Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

Identifying Style of 3D Shapes using Deep Metric Learning

Isaak Lim Anne Gehre Leif Kobbelt

Visual Computing Institute, RWTH Aachen University

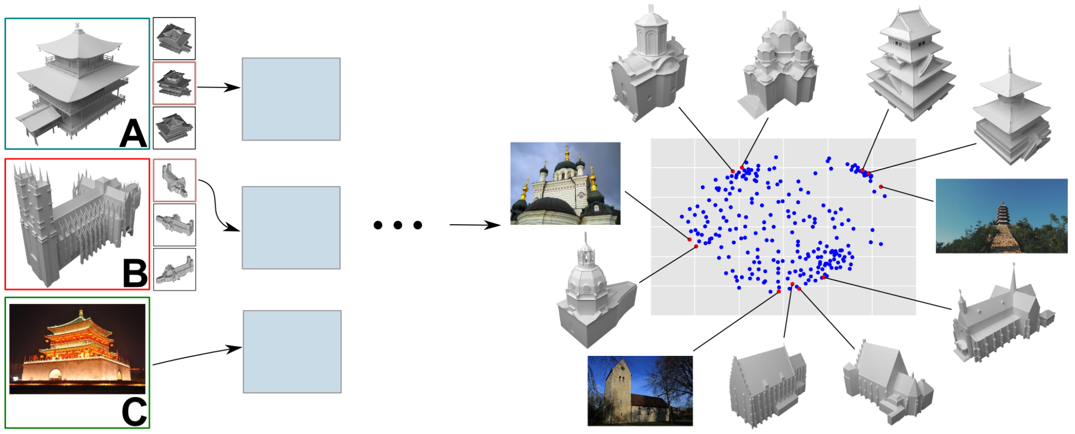

Figure 1: *A high-level visualisation of our approach to style similarity learning using deep metric learning. To the left a triplet consisting of a query sample (A), a dissimilar (B) and a stylistically similar sample (C) is shown, i.e. B and C are compared to A. Triplets of this type are fed into a neural network consisting of three convolutional networks (blue boxes), which computes an embedding (right) of these samples. The embedding space (here illustrated by multi-dimensional scaling) is such that stylistically similar samples are placed close together while dissimilar samples are placed further apart. By using rendered images as a representation for 3D shapes we are able to incorporate photos found online [pix]. This reduces the cost and effort to compile the training set since annotated images are broadly available while annotated (and consistent) 3D models are more difficult to get.*

Abstract
We present a method that expands on previous work in learning human perceived style similarity across objects with different structures and functionalities. Unlike previous approaches that tackle this problem with the help of hand-crafted geometric descriptors, we make use of recent advances in metric learning with neural networks (deep metric learning). This allows us to train the similarity metric on a shape collection directly, since any low- or high-level features needed to discriminate between different styles are identified by the neural network automatically. Furthermore, we avoid the issue of finding and comparing sub-elements of the shapes. We represent the shapes as rendered images and show how image tuples can be selected, generated and used efficiently for deep metric learning. We also tackle the problem of training our neural networks on relatively small datasets and show that we achieve style classification accuracy competitive with the state of the art. Finally, to reduce annotation effort we propose a method to incorporate heterogeneous data sources by adding annotated photos found online in order to expand or supplant parts of our training data.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

Keywords: style similarity, deep metric learning

1. Introduction

Quantifying human perception of the style of 3D shapes is key for many applications that aim to guide and support designers. However, the conversion of a perceived style similarity to a concrete distance measure is not straight forward. Solving this challenge allows for recommendation systems that suggest items that are rated to be stylistically similar to a query. In the setting of interior design, applications can suggest furniture, utensils, and decorations that form one stylistically consistent composition without the need for the user to have expert knowledge. Furthermore, designers can be supported by filtering and ranking database queries for items based on stylistic similarity.

However, identifying style in a wide range of settings automatically is a challenging task. Often, one is interested in the stylistic similarity of objects from many different categories, e.g. across items of furniture. This means that style has to be identified and compared for objects whose structure cannot be matched in a straight forward manner. The semantic segmentation of a sofa greatly differs from that of a lamp for example. It is important to note that object similarities do not necessarily correspond with style similarities. Objects that are stylistically similar can have a strong variance in the overall shape. Therefore, this notion of style similarity is hard to formalize.

Recently, Liu et al. [LHLF15] and Lun et al. [LKS15] have tackled this challenge of identifying perceptual shape style similarity. They rely on a set of geometric feature descriptors and meaningful segmentations of shapes to compare 3D models and infer the stylistic distance. However, it is hard to find a weighted set of explicit descriptors that capture the style of shapes well. Preferably, we do not just want to learn the weights of the descriptors but the descriptors themselves. The computation of the shape descriptors is often dependent on the consistency of the input 3D models (e.g. no holes, connected meshes). This leads to some amount of pre-processing. We can weaken these requirements by using an image based representation of the input, which allows us to use inconsistent and incomplete 3D shapes. Moreover, we are able to draw additional annotated data from the vast image repositories found online.

