SketchZooms: Deep multi-view descriptors for matching line drawings

Pablo Navarro, José Ignacio Orlando, Claudio Delrieux, and Emmanuel Iarussi

Abstract—Finding point-wise correspondences between images is a long-standing problem in computer vision. Corresponding sketch images is particularly challenging due to the varying nature of human style, projection distortions and viewport changes. In this paper we present a feature descriptor targeting line drawings learned from a 3D shape data set. Our descriptors are designed to locally match image pairs where the object of interest belongs to the same semantic category, yet still differ drastically in shape and projection angle. We build our descriptors by means of a Convolutional Neural Network (CNN) trained in a triplet fashion. The goal is to embed semantically similar anchor points close to one another, and to pull the embeddings of different points far apart. To learn the descriptors space, the network is fed with a succession of zoomed views from the input sketches. We have specifically crafted a data set of synthetic sketches using a non-photorealistic rendering algorithm over a large collection of part-based registered 3D models. Once trained, our network can generate descriptors for every pixel in an input image. Furthermore, our network is able to generalize well to unseen sketches hand-drawn by humans, outperforming state-of-the-art descriptors on the evaluated matching tasks. Our descriptors can be used to obtain sparse and dense correspondences between image pairs. We evaluate our method against a baseline of correspondences data collected from expert designers, in addition to comparisons with descriptors that have been proven effective in sketches. Finally, we demonstrate applications showing the usefulness of our multi-view descriptors.

Index Terms—Sketching, Line drawing descriptors, Image registration, Deep learning.

I. INTRODUCTION

Humans excel at perceiving 3D objects from line drawings. Therefore, freehand line drawings are still the preferred way for artists and designers to express and communicate shape without needing to effectively materialize a real object. Unlike humans, computers struggle to interpret a 2D sketch as a complex abstraction of our 3D world. For instance, the straightforward task of finding correspondences between a pair of images or an image and a 3D model has been an important problem in computer vision and graphics for decades. Compared to photographs, dealing with sketches is more challenging, as line drawings lack key shape cues like shading and texture, projections are imprecise, and the shapes are often composed by several sketchy lines (Figure 2). Consequently, when a target object is viewed from different angles, traditional image descriptors fail to map similar points close together in the descriptor space.

In this work we present the first data-driven framework to compute local sketch descriptors that can deal with significant changes in style, shape, and viewpoint. The main goal is to capture the domain semantics and object parts characteristics despite the heterogeneous nature of hand-drawn images (see Figure 1). Our main contribution is to adapt recent Multi-View Convolutional Neural Networks (MVCNN) [1], [2] architectures to learn image descriptors that map the local and global drawing context onto a multi-view embedding. In our setup, a query point is represented by a set of 2D views captured from the point’s immediate neighbourhood from different distances. The network then produces feature vectors for each view, aggregating them to construct a single descriptive vector for the point. In our experiments, we empirically show that the proposed approach is able to deal with significant changes in style, shape, and viewpoint, generalizing well to non-synthetic inputs.

To date, finding local sketch correspondences with deep learning techniques is an unexplored research topic. This is
likely due to the fact that learning meaningful and consistent features using such high capacity models requires a large data set of complex line drawings, paired semantically at a dense, pixel-wise level. Our second contribution is to overcome this difficulty by generating a vast collection of more than 5,000,000 synthetic sketches, distributed in four semantic categories: bag, chair, mug, and headphone. To this end, we built on top of the semantic, non-rigid registration method from [3] and render more than 350 3D models using non-photorealistic techniques that mimic design sketches. We capture them from the three main orthographic views commonly adopted by engineers and designers: side, front and 3/4 angle (isometric). Our hypothesis is that learning from such a large database results in a general model that overcomes the covariate shift between artificial and real sketches.

While literature provides descriptors robust to shape variation and affine distortions, we are the firsts to cope with part semantics and 3D viewport changes in sketches. Therefore, we evaluate and compare SketchZooms against well-known state-of-the-art techniques extensively used in line drawings’ applications. Furthermore, the generalization ability of our network is assessed by evaluating the proposed approach using 40 sketches rendered by designers in different styles and viewports. We manually annotate correspondences among images on this data set and without any prior fine-tuning, compute quantitative metrics along with the qualitative results shown in Section [7]. We also evaluate the performance of the method in a significantly larger test set of densely annotated artificial sketches, to assess the robustness of the method to specific changes in a controlled environment. Finally, we demonstrate the usefulness of our descriptors for graphics applications such as sketch-based shape retrieval and image morphing.

II. RELATED WORK

Finding image descriptors that effectively represent image data is a classic problem in computer graphics and vision. A comprehensive summary of such a vast literature is out of the scope of this paper. Instead, in this section we focus on descriptors involving drawings, either for registration or retrieval tasks on images and 3D models. We briefly classify them into two main groups: hand-crafted and learned descriptors.

Hand-crafted descriptors consist in applying a custom transformation over some input data in order to obtain a global or a local representation of it. Many applications working with raster input employ pixel-based descriptors. For instance, ShapeContext [4] is a well known descriptor that captures the distribution of points on a given neighborhood and has proven to be effective for corresponding feature points in sketches [5], [6]. Combined with cycle consistency methods like FlowWeb [7], some authors have boost ShapeContext performance and benefit from the availability of multiple similar sketches [8]. In the context of vector graphics, several authors proposed to quantify stroke similarity in order to generate in-between frames for character animation [9], [10], auto-complete line drawings repetitions [11], selection and grouping [12], [13] and sketch beautification [14], [15]. As the number of available 3D models and images steadily increases, effective methods for searching on databases have emerged. Using non-photorealistic rendering methods, meshes are transformed into sketches and search engines compute image descriptors that summarize global properties, such as contour histograms [16], stroke similarity distance [17], Fourier transform [18], diffusion tensor fields [19] and bag-of-features models [20]. More recently, the highest data availability is giving rise to automatically learned features that offer greater generalization capacity.

