# Trained 3D models for CNN based object recognition

Kripasindhu Sarkar[1,2], Kiran Varanasi[1] and Didier Stricker[1,2]
[1]*German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany*
[2]*Technical University Kaiserslautern, Germany*
{*kripasindhu.sarkar, kiran.varanasi, didier.stricker*}*@dfki.de*

Keywords: Object Recognition, Fine-tuning CNNs, Domain fusion, Training on 3D data, Graphics assisted CNN

Abstract: We present a method for 3D object recognition in 2D images which uses 3D models as the only source of the training data. Our method is particularly useful when a 3D CAD object or a scan needs to be identified in a catalogue form a given query image; where we significantly cut down the overhead of manual labeling. We take virtual snapshots of the available 3D models by a computer graphics pipeline and fine-tune existing pretrained CNN models for our object categories. Experiments show that our method performs better than the existing local-feature based recognition system in terms of recognition recall.

## 1 INTRODUCTION

In this paper, we describe a method for automatically identifying a 3D object from a data base given an input image taken from close range. We specifically address the application scenario where a machine part or a museum object needs to be precisely identified in a catalogue. The traditional approach for solving such a problem is to build a content-based image retrieval (CBIR) system that automatically recognizes a given object. However, the task is challenging due to object similarity, variations in appearance and lighting, as well as perspective. Typically, the catalogue contains only a 3D CAD model or 3D scan of the object. Creating an image database from actual photographs of each object in the catalogue is a significant overhead which may be impractical. In this paper, we assume that only the 3D model is available and automatically render a set of images using a synthetic computer graphics pipeline.

A related but different problem is automatic labelling of object categories in an image - such as the ImageNet challenge (Russakovsky et al., 2015). In this problem, a significantly large data set of images is collected that encapsulates the variation in object categories and labels: 15 million labelled high resolution images in over 22,000 categories in ImageNet. The top-performing methods for this challenge use deep convolutional neural network (CNN) that are trained on this large labelled data set (Krizhevsky et al., 2012; Girshick et al., 2014; Girshick, 2015; Ren et al., 2015). It is very hard to extend these methods to our application scenario, as the size of the training set is much smaller. However, the earlier neural layers of the CNN models that are pretrained on this larger training set learn generic image features that encode variability in perspective and lighting. In this paper, we extend these pretrained networks with a new learning mechanism that uses the rendered synthetic images of the 3D models in our catalogue. We show that our novel method of training achieves a large improvement in recognition accuracy when precise 3D models are available.

Also through this work, we want to make a conscious effort to shift our recognition system follow human behaviour. This is due to the fact that human beings do not need a lot of data to learn object categories. In fact, only a few examples per category are needed to learn and deduce about the category(Fei-Fei, 2006; Fei-Fei et al., 2006). It is estimated that a child learns almost all of the 10 - 30 thousand object categories in the world by the age of six (Biederman, 1987). These facts tells us that it is possible to achieve high detection accuracy without the need of such a large number of training examples. Second, human beings 'see' objects in 3D. We infer 3D properties of an object looking at the 2D image.

In this work, we make use of the 3D models by taking their virtual snapshots from several perspectives with different types of background. We use these generated images to fine-tune existing trained CNN model for our instances/categories.

Model based recognition techniques existed in the topic of local-feature based detection system for

a long time - since the invention of SIFT (Lowe, 2004). Here, 3D models were augmented with 2D local features computed as through Structure From Motion (SFM) algorithms (Snavely et al., 2006; Snavely et al., 2008) performed on hundreds of images taken of a real object. These feature-augmented 3D models were used for object recognition in the test time (Collet Romea and Srinivasa, 2010; Collet Romea et al., 2011). To avoid manual work of taking images and with the availability of realistic 3D models, (Sarkar et al., 2016) solved the same problem by using 3D model as the only source of training data to create the feature-augmented models. In this paper, we also consider only 3D models as input. However in contrast to (Sarkar et al., 2016), we train a CNN for the task of object recognition.

Therefore, the main contributions of our paper are as follows.

