Learning to Group Discrete Graphical Patterns

ZHAOLIANG LUN*, University of Massachusetts Amherst
CHANGQING ZOU*†, Simon Fraser University and Hengyang Normal University
HAIBIN HUANG, University of Massachusetts Amherst
EVANGELOS KALOGERAKIS, University of Massachusetts Amherst
PING TAN, Simon Fraser University
MARIE-PAULE CANI, LIX, Ecole Polytechnique, CNRS
HAO ZHANG, Simon Fraser University

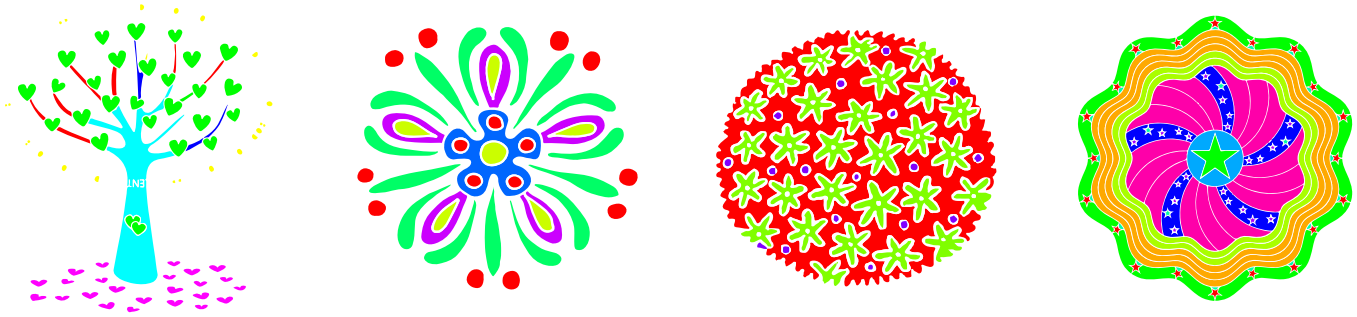

Fig. 1. We train a deep convolutional neural network which learns to group discrete graphical patterns from a large set of human-annotated perceptual grouping data. The four grouping results obtained (grouped elements share the same color), while still exhibiting various imperfections, demonstrate that our learned grouping scheme is able to handle a variety of noise and mixing of element shapes and arrangements.

We introduce a *deep learning* approach for grouping discrete patterns common in graphical designs. Our approach is based on a *convolutional neural network* architecture that learns a *grouping measure* defined over a pair of pattern elements. Motivated by perceptual grouping principles, the key feature of our network is the encoding of element shape, context, symmetries, and structural arrangements. These element properties are all jointly considered and appropriately weighted in our grouping measure. To better align our measure with human perceptions for grouping, we train our network on a large, human-annotated dataset of pattern groupings consisting of patterns at varying granularity levels, with rich element relations and varieties, and tempered with noise and other data imperfections. Experimental results demonstrate that our deep-learned measure leads to robust grouping results.

CCS Concepts: • **Computing methodologies** → **Shape analysis**;

Additional Key Words and Phrases: discrete pattern analysis, perceptual grouping, supervised learning, convolutional neural networks

ACM Reference Format:
Zhaoliang Lun, Changqing Zou, Haibin Huang, Evangelos Kalogerakis, Ping Tan, Marie-Paule Cani, and Hao Zhang. 2017. Learning to Group Discrete Graphical Patterns. *ACM Trans. Graph.* 36, 6, Article 225 (November 2017), 11 pages. https://doi.org/10.1145/3130800.3130841

*Both authors contributed equally to the paper
†Corresponding author: aaronzou1125@gmail.com

0730-0301/2017/11-ART225 $15.00
https://doi.org/10.1145/3130800.3130841

1 INTRODUCTION

Graphical patterns such as 2D vector art, facades, textile or packing designs, are often encountered in our daily lives. They have also been the subject of study in computer graphics research, i.e., in the context of pattern editing [Stava et al. 2010; Zhang et al. 2013], synthesis [AlHalawani et al. 2013; Bao et al. 2013; Yeh et al. 2013], exploration [Chen et al. 2016; Guerrero et al. 2016], and layout optimization [Xu et al. 2015], to name a few. Often, the first step towards any graphical pattern processing task is to algorithmically understand the underlying structure of the input patterns. And in turn, an understanding of pattern structures often relies on *perceptual grouping* [Köhler 1929; Palmer 1992, 1977].

While humans possess an innate ability to perceive forms and phenomena in this world as organized patterns, the main challenges in enabling a machine to group graphical patterns like humans are two-fold. Firstly, the immensely rich variations and complexities of the set of discernible graphical patterns as well as the noise and imperfections which are often associated with them in real-world data should all be accounted for; see Figure 2 (top). Secondly, and more subtly, when different perceptual grouping principles [Köhler 1929; Palmer 1977] such as similarity, proximity, continuity, symmetry, would lead to *conflicting* grouping results, there is a need to *conjoin* these principles [Kanizsa 1980; Nan et al. 2011]. In such cases, it remains unclear which grouping principles should take precedence; see Figure 2 (bottom). Resolving such conflicts is essential to obtain a high-level understanding of graphical patterns.

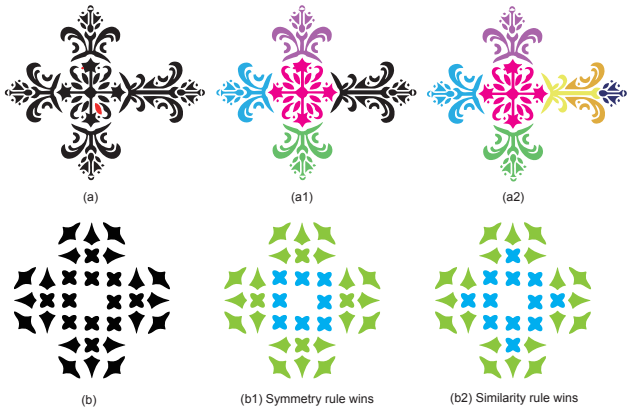

Fig. 2. Challenges in graphical pattern grouping. The top row illustrates the rich variations and complexity found in real-world patterns. (a): Red element deviating from perfect symmetry. (a1) and (a2): Which grouping is better? Discerning between complex patterns and groupings is difficult. The bottom row shows conflicting grouping principles for pattern (b), leading to different groupings: (b1) and (b2).

In this paper, we develop an algorithm which mimics the human ability of graphical pattern grouping and apply it to group such patterns formed by one or more atomic (i.e. indivisible) elements; see Figure 1. While general grouping models or principles have been known for a long time [Köhler 1929; Palmer 1977], it remains unclear how to exactly quantify the various models, how important each model is relative to the others, and what the best feature representation for pattern elements to characterize their similarity is, especially in terms of shape and structural arrangements. In our work, instead of hand-engineering rules to quantify grouping models and hand-tuning their relative importance, we resort to a data-driven approach and develop a deep learning framework.

At the heart of our learning approach is a *deep Convolutional Neural Network* (CNN) which is trained to extract a grouping measure consistent with human perception. Figure 3 provides an illustration of the CNN architecture. The CNN takes as input graphical pattern images and produces descriptors encoding element shapes, contexts, and structural arrangements, which reflect properties related to pattern similarity, proximity, continuity, and symmetry principles observed in the perceptual grouping literature. Then our CNN computes an element grouping measure as the Euclidean distance between weighted combinations of the learned descriptors. To allow our CNN-based grouping scheme to mimic human perception as well as robustly handle pattern noise and variety, we assembled a large training set of human-annotated pattern groupings to train our deep architecture. The training patterns correspond to various granularity levels, with rich varieties of compositions and variations, and are contaminated with noise and other data imperfections.

