RGB-D Object Recognition and Pose Estimation based on Pre-trained Convolutional Neural Network Features

Max Schwarz, Hannes Schulz, and Sven Behnke

***Abstract*— Object recognition and pose estimation from RGB-D images are important tasks for manipulation robots which can be learned from examples. Creating and annotating datasets for learning is expensive, however. We address this problem with transfer learning from deep convolutional neural networks (CNN) that are pre-trained for image categorization and provide a rich, semantically meaningful feature set. We incorporate depth information, which the CNN was not trained with, by rendering objects from a canonical perspective and colorizing the depth channel according to distance from the object center. We evaluate our approach on the Washington RGB-D Objects dataset, where we find that the generated feature set naturally separates classes and instances well and retains pose manifolds. We outperform state-of-the-art on a number of subtasks and show that our approach can yield superior results when only little training data is available.**

I. INTRODUCTION

The recent success of deep convolutional neural networks (CNN) in computer vision can largely be attributed to massive amounts of data and immense processing speed for training these non-linear models. However, the amount of data available varies considerably depending on the task. Especially robotics applications typically rely on very little data, since generating and annotating data is highly specific to the robot and the task (e.g. grasping) and thus prohibitively expensive. This paper addresses the problem of small datasets in robotic vision by reusing features learned by a CNN on a large-scale task and applying them to different tasks on a comparably small household object dataset. Works in other domains [1]–[3] demonstrated that this transfer learning is a promising alternative to feature design.

Figure 1 gives an overview of our approach. To employ a CNN, the image data needs to be carefully prepared. Our algorithm segments objects, removes confounding background information and adjusts them to the input distribution of the CNN. Features computed by the CNN are then fed to a support vector machine (SVM) to determine object class, instance, and pose. While already on-par with other state-of-the-art methods, this approach does not make use of depth information.

Depth sensors are prevalent in todays robotics, but large datasets for CNN training are not available. Here, we propose to transform depth data into a representation which is easily interpretable by a CNN trained on color images. After detecting the ground plane and segmenting the object, we render it from a canonical pose and color it according to distance from the object center. Combined with the image color features, this method outperforms other recent approaches on the Washington RGB-D Objects dataset on a number of subtasks. This dataset requires categorization of household objects, recognizing category instances, and estimating their pose.

All authors are with Rheinische Friedrich-Wilhelms-Universität Bonn, Computer Science Institute VI, Autonomous Intelligent Systems, Friedrich-Ebert-Allee 144, 53113 Bonn schwarzm@cs.uni-bonn.de, schulzh@ais.uni-bonn.de, behnke@cs.uni-bonn.de

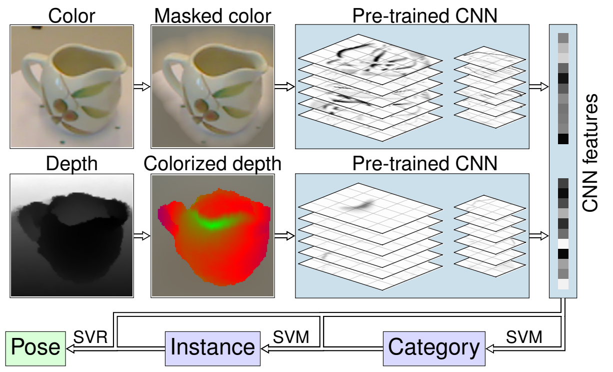


Fig. 1: Overview of our approach. Images are pre-processed by extracting foreground, reprojecting to canonical pose, and colorizing depth. The two images are processed independently by a convolutional neural network. CNN features are then used to successively determine category, instance, and pose.

In short, our contributions are as follows:

1) We introduce a novel pre-processing pipeline for RGB-D images facilitating CNN use for object categorization, instance recognition, and pose regression.
2) We analyze features produced by our pipeline and a pre-trained CNN. We show that they naturally separate common household object categories, as well as their instances, and produce low-dimensional pose manifolds.
3) We demonstrate that with discriminative training, our features improve state-of-the-art on the Washington RGB-D Objects classification dataset.
4) We show that in contrast to previous work, only few instances are required to attain good results.
5) Finally, we demonstrate that with our method even category level pose estimation is possible without sacrificing accuracy.

After discussing related work, we describe our feature extraction pipeline and supervised learning setup in Sections III and IV, respectively. We analyze the features and their performance in Section V.

