Neural Object Learning for 6D Pose Estimation Using a Few Cluttered Images

Kiru Park[0000−0002−9891−4788], Timothy Patten[0000−0003−1139−9451], and Markus Vincze[0000−0002−2799−491X]

Vision for Robotics Group, Automation and Control Institute, TU Wien, Austria
{park,patten,vincze}@acin.tuwien.ac.at

Abstract. Recent methods for 6D pose estimation of objects assume either textured 3D models or real images that cover the entire range of target poses. However, it is difficult to obtain textured 3D models and annotate the poses of objects in real scenarios. This paper proposes a method, Neural Object Learning (NOL), that creates synthetic images of objects in arbitrary poses by combining only a few observations from cluttered images. A novel refinement step is proposed to align inaccurate poses of objects in source images, which results in better quality images. Evaluations performed on two public datasets show that the rendered images created by NOL lead to state-of-the-art performance in comparison to methods that use 13 times the number of real images. Evaluations on our new dataset show multiple objects can be trained and recognized simultaneously using a sequence of a fixed scene.

Keywords: 6D pose estimation, object learning, object model, object modeling, differentiable rendering, object recognition

1 Introduction

The pose of an object is important information as it enables augmentation reality applications by displaying contents in correct locations and robots to grasp and place an object precisely. Recently, learned features from color images using Convolutional Neural Networks (CNN) have increased performance of object recognition tasks [9, 10] including pose estimation [21, 24, 27, 31, 34, 35, 45]. These methods have achieved the best performance on benchmarks for pose estimation using household objects [13, 19, 43].

However, methods using CNNs require a large number of training images to cover potential view points of target objects in test environments. There have been two approaches to create training images for pose estimation methods: rendering synthetic images using textured 3D models [21, 34] or cropping real images and pasting them on random background images [2, 27, 31]. Recently, state-of-the-art performance has been accomplished by using both synthetic and real images [23, 24, 28]. Unfortunately, both textured 3D models and large numbers of real images are difficult to obtain in the real world. Textured 3D models included in pose benchmarks are created with special scanning devices, such as

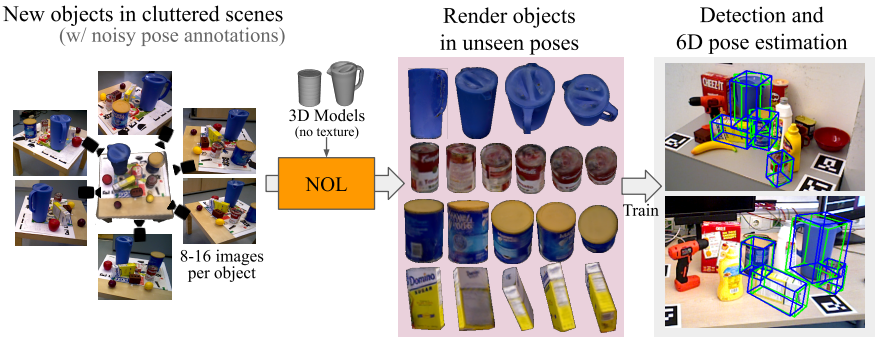

Fig. 1. The proposed method, NOL, uses a few cluttered scenes that consist of new target objects of which to render images in arbitrary poses. Rendered images are used to train pipelines for 6D pose estimation and 2D detection of objects

the BigBIRD Object Scanning Rig [3] or a commercial 3D scanner [19]. During this scanning operation, objects are usually placed alone with a simple background and consistent lighting condition. Precise camera poses are obtained using visible markers or multiple cameras with known extrinsic parameters. In the real environment, however, target objects are placed in a cluttered scene, are often occluded by other objects and the camera pose is imprecise. Furthermore, the manual annotation of 6D poses is difficult and time consuming because it requires the association between 3D points and 2D image pixels to be known. Thus, it is beneficial to minimize the number of images with pose annotations that are required to train 6D pose estimators. This motivates us to develop a new approach to create images of objects from arbitrary view points using a small number of cluttered images for the purpose of training object detectors and pose estimators.

In this paper, we propose Neural Object Learning (NOL), a method to synthesize images of an object in arbitrary poses using a few cluttered images with pose annotations and a non-textured 3D model of the object. For new objects, NOL requires 3D models and cluttered color images (less than 16 images in our evaluations) with pose annotations to map color information to vertices. To overcome pose annotation errors of source images, a novel refinement step is proposed to adjust poses of objects in the source images. Evaluation results show that images created by NOL are sufficient to train CNN-based pose estimation methods and achieves state-of-the-art performance.

In summary, this paper provides the following contributions: **(1)** Neural Object Learning that uses non-textured 3D models and a few cluttered images of objects to render synthetic images of objects in arbitrary poses without retraining the network. **(2)** A novel refinement step to adjust annotated poses of an object in source images to project features correctly in a desired pose without using depth images. **(3)** A new challenging dataset, Single sequence-Multi Ob-

jects Training (SMOT), that consists of two sequences for training and eleven sequences for evaluation, collected by a mobile robot, which represents a practical scenario of collecting training images of new objects in the real world. The dataset is available online[1]. **(4)** Evaluation results that show images rendered by NOL, which uses 8 to 16 cluttered images per object, are sufficient to train 6D pose estimators with state-of-the-art performance in comparison to methods that use textured 3D models and 13 times the number of real images.

