2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Object Detection and Pose Estimation based on Convolutional Neural Networks Trained with Synthetic Data

Josip Josifovski[1], Matthias Kerzel[1], Christoph Pregizer[2], Lukas Posniak[2], Stefan Wermter[1]

Abstract— **Instance-based object detection and fine pose estimation is an active research problem in computer vision. While the traditional interest-point-based approaches for pose estimation are precise, their applicability in robotic tasks relies on controlled environments and rigid objects with detailed textures. CNN-based approaches, on the other hand, have shown impressive results in uncontrolled environments for more general object recognition tasks like category-based coarse pose estimation, but the need of large datasets of fully-annotated training images makes them unfavourable for tasks like instance-based pose estimation.**

We present a novel approach that combines the robustness of CNNs with a fine-resolution instance-based 3D pose estimation, where the model is trained with fully-annotated synthetic training data, generated automatically from the 3D models of the objects. We propose an experimental setup in which we can carefully examine how the model trained with synthetic data performs on real images of the objects. Results show that the proposed model can be trained only with synthetic renderings of the objects' 3D models and still be successfully applied on images of the real objects, with precision suitable for robotic tasks like object grasping. Based on the results, we present more general insights about training neural models with synthetic images for application on real-world images.

I. INTRODUCTION

To recognize an object and determine its location and pose in the environment is still a challenging task in computer vision and an active research topic. Retrieving the spatial information about an object is crucial for robotic tasks like vision-based scene understanding, grasping or object manipulation. Performing such a task becomes complex if the objects and their environment are not specifically adjusted for robots by controlling lighting conditions, the positions of the objects or their textures, or placing distinctive markers on the objects.

Historically, most models for instance-based object recognition have extracted local descriptors around points of interest in images [1], [2]. While these features are invariant to rotation, scale, and partial occlusions, they are not very robust to illumination changes or partial deformations of the object. Furthermore, since they depend on interest points and local features, they only work well for objects with texture.

Recently, there has been a shift in the state of the art by using Convolutional Neural Network (CNN) approaches for

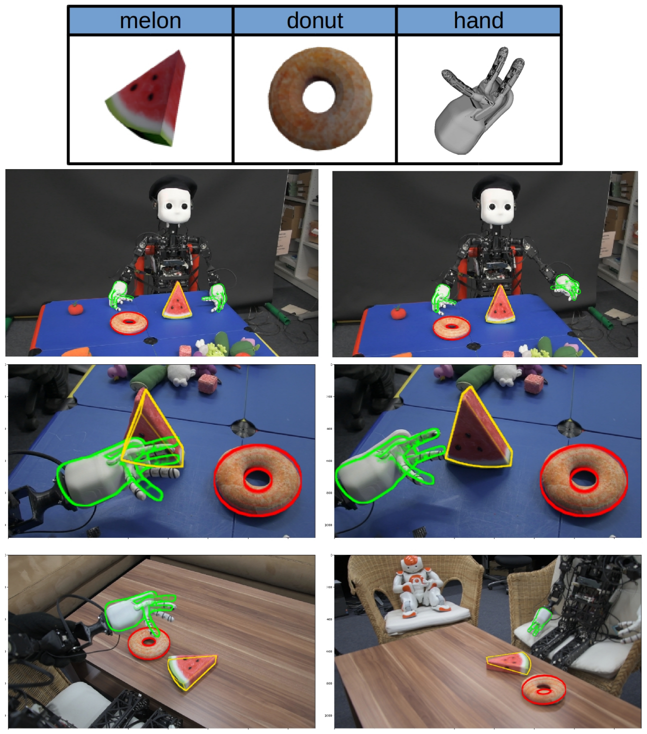

Fig. 1. Top: the 3D models used for generating synthetic training data. Bottom: the estimated 3D pose of the objects and a robotic hand using our CNN-based pose estimation model. The 3D poses are visualized by overlaying the object contours onto the image. The model can successfully handle textured or texture-less objects as well as deviations between the 3D model used for training and the real object.

computer vision tasks, such as image classification [3] or object detection [4]. The advantage of CNNs is their ability for learning discriminative, hierarchical features directly from the underlying training data, removing the need for designing domain-specific feature extractors. CNNs have shown strong generalization abilities, making them more robust to lighting conditions, object textures, partial occlusions, and even object deformations. However, the performance of CNNs relies upon the available training data, which is scarce for more complex computer vision tasks like pose estimation, due to the many variables involved. Because of this and the need for annotated data, manual preparation of a dataset becomes a time-consuming process. In contrast, advances in computer-generated imagery allow automation of the data generation process by creating fully- annotated, synthetic data. However, the benefits come at the cost of decreased realism, and it is an open research question if such data can be useful for training if the model is in the end used to process real-world data from a real vision sensor.

*The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169), and the Hamburg Landesforschungsförderungsprojekt CROSS.

[1]Knowledge Technology, Department of Informatics, University of Hamburg, Germany. `4josifov / kerzel / wermter @informatik.uni-hamburg.de`

[2]Viewlicity GmbH, Schferkampsallee 42, 20357 Hamburg, Germany `pregizer / posniak @viewlicity.com`

In this paper, we present a CNN-based approach for instance-based object detection and 3D pose estimation that uses synthetic training data. Further, we describe the procedure for creating different types of synthetic training data and an experimental setup which allows for an objective evaluation of the model's performance on real images when it has been trained with synthetic training data.