II. Related Work

Deep convolutional neural networks (CNN) [4]–[6] became the dominant method in the ImageNet large scale image classification challenge [7] since the seminal work of Krizhevsky *et al.* [8]. Their success can mainly be attributed to large amounts of available data and fast GPU implementations, enabling the use of large non-linear models. To solve the task of distinguishing 1000, sometimes very similar object categories, these networks compute new representations of the image with repeated convolutions, spatial max-pooling [9], and non-linearities [8]. The higher-level representations are of special interest, as they provide a generic description of the image in an increasingly semantic feature space [10]. This observation is supported by the impressive performance of CNN features learned purely on classification tasks applied to novel tasks in computer vision such as object detection [1], [2], subcategorization, domain adaptation, scene recognition [3], attribute detection, and instance retrieval [2]. In many cases, the results are on par with or surpass the state of the art on the respective datasets.

While Girshick *et al.* [1] report improvements when fine-tuning the CNN features on the new task, this approach is prone to overfitting in the case of very few training instances. We instead follow Donahue *et al.* [3] and only re-interpret features computed by the CNN. In contrast to the investigations of Donahue *et al.* [3] and Razavian *et al.* [2], we focus on a dataset in a robotic setting with few labeled instances.

We use pre-trained CNN in conjunction with preprocessed depth images, which is not addressed by the works discussed so far. Very recently, Gupta *et al.* [11] proposed a similar technique, where "color" channels are given by horizontal disparity, height above ground, and angle with vertical. In contrast to their method, we propose an object-centered colorization scheme, which is tailored to the classification and pose estimation task.

Previous work on the investigated RGB-D Objects dataset begins with the dataset publication by Lai *et al.* [12], who use a combination of several hand-crafted features (SIFT, Texton, color histogram, spin images, 3D bounding boxes) and compares the performance of several classifiers (linear SVM, Gaussian kernel SVM, random forests). These baseline results have been improved significantly in later publications.

Lai *et al.* [13] propose a very efficient hierarchical classification method, which optimizes classification and pose estimation jointly on all hierarchy levels. The method uses stochastical gradient descent (SGD) for training and is able to warm-start training when adding new objects to the hierarchy. While the training method is very interesting and could possibly be applied to this work, the reported results stay significantly behind the state of art.

Finally, Bo *et al.* [14] show a very significant improvement in classification accuracy and reduction in pose estimation error. Their method learns hierarchical feature representations from RGB-D Objects data without supervision using hierarchical matching pursuit (HMP). This work shows the promise of feature learning from raw data and is the current state-of-the-art approach on the RGB-D Objects dataset. However, the number of training examples must be suitably large to allow for robust feature learning—in contrast to our work, which uses pre-learned features and is thus able to learn from few examples. Of course, the feature learning process is not needed in our case which leads to a significant runtime advantage for our method. Finally, our method generates less feature dimensions (10,192 vs. 188,300) and thus is also faster in the classifier training and recall steps.

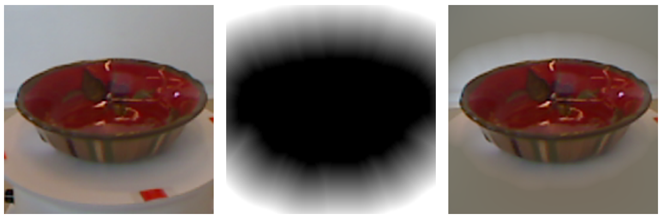

(a) Region of interest (b) Mask distance (c) Faded output

Fig. 2: Overview of the RGB preprocessing pipeline, with ROI from object segmentation, distance transform from the object mask and faded output used for CNN feature extraction.

III. CNN Feature Extraction Pipeline

In order to use a pre-trained CNN for feature extraction, it is necessary to preprocess the input data into the format that matches the training set of the neural network. *CaffeNet* [15], which we use here, expects square 227×227 RGB images depicting one dominant object as in the ImageNet Large Scale Visual Recognition Challenge [7].

A. RGB Image Preprocessing

Since the investigated CNN was trained on RGB images, not much pre-processing is needed to extract robust features from color images. Example images from the preprocessing pipeline can be seen in Fig. 2.

We first crop the image to a square region of interest (see Fig. 2a). In a live situation, this region of interest is simply the bounding box of all object points determined by the tabletop segmentation (Section III-B). During evaluation on the Washington RGB-D Objects dataset (Section V), we use the provided object segmentation mask to determine the bounding box.