Learned descriptors gained great popularity with the recent success of deep neural networks [21]. Applications involving line drawings mostly target the problem of computing global descriptors for sketch-based image retrieval. For instance, Sketch Me That Shoe authors [22], [23] train a convolutional neural network on a data set of annotated sketch-photo pairs using a triplet loss [24]. Alternatively, Qi et al. [25] proposed to train a siamese architecture network that pulls feature vectors closer for sketch-image input pairs labeled as similar, and push them away if irrelevant. Zhu et al. [26] construct pyramid cross-domain neural networks to map sketch and 3D shape low-level representations onto an unified feature space. To close the gap between 2D and 3D data, some authors train CNNs to learn embeddings from sketches and 3D models simultaneously using edge maps [27]. More generally, some authors have investigated how to learn cross-modal representations that surpass sketch images and 3D shapes, incorporating text labels, descriptions, and even depth maps [28], [29]. Other learned descriptors applications include sketch classification and recognition [30], [31]. While these methods target global features that can discriminate high level characteristics in sketches, our goal is to compute pixel-wise descriptors that capture part semantics along with local and global contexts to perform local matching.

With learning methods comes the need for training data sets. The high diversity in style and the difficulty to automate sketch annotation makes it hard to compile massive line drawing data sets. Yet, some authors have shown strong efforts in this direction. Such is the case of Eitz et al. [32], that introduced a data set of 20,000 sketches spanning 250 categories. Similarly, The Sketchy Database [33] ask crowd workers to sketch photographic objects sampled from 125 categories and acquire 75,471 sketches, compiling the first large-scale collection of sketch-photo pairs. More recently,

Fig. 2. Unlike photographs, typical design sketches lack shading, texture, and lines are often rough and incomplete.
**Quick, Draw!** [34] released an open source collection composed by 50 million doodles across 345 categories drawn by players of an online game. Nevertheless, the skills and styles disparities of contributors to these data sets make them unsuitable for our goal. In this work, we target design sketches that are approximately drawn following a particular set of rules [35]. Similar to Wang et al. [36], we exploit an existing shape collection augmented with semantic part-based correspondences data to synthesize sketches with NPR techniques. This allows us to have complete control over sketch parameters such as projection angle and sketchiness level. More importantly, it naturally provides us with 2D/3D alignment, a crucial ingredient to learn our multidimensional features. As we will show in forward sections, using synthetic sketches does not hamper our method from generalizing to sketches in the wild.

### III. Overview

Figure 3 provides an overview of the main components in our pipeline. Given a collection of 3D shapes, we automatically render images from predefined angles in a sketchy style. The generated data is then combined with semantic registration information available in the shape data set to train a convolutional neural network (depicted on the right). The final goal of our method is to learn a function that takes a succession of views centered on a point \( p \) from an input sketch and outputs a descriptor \( Y^p \in \mathbb{R}^S \) for the point, being \( S \) the descriptor dimensionality.

**Synthetic sketch data set.** Our method automatically learns the network parameters from the training data set. Our ultimate goal is to compute descriptors on sketches rendered by human artists and designers with projection inaccuracies and rough lines. Therefore, our data set must approximate the true (and unknown) data distribution, ensuring an appropriate generalization error during testing time. Similar points on each image pair must be registered, so that the network can map them nearby in descriptor space. To meet these requirements, we leverage existing repositories of 3D shapes registered and semantically segmented into meaningful parts [3]. For every sample point \( p \in \mathbb{R}^3 \) on the shape surface, we capture a set of 2D projections centered on \( p \), adopting typical camera positions suggested in the design literature (see Section V) and render them as sketches using NPR techniques.

**Multi-view convolutional network.** Our CNNs take a sequence of zoomed views centered on a particular point from a fixed viewport, providing local and global context information. This view-based representation of the point is close to the way humans perceive objects in our 3D world, and has been shown effective in reconstruction tasks involving 3D shapes and line drawing data [37]. Then, our triplet training configuration ensures that: (i) corresponding points have their embeddings close together in descriptor space, and (ii) non-corresponding points have their embeddings far away. Our network learns to combine important data from each view via max pooling, ignoring superfluous information without averaging. Since triplet training often makes the embedding to collapse into excessively small clusters, we refine the training procedure by a custom triplet selection strategy that prevents this issue (see Section IV).

**Evaluation and applications.** In Sections VI and VIII we demonstrate that our descriptors outperform previous methods while exploiting shape knowledge and part semantics. More importantly, we show evidence that SketchZooms generalizes to hand-drawn images. Additionally, we evaluate the performance on single and multi-class training setups and compute standard metrics in the field. Finally, we show the usefulness of our local descriptors for real-world applications such as part segmentation, shape retrieval and, image morphing.
IV. LEARNING MULTI-VIEW DESCRIPTORS FOR LINE DRAWINGS

We now describe the main components of our learning pipeline.

