UVStyle-Net: Unsupervised Few-shot Learning of 3D Style Similarity Measure for B-Reps

Peter Meltzer\textsuperscript{1,3}, Hoorman Shayani\textsuperscript{1,3}, Amir Khasahmadi\textsuperscript{1}, Pradeep Kumar Jayaraman\textsuperscript{2}, Aditya Sanghi\textsuperscript{1}, and Joseph Lambourne\textsuperscript{1}

\textsuperscript{1}Autodesk AI Lab, \textsuperscript{2}Autodesk Research, \textsuperscript{3}UCL

Abstract

Boundary Representations (B-Reps) are the industry standard in 3D Computer Aided Design/Manufacturing (CAD/CAM) and industrial design due to their fidelity in representing stylistic details. However, they have been ignored in the 3D style research. Existing 3D style metrics typically operate on meshes or point clouds, and fail to account for end-user subjectivity by adopting fixed definitions of style, either through crowd-sourcing for style labels or hand-crafted features. We propose UVStyle-Net, a style similarity measure for B-Reps that leverages the style signals in the second order statistics of the activations in a pre-trained (unsupervised) 3D encoder, and learns their relative importance to a subjective end-user through few-shot learning. Our approach differs from all existing data-driven 3D style methods since it may be used in completely unsupervised settings, which is desirable given the lack of publicly available labeled B-Rep datasets. More importantly, the few-shot learning accounts for the inherent subjectivity associated with style. We show quantitatively that our proposed method with B-Reps is able to capture stronger style signals than alternative methods on meshes and point clouds despite its significantly greater computational efficiency. We also show it is able to generate meaningful style gradients with respect to the input shape, and that few-shot learning with as few as two positive examples selected by an end-user is sufficient to significantly improve the style measure. Finally, we demonstrate its efficacy on a large unlabeled public dataset of CAD models. Source code and data are available at github.com/AutodeskAILab/UVStyle-Net.

1. Introduction

B-Reps are the de facto standard for industrial design, and the representation most widely used in the consumer product and automotive industries where style is of great importance. B-Reps offer unparalleled editability in a compact, memory efficient representation, they are not discrete/sampled (as per mesh/point cloud) offering precise boundaries with continuous smooth surfaces/edge curves.

Figure 1: Overview of UV-StyleNet: Grams of activations are normalized and extracted for each layer. The weights applied to each layer define the meaning of style. (a) Top-10 query results using uniform layer weights $w$ (b) Top-10 query results using $w^\ast$ based on the user-selected examples (positive in green, negative in red). In this example, $w^\ast \approx [0, 0, 1, 0, 0, 0, 0]^T$. Zoom to see fillets/stylistic details.

9690
Figure 2: Lower case examples from font ‘Viaoda Libre’. While ‘j’ and ‘r’ share some stylistic features, they are not obviously similar to ‘c’, ‘s’ or ‘z’, i.e. font classes provide a ground truth for style compatibility (as perceived by their designers) yet only a weak label for style itself.

See Appendix A for a brief introduction to B-Reps. A B-Rep style similarity measure has many use cases, i.e. finding architectural parts that are in-keeping with the style of a building, or selecting parts for a car that fit with the manufacturer’s existing range. Moreover, the gradient of a style similarity measure can be used to generate helpful visualizations or modify the input 3D shape `a la Gatys et al. [11].

Geometric style is inherently subjective and may have a different meaning in different object class domains, i.e. the boundary between style and content is unclear. For example, in the context of chair designs, number of legs could be considered either style or content depending on the particular use case. Thus, an effective geometric style measure must cater for these different interpretations of the end user.

While existing methods use hand-crafted features [24, 23] or crowd-sourcing [21, 25, 28, 26] to pre-define and measure geometric style, we propose a user-defined few-shot style metric learning method that leverages the range of style signals available in the activations of a pre-trained 3D object encoder through second order statistics (Gram matrices). The relative importance of each layer’s Gram matrix is then learnt through selection of just a few examples of what style means to an end user (see Figure 1).