The extracted image region is then scaled to fit the CNN input size, in our case 227×227 pixels. To reduce the CNN's response to the background, we apply a fading operation to the image (Fig. 2c). Each pixel is interpolated between its RGB color $\mathbf{c_0} = (r_0, g_0, b_0)$ and the corresponding ILSVRC 2011 mean image pixel $\mathbf{c_m} = (r_m, g_m, b_m)$ based on its pixel distance r to the nearest object pixel:

$$\mathbf{c} := \alpha \cdot \mathbf{c_0} + (1-\alpha) \cdot \mathbf{c_m} \,, \tag{1}$$

where

$$\alpha := \begin{cases} 1 & \text{if } r = 0, \\ 0 & \text{if } r > R, \\ (R-r)^{\beta} & \text{else.} \end{cases} \tag{2}$$

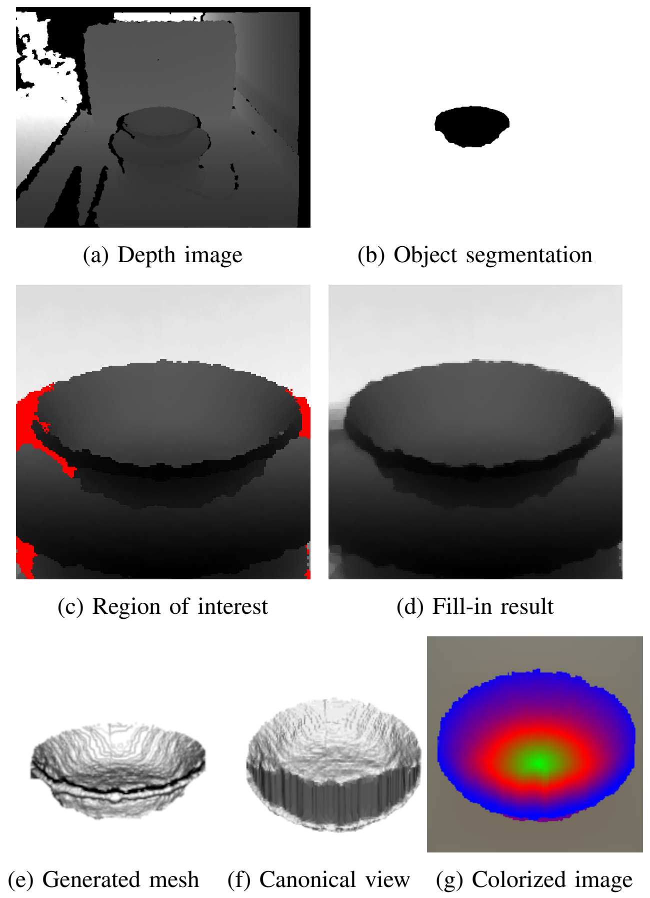


Fig. 3: Overview of the depth preprocessing pipeline. (a) input depth map, (b) extracted segmentation mask after tabletop segmentation [16], (c) region of interest containing only the object with unavailable depth pixels shown in red, (d) missing depth values filled in, (e) mesh extracted from point cloud, (f) reprojection of mesh to a canonical camera pose, (g) final image used for CNN feature extraction.

The fade radius $R = 30$ was manually tuned to exclude as much background as possible while keeping objects with non-optimal segmentation intact. The exponent $\beta = 0.75$ was later roughly tuned for best cross-validation score in the category level classification.

B. *Depth Image Preprocessing*

Feeding depth images to a CNN poses a harder problem. The investigated CNN was trained on RGB images and is thus not expected to perform well on raw depth images. To address this, we render image-like views of the object from the depth data using a five-step pipeline, which will be detailed below. Figure 3 illustrates all steps for an example.

In the first step, we perform a basic segmentation to extract the horizontal surface the object is resting on. Following Holz *et al.* [16], we estimate surface normals from the depth image, discard points from non-horizontal surfaces and register a planar model using Random Sample Consensus (RANSAC). The main objective here is to construct a local reference frame which is fixed in all dimensions, except for rotation around the vertical axis. To this end, we find points on the plane and extract object clusters with Euclidean Clustering (see Fig. 3b).

In a second step, we fill-in holes in the depth map. We employ a common scheme based on the work of Levin *et al.* [17], who investigated the colorization of grayscale images from few given color pixels. The colorization is guided by the grayscale image in such a way that regions with similar intensity are colored the same. We fill-in depth values using the same technique guided by a grayscale version of the RGB image. This has the advantage of using the RGB information for disambiguation between different depth layers, whereas the standard approach of depth image median filtering cannot include color information.