The results contribute to a better understanding of the model architecture, the training procedure, and the training data properties which are useful to "bridge the gap" between synthetic and real data for object detection and pose estimation. Finally, we summarise general advice on training neural models with synthetic images for applications based on real-world images and demonstrate the use of our approach in a robotic grasping scenario with NICO the *Neuro-Inspired COmpanion* humanoid robot [5] depicted in Figure 1.

II. BACKGROUND AND RELATED WORK

The recent success of convolutional neural networks in image classification, e.g., on the challenging ImageNet benchmark dataset [3], [6] has sparked interest to apply CNN-based methods for additional computer vision tasks. Since then, CNN-based models that perform simultaneous object classification and localization [4] were developed, with recent models [7], [8] showing considerable improvement in the accuracy of the predictions or the processing time. However, pose estimation is a harder task than localization since it subsumes classification, localization, and recognition of the correct orientation of multiple objects in an image. Due to the complexity of the problem and the need of a fully-annotated training datasets of images, which capture the variety of factors that need to be learned, very few methods tackle instance-based object pose estimation using completely CNN-based models.

Wohlhart et al. [9] introduce a CNN-based method for calculating descriptors that encode the object's class and pose for an instance-based recognition. These descriptors are generated by a CNN that is trained to generate well-separated descriptors in Euclidean space so that the distance between descriptors for the same object but different orientations is proportional to the difference in the object's pose. The training is enriched with synthetic training data using 3D models of the objects to generate images under different camera viewpoints. To determine the object and its pose in a test image, they extract a descriptor of the image and perform nearest-neighbour search to descriptors generated from predefined views for each object and its pose.

In contrast, Su et al. [10] rely exclusively on CNN-based architectures to determine an object's bounding box, class, and viewpoint for category-based pose estimation. They generate synthesized RGB training data from various 3D models within each category and use the data to train a CNN for camera viewpoint estimation. Regarding the network architecture, the authors indicate that the problem of viewpoint estimation is category-dependent; therefore, the classical CNN network topology cannot be applied when training for multiple categories. Su et al. evaluate the model when trained with synthesized-only, with real-only, and with combined training data. They report that synthesized data training is better than training with real data, while the combination of both yields slightly better results.

A recent approach by Tobin et al. [11] evaluates the transfer of knowledge from a simulated to a real-world scenario for a CNN-based model for object localization. They create a simulated environment that resembles the real world regarding 3D configuration and randomly render objects without photo-realistic appearance, while randomizing colors, lighting conditions or object positions in the generated images. They hypothesize that if the randomization is high enough, the CNN-based model should generalize well just from simulated data to perform a localization task in the real world without additional fine-tuning. The trained model is then tested in a real-world grasping task. According to their results, the task can be learned only from the simulated training images.

In contrast, Planche et al. [12] focus on creating realistic depth scans from 3D models in simulation, by modeling the error of the sensor device. They apply the synthetic depth scans to train CNN descriptors based on the method proposed by Wohlhart et al. [9]. Their experiments indicate that the added realism to the simulated data improves the performance of the task, thus making the transition from simulation to real world smoother.

Finally, in parallel to our work, two other approaches have been published very recently: the PoseCNN [13], which is a CNN-based model with complex architecture that tackles the pose estimation problem through a three-stage process, and the BB8 model [14] which is a two-stage approach. While both approaches tackle similar problems, the main difference to our approach is that they rely on real images of the objects, using synthetic data only for augmentation of the training data or improvement of the results, while our approach relies only on synthetic views generated from the objects 3D models. Both methods also treat the pose estimation as a regression problem, while we treat it as a classification problem. In this regard, our methodology is informed by the work of Su et al. [10], however, different from their approach where only the viewpoint estimator is trained with synthetic data, we train both the object detector and the viewpoint estimator with synthetic data. Furthermore, their aim has been category-based object pose estimation, and their results, while impressive for such a general problem, are still not precise enough to be practically useful in robotic tasks. Therefore, we reduce the scope of the problem to instance-based detection and pose estimation and propose different architecture with the aim to increase the precision of the model, such that it can be used for practical tasks, e.g., in robotic grasping of objects whose 3D models are known. Our underlying research question, the transferability of vision networks trained with synthetic data to a real-world scenario, is most related to the work of Tobin et al. [11]. However, where their model maps from an image of objects to their positions, we map from an image to the objects' 3D poses, so that information about the rotation is also available.