Following Lun et al. who state that a deep learning approach to the problem is of interest, we propose a method that makes use of deep metric learning. Given shapes for which we have relative stylistic information as input (e.g. object C is stylistically more similar to a query object A then another object B, cf. Figure 1), we train a neural network on rendered images of these shapes. The output are two distance values for (B,A) and (B,C), which correlate with their stylistic similarity. This comes with the benefit of not having to rely on hand-crafted feature descriptors. The question of which descriptors should be considered is moot, since the neural network will automatically construct a feature representation of the input that is fitting for the task at hand. For the same reason we also do not have to expend any effort on ensuring that the correct parts of two objects are compared stylistically in a meaningful manner.

Contribution Our contribution can be summarized as follows:

- We explore deep metric learning techniques for perceived style similarities of 3D shapes. Our training datasets are significantly smaller than the ones typically used for computer vision tasks.
- We show that rendered images of 3D geometry from multiple viewports are an appropriate representation and how salient views can be selected.
- We propose a triplet sampling method, that does not rely on style class labels and allows for an efficient learning procedure.
- We show how heterogeneous data sources in the form of 3D geometry and annotated photographs found online can be integrated into our deep metric learning method. This has the benefit that we can easily extend our training set with little effort.

2. Related Work

In the following we give a brief overview of geometry based style similarity learning and recent advances in deep metric learning.

Geometric Based Style Similarity Learning Recent investigations in style similarity learning for 3D models are based on geometric object features. The models are compared by quantifying surface properties with various feature descriptors, which allow the computation of a style metric between objects.

To learn this metric a supervised method is applied, which takes as input triplets that consist of a query, a shape that is similar and one that is dissimilar stylistically to the query shape respectively. Lun et at. [LKS15] propose a weighted combination of feature descriptor distances as a style metric for 3D models. Their computation is based on the salience and geometric similarity of patches that are extracted from the object surface. Weights for the saliency and similarity terms are learned by triplet based supervised learning.

Liu et al. [LHLF15] present an approach on supervised learning of a compatibility function for 3D furniture models. They divide the 3D objects into parts on which they specify geometric descriptors, giving feature vectors of different size for each class of models. To define a distance metric on these features they learn an embedding matrix for each class by making use of a set of labeled triplets.

In this paper we present an approach which is independent of handcrafted feature descriptors, which saves both preprocessing time and the selection of appropriate surface features. We are also not reliant on finding good segmentations of the shapes for comparison. Furthermore, due to the image based approach we are independent of mesh resolution, allowing to process models of high complexity and even include information from photos where the underlying 3D model is not accessible.

Deep Metric Learning A large body of work exists on categorization of 2D images based on deep metric learning. Usually a deep neural network learns a lower dimensional embedding of the 2D images based on labeled data. The distance between images is evaluated in this embedding space.

Cui et al. [CZLB15] developed an approach to classify images using deep metric learning with humans in the loop, which decide whether the images were labeled correctly. A challenging task in neural network training is the selection of appropriate samples (e.g. triplets). Due to the human classifier hard negatives are reliable, yet tedious to obtain.

Hoffer et al. [HA15] describe a triplet network architecture for deep metric learning. They achieve competitive results on common data sets for image classification. Wohlhart and Lepetit use a triplet network for pose estimation on RGB and RGB-D data [WL15]. Similar methods have also been used to describe image similarity. Wang et al. learn an embedding of images with a neural network architecture in order to provide fine-grained classification for objects [WSL*14]. They also use a triplet based learning approach. Bell and Bala [BB15] present a method to learn visual similarity of products, to find further appearances of this or a similar product. The parts of the image where the products appear have to be preselected.

Frequently, deep neural networks with siamese or triplet architecture have also been applied to the problem of face verification, alignment, and recognition. These methods are either based on image features, e.g. [HLT14, SCWT14] or get raw image pixels as input, e.g. [TYRW14, SKP15]. Recent results in face recognition show very high accuracies, e.g. [SKP15], which is also based on a triplet network.

Since neural networks have the ability to achieve high accuracies on raw image data, we can make use of deep metric learning techniques to learn the abstract notion of style similarity of 3D models.

Multi-View CNN Using rendered images of 3D shapes as input to neural networks has been explored before. Su et al. [SMKLM15] present a multi-view CNN architecture. 3D objects are rendered from multiple views and given as input to a set of CNNs. Their output is then combined into a single view via a pooling operation and passed on to a further network. With the resulting descriptors they are able to predict the classes of 3D objects.

Li et al. [LSQ*15] embed 3D shapes as well as 2D images of objects of the same categories into a joint embedding space. Similar to Su et al. they render images of the 3D objects from multiple views, compute Histogram of Gradients descriptors on these images, and compute an embedding space based on these. Furthermore, they train a CNN to map heterogeneous image data into this embedding space.