In detail, the objective of the fill-in step is to minimize the squared difference between the depth value $D(\mathbf{p})$ with $\mathbf{p} = (u, v)$ and the weighted average of the depth at neighboring pixels,

$$J(D) = \sum_{\mathbf{p}} \left(D(\mathbf{p}) - \sum_{\mathbf{s} \in N(\mathbf{p})} w_{\mathbf{ps}} D(\mathbf{s}) \right)^2 , \tag{3}$$

with

$$w_{\mathbf{ps}} = \exp\left[-(G(\mathbf{p}) - G(\mathbf{s}))^2 / 2\sigma_{\mathbf{p}}^2 \right] , \tag{4}$$

where $G(\mathbf{p})$ is the grayscale intensity of $\mathbf{p}$ and $\sigma_{\mathbf{p}}$ is the variance of intensity in a window around $\mathbf{p}$ defined by $N(\cdot)$. Minimizing $J(D)$ leads to a sparse linear equation system, which we solve with the BiCGSTAB solver of the Eigen linear algebra package. A result of the fill-in operation can be seen in Fig. 3d.

After the fill-in operation, we filter the depth map using a shadow filter, where points whose normals are perpendicular to the view ray get discarded. This operation is only executed on the boundaries of the object to keep the object depth map continuous.

To increase invariance of the generated images against camera pitch angle changes, we normalize the viewing angle by an optional reprojection step. The goal is to create a view of the object from a canonical camera pitch angle. To enable reprojection, we first create a mesh using straight-forward triangulation from the filled depth map (Fig. 3e). We then render the mesh from the canonical perspective to create a new depth image. Our naïve meshing approach creates a linear interpolation for previously hidden surfaces of the object (see Fig. 3f). We believe this to be a plausible guess of the unknown geometry without making further assumptions about the object, such as symmetries.

The final preprocessing step is to calculate a color C for each depth image pixel $\mathbf{p} = (u, v)$ with the 3D reprojection $\mathbf{p}^* = (x, y, z)$. We first estimate a 3D object center $\mathbf{q}$ from the bounding box of the object point cloud. The points are then colored according to their distance r from a line g through $\mathbf{q}$ in the direction of the plane normal $\mathbf{n}$ from tabletop segmentation (Fig. 4):

$$C(\mathbf{p}) = P(\mathrm{dist}_g(\mathbf{p}^*)). \tag{5}$$

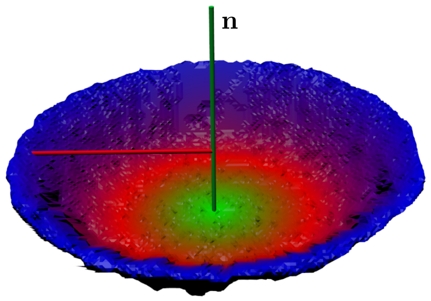


Fig. 4: Coordinate system used for colorization. The detected plane normal $\mathbf{n}$ through the object center is depicted as a green bar, while an exemplary distance to a colorized point is shown as a red bar.

We chose a fixed RGB interpolation from green over red and blue to yellow as palette function P. Since the coloring is not normalized, this allows the network to discriminate between scaled versions of the same shape. If scale-invariant shape recognition is desired, the coloring can easily be normalized.

Note that depth likely carries less information than color and could be processed at a coarser resolution. We keep resolution constant, however, since the input size of the learned CNN cannot be changed.

C. Image Feature Extraction

We investigated the winning CNN from ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2011 by Krizhevsky *et al.* [8]. The open-source *Caffe* framework [15] provides a pre-trained version of this network.

We extract features from the previous-to-last and the last fully connected layer in the network (named *fc7* and *fc8* in *Caffe*). This gives us $4\,096 + 1\,000 = 5\,096$ features per RGB and depth image each, resulting in $10\,192$ features per RGB-D frame.

Reprojection and coloring are only used for instance-level classification and pose regression, since object categorization cannot benefit from a canonical perspective given the evaluation regime of the Washington RGB-D dataset.

Figure 5 shows responses of the first convolutional layer to RGB and depth stimuli. The same filters show different behavior in RGB and depth channels. As intended by our preprocessing, the activation images exhibit little activity in faded-out background regions.

IV. LEARNING METHOD

A. Object Classification

For classification, we use linear Support Vector Machines (SVMs). We follow a hierarchical approach as in Lai *et al.* [13]: In a first level, a linear multiclass SVM predicts the object category. The next level contains SVMs predicting the instance in each particular category.

B. Object Pose Estimation

The RGB-D object dataset makes the assumption that object orientation is defined by a single angle α around the normal vector of the planar surface. This angle is consistently annotated for instances of each object category. However, annotations are not guaranteed to be consistent across categories.