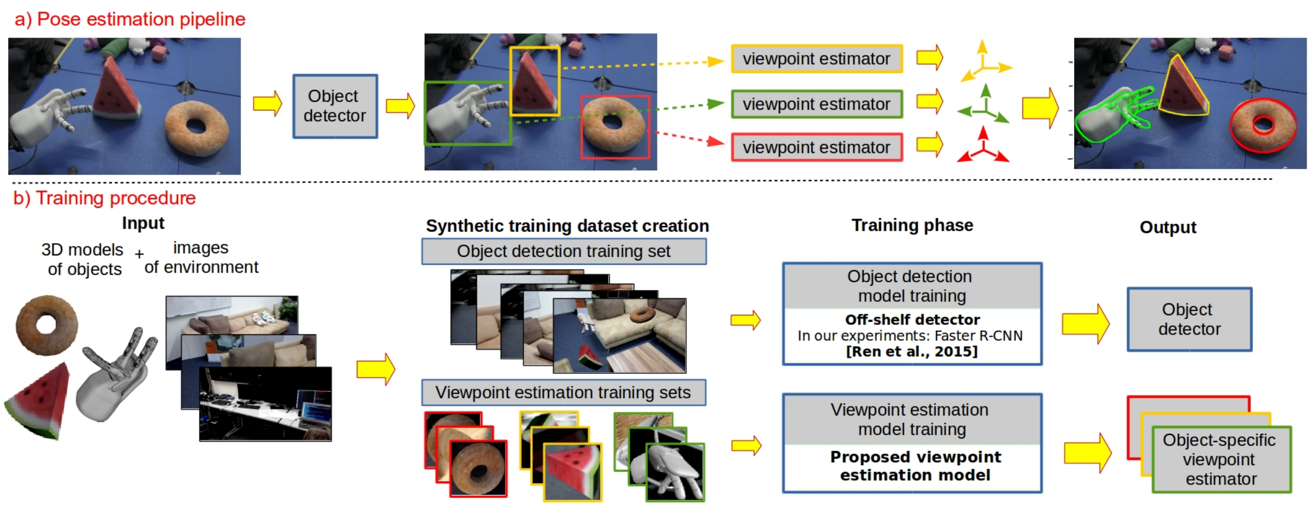
a) Pose estimation pipeline
Object detector
viewpoint estimator
viewpoint estimator
viewpoint estimator
b) Training procedure
Input
3D models of objects
+
images of environment
Synthetic training dataset creation
Object detection training set
Viewpoint estimation training sets
Training phase
Object detection model training
Off-shelf detector
In our experiments: Faster R-CNN
[Ren et al., 2015]
Viewpoint estimation model training
Proposed viewpoint estimation model
Output
Object detector
Object-specific viewpoint estimator

Fig. 2. The proposed approach: a) Pose estimation model as a sequence of object detector and viewpoint estimator. b) Procedure for training of the object detector and viewpoint estimation components.

III. METHOD

To achieve instance-based object detection and pose estimation, we use the modular architecture presented in Figure 2a. The first module of the architecture is a CNN-based detector which detects and classifies regions of the image. For a given image it outputs the bounding boxes and the labels of the detected objects. The labeled bounding box output of the detector module becomes the input for an object-specific CNN-based viewpoint estimation model. The viewpoint estimation model calculates the 3D pose under which the object appears in the bounding box by solving a related problem: estimating the viewpoint of the camera related to the object's coordinate system. The viewpoint is represented by the azimuth and elevation angle of the camera relative to the object's coordinate system and the in-plane rotation of the camera around its optical axis. These three angles are the output of the viewpoint estimation component. The final output of the pose estimation model is then a 3D pose of the object within the detected bounding box[1]. Since the architecture is modular, the detector or viewpoint estimator can easily be replaced with a different implementation. For the detector component we use the Faster R-CNN model [16]. For the viewpoint estimator component we propose the model defined in Section III-B which is inspired by [10].

A. Synthetic datasets

For training the detector and viewpoint estimation components, we use the training procedure shown in Figure 2b. The input for the training procedure is the 3D models of the objects and images representing the environment which serve as substitute background[2]. From the input data, we produce two types of training datasets: an object detection dataset and an object viewpoint estimation dataset.

The dataset for object detection is produced by importing the 3D models of the object into the Open Source 3D creation suite Blender [17], randomizing their position and orientation as well as the scene lighting conditions. The objects are then rendered on a transparent background. The generated image is overlaid on a random substitute-background image. The annotation includes the labelled bounding boxes of the objects for image detection. For viewpoint estimation, we create a separate dataset for each object. First, the 3D model of the object is imported into Blender, and the virtual camera is positioned on a random elevation and azimuth position related to the object's coordinate system. Then, the camera is rotated towards the object so that the camera's optical axis crosses through the object's origin. The camera is rotated by a random angle (in-plane rotation) around its optical axis. The lighting in the scene is randomized, and an image is rendered depicting the object on a transparent background. The generated image is cropped according to the object's bounding box and overlaid on a substitute- background image. The annotation of each image contains the azimuth, elevation and in-plane rotation angles. Since the datasets are created in simulation, the annotations are generated automatically without the need of human assistance or a complicated turn-table acquisition setup.

We use the synthetic datasets to train the object detector and the object-specific viewpoint estimators. Once trained, the object detector and object-specific viewpoint estimators can be applied sequentially to estimate the 3D pose of the objects in an image.

B. Viewpoint estimator architecture

The viewpoint estimation with a neural network can be modelled as a regression or a classification task. One problem with regression is that mapping the rotational components to numerical values suffers from discontinuities (e.g. 0 and 360 degrees represent the same angle). To overcome the discontinuity problem, some authors [18], [19] suggest a

[1]The output of our architecture does not provide a direct 6DoF pose, but it is sufficient to reconstruct the complete 6DoF pose under known camera intrinsics in combination with the object's 3D model. For this purpose, well-established approaches like PnP [15] can be used.