These multi-view approaches result in very high classification accuracy. Hence, we employ a similar method in our framework to learn style similarity.

3. Problem Statement

Given 3D shape collections with meta-data that specifies if two shapes are stylisticly similar or dissimilar, our aim is to learn a metric using convolutional neural networks that reflects this similarity. This meta-data can be obtained from a user-study (as done by Lun et al. [LKS15]) and is provided in the form of triplets $(x_i, x_i^+, x_i^-) \in X$, where x is the query shape and x^+ and x^- are the similar and dissimilar shapes respectively.

This gives rise to a set of challenges. Firstly, we have to choose a representation of 3D shapes on which to train our neural network. We chose to represent the geometric models with images rendered from cameras positioned around the objects.

Secondly, we have to design our network architecture in such a way that takes the much smaller training data sets for 3D shapes into account compared to typical data set sizes in the field of computer vision.

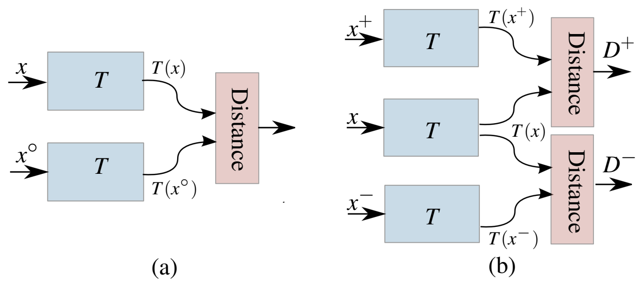

Figure 2: *Examples of siamese and triplet networks (a and b respectively) for metric learning. T stands for a non-linear transformation via a neural network, which can be interpreted as an embedding of the input. Typically, these networks share their parameters, which is equivalent to applying the same T to the inputs. The 'Distance' node compares the embedded input based on some metric, e.g. the euclidean distance. x is a neutral or query sample to which a positive or similar sample x^+ and a negative or dissimilar sample x^- is compared (x° is a sample that can be either positive or negative). Note that the siamese network compares the embedded inputs, while the triplet network compares the relative distances D^+ and D^-.*

Thirdly, we want to restrict ourselves to training with triplets consisting of one image each per element in order to allow the incorporation of real images. This requires selection of salient views of the shapes and the online generation of image triplets.

4. Deep Metric Learning

A neural network with k layers can be interpreted as a successive application of a (non-)linear transformation f_i to some input x,

$$T(x) = (f_k \circ \cdots \circ f_2 \circ f_1)(x). \tag{1}$$

For classification tasks $T(x)$ is then usually fed into a cross-entropy loss function. However, $T(x)$ can also be viewed as an embedding of the input x into some (typically lower dimensional) feature space.

Thus, we can formulate the task of metric learning in the context of this embedding space. Given our input (x_i, x_j) we would like to learn a metric of the embedded input $(T(x_i), T(x_j))$ that correlates with the perceived style similarity of (x_i, x_j), i.e. $\|T(x_i) - T(x_j)\|$ is small if they are embeddings of the same style. Otherwise, the distance between $T(x_i)$ and $T(x_j)$ should be large.

In contrast to direct classification methods with neural networks, that aim to predict the class for each image individually, e.g. the approach by Krizhevsky et al. [KSH12], metric learning requires a network architecture that allows the simultaneous relative comparison between multiple inputs. Typically, siamese or triplet network architectures (see Figure 2) are used in this context.

If the input tuple (x_i, x_j) is ordered in some manner, i.e. x_i is always an image of an upper human body and x_j is an image of a lower body, then the parameters of the siamese or triplet networks are not shared since each network is assigned an input-specific task. However, usually this is not the case. For convolutional neural networks this means that the same filters are learned and applied to each element of the input tuple.

Siamese networks (see Figure 2bluea) are typically trained by minimizing a contrastive loss function

$$\begin{aligned} L^+(x,x^+) &= \left\| T(x) - T(x^+) \right\|_2 \\ L^-(x,x^-) &= \max(0, m - \left\| T(x) - T(x^-) \right\|_2) \\ L(X) &= \sum_i L^+(x_i,x_i^+) + \sum_j L^-(x_j,x_j^-), \end{aligned} \quad (2)$$

where (x,x^+) and (x,x^-) are pairs drawn from X and m is a 'gap' or 'margin' hyperparameter, as introduced by Hadsell et al. [HCL06]. x is a reference (neutral) data sample, while x^+ and x^- are the similar (positive) and dissimilar (negative) samples relative to the reference respectively. L^+ pulls the images of the query and similar samples together in the embedding space. L^- pushes the images of the query and dissimilar samples apart. This effect vanishes once a pair of samples are sufficiently far apart in the embedding space ($\left\| T(x) - T(x^-) \right\|_2 \geq m$). Here the euclidean distance in some feature space is used to compare samples. However, other distances such as the cosine distance can also be applied.