Network input. Our network computes local descriptors from a set of zoomed sketch views \( V \) (three in our setup). These images are rendered from one of the predefined viewports front, side, 3/4 angle and centered on the point of interest \( p \). Before feeding the network, the images are normalized and scaled to 224x224 pixel resolution. Since hand-drawn designs are often not perfectly oriented with respect to camera position, we induce the network to learn rotational invariant descriptors by randomly rotating the input images. Each of the three input views have equal probability to remain untransformed or to be individually rotated by 90, 180 or 270 degrees. To keep the descriptor robust to different resolutions, we downgraded the input image size by 30% and 60% with a probability of 0.2, respectively. We then restore inputs to the original size using linear interpolation to simulate different amounts of quality degradation. In preliminary experiments we noticed that SketchZooms features were highly sensitive to the camera-target point distance. Therefore, we added noise to the zoom parameter by sampling camera displacements from a normal distribution (with \( \mu = 0 \) and \( \sigma^2 = 0.3 \), where 0.3 means 30% size increment w.r.t. the original image size). This zoom drift is applied directly over rendered images by cropping them as a post-process during training. Our data augmentation choices were iterative, and empirically guided by results obtained during the experimentation stage using a validation set.

Architecture. Our CNN is partially based on AlexNet architecture \([38]\). It is composed by five convolutional layers, followed by ReLU non-linearities and max pooling (see Appendix A). We exclude from the original AlexNet architecture the last two fully connected layers, which are related to ImageNet \([38]\) classification tasks. Instead, we use a view pooling layer aggregating the descriptors \( Y_{v,p}, v \in V \) generated for each of the three input views \( X_{v,p}, v \in V \), into a single one \( Y_p = \max(Y_{v,p}) \). The aggregation is performed by an element-wise maximum operation across the input views. In order to speed up descriptors computation and to make queries more efficient, we prepend a fully connected layer that learns to keep relevant information and transforms the 4096 dimensional output from AlexNet \((fc6)\) into a vector of size 128. This dimension reduction is performed before the pooling layer, a choice that greatly benefits max pooling computation without noticing any degradation in the descriptor quality.

Triplet loss. A key component in our approach is the learning mechanism for setting up the network parameters. While recent work like Wang et al. \([36]\) make use of a siamese architecture and the contrastive loss in their learning pipelines, we adopt a triplet loss, first introduced in FaceNet \([24]\). The main motivation is that distances gain richer semantics when put into context, and the anchor point added by the triplet loss better shapes the embedding by exploiting this relativistic approach \([39]\). Formally, we strive for an embedding from a set of sketch image views \( X_{v,p} \) centered on a point \( p \), into a descriptor \( Y_p \in \mathbb{R}^d \) (\( d = 128 \) in our setup). Ideally, two corresponding samples should have their embeddings close together in the descriptor space. On the contrary, two non-corresponding samples should have embeddings that are placed far away. The triplet loss enforces both goals by minimizing the distance between an anchor \( Y^a \) and a corresponding (also called positive) point descriptor \( Y^+ \) (see Figure 3). Simultaneously, it maximizes the distance between the anchor and a non-corresponding (negative) point descriptor \( Y^n \). Mathematically, we want:

\[
D^2(Y^a, Y^c) + \alpha < D^2(Y^a, Y^n),
\]

where \( D \) stands for the Euclidean distance between descriptors, and \( \alpha \) is a margin enforced between positive and negative pairs (\( \alpha = 1 \) in our implementation). Formulating Eq. 1 as an optimization problem over the network parameters \( w \), we have:

\[
L(w) = \sum_{i=1}^{N} \max\left( D^2(Y_{i}^a, Y_{i}^c) - D^2(Y_{i}^a, Y_{i}^n) + \alpha, 0 \right),
\]

Training with all possible triplets in our 5 million image dataset results prohibitively expensive. Moreover, naively using all triplets is highly inefficient since the more the training progresses, the more triplets are going to satisfy Eq. 1 making training slower over time \([24]\). To overcome this issue, we adaptively select semi-hard triplets on each training step satisfying:

\[
\begin{cases}
D^2(Y^a, Y^c) < D^2(Y^a, Y^n), \\
D^2(Y^a, Y^n) < D^2(Y^a, Y^c) + \alpha
\end{cases}
\]

meaning we look for training samples \( \{Y^a, Y^c, Y^n\} \) lying inside the semi-hard margin area delimited by \( \alpha \). For the sake of notation, we refer to triplets using descriptor notation symbol \( Y \). In practice, we compute \( \{Y^a, Y^c, Y^n\} \) from input images \( \{X_{i}^a, X_{i}^c, X_{i}^n\} \) using the last network training state. We build useful triplets on the fly for each training minibatch by testing whether their descriptors infringe or not Eq. 3. In our setup, we cluster individual samples in groups \( G \) to be sequentially used during each training epoch. To build a minibatch, we randomly sample positive pairs from \( G \) of the form \( \{Y_{i}^a, Y_{j}^c\} \), \( i, j \in G \). We then test the semi-hard conditions over a random number \( s \) of negative samples \( \{Y_{k}^a, Y_{k}^n\} \), \( i, k \in G \). We experimented with several values for \( s \) and found \( s = 5 \) to minimize the time spent in random search while still providing good triplets for training. Selected triplets are then packed together into a minibatch.
Visibility bias. Instead of adopting fixed camera positions, previous work on multi-view architectures select either random or a balanced set of viewports to capture features on the target shape. This prevents the network to suffer from bias to any particular view angle. SketchZooms works from a fixed set of camera positions, a restriction that simplifies the rendering and better adapts to real-world line drawing scenarios. Even if 3D shapes are sampled regularly, after projection there is no guarantee that every point will be visible from the same number of views. In practice, most points are visible from the 3/4 view (tightly related to the “most informative” definition), but far less points appear in the lateral or frontal views. A naive sampling strategy would train the network with fewer samples from the less populated views, degrading the descriptor performance on these scenarios. To compensate, we restrict our training minibatches to have approximately the same number of samples from each view.