Instead of regressing α directly, we construct a hierarchy for pose estimation to avoid the discontinuity at $\alpha = 0° = 360°$, which is hard for a regressor to match. We first predict a rough angle interval for α using a linear SVM. In our experiments, four angle intervals of $90°$ gave best results. For each interval, we then train one RBF-kernel support vector regressor to predict α. During training, we include samples from the neighboring angle intervals to increase robustness against misclassifications on the interval level.

This two-step regressor is trained for each instance. We further train the regressor for each category to provide pose estimation without instance identification, which is supported by the dataset but is not reported by other works, albeit being required in any real-world household robotics application.

V. EVALUATION

A. Evaluation Protocol

We evaluate our approach on the Washington RGB-D Objects dataset [12]. It contains 300 objects organized in 51 categories. For each object, there are three turntable sequences captured from different camera elevation angles $(30°, 45°, 60°)$. The sequences were captured with an ASUS Xtion Pro Live camera with 640×480 resolution in both RGB and depth channels. The dataset also contains approximate ground truth labels for the turntable rotation angle.

Furthermore, the dataset provides an object segmentation based on depth and color. We use this segmentation mask in our pre-processing pipeline. However, since our RGB pre-processing needs background pixels for smooth background fading (Section III-A), we could not use the provided pre-masked evaluation dataset but instead had to use the corresponding frames from the full dataset. Since our method fades out most of the background, only features close to the object remain. This includes the turntable surface, which is not interesting for classification or pose regression and the turntable markers, which do not simplify the regression problem since the objects are placed randomly on the turntable in each view pose. Thus, we believe that our results are still comparable to other results on the same dataset.

For evaluation, we follow the protocol established by Lai *et al.* [12] and Bo *et al.* [14]. We use every fifth frame for training and evaluation. For category recognition, we report the cross-validation accuracy for ten predefined folds over the objects, i.e. in each fold the test instances are completely unknown to the system.

For instance recognition and pose estimation, we employ the *Leave-Sequence-Out* scheme of Bo *et al.* [14], where the system is trained on the $30°$ and $60°$ sequences, while evaluation is on the $45°$ sequence of every instance.

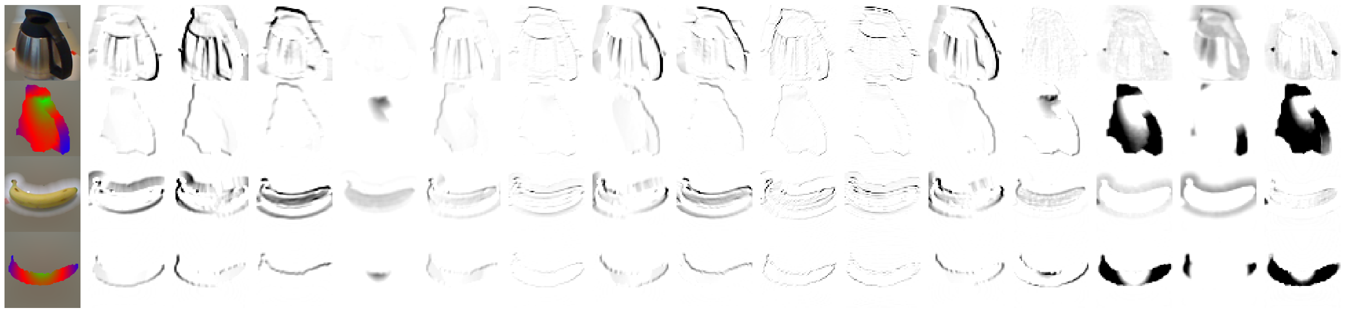


Fig. 5: CNN activations for sample RGB-D frames. The first column shows the CNN input image (color and depth of a pitcher and a banana), all other columns show corresponding selected responses from the first convolutional layer. Note that each column is the result of the same filter applied to color and pre-processed depth.

