ZERO IN ON SHAPE: A GENERIC 2D-3D INSTANCE SIMILARITY METRIC LEARNED FROM SYNTHETIC DATA

Maciej Janik, Niklas Gard, Anna Hilsmann, Peter Eisert

Fraunhofer HHI, Berlin, Germany

ABSTRACT

We present a network architecture which compares RGB images and untextured 3D models by the similarity of the represented shape. Our system is optimised for zero-shot retrieval, meaning it can recognise shapes never shown in training. We use a view-based shape descriptor and a siamese network to learn object geometry from pairs of 3D models and 2D images. Due to scarcity of datasets with exact photograph-mesh correspondences, we train our network with only synthetic data. Our experiments investigate the effect of different qualities and quantities of training data on retrieval accuracy and present insights from bridging the domain gap. We show that increasing the variety of synthetic data improves retrieval accuracy and that our system's performance in zero-shot mode can match that of the instance-aware mode, as far as narrowing down the search to the top 10% of objects.

Index Terms— 3D model retrieval, zero-shot, siamese networks, synthetic data, domain gap

1. INTRODUCTION

Reasoning about real objects in the context of computer vision is easier if we are in possession of their 3D models. Unlike 2D images, 3D models are not spoiled by factors such as viewpoint, occlusion, or lighting, and therefore provide a pure and accurate reference to the real object. In particular, we are interested in rigid objects which can be reliably recognised based solely on their geometry.

In this paper, we learn a metric (Figure 1) which measures shape similarity of objects in RGB images and as 3D models. The metric is implemented as a Siamese CNN [1] which extracts shape features from images and models and, for identical shapes, maps them close together in the latent space. We represent 3D models by a set of textureless, shaded greyscale views, rendered from viewing positions on the corners of an icosahedron enclosing the object. Features computed from every view are merged into a compact representation by max-pooling, similar to [2]. Similarity was learned by exposing the CNN to same and different image-model pairs, computing cosine distance between their extracted features and applying the contrastive loss function [3].
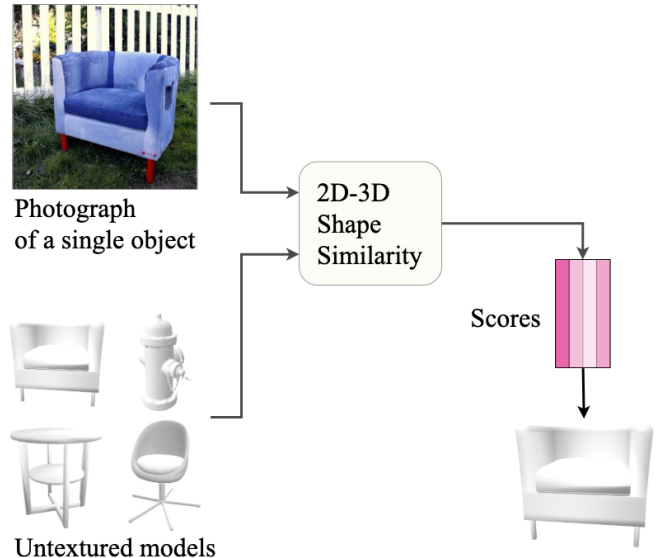

Fig. 1. Our network retrieves an unknown 3D model from an RGB image on the sole basis of object geometry.

In our work, we focus on zero-shot retrieval [4], that is, on optimizing the network to recognise shapes different from those shown in training. This ability to work with arbitrary shapes makes our network very practical, since making changes (such as adding new objects) to the retrieval set does not require any retraining of the model. To this end, we expose the CNN to a sufficient number of versatile image-to-model correspondences such that it learns a generic 2D-to-3D shape similarity function.

Due to scarcity of existing datasets featuring real images and 3D models of the exact same objects, we train our network using only synthetic data. We generate the data using Domain Randomization [5] and leverage the siamese architecture of the CNN to reduce the domain gap between the synthetic training data and the real test data.

In summary, our contributions are three-fold: 1) we design a network which finds a common embedding of two heterogeneous object representations 2) we demonstrate the network's potential for recognising unknown objects (zero-shot) 3) we present insights from training the network with only synthetic data and from bridging the domain gap.