The remainder of the paper is organized as follows. Related work is briefly introduced in Sec 2. The detailed description of NOL and the rendering process are described in Sec 3. We report results of evaluations in Sec 4 and analyze effects of each component in Sec 5. Lastly, we conclude the paper in Sec 6.

2 Related Work

In this section, we briefly review the types of training images used to train 6D pose estimation methods. The approaches that create a 3D model and map a texture of a novel object are discussed. The recent achievements of differentiable rendering pipelines are also introduced.

Training Samples for 6D Pose Estimation Previous work using CNNs requires a large number of training images of an object that covers a range of poses in test scenes sufficiently. Since it is difficult to annotate 6D poses of objects manually, synthetic training images are created using a textured 3D model of an object [21]. However, it is difficult to obtain a 3D model with high quality texture from the real world without a special device, such as the BigBIRD Object Scanning Rig [3], since precise pose information is required to correctly align texture images from different views. Using synthetic data introduces the domain gap between synthetic and real images, which should be specially treated with domain adaptation techniques [45]. It is possible to use only approximately 200 real images and apply various augmentation methods to successfully train pose estimation pipelines [2, 27, 31]. However, the performance highly depends on the range of poses in the real training samples. As discussed in [27], the limited coverage of poses in training images causes inaccurate results for novel poses. To overcome this limitation, both real images and synthetic images are used for training [23, 24, 28, 45], which currently achieves state-of-the-art performance. The advantage of using both sources is that synthetic images supplement images for novel poses that are not observed in real images while real images regularize the network from over-fitting to synthetic images. However, both textured 3D models and more than 200 real images with pose annotations are difficult to obtain from the real world. Furthermore, textured 3D models in public datasets are captured separately from constrained environments [3, 13, 15, 19] such as single objects with a simple background and precise camera pose localization tools. However, this well-constrained setup is difficult to replicate in real scenarios, e.g., a target object on a table is often occluded by other objects, camera poses are

[1] https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/smot/

noisy without manual adjustments, and lighting conditions are not consistent. Thus, it is challenging to derive training images of new objects from cluttered scenes.

3D Object Modeling and Multi-View Texturing RGB-D images have been used to build 3D models by presenting an object in front of a fixed camera while rotating a turn table [30], manipulating the object using a robot end-effector [22] or human hands [41]. Alternatively, a mobile robot is used to actively move a camera to build a model of a fixed object in [7]. Even though these methods produce good 3D models in terms of geometry, textures are not optimized or even explicitly considered. Depth images are also required to align different views. On the other hand, it is possible to map multiple images from different views to 3D mesh models using camera pose information [6, 39]. These approaches produce 3D models with high-quality textures since their optimization tries to assign continuous source images to neighboring pixels. However, these methods requires depth images for correcting pose errors, which causes misalignment of color values and disconnected boundaries when different source images are not correctly aligned. Image based rendering (IBR) has been used to complete a large scene by in-painting occluded area using multiple images from different view points [29, 33, 37, 42]. These methods re-project source images to a target image using the relative poses of view points. Then, projected images from different views are integrated with a weighted summation or an optimization based on different objective functions. However, IBR methods are designed to complete large-scale scenes and suffer from noisy estimation of the camera and object poses, which causes blurry or misaligned images.

Differentiable Rendering The recent development of differentiable rendering pipelines enables the rendering process to be included during network training [12, 20, 36]. Therefore, the relationship between the 3D locations of each vertex and UV coordinates for textures are directly associated with pixel values of 2D rendered images, which have been used to create 3D meshes from a single 2D image [12, 20]. Furthermore, it is possible to render trainable features of projected vertices, which can be trained to minimize loss functions defined in a 2D image space.

The purpose of NOL is to generate synthetic images to train CNN-based pose estimators for new objects while minimizing the effort for obtaining training data from real applications. The knowledge of 3D representation and a few observations of objects are usually sufficient for humans to recognize objects in a new environment. Likewise, NOL composes appearances of objects in arbitrary poses using 3D models and a few cluttered and unconstrained images without using depth images, which is sufficient to train a pose estimator and achieve state-of-the-art performance.

3 Neural Object Learning

The objective of NOL is to create an image X^D of an object in a desired pose T^D using K source images, $\{I^1, I^k ... I^K\}$, with pose annotations, $\{T^1, T^k ... T^K\}$,

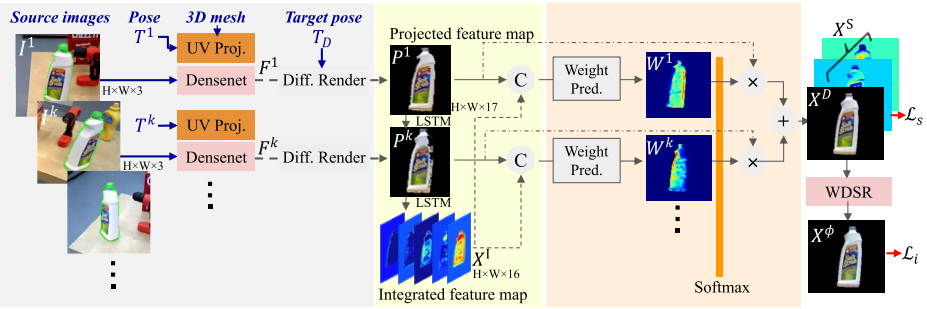

Fig. 2. An overview of the NOL architecture. X^D is used as a rendered output while X^ϕ is used to compute the image loss for training

and object masks that indicate whether each pixel belongs to the object or not in a source image, $\{M^1, M^k ... M^K\}$. We do not assume T^k or M^k to be accurate, which is common if the source images are collected without strong supervision such as marker-based localization or human annotation.

