Simple domain adaptation for CAD based object recognition

Kripasindhu Sarkar[1 2] and Didier Stricker[1 2]
[1] *DFKI Kaiserslautern* [2] *Technische Universität Kaiserslautern*
{*kripasindhu.sarkar, didier.stricker*}*@dfki.de*

Keywords: 3D object recognition; Domain adaptation; CNN

Abstract: We present a simple method of domain adaptation between synthetic images and real images - by high quality rendering of the 3D models and correlation alignment. Using this method, we solve the problem of 3D object recognition in 2D images by fine-tuning existing pretrained CNN models for the object categories using the rendered images. Experimentally, we show that our rendering pipeline along with the correlation alignment improve the recognition accuracy of existing CNN based recognition trained on rendered images - by a canonical renderer - by a large margin. Using the same idea we present a general image classifier of common objects which is trained only on the 3D models from the publicly available databases, and show that a small number of training models are sufficient to capture different variations within and across the classes.

1 INTRODUCTION

3D model based object recognition is the method for automatically identifying a 3D object from a database, given an input image taken from a close range. Often the catalogue contains only 3D CAD models or 3D scans of different object, and identifying them in images in traditional way will require creating image database from actual photographs of each object in the catalogue - which may be impractical. Therefore, existing approaches extract visual information from the 3D models and use them in a standard off-the-shelf recognition pipeline (Collet Romea et al., 2011; Sarkar et al., 2016; Sarkar et al., 2017). This is done either by extracting and augmenting 2D-local-features to the 3D location of the CAD model followed by feature-matching based recognition (Sarkar et al., 2016), or using a powerful CNN and solve the problem of classification (with recognition instances as classes) using the rendered images of the 3D models (Sarkar et al., 2017).

In spite of the fact that CNN based methods have progressed the state of the art results significantly for tasks related to images, the progress on 3D model based object recognition using CNN has not been substantial. For example, solving the task of recognition on rendered images of the 3D models in (Sarkar et al., 2017) achieved an 8% increase of recognition recall w.r.t. the local-feature base recognition. This is a big improvement over the feature-matching based methods, but is not comparable to the improvement achieved by CNN based methods in other areas where only real images are used for training (Krizhevsky et al., 2012; Ren et al., 2015; Ren et al., 2015). In this paper, we provide a simple method for bridging the gap between rendered and real images in an aim to significantly improve the results on recognition compared to the previous CNN based solutions.

Other than the problem of CAD based object recognition, learning categories of images using only 3D models as the source of training data, is also an important problem. This is because one single 3D model contains a lot more information than one particular view or image of that model. However, the rendered images of a 3D model lie in a significantly different domain compared to its real image. In this work, by using high quality rendering and a simple concept of correlation alignment, we decrease the gap between the domains of real and rendered images and provide a high performing classification system which is trained only using 3D models. We exploit the advances in computer graphics rendering techniques to improve the vision task of 3D model based recognition and classification.

Our contributions are the following:

1. We resolve the domain between synthetic images and real images by the combination of high quality rendering and a simple method of correlation alignment inspired by (Sun et al., 2015).
2. In the presence of accurate 3D models, we use our domain adaptation method and beat the existing state-of-art CAD based recognition in 2D images

by a *large margin* w.r.t. both *local-feature* based and *CNN* based recognition system.

3. In the absence of accurate 3D models, we use our method of domain adaptation and provide an accurate real-time classification system of common objects in office-desks, trained only on the 3D models from a publicly available dataset.

The demonstration video of the real-time classification/recognition system of the common objects in provided in the supplementary materials.

2 RELATED WORK

CNN based object classification AlexNet(Krizhevsky et al., 2012) was the first deep CNN model trained in GPU for the task of classification, and is still often used as the base model or feature extractors for performing other tasks. Other famous models which are often used as base CNN are VGG (Simonyan and Zisserman, 2014), GoogLeNet (Szegedy et al., 2015a), ResNet (He et al., 2015), InceptionV3/V4 (Szegedy et al., 2015b). VGG is a simple network which uses a series of small convolution filters of size 3×3 followed by fully connected layers. Due to its simplicity, we use the configuration of AlexNet in our network and fine-tune the weights based on our requirements.