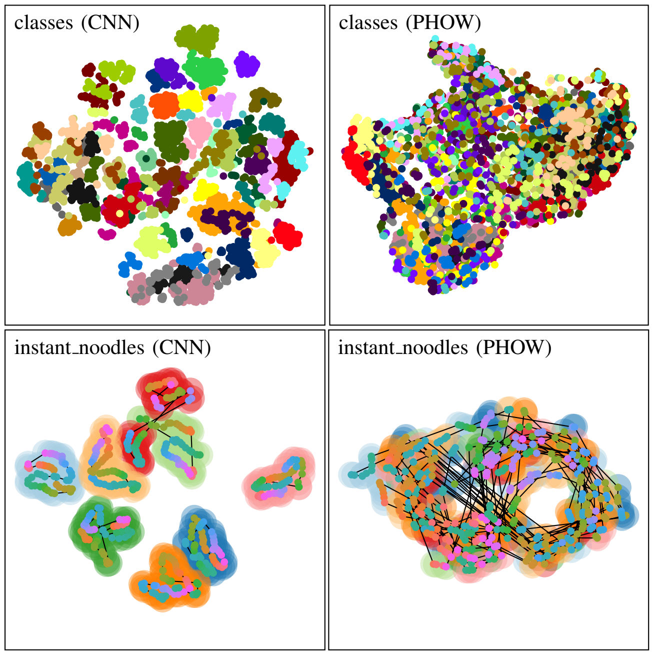


Fig. 6: Visualization of our CNN-based features. Top row shows t-SNE embedding of 1/10 of the Washington RGB-D Objects dataset using CNN and PHOW features, colored by class. CNN separates classes better than PHOW. Bottom row shows a separate t-SNE embedding of the *instant_noodles* class (45° sequence), colored and connected by pose. The CNN separates instances and creates pose manifolds.

B. Results

In addition to the work of Bo *et al.* [14], we compare our proposed method to a baseline of dense SIFT features (PHOW, [18]), which are extracted at multiple scales, quantized using k-means and histogrammed in a 2×2 and a 4×4 grid over the image. We used vlfeat[1] with standard settings, which are optimized for the Caltech 101 dataset. We then apply SVM training for classification and pose estimation as described in Section IV.

Without any supervised learning, we can embed the features produced by the CNN and PHOW in $\mathbb{R}^2$ using a t-SNE embedding [19]. The result is shown in Fig. 6. While the upper row shows that CNN object classes are well-separated in the input space, the lower row demonstrates that object instances of a single class also become well-separated. Similar poses of the same object remain close in the feature-space, expressing a low-dimensional manifold. These are highly desirable properties for an unsupervised feature mapping which facilitate learning with very few instances. In contrast, PHOW features only exhibit these properties to a very limited extent: Classes and instances are less well-separated, although pose similarities are largely retained.

TABLE I: Comparison of category and instance level classification accuracies on the Washington RGB-D Objects dataset.

	Category Accuracy (%)		Instance Accuracy (%)	
Method	RGB	RGB-D	RGB	RGB-D
Lai *et al.* [12]	74.3 ± 3.3	81.9 ± 2.8	59.3	73.9
Bo *et al.* [14]	82.4 ± 3.1	87.5 ± 2.9	**92.1**	92.8
PHOW[18]	80.2 ± 1.8	—	62.8	—
Ours	$\mathbf{83.1 \pm 2.0}$	$\mathbf{89.4 \pm 1.3}$	92.0	**94.1**

Table I summarizes our recognition results and compares them with other works. We improve on the state-of-the-art in category and instance recognition accuracy for RGB and RGB-D data. The exception is RGB-based instance recognition, where the HMP approach by Bo *et al.* [14] wins by 0.1%.

Analyzing the confusion matrix (Fig. 7), the category level classification exhibits few systematic errors. Some object categories prove to be very difficult, since they contain instances with widely varying shape but only few examples (e.g. *mushroom*), or instances which are very similar in color and shape to instances of other classes (e.g. *pitcher* and *coffe_mug*). Telling apart the *peaches* from similarly rounded but brightly colored *sponges* would likely profit from more examples and detailed texture analysis.

Classification performance degrades gracefully when the dataset size is reduced, which is shown in Fig. 8. We reduce the dataset for category and instance recognition by uniform stratified sampling on category and instance level, respectively. With only 30% of the training set available, category classification accuracy decreases by 0.65 percentage

[1]`http://www.vlfeat.org`

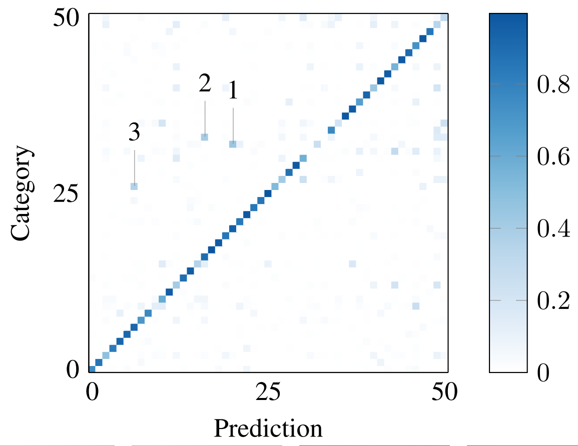


Fig. 7: Top: Confusion matrix for category recognition, normalized by number of samples for each ground truth label. Selected outliers: 1) *pitcher* recognized as *coffee_mug*, 2) *peach* as *sponge*, 3) *keyboard* as *food_bag*. Bottom: Sample images for *pitcher*, *coffe_mug*, *peach*, and *sponge*.