3.1 Network Architecture

Fig. 2 depicts an overview of the network architecture. Firstly, source images are encoded and projected to compute an integrated feature map in a target pose. Secondly, the weighted sum of projected feature maps are computed by predicting weight maps. A decoder block produces a decoded image that is used to compute the image loss.

Integrated Feature Maps Each source image, $I^k \in H \times W \times 3$, is encoded with a backbone network, Densenet-121 [17], to build feature pyramids using the outputs of the first four blocks. Each feature map from the pyramids is processed with a convolutional layer with 3×3 kernels to reduce the number of an output channel of each block to 4, 3, 3, and 3. Each feature map is resized to the size of the original input using bi-linear interpolation. In addition to the feature map with 13 channels, original color images (3 channels) and face angles with respect to camera views (one channel) are concatenated. As a result, the encoded feature map F^k of each input image I^k has 17 channels. The UV coordinates of each vertex are computed using 2D projected locations of each visible vertex in an input pose T^k. These UV coordinates are then projected to the target pose T^D using a differentiable renderer proposed in [12]. Feature values of each pixel in a projected feature map P^k are computed using bi-linear interpolation of surrounding feature values obtained from corresponding pixels from the encoded feature map F^k, which is similar to rendering an object with a separate texture image. The projected feature maps $P^{k \in K}$ are compiled by convolutional Long-Term and Short-Term memory (LSTM) layers to compute the integrated feature map X^I with 16 output channels at the same resolution of the projected feature maps. This LSTM layer enables the network to learn how to extract valuable

pixels from different source images while ignoring outlier pixels caused by pose errors. It is also possible to use a different number of source images without changing the network architecture.

Weight Prediction Block The integrated feature map X^{I} is concatenated with each projected feature map P^k to compute a corresponding weight map W^k in the weight prediction block, which implicitly encodes distances between P^k and X^{I} per pixel. The resulting weight maps $W^{k\in K}$ are normalized with the *Softmax* activation over K projected images. Therefore, the summation of the weighted maps over $W^{k\in K}$ is normalized for each pixel while keeping strong weights on pixels that have remarkably higher weights than others. The weighted sum of projected feature maps using the predicted weight maps produces the weighted feature map X^{S}. Since the first three channels of P^k represent projected color values from source images, the first three channels of X^{S} are a color image, which is referred to as the weighted rendering X^D and the output image of NOL during inference.

Decoder Block Since X^D is obtained by the weighted summation of projected images, color values of pixels X^D are limited to the color range of projected pixels. However, when training the network, the color levels of the source images can be biased by applying randomized color augmentations while maintaining the original colors for the target image. This causes the weight prediction block to be over-penalized even though color levels of X^D are well balanced for given source images. This motivates us to add a module to compensate for these biased errors implicitly during training. An architecture used for the image super-resolution task, WDSR [44], is employed as a decoder block to predict the decoded rendering X^ϕ in order to compute the losses during training. A detailed analysis regarding the role of the decoder is presented in Sec. 5.

3.2 Training

The objective of training consists of two components. The first component, the *image loss*, renders a correct image $\mathcal{L}_{\mathrm{i}}$ in a target pose. The second component, the *smooth loss*, minimizes the high frequency noise of the resulting images $\mathcal{L}_{\mathrm{s}}$.

Image Loss The image loss $\mathcal{L}_{\mathrm{i}}$ computes the difference between a target image X^{GT} in a target pose and the decoded output X^ϕ. In addition to the standard L1 distance of each color channel, the feature reconstruction loss [18] is applied to guide the predicted images to be perceptually similar to the target image as formulated by

$$\mathcal{L}_{\mathrm{i}} = \frac{1}{M^D} \sum_{p\in M^D} \lambda_i |X_p^\phi - X_p^{GT}|_1 + \lambda_f |\psi(X_p^\phi) - \psi(X_p^{GT})|_1, \tag{1}$$

where M^D is a binary mask that indicates whether each pixel has at least a valid projected value from any input $I^{k\in K}$, and $\psi(\cdot)$ denotes outputs of a backbone network with respect to the image. The outputs of the first two blocks of DenseNet [40] are used for the feature reconstruction loss. The parameters λ_i and λ_f are used to balance the losses.

Smooth Loss Even if the objective function in Eq. 1 guides the network to reconstruct the image accurately, the penalty is not strong when the computed image has high frequency noise. This is the reason why IBR and image in-painting methods [29, 37, 42] usually employ a smooth term. This minimizes the gradient changes of neighboring pixels even if pixel values are obtained from different source images. Similarly, we add a loss function to ensure smooth transitions for neighboring pixels in terms of color values as well as encoded feature values. This is formulated as

$$\mathcal{L}_{\mathrm{s}} = \frac{\lambda_s}{M^D} \sum_{p \in M^D} \nabla^2 X_p^S. \tag{2}$$

The loss function creates a penalty when the gradients of color and feature values of each pixel are inconsistent with those of neighboring pixels. In contrast to the image loss, the weighted feature map X^S is used directly instead of using the decoded output X^ϕ. Thus, the weight prediction block is strongly penalized when producing high frequency changes in the predicted weight maps $W^{k \in K}$ and the weighted feature map X^S.