Training details. The full network architecture was implemented with PyTorch 1.0 and trained on NVIDIA Titan Xp GPUs. For the category-specific setup, an individual network for each object class was independently trained during approximately 3 weeks. Alternatively, the multi-category network was trained on our four evaluation object classes simultaneously, during a period of 5 weeks. We first initialize the convolutional layers on our network using AlexNet weights trained on the ImageNet data set, as provided in Pytorch. The learning rate was set to $l = 1 \times 10^{-5}$ and trained the networks for 30 epochs. We optimize the objective in Eq. 2 using Adam optimization (β₁ = 0.9, β₂ = 0.999) and a batch size of 128 triplets. We did not use batch normalization layers or dropout in addition to those already into AlexNet (Dropout $p = 0.5$ on layer fc6). Refer to Appendix A for details on the network layers.

V. MULTI-VIEW LINE DRAWING DATA SET

We now describe the 3D model collection and the rendering techniques used to create our synthetic line drawing dataset.

Shape collection. In general, existing large scale line drawing data sets like those in [20] or [42], compile doodles created in a very limited time frame. Images from these data sets often come from contributors that not necessarily have drawing skills or design-specific knowledge, and do not provide annotated correspondences among images. Therefore, similar to recent work targeting sketches and machine learning [41]–[43], we generated synthetic line drawings directly from the semantically corresponded 3D shapes in [3]. From the few available categories, we selected those offering the largest number of models, to guarantee diversity. From the original 4932 models distributed in 24 shape categories, as a proof of concept, we selected a subset from the classes bag (79), chair (62), mug (73) and headphone (68). The models are already centered, scaled, and properly oriented, which facilitates camera positioning during rendering. More importantly, 3D models are augmented with correspondences files that provide a list of surface matching points for every possible pair of shapes within each category. Every shape is randomly sampled with around 10,000 points over the model surface. Correspondences are computed with a custom part-based registration algorithm that performs a non-rigid alignment of all pairs of segments with the same label over two target shapes.

Synthetic line drawings. While previous data-driven methods in sketching employ simple models as Canny edges [1987] or image-space contours from [44], we adopted Apparent Ridges from [45], a good approximation to artists lines as shown in Where do people draw lines? [46]. Apparent Ridges lines approximate meaningful shape cues commonly drawn by humans to convey 3D objects. Apart from rendering style, viewport selection is crucial to convey shape in sketches. Literature in design recommends to adopt specific viewports in order to reduce the sketch ambiguity and simultaneously show most of the target shape [35]. Most
3D reconstruction algorithms from sketch images relies on assumptions like parallelism, orthogonality, and symmetry [37]. Following these guidelines, we selected a set of orthographic views to be rendered from 3D models. We used a total of two accidental (object-aligned) views: front, right side, and a single isometric angle, also called informative view: front-right (see Fig. 4). Left sides were omitted since we assume that the objects are symmetric with respect to the front view (discussed in Section VII). To capture images, we centered the camera on each sample point, and shoot it from three different constant distances and three distinct viewpoints. Occluded points were discarded by comparing z-buffer data with camera-to-target point distance. In total, our data set consists of 5,053,038 images in a resolution of 512×512 pixels. It took approximately 3 months to complete the rendering stage on a PC equipped with an NVIDIA Titan Xp GPU and an Intel Xeon E5 processor. To our knowledge, this is the largest synthetic drawing database ever created, and we aim to publicly release it by the time of publication for the sake of reproducibility and to encourage further research.

Rendering parameters. We repurpose the apparent ridges algorithm incorporated into RTSC software from [48]. In addition to the apparent ridges switch, we activated contour and body lines for rendering. We switched off all the other default controls in the software. We further changed the camera model to orthographic mode and place it at a distance of three world units for 3/4 views, and 1.5 world units for side and front views (models are normalized inside a cube of side = 1). We took successive zooms at 1.0x, 1.5x and 2x. To smooth out discontinuities, we run two iterations of the subdivide mesh and curvature smoothing algorithms implemented in the triangular mesh manipulation library Trimesh.

Hand-drawn benchmark. Transferring cues learned on synthetic data to real inputs has been shown effective before to overcome the difficulties of compiling a densely registered massive dataset of line drawings [36], [41]. To validate our performance over sketches in the wild, we collected a set of 40 hand-drawn sketches freely available online or from repositories like Shutterstock. We selected 10 sketches per category, a number of images feasible to be manually matched by humans. The compiled set includes line drawings with significantly different styles, shading, construction lines, decorations and projection/camera noise. We post-processed them to standardize scales and pixel resolutions, yet preserving construction lines, decorations and other styling features. For all collected images, we selected four random points and asked users to manually match each pair of sketches within the same category. To collect correspondences, we took all possible origin-target image pair combinations for each object class. Then, for every origin image we randomly selected 4 points inside the object mask region, and asked users to manually match them on the target image. We split the full task into four users without overlap and obtained a total of 1440 correspondences. The full image set and matching data is provided as supplemental material.

VI. RESULTS AND EVALUATION

We experimentally evaluate multiple aspects of our approach: (i) we tested SketchZooms on a number of hand-drawn examples to assess the generalization power of the network (see Figure 3), (ii) we examine the ability of our learned embeddings to properly distribute descriptors in the feature space, (iii) we compute correspondence accuracy metrics to evaluate matching performance over image space, and (iv) we perform a perceptual study to assess the semantic aspects on our features. Since our training framework is sufficiently general to allow learning in a multi-category setting, we additionally evaluated our model trained on the four evaluation object categories simultaneously.