Triplet networks (see Figure 2b) are a more recent neural network architecture used for deep metric learning. Hoffer et al. [HA15] showed that the notion of context when learning (dis-)similarity is vital. An input sample might be similar to another sample only when compared to yet other samples. For example two pieces of furniture might be considered stylistically similar only in comparison to another piece of furniture with a different style. In contrast to siamese networks, triplet networks can capture this context by being trained on triplets (x,x^+,x^-) and not pairs.

Wang et al. [WSL*14] proposed the following loss function for triplets

$$\begin{aligned} D^+(x,x^+) &= \left\| T(x) - T(x^+) \right\|_2 \\ D^-(x,x^-) &= \left\| T(x) - T(x^-) \right\|_2 \\ L(x,x^+,x^-) &= \max(0, m + D^+(x,x^+) - D^-(x,x^-)) \\ L(X) &= \frac{1}{n} \sum_i L(x_i,x_i^+,x_i^-), \end{aligned} \quad (3)$$

where X is the training data consisting of n triplets. Here the gap parameter m has the effect that if the distance from an embedded dissimilar sample to the embedded query D^- compared to the distance of an embedded similar sample to the embedded query D^+ is smaller than m, the embedding of the similar sample will be 'pulled' closer to the query and the embedding of the dissimilar sample will be 'pushed' further away (see Figure 3). If this margin is not violated, i.e. $D^- - D^+ \geq m$, no forces act on the embedding of this specific triplet and it remains unchanged.

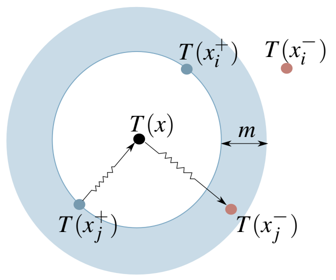

Figure 3: *The black, blue and red dots represent the embedding of elements of two triplets $t_i = (x,x_i^+,x_i^-)$ and $t_j = (x,x_j^+,x_j^-)$. For t_i the difference of the distance of the dissimilar and similar sample to the query is greater than the margin m. Therefore, their embedding is not further changed. This is not the case for t_j. By minimizing the loss function (3) the embedding of the positive sample will be pulled closer and the one of the negative sample will be pushed further away.*

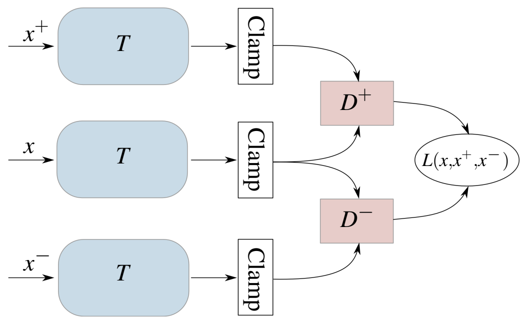

Figure 4: *Our triplet network architecture consists of three convolutional neural networks T with shared parameters. Their outputs are then clamped such that $\|T(x)\|_2 \leq 1$ for any query, positive or negative sample, such that the embedding space is confined to a hyper-sphere. The embeddings of the similar and dissimilar sample are compared to the query in D^+ and D^- respectively. These relative distances are then compared in the triplet loss function (3).*

4.1. 3D Shape Style Similarity Learning via Triplet Networks

In our case the training data originally consists of triplets of 3D shapes and due to advantages stated above we opt for a triplet network architecture. However, we choose to train our triplet network on images of the shapes. This has the advantage is that this representation has weaker consistency requirements compared to 3D shapes. Here, we do not have to ensure that the correct segments of models are matched and compared correctly. This representation of the 3D geometry also makes it possible to make use of the rich annotated databases of real-life images found online. By making use of this heterogeneous data (photos and 3D shapes) we can further expand our training data, leading to better generalization of our network.

Network Architecture In order to learn a metric based on style similarity we make use of such a triplet network architecture. Our architecture for each network (see Figure 4) is inspired by the VGGNet architecture proposed by Simonyan and Zisserman [SZ14]. The construction is such that for an input image with 112×112 pixels the resulting embedding vector will be 512 dimensional. The architecture is defined as: Conv 3@32, Max 2, Conv 3@64, Max 2, Conv 3@128, Max 2, Conv 3@256, Max 2, Conv 1@512, Flatten. Conv $f@n$ stands for a convolutional layer with bias with n filters of size f. Padding is applied to the feature maps so that no size reduction is performed with the convolutional layers. Max are max pooling layers with window size 2 and stride 3. Furthermore, we apply increasing dropout of (0.1, 0.2, 0.3, 0.4) after each pooling layer. Finally, we flatten the output of the last convolutional layer into a vector. We achieved the best results during training with the exponential linear units (ELUs) proposed by Clevert et al. [CUH15] as our non-linear activation function for the convolution layers. Contrary to the results presented by Clevert et al. we found that applying batch normalization, as introduced by Ioffe et al [IS15], after every convolution layer with non-linear activation improved training performance in our case.