Our main contributions can be summarized as follows:

- The first data-driven method trained via a deep CNN for perceptual grouping of discrete graphical patterns.
- Learned shape-, context-, and structure-aware descriptors encoding graphical elements in pattern designs.
- A large, annotated dataset of pattern groupings encompassing rich pattern varieties and relations, which should benefit future research on pattern analysis and processing.

We used both standard numerical error measures and a perceptual user study to compare the grouping results produced by our method with those produced by humans on test pattern images. The comparative studies demonstrate that our method is capable of inferring groupings often consistent with human perception. We also tested our method against several competing alternatives, and found that it consistently produces more meaningful grouping results.

2 RELATED WORK

While our work proposes a data-driven approach to learn perceptual grouping, it is inspired by previous model-driven approaches, e.g., those on Gestalt-based perceptual grouping, symmetry-based grouping, and affinity-based grouping of visual patterns. In this section, we briefly discuss the related approaches.

Gestalt-based pattern grouping. Gestalt psychologists have tried to identify principles, or laws, related to perceptual grouping of organized patterns or objects in general [Palmer 1977; Wertheimer 1938]. For example, patterns tend to be grouped together when they are similar or proximal to each other, form closed figures, are continuous or symmetric. Early approaches attempted to quantify Gestalt laws; see [Desolneux et al. 2002] for a survey. For example, one could account for the degree of proximity or similarity [Kubovy and van den Berg 2008], then apply these numerical measures to a probabilistic model for perceptual grouping. A complication with Gestalt-based approaches is that different laws may interact and conflict with each other on the same stimuli [Kanizsa 1980]. A common approach to resolve such conflicts is to select a grouping mechanism based on the "simplest interpretation" [Feldman 2003]. In the context of discrete graphical patterns, Nan et al. [2011] present an energy minimization method using graph cuts for conjoining Gestalt laws of similarity, proximity, and regularity, where the quantification of these laws and the optimization objective are heuristically defined. The grouping method with conjoining Gestalt rules is then applied to the progressive simplification of architectural drawings.

In contrast to Gestalt-based approaches, which are largely model-driven and hand-engineered, we follow a *data-driven, learning* approach towards discrete pattern grouping. We loosely consider the Gestalt principles of proximity, continuity, symmetry, closure in the design of our learning architecture, yet we do not explicitly attempt to quantify them or hand-engineer their conflict resolutions.

Symmetry-based pattern grouping. Symmetry-based grouping has been a dominant analysis tool for pattern understanding in computer graphics. Many techniques have been developed for exact and approximate symmetry detection of patterns found in natural images and shapes, e.g., [Barnes et al. 2010; Graham et al. 2010; Lukac et al. 2017; Podolak et al. 2006]; see also surveys by [Liu et al. 2010; Mitra et al. 2012]. Several techniques have also investigated hierarchical pattern grouping in images or shapes by building symmetry hierarchies [Simari et al. 2006; van Kaick et al. 2013; Wang et al. 2011; Zhang et al. 2013]. Related to structural hierarchy analysis methods are also inverse procedural modeling techniques that use grammars

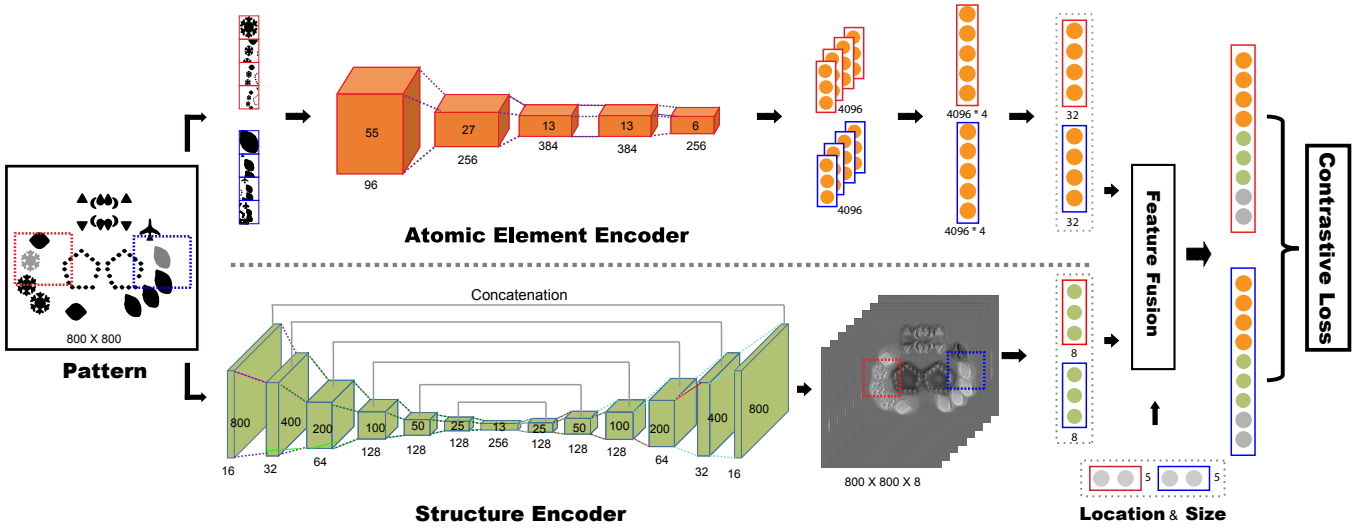

Fig. 3. Given an input image of discrete graphical patterns (*left*), our method processes the image and its elements through a deep CNN to extract shape-, context-, and symmetry-aware pattern element descriptors. The network is trained to produce a grouping measure that compares pattern elements such that the produced comparisons agree as much as possible with human-annotated ones, yielding meaningful pattern groupings.

to parse symmetric arrangements of graphical elements [Bokeloh et al. 2010; Stava et al. 2010; Wu et al. 2014].

In general, these techniques are largely model- or rule-driven, often with handcrafted precedence rules. Furthermore, while symmetries can be classified into a limited number of categories [Conway et al. 2008; Weyl 1952] and can all be well-defined mathematically, fluctuating symmetries [Graham et al. 2010], along with other grouping criteria such as continuity and proximity, are imprecise and can result in a much wider variety of pattern variations.

The recent work of Guerrero et al. [2016] on pattern exploration could account for a rich set of discrete patterns but opted for *manual* pattern analysis. It is unlikely that existing model-driven methods or their variants can cover a close-to-full spectrum of pattern variations. This has motivated us to develop a data-driven framework to learn discrete pattern grouping. Instead of hand-crafting rules for detecting symmetries and determining their precedence, we use a deep learning architecture that identifies and prioritizes symmetries for discrete pattern grouping, combining them with other perceptual grouping principles into a grouping measure to be learned from human-annotated data. The basic premise is that deep learning with rich sets of annotated pattern grouping data can lead to significant boost in discrete pattern grouping performance.

Affinity-based pattern grouping. The preeminent approach to visual pattern grouping or segmentation is to define a metric, or in other words an affinity measure, to compare patterns in the form of edges, patches, parts, or objects in natural images, then use this affinity to guide clustering algorithms (e.g., [Arbelaez et al. 2011; Shi and Malik 2000; Zitnick and Dollár 2014]). These methods typically rely on hand-engineered grouping cues, and as a result, lack robustness and generality. They have largely been superseded by data-driven and machine learning techniques by now.