Training using Synthetic Images with Pose Errors Synthetic images are created to train the NOL network. 3D models from the YCB-Video dataset [43] are used while ignoring original textures and instead applying randomly sampled images from MS-COCO [26] as textures. After sampling a 3D model and a texture image, 10 images are rendered as a batch set in different poses with random background images. During training, one image from a batch set is chosen as a target image X^{GT} and its pose is set to a desired pose T^D, and the other K images are assigned as input images $I^{k \in K}$. To simulate different lighting conditions, color augmentations are applied to the input images while no augmentation is applied to the target image. Pose errors are also simulated by applying random perturbations to the actual poses $T^{k \in K}$ of the input images during training. As a result of the perturbations, vertices are projected to wrong 2D locations, which produces wrong UV coordinates per vertex and outlier pixels in projected feature maps at a desired pose. This pose augmentation forces the network to be robust to pose errors while attempting to predict an accurate image in the target pose. A total of 1,000 training sets, consisting of 10 images per set, are rendered for training. The same weights are used to render objects in all evaluations in the paper after training for 35 epochs. Detailed parameters used for data augmentation are listed in the supplementary material.

3.3 Gradient Based Pose Refinement and Rendering

The error in an input pose T^k causes crucial outlier pixels in the projected feature map P^k at the desired pose. Fig. 3 shows an example of wrong pixels in the projected feature map obtained from the ground plane (blue) in the source image due to the error of the initial pose $T^k_{t=0}$. As discussed in Sec. 2, the differentiable renderer enables derivatives of 3D vertices of a 3D model to be computed with respect to the error defined in 2D space. Since 3D locations of vertices and UV

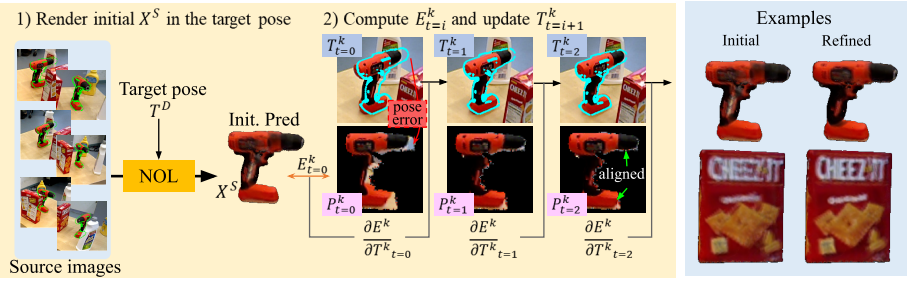

Fig. 3. An overview of the proposed pose refinement process and example results. The partial derivative of the projection error E^k is used to update each input pose T^k

coordinates in the desired pose are derived by matrix multiplications, which is differentiable, the gradient of each input pose T^k can be derived to specify a direction that decreases the difference between a desired feature map and each projected feature map P^k. The first prediction of NOL, $X^S_{t_0}$, without refinement is used as an initial desired target. The goal of the refinement step is to minimize the error, E^k, between the initial target and each projected feature map P^k. In every iteration, the partial derivative of the projection error E^k with respect to each input pose $T^k_{t=t_i}$ is computed by

$$\Delta T^k_{t_i} = \frac{\partial E^k}{\partial T^k_{t_i}} = \frac{\partial |X^S_{t_0} - P^k_{t_i}|}{\partial T^k_{t_i}}, \tag{3}$$

and the input pose at the next iteration $T^k_{t_i+1}$ is updated with a learning step δ, i.e. $T^k_{t_i+1} = T^k_{t_i} - \delta \Delta T^k_{t_i}$. In our implementation, translation components in T^k are directly updated using $\Delta T^k_{t_i}$. On the other hand, updated values for rotation components, $\mathbb{R}^{3\times3}$, do not satisfy constraints for the special orthogonal group, $SO(3)$. Thus, the rotation component of $\Delta T^k_{t_i}$ is updated in the Euler representation and converted back to the rotation matrix. As depicted in Fig. 3, the iterations of the refinement step correctly remove the pose error so that the projected image no longer contains pixels from the background, which decreases blur and mismatched boundaries in the final renderings. After refining every input pose $T^k_{t=t_0}$ until the error does not decrease or the number of iterations exceeds 50, the final output X^D is predicted using the refined poses $T^k_{t=t_f}$.

4 Evaluation

This section presents the evaluation of the proposed NOL approach in relation to the task of 6D object pose estimation. We introduce datasets used in the evaluation and provide implementation details of NOL. The evaluation results show that the quality of NOL images are sufficient for pose estimation and leads to outperforming other methods trained with synthetic images using textured 3D models or real images.

4.1 Datasets

Three datasets are used for evaluation: LineMOD [13], LineMOD-Occlusion (LineMOD-Occ) [1], and a novel dataset created to reflect challenges of real environments. LineMOD and LineMOD-Occ have been used as standard benchmarks for 6D pose estimation of objects. LineMOD provides textured 3D models and 13 test sequences that have an annotated object per image. The 3D models are created by placing each object alone on a plane and performing a voxel-based 3D reconstruction [13]. LineMOD-Occ is created by additionally annotating eight objects presented in a test sequence in LineMOD. Previous works reporting results on these dataset have used either synthetic images using given 3D models [21, 34] or 15% real images obtained from test sequences (183 images per object) [2, 27, 31, 35] for training. In contrast to previous work, images created by NOL are used to train both a pose estimator and a 2D detector.