[2]These images are an optional input since we can also use a synthetically-generated background. However, we expect that real-world images similar to the environment where the model is applied, e.g. "indoor" or "outdoor", would reduce the number of false positives by the off-shelf detector component.

prior coarse estimation of the angle, where the angle is classified to belong to one of the four 90-degree intervals to eliminate the discontinuity, and then a regressor for each interval is trained. However, another problem arises with symmetrical objects, where special care has to be taken to identify the symmetries of the object to be able to train the regressor, as it has been done in [14].

Because of these drawbacks, we define the viewpoint estimation problem as a classification task, similarly to [10], with rotation represented by 10-degree bins for azimuth, elevation and in-plane rotation of the object w.r.t to the camera. Unlike [10], where independent output units for azimuth, elevation and in-plane rotation are used, we propose an alternative architecture, where the azimuth and elevation outputs are paired together into a two-dimensional azimuth-elevation array of output units, each representing a unique viewpoint around the object, while the in-plane rotation units are kept as an independent one-dimensional array. The reason for this change is that the azimuth and elevation angles are conceptually different from the in-plane rotation angle: when the azimuth or elevation is changed we effectively see a different region of the object, while when the in-plane rotation is changed and the other angles are kept constant, we still see the same region of the object but it appears rotated in the image. We expect that such organization of the output layer would lead to learning more appropriate high-level features for viewpoint estimation by the network.

For the convolutional part of the model, we use the VGG-16 [20] architecture. In contrast to Su et al. [10], where the VGG-16 convolutional base is shared among all objects, and the final fully-connected and output layers are category-specific, we train a complete model for each object, to make sure that the accuracy is maximized by learning only single-object-specific features. While this decision might influence the scalability of the model if many objects need to be estimated at once, our analyses have shown that for each object, retraining just the top convolutional layers gives optimal results, therefore the lower layers of the convolutional base can be shared in order to have an optimal trade-off between scalability and accuracy.

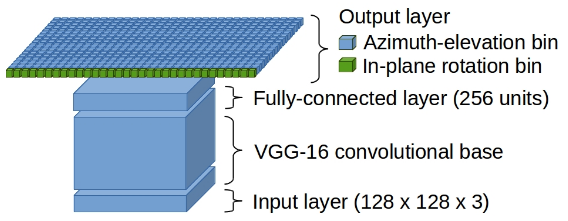
Output layer
Azimuth-elevation bin
In-plane rotation bin
Fully-connected layer (256 units)
VGG-16 convolutional base
Input layer (128 x 128 x 3)

Fig. 3. Architecture of the proposed CNN-based viewpoint estimator

The proposed architecture of the viewpoint estimator is presented in Figure 3. At the bottom is the VGG-16 convolutional base, followed by a single fully-connected layer with rectified linear activation that serves as an intermediate between the convolutional base and the output, and a soft-max output layer where the output units represent 10-degree bins for the elevation-azimuth pairs and the in-plane rotation. We have restricted the range of elevation to 180 degrees since only one of the azimuth or elevation angles should be defined in the full 360-degree range to cover all possible viewpoints on an imaginary sphere around the object. The range of the in-plane rotation angle is 360 degrees as the camera can be rotated fully in any azimuth-elevation point.

To train the viewpoint model, we use categorical-cross entropy. As indicated by Su et al. [10], the output units of the viewpoint model represent geometrical entities and not real classes. Therefore, when encoding the geometrical information in the class labels, we use soft-classification by allowing one input to belong to multiple classes (multiple neighboring bins) independently with a different probability. For each training image, we generate the probability for each bin by centering a normal distribution on the bin corresponding to the ground truth angle values of that image. We sample the distribution over the neighboring bins to get probabilities for each bin. For the azimuth-elevation part we sample a 2D normal distribution, and for the in-plane rotation part, we sample 1D normal distribution.

IV. EXPERIMENTS

We conduct several experiments to evaluate the proposed architecture and the effects of training with synthetic data. For this purpose we use the T-LESS dataset [21], which provides 3D models of various objects taken from the electricity installation domain, as well as RGB images of the real objects and a precise 6DoF annotation of the objects in the images. The dataset is challenging since the objects are texture-less, symmetrical and similar among each-other, with test images which contain high degree of object occlusion.

For the experiments, we always train the model with synthetic training images, created as described in Section III-A. However, we test the model with two types of test images: real and synthetic. The synthetic test images are carefully created to resemble the real test images. We achieve this using the rendering procedure described in III-A, but instead of randomizing the object positions, to create a synthetic test image we use the ground truth information of a corresponding real test image from the T-LESS dataset. This way, in the synthetic test images the actual objects are overlaid with the renderings of their 3D models. An example of a real and synthetic test image is shown in 4.

This leads to two test sets with the following difference: in the real test set, we have images of the real objects while in the synthetic test set we have images of the rendered objects based on their 3D models. By comparing the performance of the model on the two test sets, we can see to what degree the transfer from synthetic to real data influences the performance of the model. Using this approach, we conduct experiments with the detector and viewpoint estimator components as separate modules, as well as with the detector-estimator architecture as a whole.