1. We create a training system for object recognition in 2D images, which only uses 3D models as the training data.
2. We show that in the presence of accurate 3D models, the recognition accuracy of our system is better than the state-of-the-art local-feature based recognition system.
3. We perform a systematic evaluation with rendering parameters of our method and show that background, texture and number of training images have significant role in the training process in terms of recognition accuracy.

It is to be noted that our recognition system presented in this paper performs extremely well in the presence of accurate 3D models to be detected in the scene. In the present scenario, due to the availability of cheap and accurate 3D scanners in the robotics and vision labs, it is possible to easily acquire accurate 3D models; which makes our recognition system quite effective. Also, other than the large variety of easily available 3D scanning hardwares (D'Apuzzo, 2006), simple software solutions for 3D acquisition are available where 3D models can be acquired using off-the-shelf hardware (3Digify, 2015).

# 2 RELATED WORK

We provide in this section the literature review about topics which touches different aspects of our work. We start by providing details of CNN based classification systems followed by works which augment the existing dataset by the rendering of 3D models - the inspiration of our work. We then shortly describe the classical work in the direction feature based recognition. We finish by providing some details about systems which purely focus on shape classification. They are technically very similar to our work, but instead of the recognition of 3D models in 2D images, they perform shape or 3D model classification.

## 2.1 CNN based object classification

Convolutional neural networks (CNNs) are feedforward neural networks with convolutional layers (a layer with a small filter sharing its weights throughout its input volume), introduced in (Lecun et al., 1998). With the availability of larger training data such as ImageNet (Russakovsky et al., 2015), it became possible to train deeper networks. The first popular CNN was the AlexNet (Krizhevsky et al., 2012) which won the ImageNet challenge by a large margin inspiring a large amount of work in this direction. For object detection CNNs were applied to object regions for the presence and absence of an object (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015). These CNN based techniques remain the current state of the art for object detection in 2D images and image classification. Because of its high success in image classification, we use deep CNN in our training architecture. In particular, due to its simplicity, we use the configuration of AlexNet in our network and fine-tune the weights based on our requirements.

## 2.2 Data augmentation for learning

These methods render 3D models using computer graphics pipeline and uses the rendered images as training data to 2D techniques. Popular work in this direction is chair detection using exampler SVMs (Aubry et al., 2014). The more recent works use this rendering technique to augment existing 2D image dataset and perform CNN based detection on the augmented dataset. (Peng et al., 2014) used rendering of similar looking 3D models of some of the categories of PASCAL VOC dataset (Everingham et al., 2010) to augment the training dataset, and found it to perform superior to training on only the given training set. (Su et al., 2015b) used image of rendered 3D models to augment PASCAL 3D+ dataset to improve results on viewpoint detection. Our contributions here are deeply inspired by these works. But instead of augmenting existing training dataset with rendered images to improve accuracy, we create the entire training dataset out of the rendered images and perform training on them for the recognition task. We show in the this paper that when the 3D models for the recognition system is highly accurate resembling