Learning measures for grouping. Instead of hand-engineering grouping cues, recent analysis methods have employed deep learning architectures to group visual patterns appearing in *natural images*, in the context of image clustering [Xie et al. 2016; Yang et al. 2016], image classification [Greff et al. 2016], figure-ground segmentation [Maire et al. 2016], and contour detection [Maninis et al. 2016]. Our approach is more related to methods that learn affinities for comparing image patches, since graphical patterns can be represented as 2D rasterized image patches (alternatively to vector art). Several recent methods propose Siamese CNN architectures to learn patch-based affinities in natural images by training them in a supervised manner on patch correspondence datasets [Han et al. 2015; Simo-Serra et al. 2015; Yi et al. 2016; Zagoruyko and Komodakis 2015]. Isola et al. [2016] instead trains a CNN architecture for patch similarity in an unsupervised manner by making the CNN predict whether two visual primitives occur in the same spatial or temporal context.

These patch-based learning methods compare natural image patterns only based on their local appearance, since their CNNs exclusively operate on local patches around these patterns. Our method instead groups pre-segmented graphical elements based on their local appearance, context, and most importantly, global structures (e.g., symmetries, layouts) that are often present in graphical designs. For example, similar elements placed on four corners of the image are less likely to form a group in image segmentation tasks, while in our case they will be grouped due to the pattern regularity. As we discuss in our experiment section, existing CNNs for grouping patches in natural images do not generalize well to discrete graphical patterns.

3 GROUPING MEASURE

Overview. Given an input image of discrete graphical patterns, our method applies a learned CNN-based measure that compares atomic elements in the image, then clusters them into pattern groups.

Our CNN architecture is visualized in Figure 3. It is composed of two "sub-networks". The top sub-network, which we call *atomic element encoder* (in orange color), takes as input a pair of atomic elements and for each element, it extracts a learned descriptor encoding its overall shape and local context. We refer to this descriptor as *shape- and context-aware* element descriptor. The bottom sub-network, which we call *structure encoder* (in green color) takes as input the whole image, and attempts to detect structural arrangements in it, such as the presence of symmetries or figures. It then outputs a per-pixel descriptor that encodes where extracted structural arrangements are located in the input image. We refer to this descriptor as *structure-aware* descriptor.

To compare a pair of atomic elements, we consider (a) their shape- and context-aware descriptors extracted by the atomic element encoder (orange-colored vectors in Figure 3), (b) the structure-aware descriptors encoding the presence of structural arrangements within their area, as extracted by the structure encoder (green-colored vectors), and finally (c) their location in the image and size, which we explicitly provide as additional inputs to our architecture (grey-colored vectors). From our experiments, we found that all the descriptors, and in turn, both sub-networks are important so as to compute an accurate similarity measure between atomic elements. In addition, we found that the relative importance of the descriptors in our measure should not be the same, instead, they should be learned from training data. This is not surprising based on the perceptual pattern grouping literature [Palmer 1977], which indicates that different grouping principles do not necessarily share the same priority. Thus, our CNN architecture incorporates a linear transformation layer that re-weights the above descriptors before computing the final measure. Mathematically, given a pair of atomic elements (E, E'), our measure is evaluated as the Euclidean distance between their weighted descriptors $\mathbf{f}(E)$, $\mathbf{f}(E')$:

$$D(E, E') = ||\mathbf{W}(\mathbf{f}(E) - \mathbf{f}(E'))|| \quad (1)$$

where $\mathbf{W}$ represents a learned diagonal weight matrix used in our transformation layer. The element descriptor $\mathbf{f}(E)$ is produced by fusing its shape- and context-aware descriptor $\mathbf{f}_a(E)$ (32-D), its structure-aware descriptor $\mathbf{f}_s(E)$ (8-D), and its location and size descriptor $\mathbf{f}_l(E)$ (5-D) into a single column vector (45-D); $\mathbf{f}(E')$ is similarly defined. In the rest of this section, we describe how these different descriptors are computed through our architecture, then the learning method is discussed in Section 4.

Atomic element encoder. The element encoder aims to extract a shape- and context-aware representation capturing an atomic element along with its context at multiple scales. The architecture of this encoder is a modified version of the widely popular image-based CNN, known as AlexNet [Krizhevsky et al. 2012] (see also Figure 3, in orange color). The original AlexNet consists of two convolutional layers, followed by two pooling layers, then three additional convolutional layers and three fully connected layers. The convolutional layers apply a set of learned convolution filters to produce "feature maps" that are further non-linearly transformed through rectified linear units (ReLUs). The pooling layers employ max-pooling units that summarize the feature maps through subsampling, while the fully connected layers apply non-linear transformations operating on the whole feature maps produced in the previous layer. The last fully connected layer (called "fc8") outputs image classification probabilities related to categorization in ImageNet, thus we exclude it from our architecture.

We further modified AlexNet to process individual elements and local multi-scale context around them. Specifically, we first compute the oriented bounding box around the element, then crop the input image at 100%, 200%, 300%, and 400% of its box size. The resulting crops are resized to images of size 227x227 pixels, which is the input image resolution used in AlexNet. The four images are processed through four identical branches of AlexNet (without "fc8"), which in turn produce four $4,096$-dimensional features. These feature representations are concatenated into a single, multi-scale $16,384$-dimensional pattern representation. The representation is extremely high-dimensional, which can lead to unreliable distances when comparing elements [Zimek et al. 2012]. In addition, individual features encoded in this representation can have varying importance in evaluating atomic element and context similarity. Thus, we reduce the dimensionality and re-weight these features through a linear transformation, implemented as an additional layer after AlexNet.

Given the above-mentioned $16,384$-dimensional feature representation $\mathbf{g}(E)$ for an element E, our architecture outputs a compact 32-dimensional representation $\mathbf{f}_a(E)$ by applying the transformation $\mathbf{f}_a(E) = \mathbf{P} \cdot \mathbf{g}(E)$, where $\mathbf{P}$ is a weight matrix learned during the CNN training. We experimented with different dimensionalities for the output representation (4, 8, 16, 32, 64) - we found that 32 dimensions yielded highest agreement with ground-truth groupings in a hold-out validation set in our experiments. The output representation $\mathbf{f}_a(E)$ is the shape- and context-aware element descriptor used in the fused descriptor of Equation (1). In Section 5, we discuss results when this descriptor is omitted from our measure, and we also provide visualizations that indicate that the atomic element encoder captures useful shape information about pattern elements.

Structure encoder. In contrast to the atomic element encoder that focuses on representing individual elements and their local neighborhood, the structure encoder attempts to capture large-scale structures in the pattern. For example, if a set of elements form a heart-shaped figure, or are related under a global rotational symmetry, then these non-local structures provide strong cues to aggregate these elements into a single group, rather than splitting them into smaller groups. To capture a structure-aware element representation, one option would be to provide the atomic element encoder with very large input neighborhoods, e.g., including the whole pattern image (global context) around each element. However, such strategy would result in a highly redundant representation, since all elements would share the same global context. Instead, our structure encoder takes as input the whole image, and yields a single intermediate representation encoding the presence of large-scale structures in the input image.

The architecture of the structure encoder is shown in Figure 3 (in green color). It receives the whole input pattern image at 800x800 resolution and processes it through a series of convolution and pooling layers. Specifically, the structure encoder consists of two convolutional layers, followed by ReLU transformations and two pooling layers, which provide invariance to small perturbations of elements in the input image (similarly to AlexNet). Then the structure encoder includes five more convolutional layers followed

by ReLUs. Each convolutional layer applies a set of local filters in the feature map produced at the previous layer, resulting in feature maps of increasingly smaller size, yet encoding increasingly larger-scale structures in the input image. The output of the last convolutional layer is a set of 256 feature maps with size 13×13. Once the filters are trained on our dataset, we observed that the resulting feature map values are correlated with the presence of global symmetries or forms in the image (see Section 5 for related visualizations). Yet, these feature maps do not readily provide information on where exactly in the input image these large-scale structures exist, or which elements participate in these large-scale structures. As a result, we cannot use these maps as-is to extract element-level structure descriptors.