Despite the abundant use of B-Reps in industrial settings, there is a fundamental lack of publicly available B-Rep data for training machine learning models — in particular, there are no existing B-Rep datasets that include a reliable ground truth for style. To overcome this challenge, we provide an adaptation to SolidLetters [16], which improves the style consistency within font classes for the evaluation test set. The font classes, however, still provide only a weak label for style (see Figure 2), and as such we propose an unsupervised method and use the font labels purely for quantitative evaluation to justify design choices of our method. For comparison against existing SOTA on real-world data we also provide evaluation with the unlabeled ABC dataset [20] of CAD models and a manually labeled subset of it.

The main contributions of this work are as follows:

- We introduce a general few-shot learning method for capturing a subjective end-user’s definition of 3D style and demonstrate its effectiveness on B-Reps
- We show quantitative efficiency and performance advantages of using UVStyle-Net architecture with B-Reps over similar approaches on meshes and point clouds using a new synthetic public dataset (SolidLetters) and a small subset of ABC labeled for style
- We verify our method on the ABC dataset with no style or content labels for pre-training, and demonstrate the effectiveness of our few-shot learning process to capture subjective user-defined style similarity measures

In summary, we introduce a geometric style similarity measure for 3D solids that may be used in completely unlabeled settings for arbitrary object classes, with user subjectivity handled by few-shot learning given only a few examples. While our method is adaptable for all 3D input types, we demonstrate the benefits of our approach with B-Reps both quantitatively and qualitatively.

2. Related Work

Geometric Feature Learning. Geometric feature learning has seen many successes for both Euclidean representations, i.e. multi-view [32], projections [7], volumetric [38], and non-Euclidean representations, i.e. point clouds [35, 30, 13] and mesh [14, 10]. For a detailed review of geometric feature learning we refer the reader to [5, 12, 1]. Despite the prevalence of B-Reps in industrial and creative design applications, however, geometric feature learning for parametric representations remains largely unexplored.

In addition to their wide use, there are many advantages to working with B-Reps as 3D geometric representations. Not only do B-Reps typically require less memory than point clouds or meshes (depending on the sampling resolution/detail of the model), but they also provide richer information about a solid, including the precise boundaries of every surface and the topology of these surfaces.

The benefits of B-Reps over discrete representations are demonstrated in Jayaraman et al. [16], where each face is sampled uniformly in its parameter domain to form a regular grid then passed through a 2D CNN. The CNN face representations are then fed to a Graph Neural Network (GNN) which uses the face adjacency matrix of the original B-Rep.

Geometric Style Similarity. Existing geometric style similarity learning methods are typically trained in a supervised setting, requiring a set of hand-labeled triplets in which one pair is believed to be closer in style than the other [23, 24, 21, 25, 26, 28]. To account for style subjectivity, examples are labeled through crowd-sourcing methods and thus result in a generally accepted definition for style. For example, Liu et al. [23] use hand-crafted features (i.e.
curvature histograms) with a supervised triplet loss to learn furniture compatibility, while Lun et al. [24] apply a similar method by first segmenting input models into sub-parts to compute geometric features for independently.

For geometric style feature learning, Lim et al. [21] and Pan et al. [25] project 3D meshes into multiple 2D views which are fed into a triplet image CNN. Polania et al. [28] adopt a similar approach, where the learned style representations are then passed to a GNN for compatibility prediction. Rendering 3D solids into 2D (even with multiple views) is problematic since stylistic features can be lost or occluded and selecting the best views without making assumptions on the orientations of the data is non-trivial. Pan et al. [26] overcome this using curvature-guided sampling directly from the solids to generate element-level style features which are then aggregated to global style representations using a triplet network.

The reliance of these methods on crowd-sourced, hand-labeled style triplets creates two problems: Firstly, there is limited labeled data available in the 3D style domain, and no labeled B-Rep data. Secondly, and more importantly, the definition of style (an inherently subjective concept) is pre-defined according to a consensus, hence may not be compatible with an end-user’s particular taste or application.

**Style Transfer.** Contrary to the geometric style learning methods above, the style transfer literature has largely adopted the use of first and second order activation statistics from deep pre-trained image classifiers in order to represent and quantify style. Gatys et al. [11] showed that feature co-occurrence in the different layers of a CNN effectively captures elements of style at different scales of abstraction. In the finest layers where features are most local, the style representation given by the Gram matrix captures colour and texture information, yet deeper into the network, the Gram matrices capture higher level structure and patterns eventually crossing into semantic content. Following from this, Huang et al. [15] and Babaeizadeh et al. [4] demonstrate that first order activation statistics (channel-wise mean and variance) can also capture elements of style through the use of Adaptive Instance Normalization (AdaIN). Karras et al. [18] illustrate the relationship between layer depth and the style/content trade-off by swapping the inputs to a generator at varied depth. Swapping at lower layers renders image interpolations of low level texture/colour information, and swapping at deeper layers interpolates semantic content.