2. RELATED WORK

Image-Based 3D Model Retrieval   In order to match images with 3D models we must first design suitable embeddings of these media. Compared to image descriptors, shape descriptors are still relatively unexplored. While shape descriptors can be computed from native model formats, such as a polygon mesh or a point cloud [6, 7], most of the existing approaches instead employ intermediate representations of models, such as grayscale renderings [2], depth maps [8], or location fields [9]. Such descriptors were shown to perform better in both model-to-model and model-to-image comparison tasks [2]. Besides, operating on 2D projections, rather than on native 3D formats, is computationally more efficient and allows leveraging existing image descriptors pretrained on large datasets such as ImageNet [10].

Another question is how to jointly embed shapes and images in a meaningful way. Our work is most similar to the one of [11] who also aggregate greyscale views of each model into a single vector, and map them together with RGB images with a siamese network. Different from us, they train their network using real images and corresponding shape instances from 40 different classes. As their training pairs use the closest available shape, rather than the exact same shape, their system effectively works on a class-level, rather than on the instance-level.

Grabner et al. propose to match images and shapes by computing location field descriptors from both media [9]. The authors train their network for instance-level recognition and achieve state of the art results on the Pix3D dataset [12]. However, the small size of the dataset restricts them to learning only a small number of shape classes. In contrast, in our work we extend the recognition to unknown classes. For this we employ (unlimited in samples) synthetic data, which shifts a portion of our work towards bridging the synthetic-to-real domain gap.

Synthetic Data   In recent years, visual deep learning saw a growing trend for using synthetic images [13, 14, 15], driven by the scarcity of available real data. Image synthesis allows to quickly generate fully annotated images in potentially unlimited quantities. Recently emerged tools such as BlenderProc [16] and NDDS [17] provide ready-made dataset generation pipelines and example maps for rendering custom objects. On the downside, synthesized images have different characteristics from the real ones. This results in a 'domain gap' – systems trained with synthetic data to show worse results on real data than if they had been trained with real data.

Two of the common approaches to mitigating this problem are Domain Randomization (DR) and Domain Adaptation (DA). DR, introduced by Tobin et al. [5] for training an object detector, proposes to augment the simulated scene in a large number of ways, such as by randomising object poses, textures, camera poses, and lighting, intending that the real scene

would appear to the network just as one of the many synthetic variations.

Although most works use synthetic data only for pretraining or complementing real data, Tremblay et al. [14] attempted to bridge the domain gap without using any real data. Their training set featured 50% non-photorealistic DR renderings and 50% photorealistic renderings. The authors showed that mixing these two types of data results in better performance then using either of the data types alone. Their all-synthetic dataset was used for 6D pose estimation of 21 textured objects, and it has shown competitive results against detectors trained with both real and synthetic data.

While DR is applied to the generation of data, DA aims to make up for different distributions of computed image features, usually in the direction from real to synthetic. For example, Massa et al. [18] add to their system an adaptation layer pretrained to map synthetic images of certain objects to their real counterparts. Lee et al. [11] adopt a similar approach, but their adaptation function is learned together with the rest of the CNN in an end-to-end manner.

Our approach follows the DR principle, such that it varies all parameters of the simulated scene except for the 3D shape of interest rendered in the foreground. This prompts the CNN to extract the pure shape information from complex real images. We follow Tremblay et al. [14] in using only synthetic training data, as well as making our training set half-photorealistic and half-randomised. Unlike them, we do not restrict ourselves to classifying a small set of textured shapes, but aim for learning a general idea of shape. On top of that we introduce a novel adaptation strategy of splitting the siamese networks in the few beginning layers allowing the CNN to extract the shape information differently from different types of data.