A. *Object detection experiment*

For the detection of the bounding box of an object, we rely on the Faster R-CNN model [7], which has been proposed

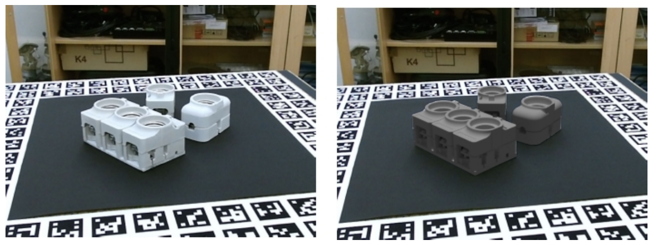
Fig. 4. Example test images. Left: real test image from the T-LESS dataset. Right: Synthetic test image created from the corresponding real image by overlaying rendered 3D models of the objects on the place of the actual objects.

as a category-based object detector and is usually trained on real images. However, it is unclear if the Faster R-CNN model will be able to successfully detect the objects in real images if it is trained with synthetic images of their 3D models. Also, it is unknown whether it is necessary to generate photo-realistic synthetic images or the detector can learn important features of the objects even if the objects are rendered without high realism. To address these questions, we perform an experiment.

1) Experimental setup: Using the procedure outlined in Section III-A, we create two synthetic datasets for object detection, which we refer to as the "high realism" and "low realism" synthetic dataset. In both datasets, all variables like object positions and substitute backgrounds are the same, the only difference is the realism in the rendering of the 3D models. For the "low realism" dataset we render the objects using the Blender Internal [17] rendering engine under a light source that generates a high contrast between different parts of the object, while for the "high realism" dataset we use the Blender Cycles [17] rendering engine under random lighting conditions and random film exposure time. The difference in realism from the two renderers emerges from the concept of how they render an image within a virtual scene, with the Cycles renderer producing more photo-realistic images, as indicated in [22].

For each dataset, we create 5000 synthetic training images and 2016 synthetic test images[3], each synthetic test image corresponding to a real test image from the T-LESS. Sample training images from the two synthetically-generated datasets are presented in Figure 5.

For the object detector component, we use the implementation of the Faster R-CNN model available from the Tensorflow Object Detection API [16]. The model is pre-trained on the COCO dataset [23] of real images. We use the default configuration parameters [24] and we initialize the model with pre-trained weights. For a single training run, starting from the pre-trained state, we fine-tune the model on synthetic training data for 5000 steps. After 5000 steps we measure the performance of the model on the synthetic test data and the real test data. The metric we use is mean Average Precision (mAP), as it has been defined for the PASCAL Visual Object Classes (VOC) Challenge [25] and implemented in the Tensorflow Object Detection API [24].

[3]The T-LESS dataset does not provide images of the environment without any objects in them, which we need as an input in order to use them as substitute backgrounds for the synthetic training data. To overcome this, we split the T-LESS test images into two parts. The first part contains the scenes 1,5,6,7,8,9 and 10, and the second part contains the scenes 2,3,4 and 11. We use the first part for creating substitute environment backgrounds without objects, where the environment background image is generated by covering the objects in the image with a random region of the image that does not contain any objects. The second part contains a total of 2016 images which we use as real test images and as a basis for generating synthetic test images.

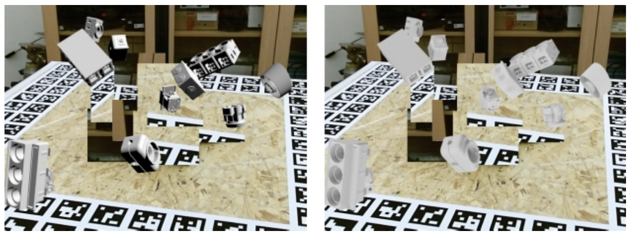
Fig. 5. Synthetic training images for object detection experiment. Left: "low realism" condition; Right: "high realism" condition.

We perform 20 independent experiment runs, 10 times using the "high realism" synthetic dataset, and 10 times using the "low realism" synthetic dataset. Before each run, we randomize the order of the training images.

TABLE I
mAP RESULTS OF THE OBJECT DETECTION EXPERIMENT.

Training data	mAP on real test images	mAP on synthetic test images
"low realism" (10 runs)	0.584, 0.598, 0.585, 0.593, 0.622, 0.594, 0.667, 0.56, 0.607, 0.605	0.605, 0.67, 0.616, 0.636, 0.697, 0.624, 0.692, 0.615, 0.656, 0.654
"high realism" (10 runs)	0.631, 0.634, 0.585, 0.603, 0.632, 0.64, 0.576, 0.666, 0.679, 0.626	0.734, 0.749, 0.723, 0.678, 0.71, 0.733, 0.666, 0.732, 0.755, 0.733
average	0.614 $\pm$ 0.032	0.683 $\pm$ 0.048

2) Results: The mAP results of the detector, given in Table I, indicate that the detector is able to learn from the synthetic training data and correctly detect and classify the bounding boxes of the objects, overcoming the challenges like high object similarity and object occlusion in the test images. Sample results of the detected bounding boxes are shown in Figure 6 (left).

To see if more "realistically-appearing" rendering of the training data changes the detector's performance on the real images, we use the mAP values of each experiment run to perform a two-tailed independent samples t-test under the null hypotheses that there is no change in the detector's performance under the two conditions. The results (t(10)=-1.8912, pvalue=0.0748) indicate that we cannot reject the null hypothesis that assumes equal performance under the two conditions. This result is somewhat surprising because it indicates that the detector can generalize from unrealistically rendered objects as good as from ones that are realistically rendered. This is an important observation, because producing a "high-realism" training image is more resource-consuming than producing a "low-realism" training image, e.g., in our experiments it takes about seven times longer to produce "high-realism" image.