Use of rendered images in vision These methods render 3D models using computer graphics pipeline and uses the rendered images as to augment the training data for the problem involving 2D images. (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. (Sarkar et al., 2017) uses only the rendered images as training set for 3D model recognition using CNN. In contrast to their work a) we render highly realistic images using powerful rendering engine b) bridge the gap further using correlation alignment of CNN feature c) achieve a large margin of improvement on recognition accuracy in comparison to them.

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. In contrast we use high quality rendered images to train CNN and outperform the result of recognition accuracy of this work.

Domain adaptation There has been several work on the line of adapting domain from different distributions. Geodesic methods bridge the source and target domain by projecting source and target onto points along a geodesic path (Gopalan et al., 2011). DLID trains a joint source and target CNN architecture for domain adaptation (Chopra and Balakrishnan, 2013). DAN (Long et al., 2015) and DDC (Tzeng et al., 2014) directly optimize the deep representation for domain invariance. Correlation Alignment or CORAL (Sun et al., 2015) is one of the simplest domain adaptation system where the whitened source features are recolored using target covarience. Because of its simplicity we adapted our method from this idea.

3 APPROACH

The focus of our work is to perform the task of object recognition of a dataset of 3D models in 2D query images. That is, we train only using the 3D models in the database and use the trained model for recognition during the test time (in query 2D images). As mentioned in the previous section we use a 2D CNN based pipeline (over local feature based pipeline) for this task because of its huge success in the general tasks of classification, recognition and detection.

Given a database of 3D models we take virtual snapshots of each model from different views with high quality rendering settings. Using this set of high quality images and a simple adaptation of the final features by correlation alignment (with the test images) we fine-tune a pretrained model for the task of recognition of the 3D instances. The following subsection describes the rendering settings in details

for the aim of bridging domain gap. Section 3.2 introduces the idea of correlation alignment, which is followed by Section 3.3 containing the details of the CNN framework to work with and without correlation alignment.

Assumption of 3D models We assume that the 3D models in the database are upright oriented along a consistent axis. This assumption holds true for most of the publicly available databases (Wu et al., 2015; Chang et al., 2015a), and has been use extensively by popular shape analysis methods (Su et al., 2015a; Johns et al., 2016; Qi et al., 2016; Maturana and Scherer, 2015). The information from the gravity direction is used in different settings for rendering - like lights and viewpoints.

3.1 Rendering scheme

We train a CNN on rendered images to use it for the classification of real images. The main goal is to have the rendering as realistic as possible. Therefore, we use a dedicated rendering engine (Blender) with the following aim. (1) The rendered images are realistic. (2) The collection of the rendered images are overfit-resistant. The rendering settings with their motivations are outlined in the following paragraphs.

General rendering ideas Rendering is the process of creating a 2D image from a 3D scene. The final image is based on factors such as camera settings, lighting settings, material and the render settings. First we start with the light settings and later we describe the different types of material settings (shaders). Please note that the lights work together with the shaders and therefore the following sections are not independent.

3.1.1 Lighting

Lighting/shading is one of the most important factors for realistic rendering. Even though rotation and scale invariance are taken care by data augmentation with random crops and rotations of the training images (Krizhevsky et al., 2012), incorporation of lighting invariance is not technically feasible when training is performed with real images. To mitigate this problem, the training set containing real images are often increased to cover instances with different lighting conditions. The presence of 3D models along with an advanced rendering pipeline gives us a big advantage on generating training images with different lighting conditions. Observing the lighting patterns in general scenarios, we experimented with the following lighting settings.