To extract these descriptors, our architecture employs seven "deconvolutional" layers consisting of convolution filters and ReLus that progressively transform and upscale the feature maps towards the original image resolution. As in the case of other deep architectures for image-to-image translation tasks [Isola et al. 2017], the configuration of the deconvolution layers, i.e., the size and number of their filters, is symmetric with the configuration of the corresponding convolutional layers. For example, the first deconvolutional layer produces feature maps of the same size and number as the sixth convolutional layer, the second deconvolutional layer produces feature maps of the same size and number as the fifth convolutional layer, etc. In addition, similarly to image-to-image translation architectures [Isola et al. 2017], we observed that better performance is achieved when each deconvolution layer receives as input the feature maps produced by the immediately previous layer in the architecture as well as the feature maps produced in its corresponding symmetric convolutional layer (see Figure 3, grey connections). This is due to the potential loss of fine-grained structure information in the produced $13 \times 13 \times 256$ feature map of the the last convolutional layer.

The output of our structure encoder is a set of 8 feature maps of size 800x800 that provide information on which and where large-scale structures are present in the input pattern image at the pixel level (see Section 5 for related visualizations). To compute the structure-aware descriptor $\mathbf{f}_s(E)$ for an element E in the input image, we find each pixel $p \in E$ covered by the element, and average the pixel feature values $h_m(p)$ per each output map m ($m = 1 \ldots 8$):

$$f_{s,m}(E) = \underset{p \in E}{avg}\, h_m(p) \tag{2}$$

The resulting 8-dimensional descriptor is incorporated into the fused descriptor used in Equation (1). We employed an aggregate function (averaging) here since each element has a different number of pixels and pixel ordering. A general learnable function cannot be readily applied to such unordered input. Instead of using the average feature values, we also tried using the maximum, but yielded slightly poorer performance. We also experimented with different number of output feature maps (4, 8, 16, 32, 64) as well as different number of layers and filters in the structure encoder, at the end, the architecture of Figure 3 offered the best performance in a hold-out validation set. Note also that producing 8 feature maps does not mean that up to 8 layouts are encoded; the features are continuous and can jointly encode a larger number of layouts. Finally, one might argue that the structure encoder may also capture information relevant to the shape or context of each element. Based on the visualizations of the output feature maps (Section 5), we found that structure encoder mainly captures large-scale structures and arrangements of elements rather than element shape or contexts. Using the structure encoder alone without the atomic element encoder significantly degraded the grouping performance (and vice versa).

Element location and size descriptors. Motivated by the "proximity" principle that states that nearby elements tend to be grouped together, we incorporated the location (x, y) of each element in the input pattern image as an additional 2-dimensional element descriptor. Computing this descriptor is straightforward and does not require any particular "hand-engineering"; we simply compute the centroid coordinates of each element in the image and normalize them between $[0, 1]$. Also motivated by observations in the perceptual pattern grouping literature stating that size influences the perception of pattern similarity [Palmer 1977], we incorporate the radius of the element bounding sphere with the width w and height h of the element's axis-aligned bounding box (relative to image size) into an additional descriptor, which is also straightforward to compute. Concatenating location and size yields the 5-dimensional descriptor $\mathbf{f}_l(E)$ used in the fused descriptor of Equation (1).

It is worth noting that the atomic element encoder is agnostic to element location or size, since the input elements are re-scaled and centered in the pattern image crops fed into it. Similarly, the structure encoder focuses on capturing large-scale structures rather than fine-grained location and size information. We found that explicitly incorporating location and size in our measure improved the grouping performance, as discussed in Section 5.

Clustering. The last step in our analysis pipeline is to apply a clustering technique to group patterns based on the measure of Equation (1). In general, given our distance measure between element pairs, one can apply any clustering technique which relies on element distances or similarities based on transformations of these distances, as input. We experimented with several clustering techniques including k-means, Gaussian Mixture Models, normalized cuts [Shi and Malik 2000], affinity propagation [Frey and Dueck 2007], as well as agglomerative clustering variants [Hastie et al. 2001] (see Section 5 for comparisons and parameter discussion). We found that affinity propagation offered the best grouping results.

4 LEARNING

Our learning procedure aims to produce a grouping measure that is consistent with human perception. To this end, all the parameters of our grouping measure, including the diagonal entries $\mathbf{w} = diag(\mathbf{W})$ of the descriptor weight matrix $\mathbf{W}$, as well as all the CNN parameters $\mathbf{v}$ (including the convolution/deconvolution filter parameters and the dimensionality reduction parameters) are learned from a training set of *procedurally generated* pattern images with human-annotated groupings. Below we explain our objective function used to train our network, the optimization procedure, and finally the training set.

Objective function. Our network parameters are learned by minimizing an objective function defined over a training set of element pairs. We use a function, known as contrastive loss [Hadsell et al. 2006], commonly used for learning distance functions. The loss

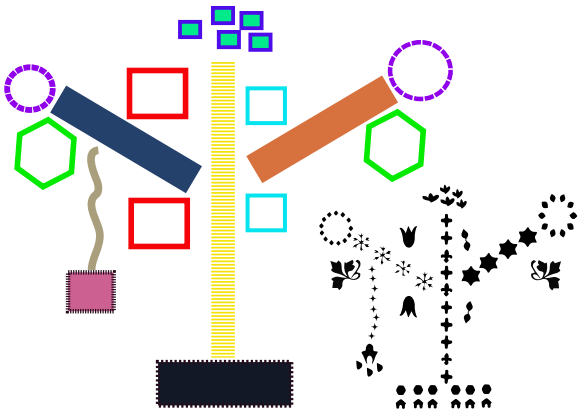

Fig. 4. An example of a layout template (left) and the pattern (right, in black) instantiated from it. The template is drawn using shape or curve primitives and annotated with desired element, orientation, size, spacing, symmetries, and stochastic parameters. The pattern is procedurally generated based on the layout and given specifications. Each colored primitive in the template corresponds to a different group.

function is composed of the following terms. The first term penalizes large distance values for elements that should be aggregated according to the training data, i.e., they belong to the same (human-annotated) group. We call these "positive" pairs. The second term penalizes small distance values for conflicting elements, i.e., ones that should not belong to the same group according to the training dataset. These are called "negative" pairs. Finally, we include a regularization term that prevents the parameter values from becoming arbitrarily large, which often happens when the learned function overfits the training dataset, thus has less chance to generalize to new inputs. The objective function is formulated as follows:

$$C(\mathbf{w},\mathbf{v}) = \sum_{P,P'\in\mathcal{P}} D^2(P,P') + \sum_{P,P'\in\mathcal{N}} \max(margin - D(P,P'),0)^2 \quad (3)$$
$$+\lambda_1||\mathbf{w}||^2 + \lambda_2||\mathbf{v}||^2,$$

where $\mathcal{P},\mathcal{N}$ are sets containing positive and negative pairs of elements or groups, respectively. The regularization parameters λ_1,λ_2 are set to 10^{-2} and 10^{-3} respectively through cross-validation. The *margin* value is set to 1 — its absolute value does not affect the learned parameters, but only scales distances such that negative pairs tend to have a margin of at least one unit distance.