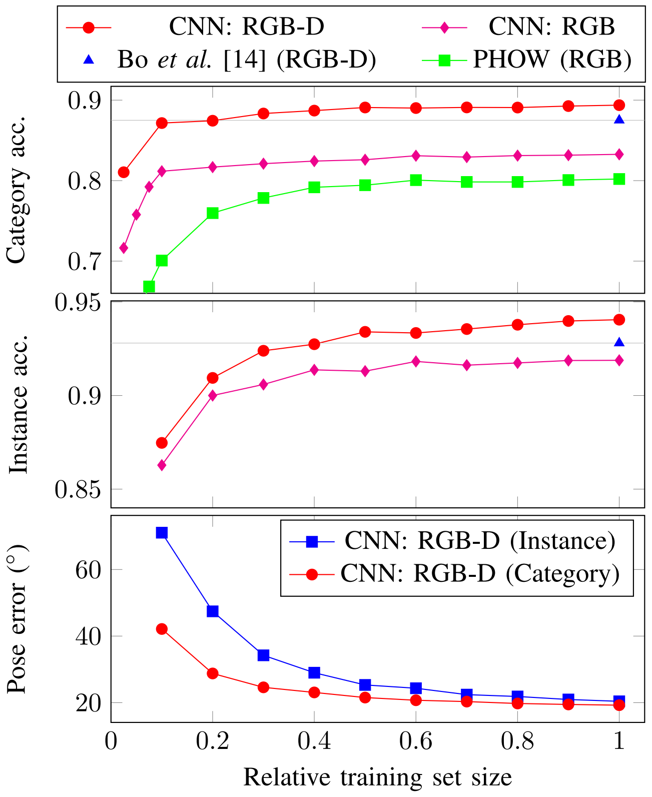


Fig. 8: Learning curves for classification accuracy (top and center) and median pose estimation error (bottom). We report cross validation accuracy for category recognition and accuracy on the 45° sequence for instance recognition.

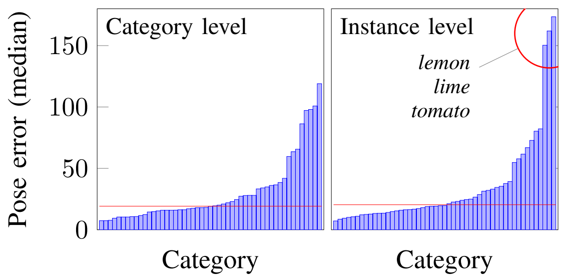


Fig. 9: Distribution of median pose error over categories. Left plot shows median pose error over categories, right plot over instances. Median over all categories is shown in red. Some objects of type *lemon*, *lime*, and *tomato* exhibit high rotation symmetry and do not support pose estimation.

TABLE III: Runtimes of various algorithm steps per input frame in seconds. We measured runtime on an Intel Core i7-4800MQ @ 2.7 GHz and a standard mobile graphics card (NVidia GeForce GT 730M) for CUDA computations. Timings include all preprocessing steps.

Step	Our work	Bo *et al.* [14]
Feature extraction (RGB)	0.013	0.294
Feature extraction (depth)	0.173	0.859
Total	**0.186**	1.153

points only (PHOW: 2.2%), while instance classification decreases by roughly 2% (PHOW: 25.2% from 62.6%, not shown). This supports our observation that the CNN feature space already separates the categories of the RGB-D objects in a semantically meaningful way.

We also performed an categorization experiment with the *Leave-Sequence-Out* evaluation protocol. When training on the two camera pitch angles of 30° and 60° and testing the same objects for the pitch angle of 45°, our method achieves near perfect category recognition accuracy (99.6%).

We also improve the state-of-the-art in pose estimation by a small margin. Table II reports the pose estimation error of the instance-level estimation and the category-level estimation. Notably, our average pose error is significantly lower than the pose error of the other methods. We were not able to produce reasonable accuracies for pose based on the PHOW features, since the large instance classification error strongly affects all pose estimation metrics.

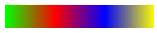
Surprisingly, our category-level pose regression achieves even lower median pose error, surpassing the state-of-the-art result of Bo *et al.* [14]. The category-level estimation is less precise only in the MedPose(I) and AvePose(I) categories, where its broader knowledge is not as useful as precise fitting to the specific instance. Figure 9 shows the distribution of pose errors over categories. We note that the dataset contains objects in at least three categories which exhibit rotation symmetries and do not support estimating pose. This effect is mitigated by category level pose estimation, which shows that pose estimation can greatly benefit from the generalization across instances provided by category-level training.