The second important question is the possibility of transferring the detector trained with synthetic images on real images. If we take the detector's performance on synthetic test images as a reference performance when there is no difference in the nature of the training and test data, the question is how close to that performance can the detector come when tested on real images. Under the null hypothesis that there is no difference in the performance between the two types of test images, we perform dependent samples t-test, using the column-vice mAP values from Table I. According to the results of the paired samples t-test for the "low realism" condition (t(10) =3.270133, pvalue=0.00425) and for the "high realism" condition (t(10) = 6.842536, pvalue=0.0000021) we can reject the null hypothesis that the performance of the model when it is trained with synthetic and applied on synthetic images is as good as when it is applied on real images. Comparing the average mAP on real test images to the average mAP on synthetic test images in Table I, we can conclude that the there is a drop of about 6 percent in the performance of the detector model when it is trained with synthetic and applied on real data.

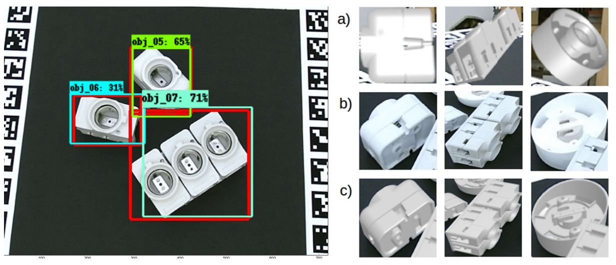

Fig. 6. Left: Results of the object detection experiment on test image from the T-LESS dataset. The predicted bounding boxes that have met the *True Positive* (TP) condition according to the mAP metric are shown together with the ground truth bounding boxes(red color).
Right: Sample images used in the viewpoint estimation experiment: a) Synthetic training images. b) Real test images of the objects within their bounding box. The synthetic test images generated from the corresponding real test images.

B. *Object viewpoint estimation experiment*

In this section, we evaluate the viewpoint estimation model proposed in Section III-B. The experiment evaluates how precisely the model can estimate the object's 3D-pose given its 2D-view in a bounding box, and whether it can be successfully applied on real images after training solely with synthetic images. The experiment is carried out assuming an ideal bounding box detector, i.e., the training and test input to the viewpoint estimator are the ideal bounding boxes of the objects. We address a non-ideal viewpoint estimation in Section IV-C, where we use the predicted bounding boxes of the detector component to test the detector-estimator architecture as a whole.

1) Experimental setup: In this experiment, we use the 3D models of objects 5, 8 and 18 from the T-LESS dataset. These objects have different shape, different level of symmetry and different level of detailed features, providing enough variability to test if viewpoint estimator architecture is robust under different object properties. For each object, we generate a synthetic dataset of approximately one million synthetic training images using their 3D models as described in Section III-A. Examples of the synthetic training images, as well as the synthetic and real test images, are shown in Figure 6 (right).

To train the viewpoint estimators, we first load the weights of the VGG-16 convolutional base [26] pre-trained on the ImageNet [6] dataset and randomly initialize the weights of the newly-attached intermediate fully-connected layer and the output layer. For each epoch, we randomly select about 50 000 synthetic training samples and train the model with batch size of 32 samples[4]. The training procedure has two stages: transfer-learning stage and fine-tuning stage[5]. In the transfer-learning stage, we train the model for 20 epochs by only adjusting the weights of the fully-connected layer while keeping the weights of the VGG-16 convolutional base fixed. For this phase we use rmsprop [27] optimizer with default parameters. After 20 epochs we perform a fine-tuning phase until the model converges, in which besides the fully-connected layers, we also adjust the top 6 convolutional layers of VGG-16. For the fine-tuning phase, we perform Stochastic Gradient Descent (SGD) optimization with a learning rate of 10^{-4} and a momentum of 0.9. At the end of each epoch, we evaluate the model on both the synthetic and the real test images. We perform 5 independent training runs for each object, to account for the effects of randomness in the results.

2) Metric for viewpoint estimation: To measure the performance of the model, we use the guidelines proposed by Xiang et al. [28]. The output prediction of our viewpoint estimator for a given input image is the center of the 10-degree azimuth-elevation and in-plane rotation bins with the highest probability. We consider a predicted output to be correct if the difference between the ground truth value and the predicted value is within $\pm x$ degrees for each of the three angles. The accuracy is then the portion of correct predictions and we refer to this metric as "accuracy with x degrees tolerance". Due to the discretization, the maximum precision we can measure is accuracy with a tolerance of 5 degrees[6]. We also show the accuracy for each of the azimuth, elevation and in-plane rotation angles independently, where the prediction is correct if the predicted value of the angle is within 5 degrees of difference from the true value.

3) Results: In Table II the accuracy results for 5 runs per object are presented. The results indicate that the model can generalize and learn features that are relevant to the viewpoint estimation task independently of the object's properties, and that the features learned from synthetic images are relevant and transferable to real-world data. It is also noticeable that the model still performs better on the synthetic test images compared to the real test images. While one of the

[4]The model and training hyper-parameters, like the number of units of the hidden layer, batch size, the covariance of the normal distribution for generating the bin probabilities, or the number of VGG-16 convolutional layers to fine-tune have been determined by additional hyper-parameter optimization analysis.