Initialization and optimization. We initialize the feature weights $\mathbf{w}$ and structural encoder parameters to small random values. The parameters of the atomic element encoder are initialized based on the AlexNet parameters pre-trained on ImageNet [Russakovsky et al. 2015]. The convolution filters trained on ImageNet already partially capture useful shape information [Su et al. 2015], thus starting our optimization based on this initialization helps it converge to a better local minimum resulting in improved grouping performance. To minimize the cost function, we use batch gradient descent based on the Adam update rule [Kingma and Ba 2014] to iteratively optimize the parameters. To compute the required analytic gradients with respect to all the parameters, we use standard backpropagation.

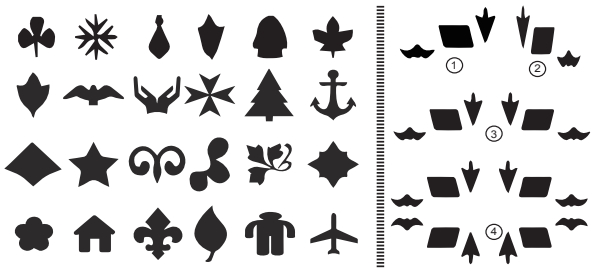

Fig. 5. We collected 86 atomic elements (some of them shown on the left) for pattern generation. During generation, the elements can undergo various affine deformations or symmetry transformations (right: ① → ② shows deformation; ① → ③ and ③ → ④ show different reflectional symmetry transformations), among others.

Training dataset. To train our network, we produced an image dataset containing a variety of 2D discrete graphical pattern arrangements. To create this dataset, we first collected 820 pattern layout templates drawn by 11 designers we hired. An example of such a template is shown in Figure 4. Each layout template consists of a set of primitives such as hexagons, rectangles, circles, and parametric curves. Each primitive is filled with atomic elements, randomly selected from a collection of icons containing various shapes (Figure 5). The atomic elements are placed following a stochastic procedure, which takes as input element placement attributes specified by the users, including element size, spacing, and orientation. For additional variety, in our procedural generation step we introduce noise on element placement including stochastic perturbations on input attributes, random occlusions between elements, and random affine transformations on the element shapes.

The ground-truth groups are specified by users through a simple UI where users simply assign different colors to primitives to indicate grouping. Each unique color corresponds to a different pattern group. Each time the stochastic procedure is executed, different pattern images with annotated groups are generated arising from the randomly selected element types, deformations, and perturbed orientation, size, and spacing attributes. Given the initial 820 pattern layouts, our procedure yielded 7,891 pattern images, which are provided in the supplementary material. We believe that the 820 templates contain interesting varieties and that our method was able to generalize well from those to different patterns, including real designs (see Section 5). From each of these images, we sampled 500 positive and 500 negative pairs of elements, which in total yielded a dataset of ~8M training pairs used to train our network.

5 RESULTS AND VALIDATION

We validated our method on a test dataset of graphical patterns mined from the web, through both a perceptual user study and standard numerical error measures. In what follows, we discuss the test dataset, show pattern grouping results obtained by our method, provide comparisons with alternative methods, present an analysis of the representations learned by the network, and report timings.

Test dataset. Graphical patterns are commonly encountered in coloring books for kids. Thus, we mined graphical pattern designs from the web using google and bing image search based on the phrase "coloring page". We note that we skipped coloring page images containing only organic shapes without any regularity. Each

	preset #group		auto #group	
	Rand index	purity	Rand index	purity
geometry distance	77.62%	73.13%	76.47%	76.64%
AlexNet	76.60%	70.96%	74.41%	71.16%
fine-tuned AlexNet	77.72%	73.12%	77.59%	80.21%
element encoder	78.34%	74.59%	77.72%	78.82%
structure encoder	78.79%	73.24%	77.35%	74.68%
element+structure enc.	80.28%	75.75%	80.03%	81.84%
our full measure	**83.05%**	**80.24%**	**83.58%**	**85.76%**

Table 1. Purity and Rand index values for different measures.

	Rand index	purity
affinity propagation	**83.05%**	**80.24%**
agglomerative (average linkage)	75.93%	71.13%
agglomerative (single linkage)	71.11%	68.38%
agglomerative (complete linkage)	76.76%	71.79%
k-means	80.85%	75.58%
Gaussian Mixture Models	80.92%	74.91%
normalized cuts	77.48%	71.72%
Tagger [Greff et al. 2016]	66.54%	55.21%

Table 2. Purity and Rand index for different clustering techniques.

contiguous region in these images often corresponds to a discrete graphical element. We then hired 8 human subjects (adults) to provide groupings for the downloaded images by using a "paint bucket" tool to flood image regions over elements belonging to the same perceived group with the same color. The resulting test set contains 214 images of annotated graphical pattern groups, which is provided in our supplementary material. The images in this test set frequently contain noise, such as occasional text on elements (e.g. "ENT" in Fig 1, left), element occlusions, or missing elements in a conjoining group.

Grouping results. Figure 6(a-h) displays groupings produced by our method for a variety of procedurally generated pattern images (not included in our training dataset), while Figure 6(i-p) displays produced groupings for downloaded pattern images belonging to our test set (see also Figure 1 for more test cases). In the last row, we show corresponding human-annotated groupings for several challenging cases (m-p). Plausible groupings are achieved under various scenarios: elements under rotational symmetries (a-b, f-g, i-j, l, etc.), translational symmetries (a, c-e, g-h, n, etc.), reflectional symmetries (a, d, e-f, h, k, m, etc.), elements laid out along various continuous curves (b-c, g-h, etc.) or geometric primitives (a, d-e), element occlusions (d, f, l, o, etc.), and graphical element designs depicting organic (i, m, p, etc.) or man-made objects (m-n). There are also imperfections in our groupings especially for organic objects (e.g., bears), which are challenging for our method since it was trained on non-organic-like shapes. Imperfections also exist when graphical elements do not strictly form contiguous regions (church entablature) or incur significant occlusion (e.g., ferris wheel, bear).

Clustering evaluation measures. Pattern grouping is an instance of clustering, i.e., each group of patterns corresponds to a distinct cluster. Thus, to numerically evaluate our method and alternatives against human-annotated groupings on our test set, we used two standard numerical error measures commonly used in clustering analysis: purity and Rand index. To compute clustering purity, each cluster produced by a grouping method under evaluation is assigned to the human-annotated cluster with the largest number of common atomic elements. Then we count the number of common atomic elements across all produced clusters divided by the total number of elements. The Rand index is an alternative measure of clustering similarity. We count the number of element pairs that are either in the same group or in different groups in both the produced and human-annotated groupings, divided by the total number of pairs.

Comparisons with alternative grouping measures. We compared our method with the following grouping measures based on prior work and "weaker" versions of our architecture:

- geometric distance: a first simple baseline is to define a geometric-based distance between two elements by first aligning them through the best rigid transformation, then measure the average Euclidean distance between nearest corresponding element silhouette points. Elements that are identical under translational, rotational, or reflective translation symmetry (isometries) have zero distance.
- AlexNet distance: a CNN-based baseline is to compute Euclidean distances between raw AlexNet features of two elements. To do this, we extract the image patch containing each element, resize these patches to 227×227 resolution, pass them through AlexNet trained on ImageNet (i.e., natural images), then extract the 4,096-dimensional feature vector per element based on its penultimate layer "fc7"; this layer offered best performance.
- fine-tuned AlexNet distance: as confirmed from our experiments, using raw AlexNet features trained on natural images does not generalize well to comparing discrete graphical patterns. A better CNN-based baseline is to instead compute Euclidean distances between their AlexNet "fc7" features fine-tuned on our training dataset under the same siamese configuration and objective.
- element encoder distance: a "weaker" version of our architecture is to use the atomic element encoder alone (i.e., AlexNet branches operating on multi-scale patches together including the dimensionality reduction layer). We compute Euclidean distances between the element 32-dimensional descriptors, as extracted by our atomic element encoder trained alone on our dataset.
- structure encoder distance: an alternative "weaker" version of our architecture is to use the structure encoder only. We compute Euclidean distances between the 8-dimensional element descriptors, as extracted by the structure encoder trained alone on our dataset.
- structure+element encoder distance: another "weaker" version of our architecture is to compute distances between the 40-dimensional element descriptors extracted by using both the structure and element encoder trained on our dataset. The difference from our full method is that we do not use here the element size and location.