SMOT A new dataset, Single sequence-Multi Objects Training, is created to reflect real noise when training images are collected from a real scenario, i.e. a mobile robot with a RGB-D camera collects a sequence of frames while driving around a table to learning multiple objects and tries to recognize objects in different locations. The dataset consists of two training sequences and eleven test sequences using eight target objects sampled from the YCB-Video [43] dataset. Two training sequences, that include four target objects per sequence, are collected by following trajectories around a small table. Camera poses of frames are self-annotated by a 3D reconstruction method [47] while building a 3D mesh of the scene. 3D models provided in YCB-Video are aligned to the reconstructed mesh and corresponding object poses are computed using camera poses. No manual adjustment is performed to preserve errors of self-supervised annotations. On the other hand, test images are collected with visible markers to compute more accurate camera poses while moving the robot manually in front of different types of tables and a bookshelf. As a result, each object has approximately 2,100 test images. The supplementary material includes more details of the dataset.

4.2 Implementation Details

For the NOL network, the resolution of input and target images are set to 256×256. A number of source images, K, is set to 8 for training and 6 for inference. The loss weights are set to, λ_i=5, λ_f=10, and λ_s=1. Detail parameters are reported in the supplementary material.

Sampling of Source Images To render NOL images for training a pose estimator, source images are sampled from the training sequences of a dataset. For LineMOD and LineMOD-Occ, a maximum of 16 images per object are sampled from the same training splits of real images used in previous work [2, 27, 31, 35]. Since objects are fully visible in the training set, images are simply sampled using pose annotations. In each sampling iteration, an image is randomly sampled and images that have similar poses (less than 300mm translation and 45° rotation), are removed. The sampling is terminated when no more images remain. In contrast to LineMOD and LineMOD-Occ, the visibility of each object

varies in the training set of SMOT. In order to minimize the number of source images, a frame with the highest value is selected at each sampling iteration by counting the number of visible vertices that have not been observed in the previously sampled frames. The sampling iteration is terminated when no frame adds additional observed vertices.

Rendering NOL Images Each target object is rendered using NOL in uniformly sampled poses defined over an upper-hemisphere for every 5° for both azimuth and elevation. For each target pose, 6 images are chosen from sampled images using the same image sampling procedure while limiting the target vertices to visible vertices in the pose. As a result, 1296 images are rendered. The in-plane rotation is applied to rendered images by rotating them from -45° to 45° for every 15°. For LineMOD and LineMOD-Occ, synthetic images are also rendered in the same sampled poses using given 3D models to train a pose estimator for comparison. Rendering takes approximately 3.6 seconds per image, which consists of 19ms for the initial prediction, 3597ms for the pose refinement with maximum 25 iterations, and 19ms for the final prediction.

Training Recognizers To show whether NOL images are sufficient to estimate poses of objects in arbitrary poses using a recent RGB-based pose estimation method, an official implementation of state of the art for 6D pose estimation using color images, Pix2Pose [27], is used. This method is one of the most recent methods [24, 27, 45] that predict objects' coordinate values per pixel. To increase the training speed and decrease the number of training parameters, the discriminator and the GAN loss are removed. All other aspects are kept the same except for the number of training iterations, which is set to approximately 14K because of the decreased number of trainable parameters. Resnet-50 [10] is used as a backbone for the encoder and weights are initialized with pre-trained weights on ImageNet [4]. Since the online augmentation of Pix2Pose uses cropped patches of objects on a black background, no further image is created for training. On the other hand, images that contain multiple objects are created to train a 2D detector by pasting masked objects onto random background images. A total of 150,000 training images are created and used to train Retinanet [25] with Resnet-50 backbone.

4.3 Metrics

The AD{D|I} score [13] is a common metric used to evaluate pose estimation. This metric computes the average distance of vertices between a ground truth pose and a predicted pose (ADD). For symmetric objects, distances to the nearest vertices are used (ADI). The predicted pose is regarded as correct if the average distance is less than 10% of the diameter of each object.

The recent challenge for pose benchmark [14] proposes a new metric that consists of three different pose errors (VSD, MSSD, and MSPD) [16] and their average recall values over different thresholds. The metric is used to evaluate the performance on LineMOD-Occ because it is more suitable and also allows comparison with the results of the recent benchmark.