[5]We followed https://github.com/ DeepLearningSandbox/DeepLearningSandbox/blob/master/transfer_learning/finetune.py) to implement two-stage training with Keras [26], accessed on: 28.08.2017

[6]A random-guess predictor with this metric under a tolerance angle of 5 degrees would have a 1 in 18*36*36 or 0.004% chance to be correct.

TABLE II
ACCURACY RESULTS OF THE VIEWPOINT ESTIMATION EXPERIMENT.

object	metric	test images (synthetic)	test images (real)
obj_05	accuracy (5° tolerance)	0.376 ± 0.009	0.196 ± 0.015
	accuracy (15° tolerance)	0.881 ± 0.005	0.741 ± 0.014
	accuracy (25° tolerance)	0.909 ± 0.005	0.829 ± 0.009
	azimuth accuracy	0.632 ± 0.006	0.4 ± 0.018
	elevation accuracy	0.69 ± 0.011	0.574 ± 0.019
	in-plane rotation accuracy	0.746 ± 0.007	0.59 ± 0.01
obj_08	accuracy (5° tolerance)	0.276 ± 0.008	0.135 ± 0.007
	accuracy (15° tolerance)	0.713 ± 0.008	0.509 ± 0.011
	accuracy (25° tolerance)	0.78 ± 0.005	0.595 ± 0.009
	azimuth accuracy	0.523 ± 0.015	0.352 ± 0.014
	elevation accuracy	0.546 ± 0.011	0.409 ± 0.021
	in-plane rotation accuracy	0.576 ± 0.007	0.431 ± 0.012
obj_18	accuracy (5° tolerance)	0.261 ± 0.018	0.09 ± 0.011
	accuracy (15° tolerance)	0.76 ± 0.014	0.388 ± 0.034
	accuracy (25° tolerance)	0.863 ± 0.008	0.553 ± 0.03
	azimuth accuracy	0.46 ± 0.018	0.229 ± 0.009
	elevation accuracy	0.666 ± 0.014	0.495 ± 0.021
	in-plane rotation accuracy	0.566 ± 0.021	0.443 ± 0.043

obvious reasons for this is the different nature of the training and test data, we assume that the result is also influenced by the properties of the 3D models: the models we use from the T-LESS dataset are manually reconstructed and do not contain all of the fine details of the real objects. We assume that the performance would increase if a more detailed 3D model of the object is available.

An important property of our viewpoint estimator is the ability to handle objects that look similar from different angles. An example is presented in Figure 7, where the probability distribution of the output layer has encoded two distinct viewpoints of the object that look identical. This shows that the model can handle symmetrical objects and that information for the object's appearance and symmetries can be inferred from the activations of the output layer.

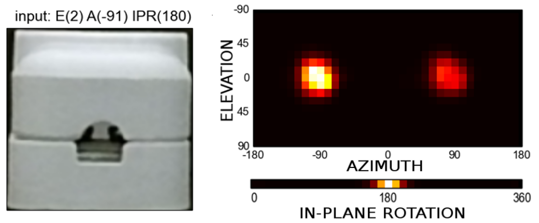

Fig. 7. Prediction of the model for a symmetrical object. Left: Image of object 05 from the T-LESS dataset. Right: The probability distribution at the output layer for the given input: lighter color indicates higher probability. The viewpoint (elevation-azimuth) distribution has two maxima: one corresponds to the actual ground truth viewpoint, and one corresponds to the viewpoint under which the object looks similar.

C. Combined object detection and viewpoint estimation

There are two additional sources of error in the accuracy when predicted bounding boxes are used for the viewpoint estimation instead of the ideal ones. First, the object might not be detected at all from the bounding-box detector. Second, the predicted bounding box would rarely exactly match the borders of the object in an image as opposed to an ideal bounding box, complicating the prediction of the correct viewpoint.

To evaluate the pose estimation model as a whole, we use the *True Positive* (TP) bounding boxes predicted by the detector trained with "low-realism" synthetic data from the experiment outlined in Section IV-A as an input to the trained object-specific viewpoint estimators from Section IV-B.

The comparative results presented in Figure 8 show that if the bounding boxes predicted from the detector are used instead of the ideal ones, the viewpoint estimator can still correctly recover the 3D pose without a significant drop in accuracy. Furthermore, from the gap in accuracy between real and synthetic test images under different tolerance angles, we can conclude that the benefit of effortless creation of synthetic training data comes at a cost of decreased accuracy and is dependent on the quality of the 3D model of the objects. Still, the accuracy achieved in our experiments on the challenging test images from T-LESS, which is about 70 percent for tolerance of 25°, indicates that the proposed architecture and training procedure can be successfully applied in real-world tasks. Sample results of the estimated 3D pose by combining the predicted bounding box and the estimated viewpoint are shown in Figure 9.

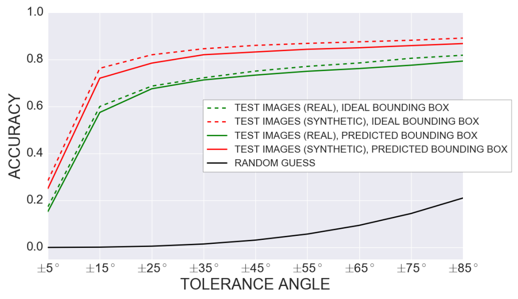

Fig. 8. Comparison of the accuracy in pose estimation of synthetic and real test images, for predicted and ideal bounding boxes, under different tolerance angles. The accuracy is calculated as the average of three objects we use in the experiment.