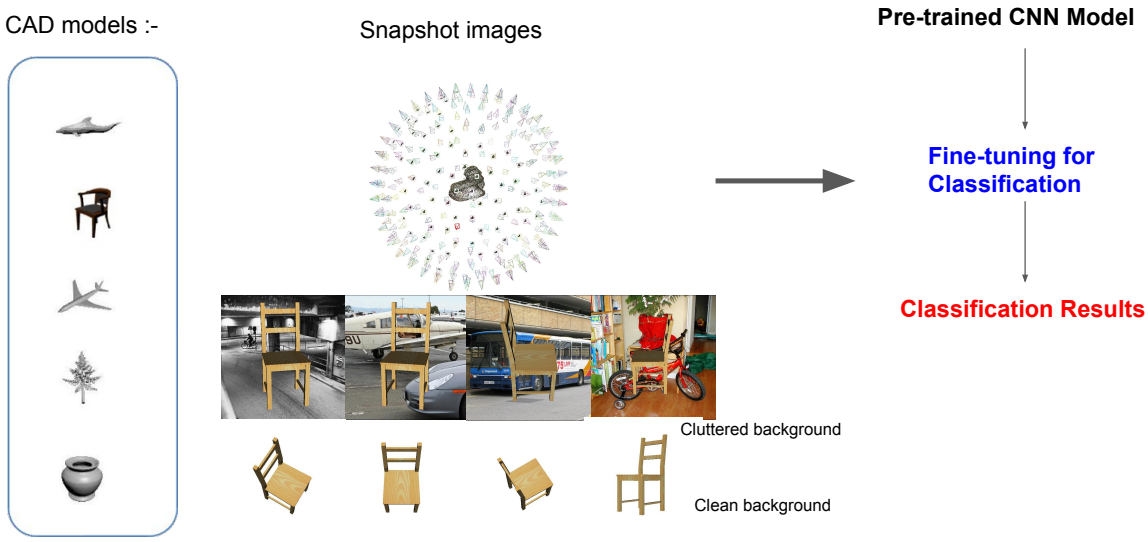



Figure 1: Summary of CNN based detection of CAD models in 2D images

the real object - which is the case for feature based object recognition system, performing training only on the synthetic images is sufficient for achieving a good recognition accuracy.

### 2.3 Feature based object recognition

Feature-augmented 3D models are created by performing Structure From Motion (SFM) on the training images taken of the object to be recognized. This association of 3D points - to - 2D descriptors, as a result of SFM, forms the pillar of most of the feature based detection, where the features extracted from a given input image, are matched to that of the feature augmented 3D models and subsequently, a *6 DOF recognition* is made (Skrypnyk and Lowe, 2004; Hao et al., 2013; Collet Romea et al., 2011; Collet Romea and Srinivasa, 2010; Irschara et al., 2009). (Sarkar et al., 2016) provided a new method for creating feature-augmented models in the presence of accurate 3D models at the training time. They used the texture-map of the 3D models to assign 2D features to the 3D points of the model, and in the second method, took rendered virtual snapshots to group 2D features assigning to the 3D points. Our method is logically similar to this work, because of the fact that we only use 3D models for training for the object recognition. In contrast, we use CNN for our training and outperform the result of recognition accuracy of this work.

### 2.4 Shape classification using rendered images

Shape classification is the problem of classifying shapes (3D models) from a database of 3D models. MVCNN (Su et al., 2015a), the state-of-the-art method for shape classification, renders 3D models or shapes from different views, and performs training on the rendered images for the task of classification. This work demonstrated that training just on the rendered images can be powerful. We use a similar training technique, but instead of performing shape classification, we solve the problem of object recognition in given 2D images.

## 3 TRAINING 3D MODELS

We perform the task of object recognition in images. Given a dataset of 3D models representing the object to be recognized, and an input query image containing an instance of an object, the problem is to find the correct 3D model present in the image. Here, the input query images are available only during the test time. We process on the database of 3D models, perform training, and use the trained model for performing recognition during the test time.

In summary, our procedure consists of taking virtual snapshots of each object from different views and using the snapshots to fine-tune an existing CNN models. Figure 1 summarizes the procedure.

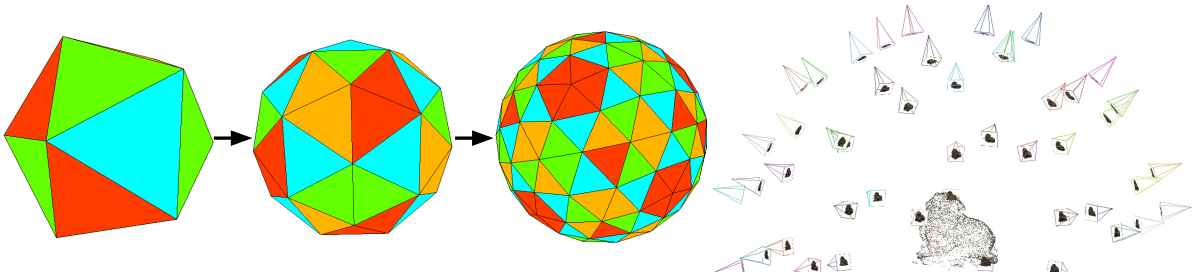


Figure 2: Rendering technique used for generating virtual snapshots. An icosaheadron is recursively subdivided, virtual cameras are placed in the center of its face and is pointed towards the 3D model placed at the center of the polyhendron to take virtual snapshots. [1]