Fig. 4. Rendered results of SMOT objects using a training sequence. NOL successfully removes pixels from the background and other objects due to pose refinement

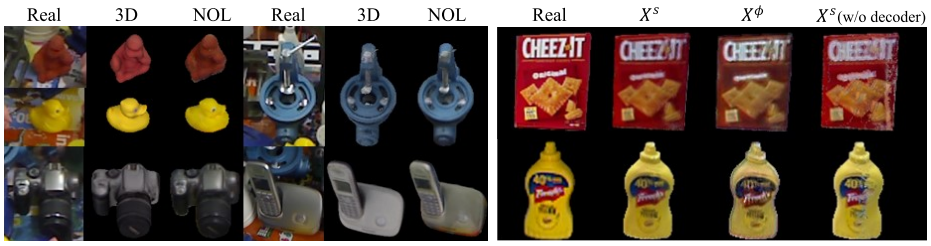

Fig. 5. Left: examples of rendered images of objects in LineMOD using NOL. Right: outputs of weighted renderings, decoded renderings, and weighted renderings after training without the decoder block

4.4 Quality of Rendered Images

Fig. 4 shows the rendered images using a training sequence of SMOT. Renderings in *3DRecont* show object models extracted directly from a reconstructed 3D mesh of the training scene [47]. Both MVS [39] and G2LTex [6] use the same images sampled for rendering NOL images with the same pose annotations. Multi-view texturing methods create less blurry textures than NOL for planar surfaces since they try to map an image to a large area without combining pixels from other images. However, this induces misaligned results when the input poses are inaccurate even if depth images are used to optimize poses as in [6], e.g. doubled letters on the cheeze-it box. On the other hand, results of NOL after pose refinement (last column) removes these doubled textures by correcting pose errors using color images only, which is robust to depth registration errors. Furthermore, NOL successfully rejects outlier pixels from other objects and the background.

4.5 Pose Estimation: LineMOD

The left side of Table 1 shows the results when RGB images are used for pose estimation. Since no real image is directly cropped and used to train the pose estimator, the results are mainly compared against methods that use synthetic images only for training. The method trained with NOL images outperforms the

Table 1. Evaluation results on LineMOD. The ADD score is used except for *Eggbox* and *Glue* that use the ADI score

Types	Required data for training										
Texture	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓
No. real images	14	-	-	-	-	183	14	-	-	183	183
GT 6D pose	✓	-	-	-	-	-	✓	-	-	✓	✓
Depth image	-	-	-	-	-	✓	-	-	-	✓	✓
Test type	RGB w/o refinement						RGB + ICP			RGB-D	
Training on	**NOL**	synthetic images				real+syn	**NOL**	syn		real+syn	
Method	[27]	[27]	[21]	[34]	[45]	[32]	[27]	[27]	[34]	[40]	[11]
Ape	35.4	10.0	2.6	4.0	**37.2**	19.8	**95.2**	92.7	20.6	92.3	**97.3**
Benchvise	55.6	13.4	15.1	20.9	66.8	**69.0**	**99.0**	90.4	64.3	93.2	**99.7**
Camera	37.5	4.4	6.1	30.5	24.2	**37.6**	**96.6**	77.9	63.2	94.4	**99.6**
Can	**65.5**	26.4	27.3	35.9	52.6	42.3	**97.6**	85.8	76.1	93.1	**99.5**
Cat	**38.1**	24.8	9.3	17.9	32.4	35.4	**98.6**	90.1	72.0	96.5	**99.8**
Driller	52.2	9.1	12.0	24.0	**66.6**	54.7	**98.0**	66.1	41.6	87.0	**99.3**
Duck	14.7	3.7	1.3	4.9	26.1	**29.4**	**89.1**	82.3	32.4	92.3	**98.2**
Eggbox	**93.7**	34.6	2.8	81.0	73.4	85.2	**99.2**	88.7	98.6	99.8	**99.8**
Glue	63.1	35.1	3.4	45.5	75.0	**77.8**	**96.5**	92.2	96.4	**100**	**100**
H.puncher	34.4	3.7	3.1	17.6	24.5	**36.0**	**93.2**	46.3	49.9	92.1	**99.9**
Iron	57.9	30.4	14.6	32.0	**85.0**	63.1	**99.3**	93.5	63.1	97.0	**99.7**
Lamp	54.2	6.7	11.4	**60.5**	57.3	75.1	**96.6**	39.3	91.7	95.3	**99.8**
Phone	41.8	13.8	9.7	33.8	29.1	**44.8**	**92.8**	79.1	71.0	92.8	**99.5**
Average	49.5	16.6	9.1	28.7	50.0	**51.6**	**96.3**	78.8	64.7	94.3	**99.4**

same method trained with synthetic images using the given 3D models. This verifies that the quality of NOL images are more similar to appearances of real objects. The results of objects with metallic or shiny surfaces, e.g., *Camera*, *Phone* ,and *Can*, show significant improvements against other results obtained with synthetic training images without any real observation. As depicted in Fig. 5, NOL realizes the details of shiny and metallic materials by optimizing colors of each view separately. The performance is competitive to the best method that uses real color and depth images of objects for domain adaptation.

NOL images tend to contain noisy boundaries especially around the lower parts of objects where NOL mistakenly extracts pixels from the background table (see the bottom of *Phone* in Fig. 5). This limits the translation precision of predictions along the principle camera axis (z-axis). To decrease the translation errors, ICP refinement is applied to refine poses using depth images as reported on the right side of Table 1. The method trained with NOL images outperforms state-of-the-art trained with synthetic images. The result is competitive to state-of-the-art results in the last two columns even though the methods [11, 40] use more than 13 times the number of real images for training.

4.6 Pose Estimation: LineMOD-Occ

The same models used in the LineMOD evaluation are used to test on LineMOD-Occ as reported in Table 2. Similar to the LineMOD evaluation, methods trained by synthetic images are mainly compared. The evaluation protocol used in the recent pose challenge [14] is applied with the same test target images. The result of [27] using synthetic images is obtained by re-training the network with Resnet-50 backbone, which performs better than the official result in the challenge [14].