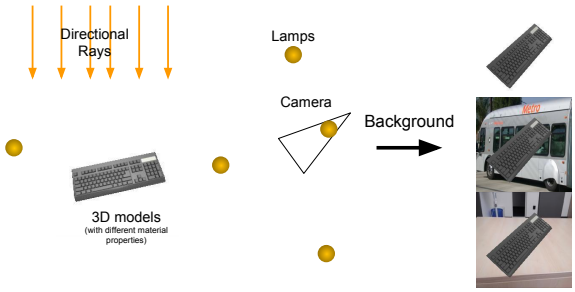

Figure 1: Overview of our rendering pipeline. See Section 3.1 for details

- *Uniform directional light*: We place a directional light of moderate intensity (from the top to bottom) which provides a uniform light from the top on every objects. We keep this light on for all rendered images.
- *Point light at the camera*: Point light is a light which radiates the same amount of light in all directions. The light intensity/energy decays based on the distance from the light to the object. We fix a low intensity point light at the location of the camera for all the renderings. This light is added to better highlight the textures and provide a well illuminated environment. In highly accurate 3D models which contains significant textures, this light has been crucial. If we just use this light (and ignore all other lights), the setting will reduce to the default rendering settings of popular rendering toolkits such as VTK and MeshLab.
- *Random point lights* Along with the two aforementioned lights which are fixed for all the rendering, we add 0 to 6 moderate intensity point lights (the actual number chosen at random) at random location at a distance 8 to 20 times the size of the object. We found this range of distance to add soft shadows similar to the real images of a model.

All the light sources are added to produce both specular highlights and diffuse shading. In addition, we used ray-traced mechanism for computing soft shadows.

3.1.2 Materials

In the presence of accurate 3D models (eg. in Wavefront .obj model), we use the material properties present in the models - which include the diffuse colors/intensity, specular exponent, transparency etc. Also based on our observation, we found that a flat shading is realistic (over the default smooth shading) in the presence of precise 3D models (scanned through an accurate 3D scanner). This is due the

high amount of details present in the models which needs no further smoothing, and the normal map in the faces are good enough to produce high quality shading. Therefore, we use the following configurations:

Accurate 3D scans: With accurate 3D scans (Eg. high texture dataset from (Sarkar et al., 2016)), we use flat shading with the material properties taken from the settings available in the scans. In the absence of material properties, we use a Lambert diffuse shader (factor of 1 - or max intensity) and a low intensity specular shader of high hardness. This is in congruent to the test images of the corresponding 3D models as well as the observation common objects.

Models from publicly available datasets: The models from the publicly available databases are often hand designed and not accurate as the real objects (in comparison to the 3D models which are scanned through a 3D scanner). Therefore smooth shading (which is also the default settings in many rendering engines) is more suitable for these kind of models. We use an 'auto smooth' functionality to combine both smoothing of normal and preserving sharp edge - edges where an angle between the faces is smaller than 30 degrees are smoothed. All the materials are chosen to cast and receive shadows.

3.1.3 Background images

Along with the aim of making the rendering realistic, we also make the rendered images overfit resistant. We do this by adding different backgrounds to the rendered image with the above mentioned settings. Following (Sarkar et al., 2017), we use 3 different backgrounds for one rendered image (with transparent background) - a) white background, b) random background from PASCAL dataset (Everingham et al., 2010) (without having any conflicting classes with our instances) c) background involving a table - which resembles test images. Therefore we get three times the number of images in the training set as the rendered images.

3.1.4 Views

When the class or the class of recognition instance belongs to one of the classes in PASCAL3D dataset (Xiang et al., 2014) (for example *chairs*), we sample the azimuth, elevation and in-plane rotation of the camera from a distribution estimated from the real image training set of PASCAL3D. In the absence of such categories sample it from a uniform distribution with the range adjusted to the categories. For example, for the instance *Totem* which is thin and tall in shape, the elevation angles are chosen between 0 and 45 degrees (so that the extreme top views are avoided), whereas for keyboard, the elevation angles are chosen between 30 and 80 degrees (so that the extreme side views are avoided).

3.2 Correlation Alignment

In the previous section, we described the details of creating realistic and overfit resistant set of images for decreasing the domain gap. We further apply a very simple, yet effective, method of Correlation Alignment of source and target feature distribution to further minimize the domain gap between the rendered images and real images. The method which is motivated by (Sun et al., 2015), aligns the input feature distributions of the source and target domains by minimizing the difference between their second-order statistics. Simply put, given the source feature set S, and target T, we perform the following steps to align the correlation of S to T to get the adapted feature set S'.

Correlation Alignment algoritm.

1. Compute covariance matrix of both S and T.
2. Whiten S using its covariance matrix to get S_w.
3. Recolor S_w with target covariance to get S'.

Even though the method is simple, it is shown to be as good as other specialized methods (Sun et al., 2015). Since we are using CNN for solving the task of classification, we can either treat the base CNN as a feature extractor to get source and target features for domain adaption, or use a specialized version of Correlation Alignment for CNN where we add a specialized loss (which is the frobenius norm between source and target features) along with the classification (or regression) loss while training the CNN. The exact method of using the adaptation in CNN is described in the next section.