A. Hand-drawn images

Figure 3 illustrates SketchZooms results on our hand-drawn testing data set under both sparse and dense matching scenarios. We observed that our method is able to successfully exploit the features learned from a synthetic training set when working with hand-drawn images. In particular, our features’ matching performance remains stable on these images, even if camera position is not perfectly aligned to the objects’ axes. More importantly, decorations and shading lines do not introduce significant discontinuities over dense mappings. Our predicted correspondences are smooth and visually plausible. Notice that most pairs correspond to line drawings rendered from different camera positions. To speed up dense map computation, we used a binary mask eliminating pixels outside the region of interest. We visualize dense color maps by projecting our descriptors onto a full color basis in $\mathbb{R}^3$.

Computing dense metrics like CMC (see Section VI-B) is not appropriate for this evaluation scenario where the dataset is annotated with sparse correspondences. Instead, we found more informative to report average error for each method. We computed matchings using features from our single-category training networks and compared L2 distances w.r.t. this silver standard data set of 1440 non-dense correspondences. SketchZooms performed better than all considered methods: SketchZooms: 0.069, ShapeContext: 0.074, GALIF: 0.195, and PCA: 0.206. It is important to highlight that methods like ShapeContext are model-agnostic and some times more general. Our goal is not to replace traditional descriptors but to introduce a new one dealing with problems involving 3D views and part information directly in 2D. However, we believe learned descriptors like SketchZooms do have the potential to generalize to unseen categories, as demonstrated in Section VII. We expect that the availability of bigger data sets and better machine learning techniques will push forward this research direction. The results presented in this section highlight the potential of our synthetic dataset to learn meaningful characteristics and generalize to sketches in the wild.

B. Single-category setup

Embedding quality. We tested our embedding space using Cumulative Match Characteristic (CMC), a standard quality
measure for image correspondences [49], [50]. This metric captures the proximity between points inside the embedding space by computing distances over descriptor pairs on two target sketches: given a point on one of the input images, a list of corresponding candidate matchings on the other image is retrieved; then, candidates are ranked using a proximity measure, e.g. the Euclidean distance in descriptor space. By repeating this process and accumulating ranks over all our testing image pairs, a plot is created in which the Y-axis accounts for the number of ground-truth matching points whose rank is below the number depicted on the X-axis. We compare our method against state-of-the-art descriptors...
commonly used for local sketch matching tasks, including the radial histograms from ShapeContext (SC) [4] and the GALIF descriptor, based on Gabor filters by Eitz et al. [20]. In the presented evaluation, we avoid performing against general image descriptors like SIFT [51]. Originally designed for photographs, it has been shown that these features do not cope well with the sparse stroke orientations in sketches [20]. We additionally consider a hand-crafted descriptor consisting on a Principal Component Analysis (PCA) over a small neighborhood of pixels surrounding the target point. This provides a simple comparison baseline for all tested methods. To the best of our knowledge, no other deep learning based approaches have been introduced for local sketch matching tasks. We computed CMC using our test data set consisting of multi-view synthetic sketches taken from models not used during the training phase. Given two shapes, we consider all matching combinations among the view set (front-front, front-side, front-3/4, side-front, side-side, side-3/4, 3/4-front, 3/4-side, 3/4-3/4). Since methods like ShapeContext are sometimes not well defined over empty regions of the sketch, we only considered points lying on black pixels to ensure a fair comparison. However, a strength of SketchZooms features is that they can be computed even on areas far away from black pixels, as shown in Figure 5. In total, our test samples consist of 2,000 corresponding points for each selected object category. Figure 6 demonstrates the performance of the evaluated descriptors on our synthetic test set. We report numerical measures on Table I over 100 retrieved matches. In all cases, our learned descriptors outperformed the other methods, better capturing the semantics of the target points across changes in viewports.

Image space accuracy. Similar to the embedding metric, we also evaluated the accuracy of our descriptors on the image space. In particular, we used a standard evaluation metric, namely the correspondence accuracy from [52], over our set of test samples. This metric evaluates the accuracy of predicted correspondences with respect to the ground truth. Given a point on one of the input images, we retrieve a candidate matching point on the other image. We measure the L2 distance between the retrieved matching point and the ground truth. By registering all distances over our test data set, we create a plot accounting for the percentage of correspondences (Y-axis) below a given threshold error (X-axis). Figure 6 shows the performance of each descriptor on our synthetic test set. Notice SketchZooms always retrieves 50% of the correspondences with less than 7% image side error, outperforming its predecessors. In all the evaluated semantic categories our descriptors succeed in matching most of the target points with the less relative error.

C. Multi-category setup

In an attempt to construct a more generic descriptor, we performed an additional experiment to study the viability of learning our neural network in a multi-category setting. To this end, we trained our method using a unified data set comprising our four object evaluation categories altogether. These models present significant shape variability, with over 350 different objects considered for this setup. Figure 6 (dashed lines) includes the CMC and the correspondence accuracy plots for

| Model  | SC    | GALIF | PCA    | Ours (single) | Ours (multi) |
|--------|-------|-------|--------|---------------|--------------|
| Bag    | 67.47%| 74.40%| 64.51% | 84.39%        | 86.51%       |
| Chair  | 72.91%| 70.4% | 56.67% | 84.60%        | 87.25%       |
| Mug    | 72.61%| 66.71%| 59.07% | 74.71%        | 81.65%       |
| Headphone | 70.11 % | 76.96% | 66.61% | 81.75%        | 86.30%       |
our 8,000 test samples, compared to our baseline methods: ShapeContext, GALIF and PCA. Both training setups showed similar behavior, overcoming the performance of the reported descriptors. More importantly, multi-category training outperformed single training when computing embedding space quality metrics (see Table I). Despite the increased complexity of the learned embedding for the multi-class scenario, our network can produce fairly general local shape descriptors that perform favorably compared to other hand-crafted alternatives. Nevertheless, we observed that category-specific models ensure more reliable matchings between object components with the same functionality, even if their shape or appearance differs significantly.