3. METHOD

Network Architecture   Figure 2 presents our network architecture. In each training or evaluation step the network compares one untextured 3D model with one colour image. We use ResNet-34 [19] to compute the image descriptors, and reduce the number output features from 1000 to 128. Shape features are computed from twelve all-around views. As shown in Figure 3, the same *CNN-view* computes features for all views, which are then merged into a single shape representation via max-pooling. In parallel, the *CNN-img* computes features of the colour image (either synthetic or real). *CNN-view* and *CNN-img* share the parameters from the ResNet's 7th layer on; this allows to better adjust to pixel-level differences, while still jointly extracting the relevant shape information in the later layers. Finally the shape vector and the image vector are L2-normalized and the cosine distance computes their similarity score. Based on the pair label and the similarity score, we compute contrastive loss [3], whose margin we set to 1.0.
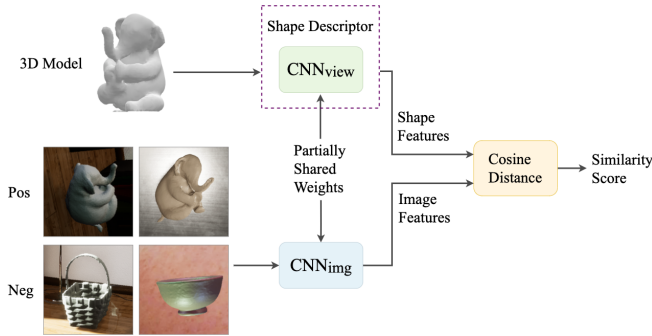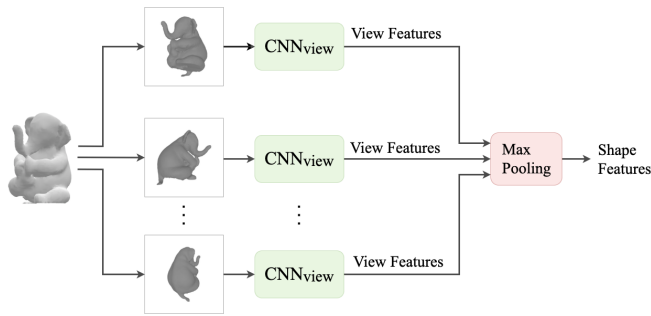
Fig. 2. Network Architecture

Fig. 3. Shape Descriptor

The training sets were designed such that each object was represented by a roughly equal number of colour images. While the *CNN-view* processed 12 views of an object, the *CNN-img* processed 6 positive and 6 negative images. On average, in one epoch every colour image was fed to the network twice, once as a positive and once as a negative example. In addition, the colour images were augmented with vertical and horizontal flip, random translation, rotation, and scale, as well as with varying contrast and brightness.

Data Generation Experimental data was made up of 3 types of images, all depicting a single object: 1) plain, greyscale renderings, 2) colour renderings, and 3) real photographs. Greyscale renderings, or views, (part of Figure 3) were generated using Pyrender [20]. Each mesh was given the default metallic material parameters and was placed at the centre of an icosahedron. The camera was placed at each of the 12 vertices and oriented at the object centre. Each captured view was cropped such that the object's longer dimension occupied 70% of the image. In addition, we tried representing 3D objects with sets of 20 and 42 views, but noted no advantage compared to using just 12.

Colour renderings served to imitate real images and were generated using BlenderProc and NDDS. One example of each is shown in Figure 4. In BlenderProc, we focused on photorealism of the images. The objects were rendered inside a cube, whose inner walls were assigned varying textures [21], and where the illumination was frequently randomised.
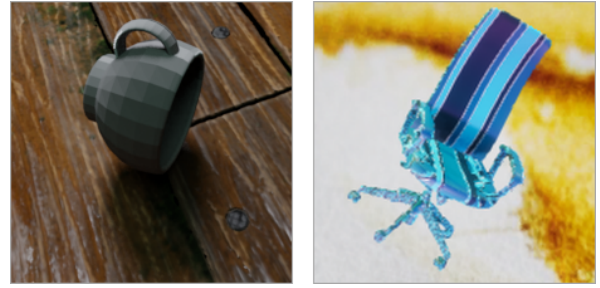

Fig. 4. Photorealistic renderings (BlenderProc, left) draw a direct link with the real world, while non-realistic ones (NDDS, right) provide more background and texture variation.

Objects were also assigned varying textures and placed on the floor in randomly sampled poses. Renderings were captured from an upper half sphere centred on the object, with a varying radius.

In NDDS, the objects were captured in a fully-randomised scene, by rendering objects in varying poses on many different backgrounds. In this case, the realistic look was traded for more variation in object textures [22], backgrounds [23] and lighting, the simplicity of the scene and the speed of rendering.

The last type of images are photographs which serve as test data only. The test set features 325 images picked from two datasets: Pix3D [12] and Toyota-Light (a subset of BOP [24]), and in small part found on web search engines. Each image is annotated with the name of the object it depicts. In order to assure that the objects are well visible, the images were cropped accordingly. We retrieve from a set of 50 models of distinct shapes representing furniture and other household objects. A link to the test set is given in the supplementary material.