D. Application of the pose estimation model in robotic tasks

To demonstrate that the proposed pose estimation approach can be extended to real-world tasks, we train the detector and view-point estimator using 3D models of toy-objects and a three-fingered robotic hand[7]. The results show that the model can successfully estimate the 3D pose of the objects without strict control of the environment in which it is applied and without the need of images from the real objects. The 3D models used for training and sample results are shown in Figure 1, while further results are available in the accompanying video attachment.

V. CONCLUSION

We present and evaluate a novel approach for instance-based pose estimation using CNNs, suitable for determining the 3D pose of objects in robotic manipulation tasks in loosely controlled environments. We use automatically generated synthetic training data based on 3D models to satisfy the high training data requirement of CNN-approaches. Results in real-world images indicate that photo-realistic renderings are not necessary for good performance. We hypothesize that a key to this result is the use of a deep

[7]The hand is the RH4D from http://www.seedrobotics.com/

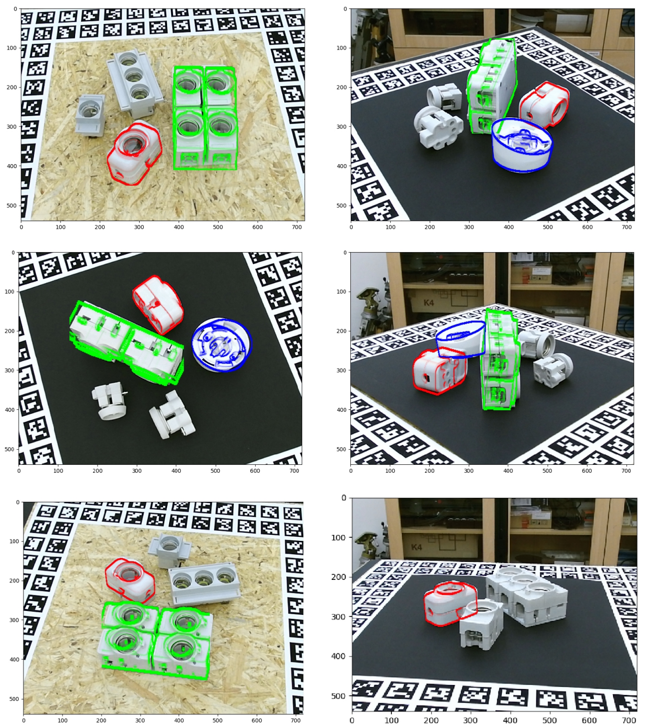
Fig. 9. Test images from the T-LESS dataset showing the estimated object pose of the three objects, by combining the bounding box detection and the estimation of the viewpoint of the object within the bounding box. The 3D poses are visualized by drawing the contours of the 3D models.

convolutional architecture that is pre-trained on real images but then fine-tuned only in the top layers.

In future work, we will investigate which factors are crucial to further improve the generalization ability of the model such that the performance on real images comes closer to the one achieved on synthetic data. Since the activations of the output layer of the model can be used to infer shape-relevant information about the object, we also plan to apply the model as a low-level visual processor, whose output is used to aid visually-guided high-level task like shape-specific robotic grasping.

References

[1] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, no. [8, pp. 1150–1157, 1999.

[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[5] M. Kerzel, E. Strahl, S. Magg, N. Navarro-Guerrero, S. Heinrich, and S. Wermter, "Nico - neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2017, pp. 113–120.

[6] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Nips*, pp. 1–10, 2015.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Cvpr 2016*, pp. 779–788, 2016.

[9] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. 1, pp. 3109–3118, 2015.

[10] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 11-18-Dece, pp. 2686–2694, 2016.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 23–30.

[12] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. Zakharov, H. Kosch, and J. Ernst, "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition," *arXiv preprint arXiv:1702.08558*, 2017.

[13] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[14] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *International Conference on Computer Vision*, vol. 1, no. 4, 2017, p. 5.

[15] X. S. Gao, X. R. Hou, J. Tang, and H. F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[16] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE CVPR*, 2017.

[17] "Blender - a 3D modelling and rendering package," Blender Institute, Amsterdam. [Online]. Available: http://www.blender.org

[18] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D Object Recognition and Pose Estimation based on Pre-trained Convolutional Neural Network Features," *IEEE International Conference on Robotics and Automation (ICRA'15)*, no. May, pp. 1329–1335, 2015.

[19] P. Fischer, A. Dosovitskiy, and T. Brox, "Image orientation estimation with convolutional networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9358, pp. 368–378, 2015.

[20] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint*, pp. 1–10, 2014.

[21] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 880–888.

[22] E. Valenza, *Blender 2.6 Cycles: Materials and Textures Cookbook*. Packt Publishing Ltd, 2013.

[23] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[24] "Tensorflow Object Detection API," 2017. [Online]. Available: https://github.com/tensorflow/models/tree/master/object_detection

[25] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[26] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[27] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." pp. 26–31, 2012.

[28] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond PASCAL: A benchmark for 3D object detection in the wild," *2014 IEEE Winter Conference on Applications of Computer Vision, WACV 2014*, pp. 75–82, 2014.