D. Perceptual study

Humans possess the extraordinary ability to resolve semantic correspondences in multi-view scenarios thanks to their previously-acquired knowledge about our 3D world. To better understand our descriptors potential to capture semantics, we conducted a perceptual study. Each of the 10 study volunteers was presented with \( m = 4 \) points on the first image and was instructed to find \( m \) corresponding points on the second image. We used a total of 40 image pairs distributed in our four evaluation categories: bag, chair, mug and headphone. Images show synthetic sketches from random viewports in our data set. Points were randomly pre-selected from a larger list of feature points computed over all the study images using the corner detector Good Features to Track from \([53]\). Similar to the study presented in BestBuddies \([54]\), we fitted 2D Gaussian distributions over the point coordinates annotated by users. We define a similarity measure by evaluating the fitted probability density function on our query points. Higher similarity scores are then assigned to regions where the consensus among users is strong, and vice-versa. We averaged the scores for all the points and summarized the results in Table II. On all tested methods, we filtered outliers outside an error threshold equal to \( \text{mean} \pm \text{std} \) measured in pixels. Heatmaps in Figure 7 represent areas where the matching consensus is stronger. Correspondences obtained with our descriptors are closer to hot areas than those produced with other methods. We used single-category trained networks to compute SketchZooms correspondences on Figure 7. The full set of images from user’s data is available as supplemental material.

VII. Robustness and Limitations

We now discuss the overall behavior of our method under challenging scenarios and its main limitations.

Camera noise. We performed a controlled study using synthetic sketches in which the camera angle was perturbed with different levels of noise. We re-rendered our emphphone test data set introducing three different levels of random noise in the form of a displacement in the camera position. This displacement represented a 5%, 10%, and 20% movement w.r.t. the distance to the sample point over the model surface. In other words, a random noise vector was added to the originally view-aligned camera position, scaled proportionally to the camera-target distance. Figure 8 shows Cumulative Match Characteristic and Correspondence Accuracy for the noisy datasets. Despite a drop in performance, we noticed this drop does not scales significantly when more noise is added.

Robustness to sketchiness. Adopting Apparent Ridges as our data set rendering engine allowed our method to be robust to typical concept drawings sketchiness. Synthetic images rendered with this method often contain wiggly lines and several other imperfections. Figure 9 shows dense mappings for inputs with different levels of sketchiness and decorations. To create this figure, we ask a designer to draw several versions of the same sketch, with an increasing density of strokes and level of detail. Then, we compute dense correspondences from the evaluated methods.

Table II: Perceptual study metrics.

|                | SC     | GALIF  | PCA    | Ours (single) | Ours (multi) |
|----------------|--------|--------|--------|---------------|--------------|
| Score          | 0.045  | 0.01   | 0.009  | 0.047         | 0.048        |
| NRMSE          | 0.068  | 0.164  | 0.17   | 0.062         | 0.067        |

Fig. 7. Users’ matching heatmaps. On each pair, left images show the reference points to be matched on the right images. Hotter colors indicate strong users’ matching consensus on a given image area. Symbols indicate correspondences from the evaluated methods.

Fig. 8. Cumulative Match Characteristic and Correspondence Accuracy curves for our noisy test data sets. Left: Y-axis accounts for the percentage of matchings retrieved below the raking position indicated on X. Right: X-axis shows normalized euclidean distance error and Y-axis accounts for the percentage of matchings retrieved below the error margin indicated on X. Notice we report single-category training results.
correspondences maps among all possible pair combinations. SketchZooms produces fairly stable results, with a slightly drop in smoothness for inputs with incomplete strokes.

Symmetry. Most man-made objects exhibit symmetry with respect to some plane in the 3D space. We noticed our features can sometimes mismatch on symmetric points on the target sketch (see Figure 10). This means descriptors from symmetric points are being mapped close together in the learned embedding, a common problem also suffered by other descriptors like ShapeContext. Symmetry mismatches happen more frequently when trying to correspond extreme viewports, like the side and front of two target objects. However, simultaneously matching several points can help disambiguating these symmetries –i.e. a combinatorial optimization method like the Hungarian algorithm [55] could help to refine more coherent matchings than using a simple strategy of matching closer points in descriptor space. Alternative solutions could incorporate orientation tags into the training phase or involve users actively in refining correspondences on the fly.

Zoom parameter sensitivity. As mentioned in Section IV, in order to compute a descriptor for a given image point, we need to successively crop three zoomed images surrounding it. These images are aggregated and transformed by the SketchZoom network to produce a descriptor of the point. We pick the zoom parameter value in order to include some information of the strokes composing the target image–since providing three empty images to the network would produce undesired outputs. In particular, for all results presented in this paper we fixed zoomed images sides to be 10%, 20%, and 40% of the total image length (512 pixels in our experiments). To demonstrate the effect of different zoom magnitudes, we computed additional dense mappings in two alternative scenarios: twice and half the reported sizes. Figure 11 shows an example of dense mappings with varying zoom window size. Even if we did not notice significant changes and sensitivity to this parameter over the reported value range, zooming too much can lead to situations in which the three cropped images have any stroke information. On the contrary, zooming too little could miss details, degrading the output descriptor quality.