Embedding Space Applying each triplet network T to some input image x (of size 112×112 pixels) yields a 512 dimensional embedding $T(x)$. $T(x)$ is often normalized (cf. [BB15]) such that the embedding space is the surface of a hyper-sphere. Ensuring that $\|T(x)\|_2 = 1$ prevents unbounded expansion in the embedding space. However, we found that relaxing this constraint such that $\|T(x)\|_2 \leq 1$ (only vectors with length greater than 1 are normalized) and using the euclidean distance performed even better, resulting in a larger embedding space that consists of the whole hyper-sphere.

Training and Regularization For training we minimize the loss function (3) using the ADADELTA method proposed by Zeiler [Zei12], which adapts the learning rate during gradient descent based on first order information.

Since we are mainly interested in learning the style similarity of 3D objects, we have much smaller training data sets when compared to typical data sets for pure image analysis tasks (e.g. ImageNet [RDS*15]). This emphasizes the need for regularization of the neural network in order to avoid overfitting. We address this issue by applying several established techniques.

First, we generate the data by rendering images of the same 3D shape from multiple camera positions on a sphere around the object. This forces the network to capture the variance of stylistically relevant features from multiple views. These views are captured in 128×128 pixel images. Following Krizhevsky et al. [KSH12] we then randomly crop the images to 112×112 pixels and randomly mirror them along each image axis during training. These transformations (data augmentations) artificially expand the dataset and thereby combat overfitting.

Furthermore, we limit the number of parameters in the network by only using filters of at most 3×3 pixels as proposed by Simonyan and Zisserman [SZ14]. In addition to these steps we also apply dropout after the max-pooling layers as proposed by Srivastava et al. [SHK*14].

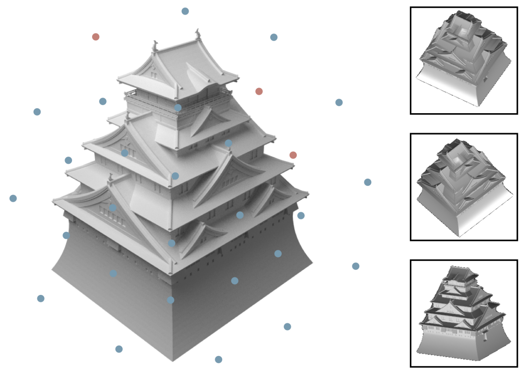

Figure 5: *Here we illustrate our salient view selection. The dots represent camera positioned on a sphere around the shape. The red dots symbolize 3 of the most salient views, estimated with the entropy of the image intensities. The views are shown on the right.*

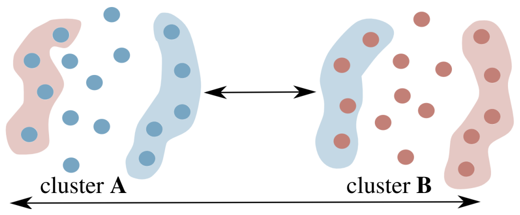

Figure 6: *The blue and red dots represent the embedding of two style clusters A and B. Dots with a red underlay show embedded samples that have maximal distance to each other, while those with a blue underlay have minimal distance to each other. By retaining the underlayed samples for training, we approximately capture the learned boundary and shape of the style clusters in the embedding space.*

We implemented our network with the help of the lasagne library provided by Dieleman et al. [DSR*15].

4.2. Salient View Selection

Ideally, we would like to avoid unnecessary assumptions about the alignment or upright orientation of the input data. However, by capturing images from camera positioned on a sphere around the shapes, we cannot guarantee that every image captures the stylistic details necessary to differentiate between them (see Figure 5).

In order to solve this issue, we take an information theoretic inspired approach. For this we apply an entropy filter to each image, which computes the entropy (over the intensity values) of an image patch, and integrate it over the image. For images with only one intensity value the entropy will be minimal, while it will be maximal if the distribution of the intensity values is uniform. We then select the k (5 in our case) images with the highest entropy, capturing those views which hold the most information.

4.3. Triplet Sampling

While the triplet training data provided by Lun et al. [LKS15] is compact in terms of memory, we have to generate several image triplets for each object triplet. In essence we have to generate tuples of the form $((x, v_i(x)), (x^+, v_j(x^+)), (x^-, v_k(x^-))$, where $v_i(x)$ corresponds to the i-th view of the shape x. A naive triplet generation strategy based on the training data would create data that grows with cubic complexity with the size of the original training data. This is also an issue if triplets have to be generated automatically from multiple categories (elements of the same category are labeled as similar) instead of obtaining them from user studies.

In order to keep training efficient, we have to restrict ourselves in terms of the number of generated triplets. Simultaneously, the triplets should capture the similarity and dissimilarity information as good as possible.

We start out by randomly sampling views for each triplet in the training set until we have reached some specified limit (10000 in our case). Ideally, we would regenerate the triplet training set for each new training epoch. For performance reasons this is not feasible. We have found that regenerating the training set every 10 epochs is good trade-off between exploration of the training set and training time.