TABLE II: Median and average pose estimation error on Washington RGB-D Objects dataset. Wrong classifications are penalized with 180° error. (C) and (I) describe subsets with correct category/instance classification, respectively.

Work	Angular Error (°)					
	MedPose	MedPose(C)	MedPose(I)	AvePose	AvePose(C)	AvePose(I)
Lai *et al.* [13]	62.6	51.5	30.2	83.7	77.7	57.1
Bo *et al.* [14]	20.0	**18.7**	**18.0**	53.6	47.5	44.8
Ours – instance level pose regression	20.4	20.4	18.7	51.0	50.4	**42.8**
Ours – category level pose regression	**19.2**	19.1	18.9	**45.0**	**44.5**	43.7

The MedPose error is the median pose error, with 180° penalty if the class or instance of the object was not recognized. MedPose(C) is the median pose error of only the cases where the class was correctly identified, again with 180° penalty if the instance is predicted wrongly. Finally, MedPose(I) only counts the samples where class and instance were identified correctly. AvePose, AvePose(C) and AvePose(I) describe the average pose error in each case respectively.

TABLE IV: Color palettes for depth colorization with corresponding instance recognition accuracy.

Palette		Accuracy (%)	
		Depth Only	RGB-D
Gray		41.8	93.1
Green		38.8	93.3
Green-red-blue-yellow		**45.5**	**94.1**

The color palette choice for our depth colorization is a crucial parameter. We compare the four-color palette introduced in Section III to two simpler colorization schemes (black and green with brightness gradients) shown in Table IV and compared them by instance recognition accuracy. Especially when considering purely depth-based prediction, the four-color palette wins by a large margin. We conclude that more colors result in more discriminative depth features.

Since computing power is usually very constrained in robotic applications, we benchmarked runtime for feature extraction and prediction on a lightweight mobile computer with an Intel Core i7-4800MQ CPU @ 2.7 GHz and a common mobile graphics card (NVidia GeForce GT 730M) for CUDA computations. As can be seen in Table III, the runtime of our approach is dominated by the depth preprocessing pipeline, which is not yet optimized for speed. Still, our runtimes are low enough to allow frame rates of up to 5 Hz in a future real-time application.

VI. CONCLUSION

We presented an approach which allows object categorization, instance recognition and pose estimation of objects on planar surfaces. Instead of handcrafting or learning features, we relied on a convolutional neural network (CNN) which was trained on a large image categorization dataset. We made use of depth features by rendering objects from canonical views and proposed a CNN-compatible coloring scheme which codes metric distance from the object center. We evaluated our approach on the challenging Washington RGB-D Objects dataset and find that in feature space, categories and instances are well separated. Supervised learning on the CNN features improves state-of-the-art in classification as well as average pose accuracy. Our performance degrades gracefully when the dataset size is reduced.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv: `1311.2524`.

[2] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CVPR DeepVision Workshop*, 2014.

[3] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: a deep convolutional activation feature for generic visual recognition," in *Proceedings of International Conference on Machine Learning (ICML)*, 2014, pp. 647–655.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[6] S. Behnke, *Hierarchical neural networks for image interpretation*, ser. Lecture Notes in Computer Science (LNCS). Springer, 2003, vol. 2766.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.* (2014). ImageNet large scale visual recognition challenge. arXiv: `1409.0575`.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[9] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks (ICANN)*, Springer, 2010, pp. 92–101.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 818–833.

[11] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *Europ. Conf. on Computer Vision (ECCV)*, 2014, pp. 345–360.

[12] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1817–1824.

[13] K. Lai, L. Bo, X. Ren, and D. Fox, "A scalable tree-based approach for joint object and pose recognition.," in *Proceedings of Conference on Artificial Intelligence (AAAI)*, 2011.

[14] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for RGB-D based object recognition," in *International Symp. Experimental Robotics*, 2013, pp. 387–402.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. (2014). Caffe: convolutional architecture for fast feature embedding. arXiv: `1408.5093`.

[16] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using RGB-D cameras," in *RoboCup 2011: Robot Soccer World Cup XV*, 2012, pp. 306–317.

[17] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *Transactions on Graphics (TOG)*, vol. 23, 2004, pp. 689–694.

[18] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Int. Conf. on Computer Vision*, 2007.

[19] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.