3.3 CNN architecture

We use the eight layer 'AlexNet' as our neural network architecture ((Krizhevsky et al., 2012)) for training and testing because of its popularity and simplicity. Even though recent deep networks like VGGNet (Simonyan and Zisserman, 2014), or recent ResNet (He et al., 2015) should work better, our experiments on the with 5 categories (from the dataset by (Sarkar et al., 2016)), we found AlexNet to be sufficient for the recognition task. This also enabled us an easy comparison with (Sarkar et al., 2017).

Table 1: Comparison of object recognition recall of our system with local-feature based method (feature augmented models using the settings *snap2* and *tmap* in (Sarkar et al., 2016) + online matching using MOPED (Collet Romea et al., 2011)) and CNN based method without specialized rendering (Sarkar et al., 2017). Boldface numbers highlights the best and underlined numbers highlights the second best performing values.

Models	snap2 (Sarkar et al., 2016)	tmap (Sarkar et al., 2016)	(Sarkar et al., 2017)	our-r
Milk-carton	0.88	0.71	<u>0.81</u>	**0.98**
Totem	0.83	0.21	<u>0.98</u>	**0.99**
Lion	<u>0.89</u>	0.63	0.74	**0.98**
Whitener	0.48	0.63	<u>0.77</u>	**0.95**
Matriochka	0.58	0.62	<u>0.64</u>	**1**
Average	0.73	0.56	<u>0.79</u>	**0.98**

3.3.1 Finetuning with correlation alignment

There are specialized methods of using Correlation Alignment in CNN, which essentially adds a factor - correlation mismatch between the source and target domain - in the original loss (Sun and Saenko, 2016). This requires processing a batch of test images in every training iteration. Instead of processing a batch of test images, we further simplify the process by the following way. At first as a preprocessing step, we compute and store the covariance of the target CNN features (from the last FC layer - using the real test images). Then during training (using rendered images), we transform the features (by whitening and subsequently recoloring with the target distribution) to target domain as explained in Section 3.2. Finally, we compute the loss using the transformed features.

4 EXPERIMENTAL RESULTS

4.1 3D model based object recognition

In this section, we provide our results for the problem of object recognition of 3D models in 2D images when the models are of high quality and closely resemble the real objects. We use the subset of the high-texture dataset provided by (Sarkar et al., 2016) for this task. In brief, the dataset 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 a high quality 3D scanner of 3Digify (3Digify, 2015).

4.1.1 Rendered images

We took the ideas presented in Section 3.1 and performed rendering on the 5 models present in dataset. Since no view statistics (or training set containing real images) is available in this dataset, we manually define the ranges of camera extrinsics depending on the models (see Section 3.1.4). Since all the models in the dataset are provided in unit bouding box, the lights and the camera positions represented w.r.t. the bounding box unit. As mentioned in Section 3.1.3, we added a total of three background images per rendering - white, random and of a table similar to that in the test images. This idea was taken from (Sarkar et al., 2017) which also performs the tasks of 3D object recognition in 2D images using CNN.

We render 2000 images with different views per model - which results to 6000 images per category instance with different background.

4.1.2 CNN architecture and training details

In order to evaluate our method against (Sarkar et al., 2017) which uses AlexNet on rendered images, we use AlexNet as our base network. This is done to isolate the effect of our domain adaption from CNN design. Because of the simplicity of AlexNet, our training do not require large amount of GPU memory and we could easily use a batch size of 64 in GTX 1070. We finetune our network initialized from pretrained ImageNet for 30 epochs with a cross entropy loss. We use a learning rate of 0.0001 which we decreased by half after 20 epochs and Adam optimizer. The finetuneing of 30 epochs for a training set of size 30k images takes around 30 minutes.