## 3.1 Rendering techniques

### 3.1.1 Rendering Views

Our CNN model trains on the virtual snapshots of the 3D model. Therefore, the set of virtual snapshots should cover a good variations in terms of viewpoints, in order for the CNN to perform well on a real query image during the test time. Thus, we intend to take snapshots of the 3D model from all possible directions. This is unlike (Peng et al., 2014) where 4-5 manually chosen views were used to augment the training set. This is because, our training set is limited to the rendered images and we intend to capture maximum information possible from this set.

To achieve this, we place the model in the origin and point the camera towards the model from a set of uniformly discretized rotation angles. We do this by placing the virtual camera at the faces (or vertices) of a tesselated icosaheadron as icosaheadron is the largest convex regular polyhedron with 20 faces. Tesselation level of $n$ which subdivides a triangle into 4 triangles recursively $n$ times, providing us with the parameter to control the number of views. We also observed that rendering a model from the bottom of its upright position does not provide us with useful views of the object. In fact, many times the bottom of the model do not resemble the actual object at all, giving rise to confusion to the learning algorithm. Therefore, we only consider the top half of the tesselated icosaheadron for taking the virtual snapshots. Note that this assumes that the models are provided in their upright position and are normalized in a unit cube.

Considering only the top half of the tesselated icosaheadron for placing the virtual camera for generating the views, the tesselation levels of 0, 1 and 2 provide us with 10, 40 and 160 images respectively. The procedure is summarized in Figure 2.

### 3.1.2 Background Images

It is observed that the system performs better in the cluttered scene when some background is introduced in the rendered images. The effect of the background based on the dataset is more elaborated in the Results section. In our rendering, we used both white background and real images as background. Because of the fact that our models are indoor, we chose backgrounds resembling indoor scene. We considered few categories in the PASCAL dataset which are generally present indoors (Example - Television), but do not resemble any of the categories of our 3D models to be recognized to avoid conflicts. We then use the images in these categories to create background images for our training set. To show the effect of the background we perform experiments with different combination of rendered and white background as explained in Section 4.2.

### 3.1.3 Rendering scheme

We use the default rendering settings of Visualization Toolkit (VTK) (VTK, ), which uses a directional headlight located at the center of the camera and Phong shading interpolation. We render the model with and without texture to show that for highly accurate models the presence of texture do not accuracy significantly.

## 3.2 CNN framework

### 3.2.1 Framework

We use the eight layer 'AlexNet' ((Krizhevsky et al., 2012)) as our neural network architecture for training and testing because of its popularity and simplicity. Though recent deep networks like VGGNet (Simonyan and Zisserman, 2014), or more recent (He et al., 2015) should work better. In our experiments on