Table 2. Evaluation results on LineMOD-Occ. The results of other methods are cited from the last 6D pose challenge [14]

Type	RGB w/o refinement					RGB+ICP3D			Depth
Train source	**NOL**	Syn. using 3D models				**NOL**	Syn		3D model
Method	[27]	[27]	[34]	[24]	[45]	[27]	[27]	[34]	[38]
BOP score	**37.7**	20.0	14.6	37.4	16.9	**61.3**	45.3	23.7	58.2

Table 3. Evaluation results on the SMOT dataset

Type	RGB				RGB-D (ICP3D)			
3D Model	Precise			Recont	Precise			Recont
Train source	Real	G2Ltex	**NOL**	**NOL**	Real	G2Ltex	**NOL**	**NOL**
AD{D|I} score	25.0	25.7	**35.5**	22.5	86.5	82.0	**90.0**	80.6
$\text{mAP}_{IoU=50}$	88.3	90.2	**90.7**	73.9	-	-	-	-

The performance of this method is significantly improved by using images created by NOL for training with RGB inputs and with the inclusion of ICP refinement using depth images. Furthermore, using NOL images leads to the method outperforming state of the art using color images [24] and the best performing method on this dataset [38].

4.7 Pose Estimation: SMOT

The pose estimator [27] and the 2D detection method [25] are trained using crops of entire real images where each object is visible more than 50%. This is an average of 364 images per objects. For G2Ltex and NOL, up to 16 images per object are sampled as explained in Sec. 4.2 to render training images.

Table 3 shows pose estimation and 2D detection results in terms of the AD{D|I} score and the mean Average Precision (mAP) [5]. The results using NOL images outperforms other methods using real images and models textured by G2Ltex for both RGB and RGB-D inputs. This is because real images do not fully cover target poses and objects are often occluded by other objects in training images. The comparison with G2LTex provides a quantitative verification regarding the better quality of renderings created by NOL using the same source images. The results denoted with *Recont* are obtained using reconstructed 3D models instead of precise 3D models for rendering. The performance drops significantly since the NOL images are noisier and blurrier due to geometrical errors of models. This indicates that precise 3D models are important for NOL to generate high-quality images.

5 Ablation Study

This section analyzes factors that influence the quality of NOL images. The perceptual similarity [46] is used to measure quality of generated images in com-

Table 4. Perceptual similarity (smaller is better) of rendered images with different configurations

	Components			Loss functions			
Setup	All	w/o Decoder	w/o LSTM	$\mathcal{L}_1$ (RGB)	$\mathcal{L}_\text{i}$	$\mathcal{L}_\text{i} + \mathcal{L}_\text{s}$	$+\mathcal{L}_\text{GAN}$
w/o Ref	**0.181**	0.289	0.247	0.194	0.184	**0.181**	0.188
w/ Ref	**0.173**	0.279	0.241	0.184	0.177	**0.173**	0.183

parison to the real images. We sample 10 test images per object in SMOT (80 images), render the objects at GT poses, and compare them with real images.

Components Table 4 shows the most significant improvement comes from the decoder. The right side of Fig. 5 shows qualitative results of weighted renderings X^D, decoded renderings X^ϕ, and results after training the network without the decoder. As discussed in Sec. 3, the network trained with the decoder converges to produce X^D in a neutral color level as a reference image while the decoder absorbs over-penalized errors caused by randomly biased colors. The results denoted as *w/o LSTM* are derived by replacing the LSTM module with an simple average over projected features P^k. In this case, the results drops significantly since the LSTM module highlights valuable pixels among projected pixels. The refinement step consistently improves the image quality for all configurations.

Loss functions The best results are made with all proposed losses $\mathcal{L}_\text{i}+\mathcal{L}_\text{s}$. The perceptual loss in addition to the standard L1 loss significantly improves the performance by guiding the network to preserve perceptual details, like edges, with less blurry images while the smooth loss $\mathcal{L}_\text{s}$ additionally reduces the high-frequency noise. As the adversarial loss [8] provides better performance for image reconstruction tasks, the adversarial loss $\mathcal{L}_\text{GAN}$ is added to our loss function, which does not improve the result in our implementation.

6 Conclusion

This paper proposed a novel method that creates training images for pose estimators using a small number of cluttered images. To the best of our knowledge, this is the first attempt to learn multiple objects from a cluttered scene for 6D pose estimation, which minimizes the effort for recognizing a new object. Our code[2] and dataset are publicly available to motivate future research in this direction. The method can be further extended to optimize 3D models for reducing geometrical errors, which accomplishes the fully self-supervised learning of objects from cluttered scenes in real environments.

Acknowledgment The research leading to these results has partially funded by the Austrian Science Fund (FWF) under grant agreement No. I3969-N30 (InDex) and the Austrian Research Promotion Agency (FFG) under grant agreement No. 879878 (K4R).