At each re-sampling step we retain 50% of the previous training set since we want to ensure that the network does not 'forget' about previously seen data and does not overfit on only a subset of the whole training data. Ideally, we would like to retain those triplets that contain the information about the shape of the different style clusters in the embedding space. However, in our case we do not have any style class labels but only information regarding the relative (dis-)similarity of the data. Therefore, we aim to approximate the style cluster boundaries and thereby retain the regions in the embedding space they cover. Thus, we retain those triplets (x, x^+, x^-) where $d_-^+ = \left\|T(x^+) - T(x^-)\right\|_2$ is either very small or very large (see Figure 6). Concretely, we compute d_-^+ for our previous training set and then partition the data accordingly. Our retained triplets for the current training set are made up of the partitions with the 25% of triplets with the smallest d_-^+ and the 25% of triplets with the largest d_-^+ of the previous set.

In order to efficiently train our network we pick the remaining 50% as follows. We randomly sample views for triplets in our training data and keep them if they are not already included in our current set. Furthermore, we try to only select those samples that violate the gap criteria of our loss function (3) (we allow 10 tries in our experiments), i.e.

$$\left\|T(x) - T(x^-)\right\|_2 - \left\|T(x) - T(x^+)\right\|_2 < m.$$

This can lead to generated triplets that favour one stylistic class, where dissimilar samples never appear as a query or similar elements of a triplet. During training this can lead to a meaningless embedding of these dissimilar samples that is bereft of any context. In order to counteract this effect, we add a triplet for each generated triplet where the negative sample is either neutral or positive.

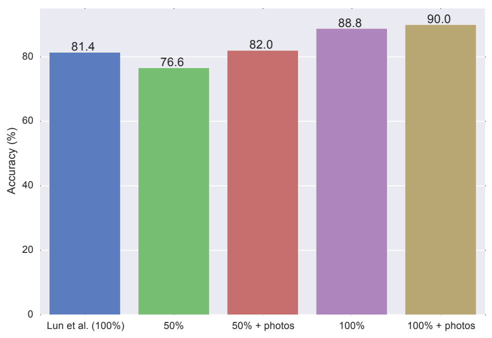

Figure 7: *The results of 10-fold crossvalidation on the building data set with varying training set sizes are shown here. The green bar shows the accuracy obtained by throwing away 50% of all training shapes. The red bar visualizes the results obtained by adding photos to the training data consisting of 50% of the shapes. Similarly, the purple and beige bars show the results obtained with 100% of the training shapes without and with photos respectively. Note, that the test set sizes remained unchanged.*

4.4. Heterogeneous Training Data

In order to expand our training data we want to incorporate data from heterogeneous sources. In our case, we want to make use of the vast image repositories found online that often contain stylistically relevant annotations.

A simple search query, e.g. 'gothic church exterior', often suffices in finding extra annotated data. With relatively small effort we can then specify which 3D shapes are stylistically similar to the found image style classes (Gothic architecture for example). Furthermore, we also have the information that images from different style classes, e.g. Romanesque and Baroque architecture, are stylistically dissimilar.

Making use of this information we extend our online triplet sampling approach as follows. In addition to the generated triplets of rendered images of the shapes, we generate triplets of photos based on their style classes. For this we randomly draw two images from one class as the query and similar sample and a third image from any other class as the dissimilar sample. In order to ensure a good coverage of the training data we generated triplets such that every class is compared to every other class. Finally, we put the 3D shapes and photos into relation with each other by generating triplets that contain both photos and shapes based on a pre-determined similarity of some of the shapes to the photo style classes.

For our triplet sampling we also apply the previously described step of preferring triplets that violate the gap criteria. In this way we expand our training set such that one half consists of rendered images and the other half incorporates photos.

Category	our (salient views)	our (upright views)	our (heterogeneous)	Lun et al. [LKS15]
building	86.4%	88.8%	**90.0%**	81.4%
coffee set	86.2%	89.2%	-	**90.6%**
column	97.7%	**98.0%**	-	90.2%
cutlery	81.2%	81.2%	-	**85.8%**
dish	90.4%	**90.8%**	-	89.5%
furniture	82.3%	86.2%	-	**91.4%**
lamp	86.5%	88.5%	-	**95.0%**
average	87.2%	89.0%	-	**89.1%**

Table 1: *The results of the 10-fold crossvalidation on the data set provided by Lun et al. [LKS15]. Our results are shown in the first three columns. The first column presents the results obtained with the salient view selection presented in Section 4.2. For the second column we assume a given upright orientation and render images only from camera positions that 'look down' on the shapes. The third column shows our results when we add photos found online to the building data set (upright views).*