Generalization to unseen categories. Finally, we assess the capacity of SketchZooms to generalize to unseen object categories. We asked a designer to produce drawings of lamps, a class with significant shape variation w.r.t. those used for training, but that still shares the characteristics of man-made objects. We then computed dense and sparse correspondences on these sketches using our multi-category trained network. Figure 12 shows exemplary outputs from this evaluation. Dense maps obtained with the top performing single-category model were also included for comparison. It can be seen

Fig. 12. Examples of pair-wise correspondences for unseen object classes with significant shape variation. (a) Dense correspondences obtained with our top-performing single-category trained SketchZooms model (bag). (b) Dense correspondences obtained with our multi-category SketchZooms model. (c) Sparse correspondences obtained with the multi-category model. Notice the increased accuracy of (b) over (a), correctly mapping the base and most of the lamps’ shades. However, the thin feature along the lamps’ bodies is still not accurately matched.
that the multi-category model is able to match sketches from unseen categories, even with significant style and shape variations between pairs. This behavior does not hold for the single-category model, which performed significantly worse. Our hypothesis is that multi-category learning, as a (heavy) form of data augmentation, increases the regularization effect, allowing better generalization with a negligible decrease in performance for the individual classes used for training (Fig. 6). On the other hand, single-category training ensures slightly better results on objects similar to those used for training, but at the cost of less generalization to unseen classes.

VIII. APPLICATIONS

We now demonstrate applications for our multi-view local descriptors. We address common problems found in the graphics and sketching literature like drawing assistance from 3D models, automatic layering, sketch-based shape retrieval, and image morphing.

Image morphing for shape exploration. Exploring different alternatives at early design stages is of paramount importance as it can catch and avoid problems that are costly later in the design pipeline. Inspired by the recent work of Arora et al. [8], we implemented an image morphing algorithm based on the image mapping obtained from the SketchZooms features. The goal is to allow interactive exploration of the continuous design space between two sketches while smoothing views and shape transitions. We start by computing motion paths between sparse corresponding points, and then interpolate them into dense smooth trajectories. In practice, we sample \( k \) (\( k = 10 \)) correspondences evenly distributed over the input-target pair. Then, we compute a Delaunay triangulation of the image space using the sampled points as input. For each triangle, we estimate an affine transformation that maps both triangulations on a number of steps \( s \) (\( s = 50 \)). Similar to the work of Arora et al., we implemented a non-linear alpha blending function to reduce ghosting effects. First, we define the confidence score as the L2 distance in feature space of two matched points. This confidence is then propagated to all other pixels via linear interpolation. Our blending function for a pixel \( p \) at a step \( s \) is defined as,

\[
\alpha_p(s) = \frac{1}{2} \left( 1 + \frac{1}{2} \tanh \left( \frac{s - \delta(p)}{\rho(p)} \right) \right),
\]

where \( \delta \) and \( \rho \) are linear functions of the pixel confidence score to keep the sigmoid outputs in the \([0,1]\) interval. This blending function ensures that well matched regions smoothly transition into other images, while regions with poor matchings disappear quickly from the image (Fig. 13).

Part segmentation. Most graphics software offer stylization, manipulation and animation capabilities for 2D drawings. However, some understanding of the content of the drawing is required to perform these high-level tasks. Achieving this understanding automatically is challenging due to the significant gap between human knowledge and the algorithm ability to derive it from pixels or vector strokes. Sketch segmentation has been addressed before as an instance of colorization [55] and simplification [13], [15]. Segmentations can then be used for different applications, like adding depth information to line drawings or applying global illumination effects [57], [58]. Similarly, SketchZooms features can be used to perform automatic semantic layering and coloring, since painting has much in common with image segmentation. Specifically, we first manually segmented hand-drawn images from the headphone category (10 in our test application). Then, we computed SketchZooms features for every pixel and used them to train a C-SVM classifier in order to learn to predict labels from our descriptors. Formally, we solve for:

\[
\min_{w, b, \zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i
\]

\[
\text{subject to } y_i \left( w^T \phi(x_i) + b \right) \geq 1 - \zeta_i,
\]

\[
\zeta_i \geq 0, i = 1, \ldots, n
\]

where \( C \) is the capacity constant (set to \( C = 1 \)), \( w \) is the vector of coefficients, and \( \zeta_i \) represents parameters for handling non-separable data. The index \( i \) labels the \( n \) training cases (\( n = 3 \) in our setup). The basic intuition behind our SVM classification is to find a separating hyperplane that corresponds to the largest possible margin between the feature points on different classes. Figure 14 shows some of our coloring results over automatically extracted segmentations.

Fig. 13. Image morphing sequences using SketchZooms descriptors for corresponding two target sketches. A non-linear alpha blending map was computed from point distances in the SketchZooms feature space.

Fig. 14. SketchZooms features repurposed for sketch segmentation. Users can decompose sketches into different layers or use the pixel-wise semantic tags for coloring.