Many further works utilise and extend the use of first order statistics of network activations to improve style transfer results, e.g. GAN based methods [19, 37, 17]; however, these methods rely on a generator to align the activations to these statistics while generating an output image, mainly focusing on the quality of the output images rather than the interpretability of the statistics in defining an explicit style distance metric for arbitrary inputs. To explicitly disentangle style and content for arbitrary inputs [27] proposes an auto-encoder that adopts the technique of swapping inputs at various layers and a GAN based encoder and discriminator that is able to effectively separate structure and texture.

Azadi et al. [3] propose a few-shot learning approach for font style transfer in which stacked conditional GANs are used to generate unseen characters in a target style from a small number of observed examples. This method is, however, specific to font generation and relies on supervised pre-training using the style labels.

**3D Style Transfer.** Recently, Liu et al. [22] showed that style could be learned from one mesh model and transferred to another using a neural subdivision surface scheme. Cao et al. [6] generalised the second order statistics approach of [11] to 3D point clouds, adopting the use of a Pointnet [29] encoder pre-trained for classification on ShapeNet [8]. Following the trend of 2D style transfer Segu et al. [31] extend this work using GAN methods to produce a generative model with better disentanglement of content and style. There are no existing style transfer/unsupervised approaches to style metric learning for B-Reps.

### 3. UVStyle-Net

Inspired by image-based style-transfer, our method uses second order statistics of the activations from a pre-trained B-Rep encoder to form a flexible style representation.

For the encoder we use UV-Net [16], which processes each face of a solid with 3 layers of 2D convolutions, and propagates the projected pooled features of each face in a face-adjacency graph using 2 GIN [36] layers. Each face is represented by a $10 \times 10$ grid (image) of 7 dimensions containing the absolute 3D position (xyz) of each UV sample, the normal for each sample, and a mask indicating whether each sample lies within or outside of the trimmed face. We use UV-Net due to its SOTA performance on B-Rep classification and its parallels to conventional 2D CNNs.

For B-Rep model $x$, we extract the normalised, flattened upper triangle of the Gram matrix (modelling the feature correlations [11]) for each layer $l$:

$$G_l(x) = \text{triu} \left( \phi_l(x)^T \phi_l(x) \right)$$

where $\phi_l(x) \in \mathbb{R}^{d_l \times N_l}$ is the normalised feature map of a pre-trained classifier given input $x$ such that $\phi_l(i,x)$ is the normalised activation of filter $i$ at position $j$ in layer $l$, $d_l$ and $N_l$ are the number of distinct filters and non-masked samples in layer $l$ respectively, and triu : $\mathbb{R}^{d_l \times d_l} \rightarrow \mathbb{R}^{d_l(d_l+1)/2}$ returns the flattened upper triangle of a matrix.

For the first (features) layer, samples corresponding to the positions that do not lie on the surface of a trimmed face are masked, and the gram matrix is calculated accordingly. In the GIN layers, we have a single vector per face (i.e. node), thus instance normalization [33] is applied across
the solid prior to computing the Grams. For each of the features (non-masked positions and normals) and activations of each convolution layer’s filters, we leverage the grouping of samples into faces which is unique to B-Rep (compared to meshes and point clouds), whereby we re-center (subtract the mean of) the UV samples by face. This can be interpreted as per-face instance normalization without division by the standard deviation.

Face re-centering/instance normalization are applied to the activations after extraction from the encoder, but the raw (un-normalised) activations are passed to the next layer of the encoder, thus imposing no requirements on the encoder architecture in terms of normalization strategies.

Analogous to style-transfer with 2D images [11], for a pair of B-Rep a and b we define the style distance:

\[ D_{style}(a, b) = \sum_{l=1}^{L} w_l \cdot D_l(a, b), \tag{2} \]

where

\[ D_l(a, b) = 1 - \frac{G_l(a) \cdot G_l(b)}{\|G_l(a)\| \|G_l(b)\|} \tag{3} \]

and \( w \) is a weight vector that controls how much each layer contributes to the style distance measure. We deviate from Gatys et al. [11] in use of the cosine distance (rather than Euclidean) due to simplified normalization and an observed improvement in our initial experiments.