[1] Generated from https://www.openprocessing.org/

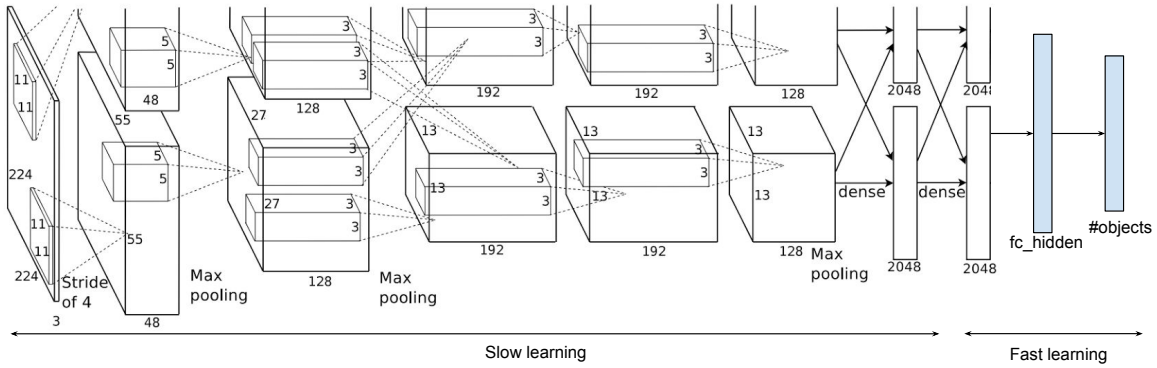



Figure 3: AlexNet architecture, which is used for our evaluation. We fine-tune a pre-trained AlexNet model for our object recognition problem by learning faster on the last fully connected layer while performing a slow weight update in the hidden layers. The image of AlexNet is taken from (Krizhevsky et al., 2012) and is modified to show its use for the classification of $n$ number of object in our pipeline. Blue blocks show the additional layers.

the dataset by (Sarkar et al., 2016) with 7 categories, we found AlexNet to be sufficient for the recognition task.

The architecture of the network is shown in the Figure 3. The network takes a fixed size $227 \times 227$ RGB images as input, feeds it through 5 convolutional layers (with max-pool layers after first, second and the last layers) followed by three fully connected (FC) layers of 4096, 4096 and 1000 neurons respectively, to produce a score for each of the 1000 class it was originally intended to use. To use it for our purpose for recognizing $n$ objects, we change the last layer to have only $n$ neurons. To replace AlexNet with another pre-trained network, one needs to follow the same step of replacing the last layer to have the number of neurons same as the number of object. We keep the SoftMax layer as used by AlexNet for classification of the object. We optionally add another layer before the last layer and call it *fc_hidden* as it shows to increase the accuracy.

### 3.2.2 Training

For training, we essentially fine-tune a pre-trained AlexNet model for the purpose of category detection. More elaborately, we take a trained model of AlexNet, replace the last fully connected (FC) layer to contain the same number of neuron as the number of our categories, and initialize them with random weights. We then learn the entire network end-to-end by backpropagation with SoftMax classification error, and update the weights of the last fully connected layers 10 times faster than the weights in the other layers. This causes the last FC layer to learn and adapt to our categories while enabling a slow adaptation of the hidden layers towards the synthetic data.

Our experimentation shows that introduction of a new FC layer (fc_hidden) before the final classifier while doing a slow update throughout the network - which is similar to learning of a fully connected neural network classifier on the FC7 features, increases the accuracy compared to the one without the new layer. This is due to the fact that the hidden layer causes the network to have non-linearity than training only the last layer. However, a fast update on FC7 without the addition of fc_hidden performs very similar. This is due to the fact that FC7 now introduces the nonlinearity providing a fully connected neural network working on the FC6 features. We chose this setting for our experiments because of its simplicity. The architecture is shown in Figure 3 where we include the hidden layer as it is a viable option. We also found the method of fine-tuning all the layers perform better than freezing all but the last one.

# 4 EXPERIMENTAL RESULTS

## 4.1 Dataset

We use the subset of the the dataset provided by (Sarkar et al., 2016) to test our system. In brief the dataset we considered contains 5 textured meshes - *Lion, Totem, Matriochka, Milk-carton* and *Whitener* and a set of test images. The test image set contains in total around 3000 images of the real objects corresponding to the provided meshes. The meshes represents accurate version of the real objects and has been reconstructed by the scanner 3Digify (3Digify, 2015).