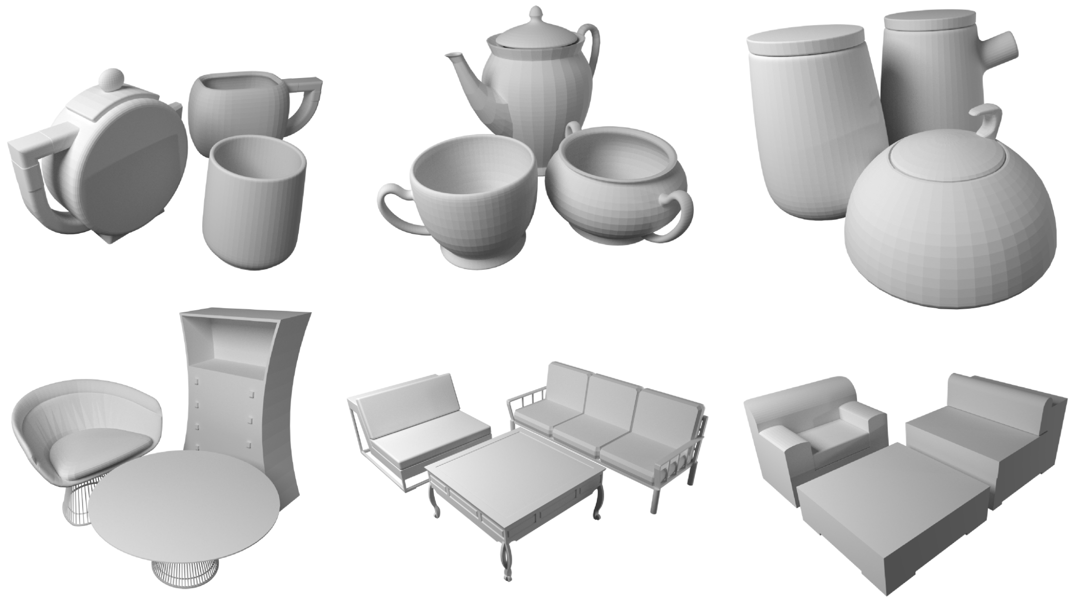

Figure 8: *Some examples of stylistically similar shapes from the coffeeset and furniture data sets are shown here (3 groups each). We constructed these groups by selecting a query object and then finding shapes of a different category (e.g. table and sofa) that are located close by in the learned embedding space.*

5. Evaluation

We evaluated our method on the dataset of shapes from many different categories with triplets acquired via a user study provided by Lun et al. [LKS15]. For this we measured the accuracy of our network correctly identifying the similar and dissimilar elements in the set of test triplets. Following Lun et al. we used 10-fold crossvalidation to test our trained network and selected the same triplets as they did.

For training we used the same set of hyper-parameters for all data sets. For ADADELTA we used the parameters suggested by the paper [Zei12] (learning rate: 1, $\varepsilon : 1 \times 10^{-6}$, $\rho : 0.95$). We trained the network to convergence, which took about 130-140 epochs for each fold of the crossvalidation. On a computer with a i7-6700K CPU and a NVIDIA Titan X GPU this took around 70 minutes for each fold.

Furthermore, we tested the benefit of using heterogeneous data for style similarity learning. In this setting we focused on the building data set and downloaded 446 annotated photos found online with simple search queries (e.g. 'orthodox church exterior') spread over 5 stylistic classes (Baroque, Gothic, Orthodox, Romanesque, and East-Asian). We then identified to which style class a 3D shape is most similar for 80 buildings. For testing we used 10-fold crossvalidation.

In order to perform a stress test for our approach, we dumped 50% of the 3D training shapes while retaining the same test set. By making use of heterogeneous data sources we are still able to beat the accuracy achieved by Lun et al. on the full training set as shown in Figure 7. In the case where we use all 3D models available in the training set and the photos found online we managed to achieve

90% accuracy. A joint embedding of shapes and photos for the building data set is shown in Figure 1.

Please see our results of training the network only on rendered images of the shapes in Table 1. In the first column we present our results obtained by using our salient view selection during triplet sampling. In the second column we show the results obtained with our network, where we assume an upright orientation. In this setting we capture 60 images of each shape with cameras positioned between 20° and 70° in relation to the up vector (up is 0°). For the building data set we also compared our triplet sampling method (Section 4.3) with a simple random sampling of triplets. In comparison to our triplet generation approach (88.8%) random sampling only managed to achieve 78.5%. A naive exploration of the triplet space leads to the network overfitting on a randomly chosen subset of triplets as expected. The third column shows the results if we use heterogeneous data sources for the building data set. Examples of groups of stylistically consistent shapes selected with our trained networks are shown in Figure 8.

Overall we obtain competitive results to Lun et al. For the building, column and dish data sets we achieve higher accuracy than Lun et al. However, if there are many views of a shape that hide stylistically relevant content we do not perform as well as Lun et al., where they can benefit from the use of hand-crafted geometric descriptors.

While we do not achieve better accuracies on all datasets compared to Lun et al., we have presented a fully automatic method, which does not rely on any special descriptors. We have also shown that improved performance with the addition of photos can be expected as we have demonstrated in the case of the building dataset.