Given a set of user-selected examples from a target style (i.e. positive samples) \( T \), and a set of user-selected counter-examples (i.e. negative samples) \( T' \), we define the user-defined loss:

\[ L_{user} = \sum_{l=1}^{L} w_l \cdot E_l \tag{4} \]

where

\[ E_l = c_1 \cdot \sum_{t_i, t_j \in T \land i \neq j} D_l(t_i, t_j) - c_2 \cdot \sum_{(t, t') \in T \times T'} D_l(t, t') \tag{5} \]

is a layer-wise energy term, \( c_1 \) and \( c_2 \) are normalization constants, and to prevent trivial solutions \( w \) is constrained such that \( \sum_{l=1}^{L} w_l = 1 \) and \( w \geq 0 \). Due to these constraints, we note that even with only positive examples \( T \) (i.e. \( T' = \emptyset \)), \( E_l \) is sufficiently determined, and in such a case the second term may be omitted. However, to reduce the risk of overfitting, a large number of negative examples may be drawn randomly from the remaining dataset. This is of particular benefit in real-world settings without access to labeled datasets, where an end user may select only a handful of positive examples that share style as they perceive it.

We find the optimal weights for an end-user

\[ w^* = \arg \min_w \sum_{l=1}^{L} w_l \cdot E_l \tag{6} \]

subject to the above constraints, and substitute them into Eq. (2) to produce the final user style distance metric.

We observe that \( E_l \) is constant w.r.t. \( w \), thus Eq. (6) is simply a linear combination and its intersection with the hyperplane \( \sum_{l=1}^{L} w_l = 1 \) results in a twice-differentiable convex optimization which we solve using Sequential Least Squares Quadratic Programming (SLSQP) [34].

**Intuition Behind the Grams.** At 0-feats (inputs), the Gram matrix models the distribution of the position and surface normal of the sampled points with 2nd order statistics. Based on the CNN receptive fields, we understand that the first layer Gram is modeling the distribution of local curvatures (i.e. flat/saddle/doubly curved), and the next levels capturing those of higher order curvatures (i.e. s-shaped), then leading into correlations of patterns of these lower level features, and eventually into content.

## 4. Experiments & Results

We first test if a method similar to image style approaches can capture 3D style, and quantify the presence of this signal at each layer. We evaluate our method for disentanglement of style from content with a gradient visualization, thus demonstrating a practical use-case in which a designer may utilize the feedback from the model. We then test few-shot learning of our style metric in its ability to capture an end-user’s subjective requirements. Finally, we assess the effectiveness of our completely unsupervised encoder pre-training approach without content labels.

For data, we start with SolidLetters [16], which is a collection of extruded letters from a variety of fonts including labels for both content (i.e. letter class) and style (i.e. font class) (Table 1). This is a good choice of data for initial validation of our design decisions, as the 2D nature of the elements of style in 3D shapes simplifies the analysis and debugging while the generation process of these 3D letters mirrors the most typical CAD modeling approach — drawing a 2D wire body, then extruding to 3D and potentially filleting/bevelling the edges. Following this, we use the real-world ABC dataset [20] of CAD models.

In all cases on SolidLetters we pre-train the classifier on the training set to predict the letter, and perform model selection for the best classifier with the validation set. Following the methodology of Cohen et al. [9] and Jayaraman et al. [16], we perform pre-training using 26 classes (combining upper and lower case examples). SolidLetters includes randomness in the fillet size, and extrusion depth and angle, so for the held-out test set used in all our evaluations, we
regenerate the letters to remove sources of randomness (extrusion angle/amount and fillet radius) within font classes, hence strengthening the style labels. For further detail, see Appendix E. After pre-training, all experiments are performed using the held-out test set. We note in particular that no examples of the test fonts are included in the training/validation sets, and that font style labels are used purely for evaluation and not during pre-training. For all experiments on ABC, we perform unsupervised pre-training using point cloud reconstruction on the complete dataset.