Table 1: Comparison of object recognition recall with our system, local feature based using MOPED (Collet Romea et al., 2011) and local feature based using textured 3D models (Sarkar et al., 2016). Section 4.2 explains the settings used in our methods. Note that we only show the results of recognition recall of these two objects while considering recognition between 5 models explained in 4.1

| | Milk-carton | Lion |
|---|---|---|
| txtmap (Sarkar et al., 2016) + MOPED (Collet Romea et al., 2011) | 0.71 | 0.63 |
| Manual SFM + MOPED (Collet Romea et al., 2011) | 0.86 | 0.70 |
| tmap (Sarkar et al., 2016) + PnP + RANSAC | 0.55 | 0.72 |
| ours - R-BG + W-BG + Full texture | **0.98** | **0.76** |
| ours - W-BG + Full texture | 0.80 | 0.42 |
| ours - C-BG + W-BG + Full texture | 0.81 | 0.74 |
| ours - C-BG + W-BG + WO texture | 0.45 | 0.20 |
| ours - C-BG + Full texture | 0.80 | 0.67 |

## 4.2 Rendering settings

**Background and texture** We performed different experiment which gives insight to the results of the learning algorithm with respect to the rendering techniques. The different methods considered are described below:

- **R-BG** Rendered with random indoor background taken from indoor categories of PASCAL dataset.
- **W-BG** Rendered with white background.
- **R-BG + W+BG** Rendered with random indoor background and white background together.
- **C-BG** Rendered with chosen background similar to that of test images. Here we took 10 - 15 images of a table and used them as background.
- **Full Texture** Rendered with full texture.
- **WO Texture** Rendered without any texture.

**Number of images** We found that when the virtual snapshots were are taken from varying distance from the camera, the results were better. Therefore, we follow the technique in Section 3.1.1 and took snapshots at 7 different distances (radius of tesselated icosaheadron) with the tesselation level of 1 - which gives a total training image set of around 300 image per background settings. We found that increasing the tesselation level to higher level do not improve the accuracy.

## 4.3 Comparing Algorithms

We use two models *Lion* and *Milk-Bottle* to compare our results with local-feature based object recognition results. For creating feature-augmented models, we considered both the Structure From Motion approach (Eg - (Collet Romea et al., 2011)) and Model based approach from (Eg - (Sarkar et al., 2016)). We limited to these two models as building feature-augmented models from SFM has a large manual overhead. Also, more images of these models in the test-image-set in our dataset made it possible to create a robust feature-augmented model from SFM.

For online phase or matching phase we considered PNP + RANSAC (solution of Perspective-n-Points problem under RANSAC iterations) and MOPED (Collet Romea et al., 2011) framework.

Table 2: Comparison of object recognition recall with our system and local feature based using textured 3D models (snap2 & tmap) (Sarkar et al., 2016). Here ours use the settings *C-BG + W-BG + Full texture*

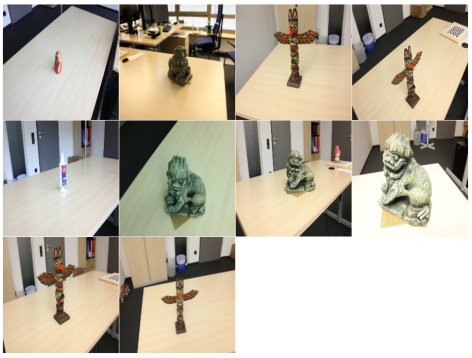
| Models | snap2 | tmap | **our** |
|---|---|---|---|
| Milk-carton | 0.88 | 0.71 | **0.81** |
| Totem | 0.83 | 0.21 | **0.98** |
| Lion | **0.89** | 0.63 | 0.74 |
| Whitener | 0.48 | 0.63 | **0.77** |
| Matriochka | 0.58 | 0.62 | **0.64** |

## 4.4 Comparison

The detailed comparison of our approach to existing approaches is shown in Table 1. The table shows the advantages of our method over the existing texture based recognition approaches. It is seen that the combination *C-BG + W-BG + Full texture* works the best in our case giving the best accuracy. Note the setting *R-BG + W-BG + Full texture* is equally effective and gives a better recall value for Lion and Milk-carton; though the overall accuracy over the dataset of this method is slightly less.

