ADD measure for the training done on synthetic data. If no refinement is used, our approach outperforms all other approaches by a significant margin on the majority of the objects. Moreover, the average percentage of correctly estimated poses (50%) is significantly higher than 28.65% of the second best approach. The accuracy gap is more prominent on small objects such as the ape and duck. The availability of a large number of 2D-3D correspondences ensures that the performance of our method is 5 times better than SSD6D's and almost 2 times better than AAE's. If deep learning-based refinement is used, we significantly outperform [22] with 66.43% of correct poses against 34.1%.

If trained on real data, our method is the second best after [25]. The right-hand side of Table 1 compares the proposed approach to the previous deep learning-based ones. If no refinement is used, the proposed approach outperforms PoseCNN and YOLO6D by a significant margin, while per-
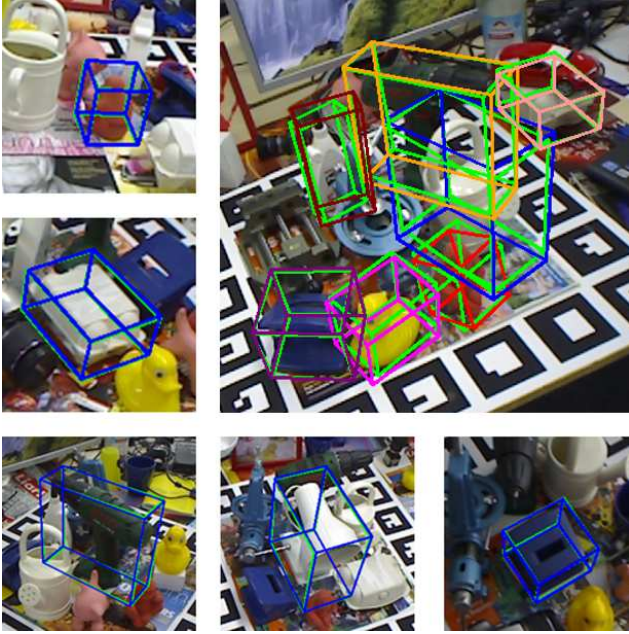
Figure 5: **Qualitative results:** Poses predicted with the proposed approach on (a) the LineMOD dataset and (b) the OCCLUSION dataset. Green bounding boxes correspond to ground truth poses, bounding boxes of other colors to predicted poses. For both datasets predicted poses are very close to correct poses.

forming on par with PVNet on most of the objects. On average, we are better than PoseCNN by 31%, YOLO6D by 23.57%. Again, our approach uses RGB data exclusively and does not rely on depth data. Figure 5 provides a visual comparison of ground truth poses versus predicted poses. Poses are visualized as projections of 3D bounding boxes of models in given poses on top of a test image. In comparison to deep learning-based refinement of [18], we perform on average better by 6.55% reaching 95.15% of correct poses. When DeepIM was applied to the poses predicted by the proposed approach, ADD improved to 91.8% which is better than the original 88.6% reported in their paper, but still worse than the result of our refiner.

In conclusion, the proposed detector achieves state-of-the-art results surpassing other detectors by a large margin on synthetic data and performs either much better or comparable to the other detectors on real data. The proposed

Table 2: **Pose estimation for multiple objects:** Comparison of our approach on real data to the other RGB detectors on the OCCLUSSION dataset. The table reports percentages of correctly estimated poses w.r.t. the ADD score.

Method	YOLO6D [33]	PoseCNN [34]	SSD6D + Ref [22]	HMap [24]	PVNet [25]	Ours	Ours +Ref
Mean	6.42	24.9	27.5	30.4	40.77	32.79	**47.25**

Table 3: **Detection performance for multiple objects:** Comparison of the state-of-the-art mean average precision (mAP) scores on the OCCLUSION dataset.

Method	SSD6D [15]	YOLO6D [33]	Brachmann [3]	Ours
mAP	0.38	0.48	0.51	0.48

refinement clearly outperforms all the competitors both on real and synthetic data. Pose quality varies from object to object, but in general poses are significantly better for larger objects since there are more 2D-3D correspondences available. On the other hand, simplicity of the proposed approach also makes it quick. On average our detector performs at 33 FPS. The runtime can be adjusted by changing the number of RANSAC iterations, as it is the bottleneck of the pipeline. One iteration of the refinement takes 5ms, excluding the rendering time, which heavily depends on the renderer used. Two refinement iterations suffice for synthetic data, one iteration—for real data.

7.4. Multiple Object Pose Estimation

Performance evaluation of the proposed detector in cases when the number of objects to detect increases and when severe occlusions are present was conducted on the OCCLUSION dataset [2]. Accuracy of object detection on the OCCLUSION dataset is conventionally reported in terms of mean average precision (mAP). The confidence score is computed based on the RANSAC inlier proportion as confidence, rendering the final score of 0.48, which is comparable the best result on this dataset (see Table 3). Table 2 demonstrates ADD scores for various detectors on the OCCLUSION dataset. Before the refinement, the proposed detector shows very competitive results in comparison to other detectors. After the refinement, the proposed approach performs substantially better and achieves the best results.

8. Conclusion

In this paper we proposed the Dense Pose Object Detector (DPOD) method that regresses multi-class object masks and dense 2D-3D correspondences between image pixels and corresponding 3D models. Unlike the best performing methods that regress projections of the object's bounding boxes [26, 33] or formulate pose estimation as a discrete pose classification problem [15], dense correspondences computed by our method allow for more robust and accurate 6D pose estimation. We demonstrated that for both, real and synthetic training data, our detector outperforms other related works, such as [33, 34], by a large margin and performs similarly to [25]. The proposed pose refinement approach also performs very well and allows for achieving a pose accuracy that surpasses all other related deep learning-based pose refinement approaches, while having a simpler and more lightweight backbone architecture.

References

[1] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[3] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016.

[4] Tommaso Cavallari, Stuart Golodetz, Nicholas Lord, Julien Valentin, Victor Prisacariu, Luigi Di Stefano, and Philip HS Torr. Real-time rgb-d camera pose estimation in novel scenes using a relocalisation cascade. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[5] Changhyun Choi and Henrik I Christensen. Rgb-d object pose estimation in unstructured environments. *Robotics and Autonomous Systems*, 75:595–613, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 international conference on computer vision*, pages 858–865. IEEE, 2011.

[12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[13] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[14] Omid Hosseini Jafari, Siva Karthik Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother. The best of bothworlds: Learning geometry-based 6d object pose estimation. *arXiv preprint arXiv:1712.01924*, 2017.

[15] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[16] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220. Springer, 2016.

[17] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A scalable tree-based approach for joint object and pose recognition. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[18] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[21] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018.

[22] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018.

[23] Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2017.

[24] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[25] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[26] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[29] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2048–2055, 2013.

[30] Juil Sock, S Hamidreza Kasaei, Luis Seabra Lopes, and Tae-Kyun Kim. Multi-view 6d object pose estimation and camera motion planning using rgbd images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2228–2235, 2017.

[31] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[32] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110. IEEE, 2012.

[33] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[34] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[35] Haoruo Zhang and Qixin Cao. Combined holistic and local patches for recovering 6d object pose. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2219–2227, 2017.

[36] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.