6. Conclusion

We have presented a competitive approach for style similarity learning of 3D shapes using deep metric learning. It comes with the benefit of not relying on explicitly pre-computed descriptors. We also avoid the issue of having to choose and then weigh the relative importance of each feature descriptor.

Furthermore, our chosen representation has less requirements with respect to the consistency of the input data. We are also robust to great variations of the overall shape and structure. Since objects with the same style can have varying geometric shapes, this is vital for style similarity. Using rendered images also allows us to make use of the vast information contained in image data sets found online. We showed that by tapping these data sources we were able to improve style identification performance in the setting of comparably small 3D shape collections. This is of benefit for shape data sets, which can be extended with photos requiring only little annotation effort.

An interesting future direction would be to alleviate the problem of stylistic content being hidden from certain views of the shapes. For this, a neural network based selection of salient views might be of interest. This would allow us to move beyond the entropy motivated selection towards a task driven one.

In summary, we presented a fully automated method without the need for expensive pre-preprocessing. Also, we do not require any expert knowledge on the selection and computation of specific geometric feature descriptors. Hence, we believe that our method can be especially useful for applications ranging from supporting 3D artists in building stylistically consistent scenes to recommendation systems that make use of heterogeneous data representations.

Acknowledgements

We thank Lucas Beyer for the fruitful discussions. Furthermore, we thank Alexander Dielen for help with the implementation. Last but not least, we would like to thank Jan Möbius for creating and maintaining the geometry processing framework OpenFlipper [MK12] as well as the reviewers for their insightful comments.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n° [340884].

References

[BB15] BELL S., BALA K.: Learning visual similarity for product design with convolutional neural networks. *ACM Trans. Graph. 34*, 4 (July 2015), 98:1–98:10. 3, 5

[CUH15] CLEVERT D.-A., UNTERTHINER T., HOCHREITER S.: Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015). 5

[CZLB15] CUI Y., ZHOU F., LIN Y., BELONGIE S. J.: Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. *CoRR abs/1512.05227* (2015). 2

[DSR*15] DIELEMAN S., SCHLÃIJTER J., RAFFEL C., OLSON E., SÃŸNDERBY S. K., NOURI D., MATURANA D., THOMA M., BATTENBERG E., KELLY J., FAUW J. D., HEILMAN M., DIOGO149, MCFEE B., WEIDEMAN H., TAKACSG84, PETERDERIVAZ, JON, INSTAGIBBS, RASUL D. K., CONGLIU, BRITEFURY, DEGRAVE J.: Lasagne: First release., Aug. 2015. 5

[HA15] HOFFER E., AILON N.: Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92. 2, 4

[HCL06] HADSELL R., CHOPRA S., LECUN Y.: Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on* (2006), vol. 2, IEEE, pp. 1735–1742. 4

[HLT14] HU J., LU J., TAN Y.-P.: Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1875–1882. 3

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015). 5

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105. 3, 5

[LHLF15] LIU T., HERTZMANN A., LI W., FUNKHOUSER T.: Style compatibility for 3d furniture models. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 85. 2

[LKS15] LUN Z., KALOGERAKIS E., SHEFFER A.: Elements of style: learning perceptual shape style similarity. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 84. 2, 3, 6, 7

[LSQ*15] LI Y., SU H., QI C. R., FISH N., COHEN-OR D., GUIBAS L. J.: Joint embeddings of shapes and images via cnn image purification. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 234. 3

[MK12] MÖBIUS J., KOBBELT L.: Openflipper: An open source geometry processing and rendering framework. In *Curves and Surfaces*, Boissonnat J.-D., Chenin P., Cohen A., Gout C., Lyche T., Mazure M.-L., Schumaker L., (Eds.), vol. 6920 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2012, pp. 488–500. 8

[pix] Pixabay. https://pixabay.com/. Accessed: 2016-04-01 Published under Creative Commons CC0. 1

[RDS*15] RUSSAKOVSKY O., DENG J., SU H., KRAUSE J., SATHEESH S., MA S., HUANG Z., KARPATHY A., KHOSLA A., BERNSTEIN M., BERG A. C., FEI-FEI L.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252. 5

[SCWT14] SUN Y., CHEN Y., WANG X., TANG X.: Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems* (2014), pp. 1988–1996. 3

[SHK*14] SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R.: Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958. 5

[SKP15] SCHROFF F., KALENICHENKO D., PHILBIN J.: Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). 3

[SMKLM15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E.: Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 945–953. 3

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014). 5

[TYRW14] TAIGMAN Y., YANG M., RANZATO M., WOLF L.: Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708. 3

[WL15] WOHLHART P., LEPETIT V.: Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3109–3118. 3

[WSL*14] WANG J., SONG Y., LEUNG T., ROSENBERG C., WANG J., PHILBIN J., CHEN B., WU Y.: Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1386–1393. 3, 4

[Zei12] ZEILER M. D.: Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012). 5, 7