[2] https://github.com/kirumang/NOL

References

1. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 536–551. Springer (2014)
2. Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., Rother, C.: 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 (CVPR). pp. 3364–3372 (2016)
3. Calli, B., Walsman, A., Singh, A., Srinivasa, S.S., Abbeel, P., Dollar, A.M.: Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. IEEE Robotics and Automation Magazine (RAM) **22**, 36–52 (2015)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255 (2009)
5. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision (IJCV) **88**, 303–338 (2010). https://doi.org/10.1007/s11263-009-0275-4
6. Fu, Y., Yan, Q., Yang, L., Liao, J., Xiao, C.: Texture mapping for 3d reconstruction with rgb-d sensor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4645–4653 (2018)
7. Fäulhammer, T., Ambruş, R., Burbridge, C., Zillich, M., Folkesson, J., Hawes, N., Jensfelt, P., Vincze, M.: Autonomous learning of object models on a mobile robot. IEEE Robotics and Automation Letters (RA-L) **2**(1), 26–33 (Jan 2017). https://doi.org/10.1109/LRA.2016.2522086
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems 27 (NeurIPS). pp. 2672–2680. Curran Associates, Inc. (2014)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2961–2969 (2017)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
11. He, Y., Sun, W., Huang, H., Liu, J., Fan, H., Sun, J.: Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11632–11641 (2020)
12. Henderson, P., Ferrari, V.: Learning single-image 3d reconstruction by generative modelling of shape, pose and shading. International Journal of Computer Vision (IJCV) **128**, 835–854 (2019). https://doi.org/10.1007/s11263-019-01219-8
13. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) Computer Vision – ACCV 2012. pp. 548–562. Springer (2013)
14. Hodaň, T., Brachmann, E., Drost, B., Michel, F., Sundermeyer, M., Matas, J., Rother, C.: Bop: Benchmark for 6d object pose estimation (2019), https://bop.felk.cvut.cz (visited on 2020-02-21)

15. Hodaň, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. IEEE Winter Conference on Applications of Computer Vision (WACV) pp. 880–888 (2017)
16. Hodaň, T., Matas, J., Obdržálek, Š.: On evaluation of 6d object pose estimation. In: Hua, G., Jégou, H. (eds.) Computer Vision – ECCV 2016 Workshops. pp. 606–619. Springer (2016)
17. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708 (2017)
18. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 694–711. Springer (2016)
19. Kaskman, R., Zakharov, S., Shugurov, I., Ilic, S.: Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In: The IEEE International Conference on Computer Vision Workshops (ICCVW) (2019)
20. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3907–3916 (2018)
21. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 1521–1529 (2017)
22. Krainin, M., Henry, P., Ren, X., Fox, D.: Manipulator and object tracking for in-hand 3d object modeling. The International Journal of Robotics Research (IJRR) **30**(11), 1311–1327 (2011)
23. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. International Journal of Computer Vision (IJCV) **128**(3), 657–678 (2020). https://doi.org/10.1007/s11263-019-01250-9
24. Li, Z., Wang, G., Ji, X.: CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 7678–7687 (2019)
25. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2999–3007 (2017)
26. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 740–755. Springer (2014)
27. Park, K., Patten, T., Vincze, M.: Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 7668–7677 (2019)
28. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting network for 6dof pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4556–4565 (June 2019)
29. Philip, J., Drettakis, G.: Plane-based multi-view inpainting for image-based rendering in large scenes. In: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. pp. 1–11 (2018)
30. Prankl, J., Aldoma, A., Svejda, A., Vincze, M.: Rgb-d object modelling for object recognition and tracking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 96–103 (2015)

31. Rad, M., Lepetit, V.: 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 (ICCV). pp. 3828–3836 (2017)
32. Rad, M., Oberweger, M., Lepetit, V.: Domain transfer for 3d pose estimation from color images without manual annotations. In: Jawahar, C., Li, H., Mori, G., Schindler, K. (eds.) Computer Vision – ACCV 2018. pp. 69–84. Springer (2019)
33. Shum, H., Kang, S.B.: Review of image-based rendering techniques. In: Visual Communications and Image Processing 2000. vol. 4067, pp. 2–13. International Society for Optics and Photonics (2000)
34. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3d orientation learning for 6d object detection from rgb images. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. pp. 712–729. Springer (2018)
35. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 292–301 (2018)
36. Thies, J., Zollhöfer, M., Nieundefinedner, M.: Deferred neural rendering: Image synthesis using neural textures. ACM Trans. Graph. **38**(4) (2019). https://doi.org/10.1145/3306346.3323035
37. Thonat, T., Shechtman, E., Paris, S., Drettakis, G.: Multi-view inpainting for image-based scene editing and rendering. In: International Conference on 3D Vision (3DV). pp. 351–359. IEEE (2016)
38. Vidal, J., Lin, C.Y., Lladó, X., Martí, R.: A method for 6d pose estimation of free-form rigid objects using point pair features on range data. Sensors **18**(8), 2678 (2018)
39. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! large-scale texturing of 3d reconstructions. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 836–850. Springer (2014)
40. Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., Savarese, S.: DenseFusion: 6D object pose estimation by iterative dense fusion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3343–3352 (2019)
41. Wang, F., Hauser, K.: In-hand object scanning via rgb-d video segmentation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 3296–3302 (2019)
42. Whyte, O., Sivic, J., Zisserman, A.: Get out of my picture! internet-based inpainting. In: British Machine Vision Conference (BMVC) (2009)
43. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. Robotics: Science and Systems (RSS) (2018)
44. Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., Huang, T.: Wide activation for efficient and accurate image super-resolution. arXiv preprint arXiv:1808.08718 (2018)
45. Zakharov, S., Shugurov, I., Ilic, S.: DPOD: 6D pose object detector and refiner. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 1941–1950 (2019)
46. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 586–595 (2018)

47. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018)