4. EXPERIMENTS

The model was trained in two different modes, which will be referred to as instance-aware and zero-shot. In the first mode the same 50 object instances were used in training and in evaluation. Effectively, the network's task was to transfer knowledge about a set of shapes from synthetic to real data in the most efficient manner.

In zero-shot mode, none of the test objects were shown to the network in training. Instead, the model learned from data rendered from several hundreds of different 3D models, in pursuit of generalising to recognition of any shape.

The network was implemented in Pytorch [25]. After experimenting with AlexNet [26], VGG-16 [27] and different depth-versions of ResNet, we chose ResNet-34 as the best backbone for our task. We used Adam optimizer [28], set the learning rate to 5e-5, weight decay to 1e-5 and the batch size to 12. After every epoch we evaluated the model on the real test set and reported results from the checkpoint with the highest retrieval rate of the single correct object (Top-1

Acc. \ Mode	Photorealistic	Randomised	Mixed
Top-1	.50	.46	.59
Top-2	.65	.61	.73
Top-5	.86	.80	.87

Table 1. Comparison of different synthetic training data types.

Acc. \ Mode	Separate Params	Shared Params
Top-1	.55	.62
Top-2	.71	.75
Top-5	.87	.90

Table 2. The effect of siamese weight sharing.

Fig. 5. Deriving a sufficient number of objects to learn generic shape similarity. For comparison we also include the retrieval rates of the best instance-aware model.

accuracy). Depending on the dataset used this proceeding required training for 10 to 25 epochs.

The initial experiments were conducted in instance-aware mode. Firstly, we investigated how well different types of synthetic data serve learning our function. For this, we rendered two training sets of 6k images, featuring 50 object instances: one using BlenderProc, with the focus on photorealism and one using NDDS, with the focus on more randomisation. Examples of these images are shown in Figure 4. The network was trained thrice: first with only realistic data, then with only randomised data, finally with the union of the datasets. The top accuracy for all trainings are shown in Table 1. We see that the photo-realistic data alone yielded better results than the randomised data alone. Mixing the datasets resulted in the best accuracies of all, which speaks for increasing the variety of synthetic data to train with.

Secondly, we looked into the effect of siamese parameter sharing between the network's arms. The model was trained once more in instance-aware mode, this time using more training samples (20k) including renderings from two extra UE4 maps, in order to further increase the retrieval rates through more varied data. Following that, the model was retrained with the same data, this time with *CNN-view* and *CNN-img* learning separately. The retrieval rates shown in Table 2 demonstrate a clear advantage from sharing weights.

The final experiments were conducted in zero-shot mode, as we looked into how increasing the number of training objects affects the retrieval rates. For this the model was trained five times, using sets of 150, 300, 450, 600, and 1800 different meshes sampled from ShapeNet [29]. The best accuracy is shown in Figure 5, together with the best accuracy from the instance-aware mode (all models were evaluated on the same test set). The graph shows an increase in accuracy caused by raising the number of objects, until using 600 unique objects. Further increase in this number does not show a clear advantage. If we couple these results with those of Experiment 1, we see that for zero-shot object recognition in the wild it is more vital to increase the versatility of synthetic training data than to increase the number of samples beyond a certain point.
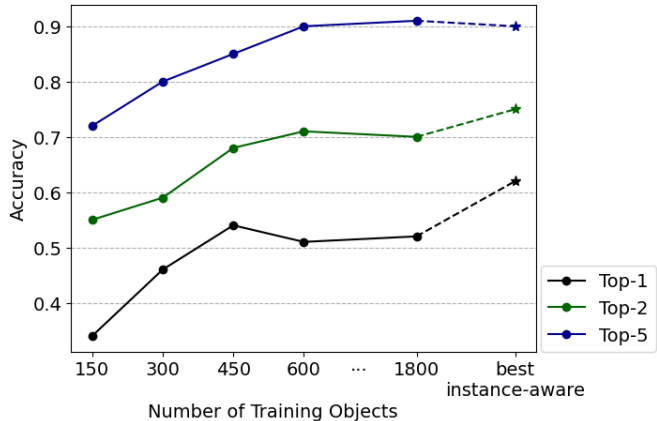
A comparison of the best instance-aware model and the zero-shot model shows that although the former's Top-1 and Top-2 accuracy are clearly better, zero-shot matches the other's performance with the Top-5 accuracy. This enables the use of our metric in applications that only require narrowing down the search to a few candidates.