4.2 Comparison algorithms

We compare our results with two very different approaches:

(1) Local-feature based recognition: We use the classical local-feature based object recognition pipeline where feature-augmented 3D models are created by (a) performing Structure From Motion (SFM) on the training images or by (b) model based approach such as (Sarkar et al., 2016). During test/query time, features extracted from the given query image are

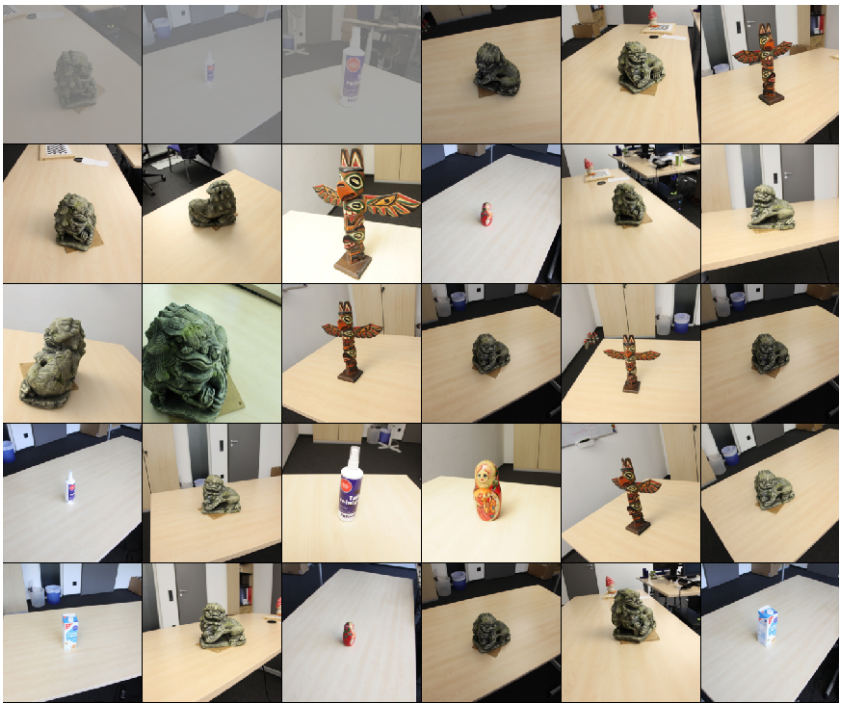

Figure 2: Some examples of the positive recognition from our system

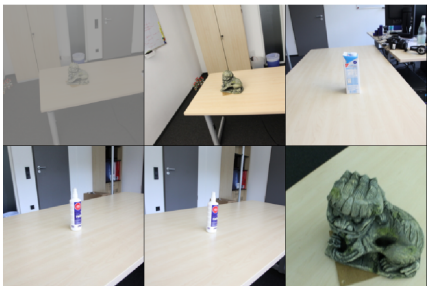

Figure 3: Some of the erroneous recognition. The predicted labels (left to right, top to bottom) are *Whitener, Whitener, Whitener, MilkBottle, MilkBottle, Matriochka.*

matched to that of the feature augmented 3D models and subsequently a recognition is made. For this query phase we used a sophisticated version of PNP + RANSAC (solution of Perspective-n-Points problem under RANSAC iterations) known as MOPED (Collet Romea et al., 2011). For creating feature augmented models we use both the technique of tmap and snap2 as explained in (Sarkar et al., 2017).

(2) CNN based recognition: We consider a baseline CNN approach where a pretrained model is fine-tuned with the rendered images with minimal rendering configurations. This minimal rendering scheme has been used by many of the existing model based systems (Hinterstoisser et al., 2017; Sarkar et al., 2017; Kehl et al., 2017). We use the results from (Sarkar et al., 2017) for this task, where the 3D models are rendered with the default rendering settings of Visualization Toolkit (VTK) - a directional headlight located at the center of the camera and Phong shading interpolation. We use their best configuration of background and texture for comparison.

The comparison of recognition recall is shown in Table 1. As seen, our simple yet effective method improve the previous CNN based recognition system by a large margin of **24%** (0.79 $\rightarrow$ 0.98). In fact, existing CNN based system (without sophisticated rendering) could only improve upon the classical local feature based system by 8% (0.73 $\rightarrow$ 0.79). This was mostly due to high quality and texture details of the 3D models in the dataset which could not create much difference in performance from feature matching based method with the CNN based approach with simple rendering. Our high quality rendering along with the domain adaptation improve on the local feature based system by a margin of **34%** (0.73 $\rightarrow$ 0.98). Figure 2 and 3 respectively shows some of the positive and erroneous recognition examples.