Table 1: Details of SolidLetters dataset [16]. The test set is regenerated without sources of randomness within font classes to strengthen the associated style labels used for evaluation.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td>40,402</td>
<td>10,100</td>
<td>13,339</td>
</tr>
<tr>
<td>Letter Classes</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Font Classes</td>
<td>1,664</td>
<td>1650</td>
<td>378</td>
</tr>
<tr>
<td>Random Extrude/Fillet</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

For comparison with other representations and encoders, we use MeshCNN [14], and Pointnet++ [30]. We use Pointnet++ over DGCNN [35] or Pointnet [29] since we are drawing upon 2D style literature. DGCNN aggregates intermediate layer activations according to locality in feature space rather than coordinate space, and Pointnet does not perform hierarchical pooling, thus Pointnet++ is a closer point cloud generalization of the 2D CNN approach used in [11]. In mesh and point cloud representations, there is no information regarding local grouping of samples, thus it is not possible to apply face-wise re-centering, so we use instance normalization for the extracted activations throughout.

For comparison against SOTA, knowing of no existing unsupervised B-Rep style learning methods, as a baseline we use the geometric style embedding of PSNet [6], without the colour inputs, which we refer to as PSNet*. PSNet performs geometric and colour style transfer on point clouds without surface normals. Its architecture allows us to pre-train its encoder using point cloud reconstruction in a completely unsupervised settings rather than content classification as proposed. See Appendix F for further details.

4.1. Measuring Style Signal

We adopt the Linear Probe methodology [2] to measure the amount of style signal present in the Gram matrices of each layer of the pre-trained network. We train a linear classifier on each layer’s Gram matrix $G_l$ with ground truth font labels on a subset of the SolidLetters test set. We select four visually distinct fonts in order to strengthen the style labels with respect to style over style compatibility (see Figure 2), and due to many fonts in the test set containing almost identical variants. Each encoder is pre-trained with only letter classes as labels, and the four test fonts used in this evaluation are previously unseen. Since the dimensions of the Gram matrices are very large (i.e. in some cases $> 2^{19}$), but we have only 137 examples, we perform logistic regression with L2 regularization and 5-fold cross-validation to prevent overfitting. We report the mean validation accuracies.

Figure 3 shows the mean validation accuracy using the extracted Gram matrices from each layer in all four pre-trained models. Compared to random baseline at 0.25, we
observe significant indication of style being present in the signals extracted from all layers (including features) for all models. For UV-Net we see the greatest amount of style information in the lowest layers, with the signal reducing deeper into the network. This aligns with our assumption that second order activation statistics transition from style to content representations as network depth increases, as shown for 2D images in [11, 18].

For a qualitative evaluation of our design choices, we perform a top-k query for an example from each font distributing all weight uniformly over the first \( \frac{1}{2} \) layers. As shown in Figure 4, with this particular style definition, the style features provided by the pre-trained Pointnet++ model suggest the ‘Z’ from another font is close in style to the query ‘L’, while all UV-Net query results match the target font, and MeshCNN makes only one less obvious mistake. PSNet* has the highest number of errors.

We hypothesize that this result may be partly due to the sampling strategy of each method. As Figure 5 illustrates, UV-Net samples a fixed size grid for each face, thus large faces (such as the long diagonal stem of the ‘Z’) will contribute less to the style features extracted than in PSNet* and Pointnet++ where the point cloud is sampled with uniform density. Therefore, the large diagonal faces have larger influence with Pointnet++ features as network depth increases. Lack of a CNN hierarchy and surface normal inputs may explain lower performance of PSNet* vs PointNet++.

Figure 6 shows the top-k query for the same letter ‘L’ using the style distance from single layers (\( l = 0, l \approx \frac{1}{4}, \) and \( l \approx \frac{1}{2} \)). Supporting our hypothesis above, we see that in this particular scenario, the font is better matched by Pointnet++ in the lower layers. Within the first layer of the network, the features extracted will contain more information about low level structure, i.e. bumpy rather than smooth surfaces. Interestingly, for \( l \leq \frac{1}{2} \), MeshCNN performs worst with the features (\( l = 0 \)). We hypothesise this is due to the rotation and scale invariance in the MeshCNN features, whereas UV-Net/Pointnet++ features contain global information.