The above competing measures are used by the affinity propagation clustering technique [Frey and Dueck 2007] to produce element groupings under two modes: (a) "preset #group" mode: we provide the affinity propagation technique with the desired (ground-truth) number of groups and force it to produce the same number of clusters per test case (using the implementation from [Wang 2010]), (b) "auto #group" mode: the default mode of affinity propagation works without specifying the target number of clusters, but uses instead an internal "preference" parameter, which controls the granularity of clustering and affects the resulting number of clusters. We

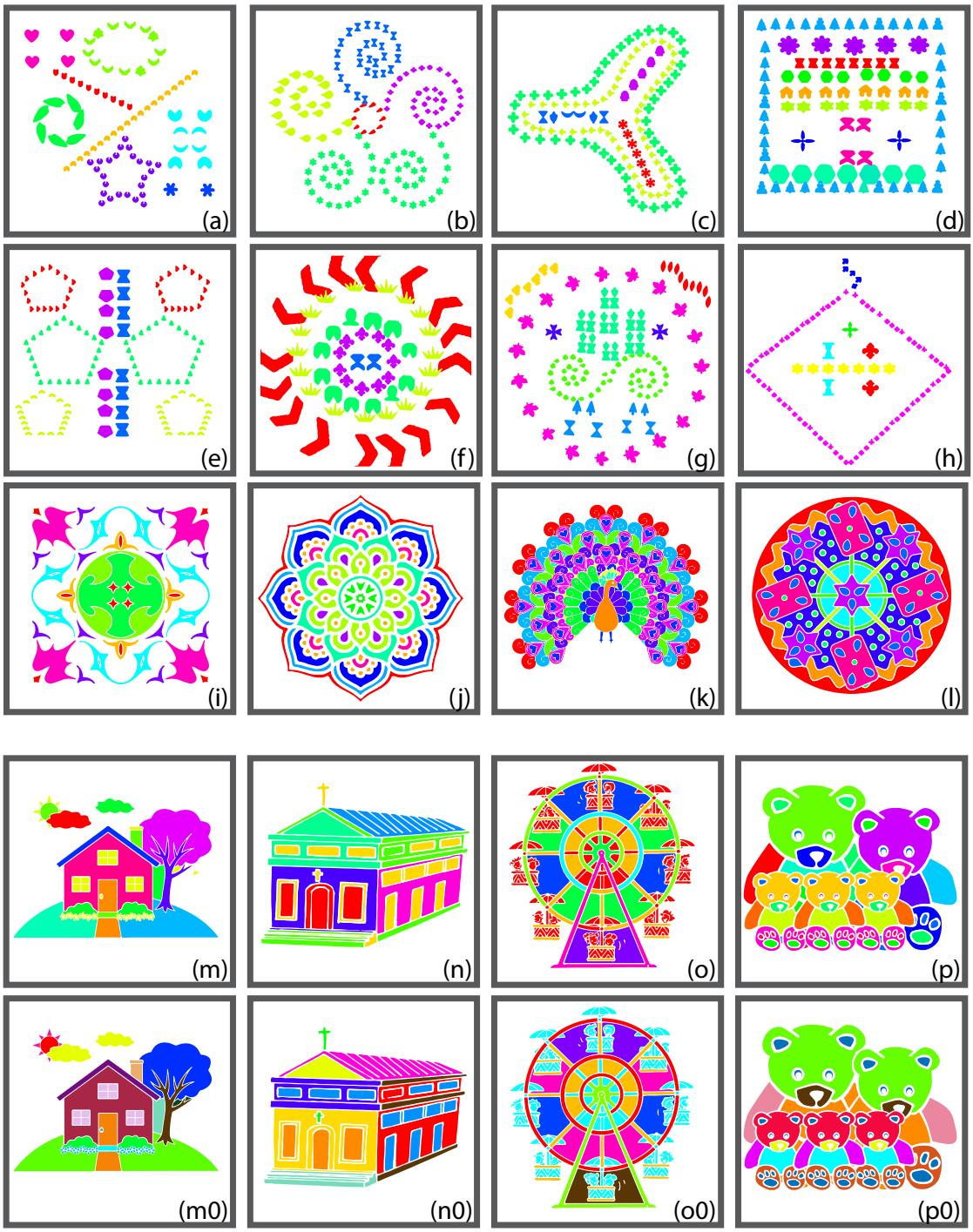

Fig. 6. Gallery of grouping results obtained by our method on procedurally generated patterns (a-h) and downloaded images (i-p). Results obtained with preset group counts include (m), (n), and (o); the rest was automatic. Some human-annotated groupings are shown (m0-p0).

greedily select the preference value that yields the highest Rand index on our hold-out validation set using grid search. The same preference value is used for clustering in our entire test set. We also tried other clustering techniques (see next paragraph) - we

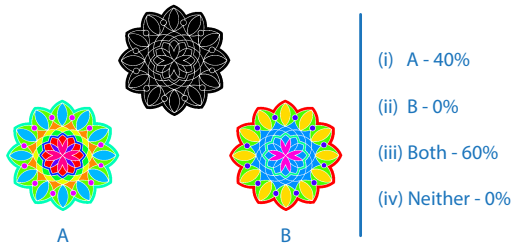

Fig. 7. Query layout shown to participants of our user study.

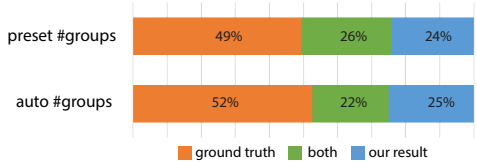

Fig. 8. Distribution of participant answers comparing human-annotated grouping versus different algorithmic choices.

similarly provided them with the target number of groups, or tuned their internal parameters on our validation set. Affinity propagation yielded the best performance. Table 1 reports purity and Rand index for our measure and alternatives under both "preset #group" and "auto #group" modes. Our full measure yielded better grouping performance compared to alternatives under both modes according to Rand index and purity. The performance differences between the "preset #group" and "auto #group" mode are subtle indicating that providing the desired number of clusters as additional input is not crucial.

Different clustering techniques. Table 2 reports purity and Rand index for alternative clustering techniques using our full measure, including k-means, Gaussian Mixture Models (initialized with k-means), normalized cuts [Shi and Malik 2000], and agglomerative clustering variants [Hastie et al. 2001]. We report performance under the "preset #group" mode, where we provide all techniques access to the target number of clusters. We also tried to tune the internal parameters (stopping criteria) of agglomerative clustering and normalized cuts using grid search on our validation set ("auto #group" mode), yet this resulted in slightly lower grouping performance for these techniques. Thus, we report results for these methods under the "preset #group" mode only.

In addition, we compared to Tagger [Greff et al. 2016], a deep neural network that jointly learns image representations and performs perceptual grouping. Tagger does not use our measure; it learns to group patterns based on its own internal representation. We trained it on the same training data as our method using their publically available implementation. The training and test images were down-sampled into the largest possible size 120×120 (6 times larger than the size of images used in the original implementation) so that we can fit their network in our GPU (12GB memory). We also tuned Tagger's hyper-parameters in our hold-out validation set. Tagger infers pixel-level probabilities for group assignments, thus for the test images, pixel-level probabilities were up-sampled to 800×800, and element-level group assignments were computed through averaging the probabilities of pixels they contained, then selecting the most likely group assignment. Table 2 shows that affinity propagation using our measure yielded the best performance in terms of purity and Rand index compared to other techniques, including Tagger.