Table 2 shows the comparison of our approach to a

(a) Positive recognition of instances



(b) Incorrect recognitions - from left to right, top to bottom - *(predicted, actual)* [('Lion', 'Matriuschka'), ('Matriuschka', 'Lion'), ('Matriuschka', 'Totem'), ('Matriuschka', 'Totem'), ('Milkbottle', 'Whitener'), ('whitener', 'Lion'), ('Totem', 'Lion'), ('Totem', 'Lion'), ('Matriuschka', 'Totem'), ('Matriuschka', 'Totem')]

Figure 4: Example recognition. (Top) Positive recognition of instances. (Bottom) Example erroneous detection. Note the effect of the background affecting recognition.

previous model based approach which uses local features. Here, we chose our best settings of *C-BG + W-BG + Full texture*.

As seen from the experiments, texture seem to have effect in our training - and so does the background. This is true specially for high textured models such as Lion and Matriochka. Figure 4 shows some of the example recognition. The erroneous cases show the contribution of the background for error. This and the quantitative analysis makes us conclude that for a cross domain learning scenario, it is best to include maximum variations in the training image set.

## 5 CONCLUSION

We have presented a method of using only 3D models for learning CNN architecture for the purpose of object detection. We showed that in case of accurate models and uncluttered test images, our method can be successfully applied with a very high accuracy in results. The problem arises in the absence of accurate 3D models, or in the case when the 3D model covers a broad category instead of a single recognition instance. In those case there is a missing domain gap which needs to be fulfilled. With this paper we have identified the following promising set of future work:

1. Identifying the domain gap between the features in rendered images and real images with some of the recent works, such as (Sun et al., 2015; Tzeng et al., 2014; Long and Wang, 2015), and apply domain transfer for improving the accuracy.
2. Use of photorealistic rendering to decrease the domain gap.
3. Finding camera viewpoint, or performing 6DOF object recognition for accurate 3D models using CNN, thereby creating a recognition system as complete as the local feature based systems like (Sarkar et al., 2016; Collet Romea et al., 2011) enabling its usage in robotics application such as grasping.

## 6 ACKNOWLEDGEMENTS

This work was partially funded by the BMBF project DYNAMICS (01IW15003).

# REFERENCES


3Digify (2015). 3digify, http://3digify.com/.

Aubry, M., Maturana, D., Efros, A., Russell, B., and Sivic, J. (2014). Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *CVPR*.

Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115.

Collet Romea, A., Martinez Torres, M., and Srinivasa, S. (2011). The moped framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research*, 30(10):1284 – 1306.

Collet Romea, A. and Srinivasa, S. (2010). Efficient multi-view object recognition and full pose estimation. In *2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*.

D'Apuzzo, N. (2006). Overview of 3d surface digitization technologies in europe. In *Electronic Imaging 2006*, pages 605605–605605. International Society for Optics and Photonics.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Fei-Fei, L. (2006). Knowledge transfer in learning to recognize visual objects classes. In *Proceedings of the Fifth International Conference on Development and Learning*.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, abs/1504.08083.

Hao, Q., Cai, R., Li, Z., Zhang, L., Pang, Y., Wu, F., and Rui, Y. (2013). Efficient 2d-to-3d correspondence filtering for scalable 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 899–906.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2599–2606.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Long, M. and Wang, J. (2015). Learning transferable features with deep adaptation networks. *CoRR, abs/1502.02791*, 1:2.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Peng, X., Sun, B., Ali, K., and Saenko, K. (2014). Exploring invariances in deep convolutional neural networks using synthetic images. *CoRR*, abs/1412.7122.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sarkar, K., Pagani, A., and Stricker, D. (2016). Feature-augmented trained models for 6dof object recognition and camera calibration. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 632–640.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Skrypnyk, I. and Lowe, D. (2004). Scene modelling, recognition and tracking with invariant image features. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 110–119.

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846.

Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80(2):189–210.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. G. (2015a). Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*.

Su, H., Qi, C. R., Li, Y., and Guibas, L. J. (2015b). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *The IEEE International Conference on Computer Vision (ICCV)*.

Sun, B., Feng, J., and Saenko, K. (2015). Return of frustratingly easy domain adaptation. *CoRR*, abs/1511.05547.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474.

VTK. Visualization toolkit (vtk), http://www.vtk.org/.