Finally, comparing with the computational costs of PSNet*, PointNet++, and MeshCNN we observe that the UV-Net encoder with only 645K parameters is 23, 85, and 96 percent faster for style inference, and the Gram matrices require 82, 94, and 35 percent less memory per solid respectively. Full details in Appendix F. Based on the above results and computational costs, we perform further experiments using the UV-Net encoder only.

4.2. Gradient Visualization

In Figure 7 we visualise our proposed pairwise style distance metric for each B-Rep \( x \) by computing

\[
\nabla_{xyz} = \frac{\partial L_{style}}{\partial x_{xyz}} \in \mathbb{R}^{N_0 \times 3}
\]

where \( N_0 \) is the grid size (number of unmasked UV samples), and \( x_{xyz} \) is the absolute positions of the UV samples. For easy interpretation, we plot the vectors \(-k \cdot \nabla_{xyz} \) centered at the samples \( x_{xyz} \) with black lines to indicate the direction in which a UV sample point should be displaced in order to better match the style between the pair, and \( k \) is a constant scaling factor that aids visualization.

In Figure 7 (left) we fix the content and compare different styles. The xyz gradients suggest that the samples of the left example should be moved outwards to match the squarish style on the right, and the samples of the example on the right should be moved inside the solid to match the curves on the left. Figure 7 (right) confirms our approach is able to disentangle style from content, as we compare different content and different style. The gradients on the left example are similar to (left), confirming that the style is matched despite a different content example to compare with.

4.3. Few-shot Learning of User-Defined Style Measure

We evaluate few-shot learning of our user-defined style loss on the complete unseen test set, by measuring the mean Precision@10 for each example from a selected font for a range of number of positive and negative user-selected examples. We evaluate on 6 visually distinct fonts (see Appendix B). Precision@10 is calculated as the proportion of
top-10 neighbors that match the target font. For baseline, we compare against the mean Precision@10 with uniform layer weights (one positive and no negative examples). For computational reasons, we reduce the dimensions of each layer’s representation $G_i$ to $\min(d_i, 70)$ using PCA.

As shown in Figure 2, the font name provides only a weak style label, and as such we are concerned more with the improvement in the mean Precision@10 score than the absolute values. We also consider upper and lower case within the same font as separate labels to further strengthen the associated label, yet also increasing the difficulty of the task as the number of classes doubles to 756.

Positive examples are randomly drawn from the same font and case, and negatives are drawn randomly from all remaining examples. For each number of positives and negatives we perform 20 trials (different positives and negatives each time). We report the mean Precision@10 across all positive examples across all trials, i.e. for each number of positives and negatives, every example of a chosen font is queried and evaluated, and this process is repeated 20 times.

Figure 8 (left) shows the result for a single font, and (right) the mean gain in Precision@10 (ratio to baseline) for a selection of fonts (further results in Appendix B). For all combinations of number of positives and negatives greater than 0, we observe a significant improvement in the mean Precision@10 score over the uniformly weighted baseline. Moreover, since negatives are selected randomly from the remaining dataset, we also confirm that providing only positive examples is sufficient to obtain a significantly improved style measure based on the end-user’s requirements.

### 4.4. Unsupervised Pre-training

The advantage of our method over existing approaches is that it may be used in unsupervised settings. This is particularly important for B-Reps, since there are no publicly available B-Rep datasets with style labels. We evaluate our approach using the ABC dataset, which contains no content or style labels. For the UVStyle-Net/PSNet* encoder pre-training we use an auto-regressive approach with point cloud reconstruction [16]. Again, we reduce the dimensions of the style representations $G_i$ to $\min(d_i, 70)$ using PCA.

Figure 9 shows a few top-5 queries in the style embedding space, using only the lowest 3 layers. For PSNet* queries we use Euclidean distance as this is the metric optimised in [6]. We observe that UVStyle-Net matches surface style with more variation in content, while in many cases PSNet* matches shapes that roughly occupy the same regions in space as the query, i.e. the content. For example, in A: UVStyle-Net finds solids with matching flat surfaces/angles, while PSNet* finds many curved surfaces not present in the query, in C: UVStyle-Net finds more solids with matching curved surfaces, and in E: UVStyle-Net finds blocks with the matching notch style (even with different block size or numbers of notches), whereas PSNet* matches similar sized blocks without the notched style. Comparison of same queries with different UVStyle-Net weights, and PSNet* distance measures are provided in Appendix C.