In a qualitative analysis of retrieval cases we noted that the network tends to have trouble classifying images with strong patterns present in the background as well as those with non-uniform texturing of the object itself.

5. CONCLUSIONS

We presented a novel approach to matching RGB images and 3D models of arbitrary objects. We showed that our network, trained with only 600 unique objects, can extend its recognition ability from seen to unseen shapes as far as the top five retrieved results are concerned. This zero-shot functionality has practical use in situations where we need to recognise obscure shapes, for which little training data exists or wherever we frequently add new objects to the retrieval set. Besides, we showed the advantage of siamese weight sharing for learning cross-dimensional similarity of shape, and of increasing the variety of synthetic training data.

In the future, we plan to explore different shape descriptors, such as MeshNet [6], as well as to backtrack network activations in order to highlight image pixels most responsible for the extraction of similar features from the two inputs [30].

6. ACKNOWLEDGEMENT

7. REFERENCES

[1] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 7, no. 4, pp. 669–688, August 1993.

[2] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015, p. 945–953.

[3] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. CVPR*, 2006, p. 1735–1742.

[4] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks," in *Proc. AAAI*, 2008, p. 646–651.

[5] J. Tobin, R. H. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IROS*, 2017, pp. 23–30.

[6] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "Meshnet: Mesh neural network for 3d shape representation," in *Proc. ECAI*, 2019, pp. 8279–8286.

[7] Q.-H. Pham, M. A. Uy, B.-S. Hua, D. T. Nguyen, G. Roig, and S.-K. Yeung, "Lcd: Learned cross-domain descriptors for 2d-3d matching," in *Proc. AAAI*, 2020, pp. 11856–11864.

[8] A. Grabner, P. M. Roth, and V. Lepetit, "3d pose estimation and 3d model retrieval for objects in the wild," in *Proc. CVPR*, 2018, pp. 3022–3031.

[9] A. Grabner, P. M. Roth, and V. Lepetit, "Location field descriptors: Single image 3d model retrieval in the wild," in *Proc. 3DV*, 2019, pp. 583–593.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009, pp. 248–255.

[11] T. Lee, Y.-L. Lin, H. Chiang, M.-W. Chiu, W. Hsu, and P. Huang, "Cross-domain image-based 3d shape retrieval by view sequence learning," in *Proc. 3DV*, 2018, pp. 258–266.

[12] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, "Pix3d: Dataset and methods for single-image 3d shape modeling," in *Proc. CVPR*, 2018, pp. 2974–2983.

[13] J. Tremblay, A. Prakash, D. Acuna, et al., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. CVPR*, 2018, pp. 1082–10828.

[14] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proc. CoRL*, 2018, pp. 306–316.

[15] S. Hinterstoißer, O. Pauly, H. Heibel, M. Martina, and M. Bokeloh, "An annotation saved is an annotation earned: Using fully synthetic training for object detection," in *Proc. ICCV*, 2019, pp. 2787–2796.

[16] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "Blenderproc: Reducing the reality gap with photorealistic rendering," in *Proc. RSS*, 2020.

[17] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," `https://github.com/NVIDIA/Dataset_Synthesizer`, 2018.

[18] F. Massa, B. C. Russell, and M. Aubry, "Deep exemplar 2d-3d detection by adapting from real to rendered views," in *Proc. CVPR*, 2016, pp. 6024–6033.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[20] M. Matl, "Pyrender," `https://github.com/mmatl/pyrender`.

[21] L. Demes, "Cc textures," `https://cc0textures.com`.

[22] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi, "Describing textures in the wild," in *Proc. CVPR*, 2014, p. 3606–3613.

[23] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *Journal of Artificial Intelligence Research (JAIR)*, vol. 47, no. 1, pp. 853–899, May 2013.

[24] T. Hodan, F. Michel, E. Brachmann, et al., "Bop: Benchmark for 6d object pose estimation," in *Proc. ECCV*, 2018, pp. 19–35.

[25] A. Paszke, S. Gross, F. Massa, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. NIPS*, 2019, pp. 8026–8037.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *CoRR*, 2014.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[29] M. Savva, A. X. Chang, and P. Hanrahan, "Semantically-enriched 3d models for common-sense knowledge," in *Proc. CVPR*, 2015, pp. 24–31.

[30] C. Seibold, A. Hilsmann, and P. Eisert, "Focused lrp: Explainable ai for face morphing attack detection," in *WACV Workshops*, 2021, pp. 88–96.