Part segmentation. Most graphics software offer stylization, manipulation and animation capabilities for 2D drawings. However, some understanding of the content of the drawing is required to perform these high-level tasks. Achieving this understanding automatically is challenging due to the significant gap between human knowledge and the algorithm ability to derive it from pixels or vector strokes. Sketch segmentation has been addressed before as an instance of colorization [55] and simplification [13], [15]. Segmentations can then be used for different applications, like adding depth information to line drawings or applying global illumination effects [57], [58]. Similarly, SketchZooms features can be used to perform automatic semantic layering and coloring, since painting has much in common with image segmentation. Specifically, we first manually segmented hand-drawn images from the headphone category (10 in our test application). Then, we computed SketchZooms features for every pixel and used them to train a C-SVM classifier in order to learn to predict labels from our descriptors. Formally, we solve for:
**Sketch-based 3d shape retrieval.** As shown in Section [15], much of the work on image features for sketches was proposed in the context of 3D retrieval applications. In order to test the potential of our features in this task, we implemented a 3D model search engine based on our local descriptors. First, we pre-computed 200 synthetic headphones sketchy renders from our three key viewports (68 models in total). Second, we took 32 point samples from each image using poisson disk sampling. Third, we computed and stored our features on each sample point, and keep them associated with the original model. At query time, we repeat the sampling strategy using hand-drawn images, and retrieve a model list sorted using L2 distance w.r.t. query points. This simple strategy retrieves similar models in the database (Fig. [15]). Additionally, our search engine can accurately determine which camera viewport best matches the query sketch in order to consistently orient 3D models.

**IX. Conclusion and Future Directions**

We presented SketchZooms, a new image descriptor for corresponding sketches. To the best of our knowledge, SketchZooms is the first data-driven approach that automatically learns semantically coherent descriptors to match sketches in a multi-view context. Aiming this with deep neural networks was unfeasible before due to data limitation, as collecting sketches from artists and designers is extremely challenging. To address this problem, we have put together a vast collection of synthetic line drawings from four human-made objects categories and camera viewports commonly adopted by designers. More importantly, the proposed technique was able to generalize to sketches in the wild directly from our synthetic data. Our results offer interesting future directions of research. On the technical side, recent approaches have proposed to use semi-supervised hand-drawn images to improve network performance [59]. Investigating whether explicit treatment of domain shifts can boost performance on our hand-drawn data set is an interesting future direction to explore. Another avenue is to investigate whether other viewpoint configurations are possible without introducing much ambiguity into the descriptor space. Finally, a deep study on how humans performs matching tasks on the sketch image domain would be very beneficial to build more accurate descriptors.

**Acknowledgments**

The financial support by Universidad Tecnológica Nacional, SIUTINBA0005139 - CrossMatch: Detección de Correspondencias de Dominio Cruzado mediante Deep Learning, is gratefully acknowledged. We thank the NVIDIA Corporation for their GPU donation.

**References**

[1] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 945–953.

[2] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai, “Triplet-center loss for multi-view 3d object retrieval,” arXiv preprint arXiv:1803.06189, 2018.

[3] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer, “Learning local shape descriptors from part correspondences with multiview convolutional networks,” ACM Transactions on Graphics (TOG), vol. 37, no. 1, p. 6, 2018.

[4] C. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” California Univ. San Diego La Jolla Dept. of Computer Science and Engineering, Tech. Rep., 2002.

[5] J. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, “Sketch2photo: Internet image montage,” in ACM Transactions on Graphics (TOG), vol. 28, no. 5. ACM, 2009, p. 124.

[6] B. Whited, G. Noris, M. Simmons, R. W. Sumner, M. Gross, and J. Rossignac, “Betweenit: An interactive tool for tight inbetweening,” in Computer Graphics Forum, vol. 29, no. 2. Wiley Online Library, 2010, pp. 605–614.

[7] J. Xing, L.-Y. Wei, T. Shiratori, and K. Yatani, “Autocomplete hand-drawn animations,” ACM Transactions on Graphics (TOG), vol. 34, no. 6, p. 169, 2015.

[8] J. Xing, H.-T. Chen, and L.-Y. Wei, “Autocomplete painting repetitions,” ACM Transactions on Graphics (TOG), vol. 33, no. 6, p. 172, 2014.

[9] X. Liu, T.-T. Wong, and P.-A. Heng, “Closure-aware sketch simplification,” ACM Transactions on Graphics (TOG), vol. 34, no. 6, p. 168, 2015.

[10] X. Liu, E. Rosales, and A. Sheffer, “Strokeseggregator: consolidating raw sketches into artist-intended curve drawings,” ACM Transactions on Graphics (TOG), vol. 37, no. 4, p. 97, 2018.

[11] J. Pu, K. Lou, and K. Ramani, “A 2d sketch-based user interface for 3d cad model retrieval,” Computer-aided design and applications, vol. 2, no. 6, pp. 717–725, 2005.

[12] J. Xing, H.-T. Chen, and L.-Y. Wei, “Complete sketch illumination,” ACM Transactions on Graphics (TOG), vol. 34, no. 6, p. 169, 2015.

[13] J. Xing, H.-T. Chen, and L.-Y. Wei, “Complete sketch illumination,” ACM Transactions on Graphics (TOG), vol. 34, no. 6, p. 169, 2015.

[14] J. Xing, H.-T. Chen, and L.-Y. Wei, “Complete sketch illumination,” ACM Transactions on Graphics (TOG), vol. 34, no. 6, p. 169, 2015.
Our network architecture (Figure 16) is based on the work of Krizhevsky et al. [38] known as AlexNet. We modified the last layers to incorporate a view pooling mechanism.

**APPENDIX**

Our network architecture (Figure 16) is based on the work of Krizhevsky et al. [38] known as AlexNet. We modified the last layers to incorporate a view pooling mechanism.

![Fig. 16. Full network architecture, ReLUs are used after each convolutional or fully connected layer.](image-url)