User study. We also validated our method through a perceptual user study executed using the Amazon MTurk service. Each questionnaire included 50 queries, each showing three images originating from our test set: an image with the patterns on top, an image with the human-annotated groups (image A), an image produced using our measure based on affinity propagation under either the "preset #group" or "auto #group" mode (image B). The images were laid out as shown in Figure 7. Queries were shown at a random order, while each page was repeated twice (i.e., 25 unique queries), with A and B randomly flipped, to detect unreliable users giving inconsistent answers. Each query included the following question: "Which of the two groupings (A or B) seems more plausible to you?". Participants were asked to pick one of the following answers: "(i) A, (ii) B, (iii) can't tell - Both A and B are equally plausible, (iv) can't tell - Neither A nor B is plausible". To avoid any individual bias, we allowed each participant to complete only one questionnaire per category. Each query was answered by 5 different, reliable MTurk participants. We filtered out unreliable MTurk participants who gave two different answers to more than 8 out of the 25 unique queries in the questionnaire, or took less than 3 minutes to complete it.

Figure 8 demonstrates how frequently participants selected the human-annotated grouping versus our method in either of the two modes, and vice versa, on average. We also demonstrate how frequently they found both displayed groupings to be equally plausible (we do not show the option "none is plausible" since it represented less than 1% of the answers). For 26% of the comparisons, participants found that the human-annotated groupings were as plausible as the groupings produced by our measure in the "preset #group" mode. For 24% of the pattern comparisons, the produced groupings were found to be better than the human-annotated ones in this mode. In contrast, for 49% of the pattern comparisons, the human-annotated groupings were found to be better than the groupings of our method. Participants found the groupings produced by the automatic stopping mode of our method ("auto #groups") a bit worse than groupings produced when the desired number of groups is given ("preset #groups). In the ideal scenario, an algorithm should produce groupings that are exactly as plausible as human groupings. The results of our user study indicate that there is still room for improvement.

Learned representations. Finally, we looked more closely on the feature representations learned by our CNN. We found that after training, the element encoder network becomes sensitive to particular element styles or types. For example, the columns of Figure 9 show the top-5 elements that cause highest feature responses for particular entries (dimensions) of its learned 32-dimensional representation. For example, we found that the 5th

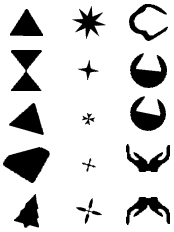

Fig. 9. Elements giving strongest feature responses at different descriptor entries of the learned CNN representation. Each column corresponds to a different descriptor dimension.

feature is sensitive to triangle-based element inputs (first column), the 8th feature is sensitive to star- or cross-like elements (second column), and the 15th feature is more sensitive to spider-like elements (third column). Interestingly, the elements that strongly activate these feature responses lie in various orientations, indicating that the learned CNN representation is likely to be invariant to rotations of the pattern elements.

Figure 10 visualizes output feature maps produced by the structure encoder network for various input pattern images. We noticed that the feature maps have similar values in areas where certain structural arrangements of elements exist. Elements with similar values in these areas will be favored to be grouped together by our measure. More specifically, the feature map shown in Figure 10(a) reveals which elements are related under a certain rotational symmetry, the one in Figure 10(b) reveals which elements are related under a reflectional symmetry, and the one in Figure 10(c) reveals which elements are layout according to a closed form (e.g., heart).

Running times. CNN training on our dataset took 30 hours using a TitanX GPU and a Xeon E5-2620 CPU. Once trained, at test time, our method computes the descriptors for all elements within an image in ~3 seconds. Computing distances for all pairs of elements and applying affinity propagation clustering needed about a second to produce the grouping per test image.

6 DISCUSSION AND LIMITATIONS

The ability to discern patterns of varying forms and complexity is one of the most fundamental human capabilities. We have developed what we believe to be the first data-driven, deep learning based approach to discrete graphical pattern grouping, enabling us to better understand the way such patterns are perceived. In particular, we learn a grouping measure from human-annotated pattern data, which allows us to produce meaningful graphical pattern groupings. Extensive experiments and validation demonstrate that within the realm of reasonable expectation of a data-driven approach, our method is able to produce robust grouping results.

Figure 6 also shows various grouping imperfections produced by our method, such as in the tree and bears images. These input images mainly consists of many organic patterns, which are stylistically quite different from the ones in our training data. Our CNN was trained on a synthetic dataset of 8K images with 86 unique elements, most of which represent non-organic-like patterns. Thus, it is not expected to generalize well to organic-like patterns or natural image patches. We expect that enriching our training dataset with organic-like patterns can help learn descriptors more suited for such elements, thus improving the generality of our algorithm. Another limitation is that our method assumes that elements are pre-segmented or form contiguous regions in the input images. Jointly segmenting an image into pattern elements and grouping them deserves future investigation, which may also be useful for natural image segmentation tasks.

In addition, our method does not incorporate explicit part labels, or "semantic" knowledge, which could help disambiguate such cases. It is also not designed to produce hierarchical groupings. Collecting hierarchical grouping data to train our network is significantly more involved than collecting "flat" grouping results. The downside is that our grouping measure leads to a single grouping, without any assurance of it being the most reasonable one. A true hierarchical grouping measure would require a global foresight when organizing a set of patterns in a top-down fashion. How to collect the appropriate data and learn such a measure is an intriguing question. Finally, our grouping measure is formulated as a weighted linear combination of shape-, context- and structure-aware descriptors, yet linearity is only a simplifying assumption. It would be interesting to enforce inherent non-linearity in the weighting mechanism through additional fully connected layers and non-linearities in our network.

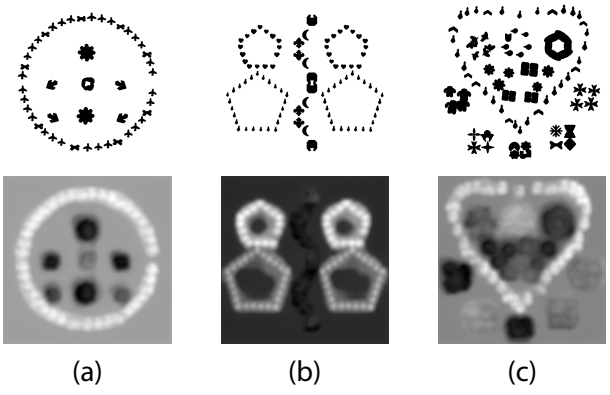

Fig. 10. Feature maps produced by the structure encoder network revealing (a) rotational symmetry, (b) reflectional symmetry, (c) closed form (heart shape).

7 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their comments and Dr. Ke Li for the help on experimental data preparation. Zou acknowledges support from the Science and Technology Plan Project of Hunan Province (Grant NO.: 2016TP1020) and the Program of Key Disciplines in Hunan Province. Kalogerakis acknowledges support from NSF (Grant NO.: CHS-1422441 and CHS-1617333), and the Massachusetts Technology Collaborative grant for funding the UMass GPU cluster. Tan acknowledges support from NSERC Canada (Grant NO.: 31-611663 and 31-611664). Zhang acknowledges support from NSERC Canada (Grant NO.: 611370 and 611649), and gift funds from Adobe Research.

REFERENCES

Sawsan AlHalawani, Yongliang Yang, Han Liu, and Niloy J. Mitra. 2013. Interactive Facades: Analysis and Synthesis of Semi-Regular Facades. *Computer Graphics Forum (Eurographics)* (2013), to appear.

Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. 2011. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 33, 5 (2011), 898–916.

Fan Bao, Michael Schwarz, and Peter Wonka. 2013. Procedural facade variations from a single layout. *ACM Trans. on Graph* 32, 1 (2013).

Connelly Barnes, Eli Shechtman, Dan B. Goldman, and Adam Finkelstein. 2010. The Generalized Patchmatch Correspondence Algorithm. In *Proc. of ECCV*.

M. Bokeloh, M. Wand, and H.-P. Seidel. 2010. A Connection between Partial Symmetry and Inverse Procedural Modeling. *ACM Trans. on Graph* 29, 4 (2010), 104:1–104:10.

Yilan Chen, Hongbo Fu, and Kin Chung Au. 2016. A Multi-level Sketch-based Interface for Decorative Pattern Exploration. In *SIGGRAPH ASIA 2016 Technical Briefs (SA '16)*. Article 26, 4 pages.

John H. Conway, Heidi Burgiel, and Chaim Goodman-Strauss. 2008. *The Symmetries of Things.* A K Peters/CRC Press.

Agné Desolneux, Lionel Moisan, and Jean-Michel Morel. 2002. *Gestalt theory and computer vision.* Springer.

Jacob Feldman. 2003. Perceptual grouping by selection of a logically minimal model. *Proc. of ICCV* 55 (2003), 5–25.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science* 315 (2007), 2007.

John H. Graham, Shmuel Raz, Hagit Hel-Or, and Eviatar Nevo. 2010. Fluctuating Asymmetry: Methods, Theory, and Applications. *Symmetry* 2, 2 (2010), 466–540.

Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Juergen Schmidhuber. 2016. Tagger: Deep Unsupervised Perceptual Grouping. In *Advances in Neural Information Processing Systems*. 4484–4492.

Paul Guerrero, Gilbert Bernstein, Wilmot Li, and Niloy J. Mitra. 2016. PATEX: Exploring Pattern Variations. *ACM Trans. on Graph* 35, 4, Article 48 (July 2016), 48:1–48:13 pages.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *Proc. CVPR*.

Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. 2015. MatchNet: Unifying feature and metric learning for patch-based matching. In *Proc. of CVPR*.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. 2016. Learning visual groups from co-occurrences in space and time. *International Conference on Learning Representations, Workshop paper* (2016).

Gaetano Kanizsa. 1980. *Grammatica del Vedere*. Il Mulino.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

W. Köhler. 1929. *Gestalt Psychology*. Liveright, New York.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*.

Michael Kubovy and Martin van den Berg. 2008. The whole is equal to the sum of its parts: A probabilistic model of grouping by proximity and similarity in regular patterns. *Psychological Review* 115, 1 (2008), 131–154.

Yanxi Liu, Hagit Hel-Or, Craig S. Kaplan, and Luc J. Van Gool. 2010. *Computational Symmetry in Computer Vision and Computer Graphics*. Foundations and Trends in Computer Graphics and Vision, Vol. 5. 1–195.

Michal Lukac, Daniel Sykora, Kalyan Sunkavalli, Eli Shechtman, Ondrej Jamriska, Nathan Carr, and Tomas Pajdla. 2017. Nautilus: Recovering Regional Symmetry Transformations for Image Editing. *ACM Trans. on Graph* to appear (2017).

Michael Maire, Takuya Narihira, and Stella X Yu. 2016. Affinity CNN: Learning pixel-centric pairwise relations for figure/ground embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 174–182.

Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. 2016. Convolutional oriented boundaries. In *European Conference on Computer Vision*. 580–596.

Niloy J. Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. 2012. Symmetry in 3D Geometry: Extraction and Applications. In *Proc. of Eurographics STAR Report*.

Liangliang Nan, Andrei Sharf, Ke Xie, Tien-Tsin Wong, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. 2011. Conjoining Gestalt Rules for Abstraction of Architectural Drawings. *ACM Trans. on Graph* 30, 6 (2011). 185:1-185:10.

S. Palmer. 1992. Common region: a new principle of perceptual grouping. *Cognitive Psychology* 24 (1992), 436–447.

Stephen E. Palmer. 1977. Hierarchical structure in perceptual representation. *Cognitive Psychology* 9, 4 (1977), 441–474.

Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. 2006. A Planar-reflective Symmetry Transform for 3D Shapes. *ACM Trans. on Graph* 25, 3 (2006).

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pat. Ana. & Mach. Int.* (2000).

Patricio Simari, Evangelos Kalogerakis, and Karan Singh. 2006. Folding meshes: hierarchical mesh segmentation based on planar symmetry. *Symp. on Geom. Proc.* (2006), 111–119.

Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. 2015. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *Proc. of ICCV*.

O. Stava, B. Benes, R. Mech, D. Aliga, and P. Kristof. 2010. Inverse Procedural Modeling by Automatic Generation of L-systems. *Computer Graphics Forum* 29, 2 (2010), 665–674.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *Proc. ICCV*.

Oliver van Kaick, Kai Xu, Hao Zhang, Yanzhen Wang, Shuyang Sun, Ariel Shamir, and Daniel Cohen-Or. 2013. Co-Hierarchical Analysis of Shape Structures. *ACM Trans. on Graph* 32, 4 (2013), Article 69.

Kaijun Wang. 2010. Fast Affinity Propagation Clustering under Given Number of Clusters. (2010). https://www.mathworks.com/matlabcentral/fileexchange/25722-fast-affinity-propagation-clustering-under-given-number-of-clusters

Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhiquan Cheng, and Yueshan Xiong. 2011. Symmetry Hierarchy of Man-Made Objects. *Computer Graphics Forum (Eurographics)* 30, 2 (2011), 287–296.

M. Wertheimer. 1938. *Laws of organization in perceptual forms*. 71–88.

H. Weyl. 1952. *Symmetry*. Princeton University Press.

Fuzhang Wu, Dong-Ming Yan, Weiming Dong, Xiaopeng Zhang, and Peter Wonka. 2014. Inverse procedural modeling of facade layouts. *ACM Trans. on Graph* 33, 4 (2014), 121:1–121:10.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *Proc. of ICML*.

Pengfei Xu, Hongbo Fu, Chiew-Lan Tai, and Takeo Igarashi. 2015. GACA: Group-Aware Command-based Arrangement of Graphic Elements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*. 2787–2795.

Jianwei Yang, Devi Parikh, and Dhruv Batra. 2016. Joint Unsupervised Learning of Deep Representations and Image Clusters. In *Proc. CVPR*.

Yi-Ting Yeh, Katherine Breeden, Lingfeng Yang, Matthew Fisher, and Pat Hanrahan. 2013. Synthesis of Tiled Patterns Using Factor Graphs. *ACM Trans. on Graph* 32, 1, Article 3 (Feb. 2013), 3:1–3:13 pages.

Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. 2016. LIFT: Learned Invariant Feature Transform. In *IEEE ECCV*.

S. Zagoruyko and N. Komodakis. 2015. Learning to compare image patches via convolutional neural networks. In *Proc. of CVPR*. 4353–4361.

Hao Zhang, Kai Xu, Wei Jiang, Jinjie Lin, Daniel Cohen-Or, and Baoquan Chen. 2013. Layered Analysis of Irregular Facades via Symmetry Maximization. *ACM Trans. on Graph* 32, 4 (2013), Article 121.

Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. 2012. A Survey on Unsupervised Outlier Detection in High-dimensional Numerical Data. *Stat. Anal. Data Min.* 5, 5 (2012).

C. Lawrence Zitnick and Piotr Dollár. 2014. Edge Boxes: Locating Object Proposals from Edges. In *Proc. of ECCV*.