4.3 Real-time classification system

In this section we provide the details of our real time AR application of object classification which uses no real images for training. We chose 2 specialized categories and 6 common categories for office desks

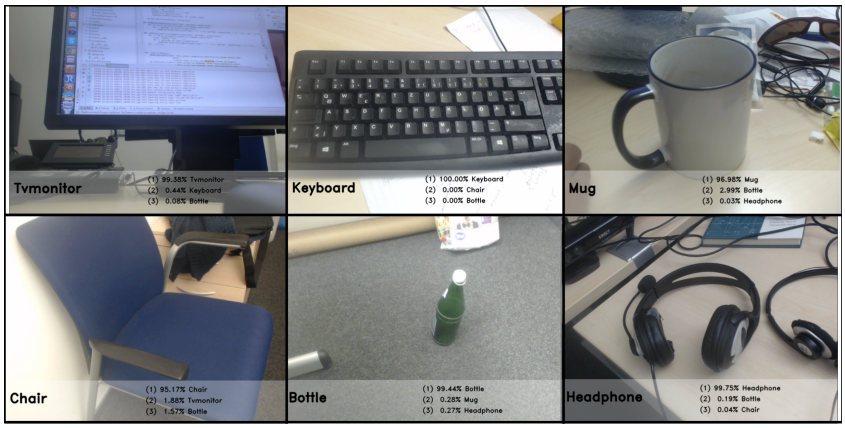

Figure 4: Snapshots of our demo application of real-time object classification using 3D models. Even with a very high background clutter our system provides good classification results.

(namely - Keyboard, Monitor, Headphone, Mug, Bottle, Chair) in the aim of having an appropriate demo application in the office/lab environment.

Our classification system is trained just using the rendered images of general 3D models with the settings mentioned in Section 3.1. *No* real images are used at any step. Also, the 3D models used are from publicly available databases and do not correspond to the exact objects from the real world.

4.3.1 Training set

We use ShapeNet (Chang et al., 2015b) to get 3D models of the respective categories. ShapeNet core contains common daily objects with alignments which we used for the gravity direction while placing lights and camera. We randomly take a maximum of 100 3D models for each category, and sample 600 different views for each 3D model - giving a total of 6000 rendered images for a category. Along with the background augmentation (3 different backgrounds - Section 3.1.3), we train using a total of 18k images per class.

The details of finetuning CNN for the classification application is similar to that of Section 4.1.2. We finetune AlexNet using the generated images explained above.

4.3.2 Testing with real images and AR application

Our application performs impressively even though our network is not trained using any real images. Figure 4 shows some screenshots of our application. As we zoom towards an object, the application shows the category class. A short video showing the output of our application is provided in the supplementary material.

5 CONCLUSION

In this work we provided a simple, yet powerful domain adaptation system by the fusion of high quality rendering and correlation alignment. Using the rendered images of our method, we significantly outperformed the existing system which uses a simple renderer but same learning technique. We also showed that this idea can be generalized to learn a classifier capturing different variations of categories only by using the 3D models of the publicly available datasets. In future we would like to extend our classification system to identify 6DOF pose by attaching a regressor when trained only using the 3D models.

REFERENCES

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

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015a). ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015b). ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chopra, S. and Balakrishnan, S. (2013). Deep learning for domain adaptation by interpolating between domains.

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)*.

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.

Gopalan, R., Li, R., and Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 999–1006. IEEE.

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.

Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K. (2017). On pre-trained image features and synthetic images for deep learning. *CoRR*, abs/1710.10710.

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.

Johns, E., Leutenegger, S., and Davison, A. J. (2016). Pairwise decomposition of image sequences for active multi-view recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 3813–3822. IEEE.

Kehl, W., Manhardt, F., Tombari, F., Ilic, S., and Navab, N. (2017). Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*, pages 22–29.

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.

Long, M., Cao, Y., Wang, J., and Jordan, M. I. (2015). Learning transferable features with deep adaptation networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 97–105. JMLR.org.

Maturana, D. and Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*.

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

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

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.

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.

Sarkar, K., Varanasi, K., and Stricker, D. (2017). Trained 3d models for cnn based object recognition. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, (VISIGRAPP 2017)*, pages 130–137.

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.

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.

Sun, B. and Saenko, K. (2016). Deep CORAL: correlation alignment for deep domain adaptation. *CoRR*, abs/1607.01719.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015a). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015b). Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567.

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/.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920.

Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*.