Evaluation of our few-shot user-defined style measure, Figure 1 shows the nearest neighbour queries for a given target after optimizing the style loss for the user selected examples shown. Selecting filleted solids for positive and a bevelled solid for negative improves the nearest neighbours to the target by pushing away the nearest neighbour of (a) which matches closely in content but not the filleted style. We provide further results in Appendix C.

For quantitative evaluation on a real-world dataset, we use subsets of ABC for which we manually curate style labels (details in Appendix D). For each model we perform logistic regression on the extracted style embeddings from the pre-trained encoders (Grams from all layers concatenated together). Again we train the encoders using point cloud reconstruction on the complete ABC dataset. We perform
5-fold cross validation with L2 regularization and report the mean validation weighted F1 scores for the best parameters, summarised in Table 2 showing UVStyle-Net significantly outperforms PSNet* on all subsets.

<table>
<thead>
<tr>
<th>ABC Subset</th>
<th>UVStyle-Net</th>
<th>PSNet*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat/Electric</td>
<td>0.789 ± 0.034**</td>
<td>0.746 ± 0.038</td>
</tr>
<tr>
<td>FreeForm/Tubular</td>
<td>0.839 ± 0.011**</td>
<td>0.808 ± 0.023</td>
</tr>
<tr>
<td>Angular/Rounded</td>
<td>0.805 ± 0.010**</td>
<td>0.777 ± 0.020</td>
</tr>
</tbody>
</table>

Table 2: Weighted F1 scores for each manually labeled styles subset of ABC. ** indicates 5% statistical significance.

4.5. Ablation

Figure 10 illustrates the impact of face re-centering and instance normalization using the complete SolidLetters unseen test set. Adopting the linear probe methodology as above, we compare the mean classification accuracy of each layer for predicting all fonts using 5-fold cross-validation. While instance normalization is tested on all layers, face re-centering is not possible beyond the third convolution layer since each face is already represented by a single vector. The significantly higher scores in the lower layers (excluding features) confirms our assumption that style transitions into content deeper in the network. We also see empirical justification for the use of instance normalization, and in particular face re-centering, which is not possible when working with meshes or point clouds. Comparison with the UV-Net content embedding shows that any of the layer-wise style representations ($G_l$) as proposed in our method are better suited to capturing style information.

Figure 11 shows the effect of PCA on the layer-wise style representations ($G_l$) to test the significance of style as a source of variation in each layer. Again, we use linear probes to quantify the style information. In line with our assumptions, the lowest layers ($l = 0 \ldots 3$) show the greatest amount of style information when the dimensions are sufficiently low, thus indicating that the font style signals are the most significant source of variance in these layers.

5. Conclusions and Further Work

UVStyle-Net is a 3D style similarity measure for B-Reps which caters to an end-user’s subjective definition of style through few-shot learning based on user selected examples and an unsupervised pre-trained encoder. As a data-driven style measure for B-Reps, which does not require style or content labels yet is adaptable to end-users’ requirements, our approach is unique from all existing methods.

Using the SolidLetters font labels for evaluation, our results have demonstrated the applicability of 2D image style principles and assumptions for 3D shapes, and quantified the advantages of our method with B-Reps over alternative methods on meshes and point clouds. In particular, we have confirmed that second order statistics of 3D encoder activations in the first few layers retain style information as the greatest source of variance. We have also shown that our method generates meaningful style gradients, and that the UV-Net sampling strategy and leveraging the face boundary information unique to B-Reps, particularly through face re-centering, significantly improves the style measure.

For a range of 3D fonts and real-world CAD models, we have demonstrated that our proposed method for few-shot learning of user-defined style is effective in improving the style measure for a specific task, even with a minimal number of positive (and optionally, negative) examples. We also demonstrate the benefits of our approach over an existing SOTA method on the real-world ABC dataset where even content labels are not available for encoder pre-training.

A limitation of our method can be seen when solids have very similar content, thus may be improved by stronger disentanglement of style from content. We hypothesize that other unsupervised methods for the encoder pre-training may capture greater detail in the network activations, and therefore improve the style measure on very similar content. We also observe that the current formulation of the few-shot learning often puts all weight on one layer. For future work, we propose investigation into regularization of the few-shot user loss and further investigation into sophisticated distance measures for comparing feature distributions, as well as the natural next step of B-Rep style transfer.
References


