To appear in ACM TOG 33(4).

A Similarity Measure for Illustration Style

Elena Garces
Universidad de Zaragoza

Aseem Agarwala
Adobe

Diego Gutierrez
Universidad de Zaragoza

Aaron Hertzmann
Adobe

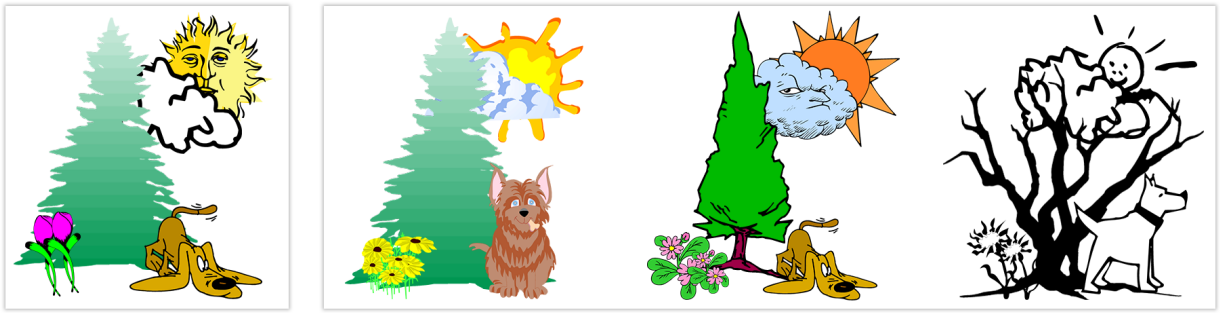

Figure 1: *The leftmost composition is generated by selecting from a dataset of 200K clip art searched with keywords: dog, tree, sun, cloud, and flower. Unfortunately, the styles of the clip art are inconsistent. Our style similarity function can be used to re-order the results of a search by style. The next three scenes are generated by fixing one element, and then searching for stylistically similar clip art with the above keywords. In each case, the additional clip art were chosen from the top twelve returned results (out of thousands).*

Abstract

This paper presents a method for measuring the similarity in style between two pieces of vector art, independent of content. Similarity is measured by the differences between four types of features: color, shading, texture, and stroke. Feature weightings are learned from crowdsourced experiments. This perceptual similarity enables style-based search. Using our style-based search feature, we demonstrate an application that allows users to create stylistically-coherent clip art mash-ups.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques— [I.3.m]: Computer Graphics—Miscellaneous;

Keywords: style, illustration, learning, crowd-sourcing

1 Introduction

Vector art is one of the most common forms of two-dimensional computer graphics. Clip art libraries contain hundreds of thousands of pieces of vector art designed to be copied into documents and illustrations. These collections are typically tagged by object categories; searches for common objects (e.g., "dog") yield huge numbers of results. However, there is another aspect of vector art that is currently much harder to search for: *style*. Clip art comes from many artists and many sources, in a vast range of visual styles, including sketches, woodcuts, cartoon drawings, and gradient-shading; some are very cartoony and whimsical, whereas others are more professional-looking. Because clip art comes from heterogeneous sources with very inconsistent tagging, these datasets lack any reliable annotation of artist or style.

While simulation of depiction style has long been a focus of non-photorealistic rendering [Gooch and Gooch 2001], little attention has been paid to understanding style, and no good tools exist for stylistic search or analysis in clip art datasets. Indeed, it is fundamentally difficult to define a simple function that describes these different styles. But, with the recent dramatic growth in the quantity of visual content available online and the rising popularity of remixed and mashup art [Lessig 2008], stylistic search could be valuable for many design applications.

This paper presents a *style similarity function* for clip art. That is, given two pieces of clip art, our function computes a real-valued measure of their style similarity, independent of content (Figure 1). We demonstrate style-based search, where clip art search results are sorted by similarity to a query artwork. We describe a clip art mashup application that uses style-based search to help users combine multiple pieces of stylistically-coherent clip art into a composition. For example, if a user has already placed a house and tree in a sketchy pen style onto the canvas, and then searches for a dog, our application re-orders search results so that dogs of a similarly sketchy pen style are shown first.

We compute our style distance function using a combination of crowdsourcing and machine learning. We gathered a stylistically-diverse collection of clip art. Then, for a large selection of clip art triplets, we gathered Mechanical Turk (MTurk) ratings of the form "Is clip art A more similar in style to clip art B or C?" We then learned a model of stylistic similarity from these ratings. The model is based on a set of features that we observe to be descriptive of the style of vector art. In total we compute 169 features in four categories: color, shading, texture, and stroke. The similarity function is a weighted L2 distance of the feature vectors; the weights are learned by maximizing the probability of the MTurk ratings. Learning with a sparsity prior produces a final weighting with 78 non-zero weights. We numerically evaluate the performance of our distance function on separate test data. We also perform user studies in which workers create mash-ups with and without style-based search. We find that raters judge mash-ups created with style-based search to be more stylistically coherent and generally preferable.

2 Related Work

Though there has been considerable effort in computer graphics on stylistic rendering, there has been less work in the analysis of artistic style. Willats and Durand [2005] provide an overview of the elements of pictorial style found in 2D illustrations. One approach to the algorithmic analysis of style is to learn a *generative* model; that is, a model that learns to create new examples

Figure 2: *For each feature category we show two pieces of clip whose style is very different. The color example contrasts a colorful illustration from a monochrome one. The shading example shows a gradient-shaded illustraton next to one with regions of constant color. The texture example shows an image with artistic patterns next to a woodcut illustration with more stochastic patterns. Finally, the stroke example pairs a sketchy illustration with lines of varying width next to smooth contours of constant width.*

of a style from scratch, such as generating new typefaces [Tenenbaum and Freeman 2000], and learning 3D hatching styles from examples [Kalogerakis et al. 2012]. A second class of approaches *transfer styles* from examples, such as transferring painting styles [Hertzmann et al. 2001], photographic styles [Bae et al. 2006], or curve styles of 2D shapes [Li et al. 2013]. Our work focuses on a different problem, namely, perceptual measures of stylistic similarity, rather than synthesis. To our knowledge, there is no previous work on perceptual similarity of vector art. Moreover, the above methods are not directly applicable. A generative model could theoretically be used to compute stylistic similarity; however, creating a generative model of clip art style from examples would be extraordinarily difficult.

Though image search and retrieval is a standard problem in vision and image analysis [Datta et al. 2008], recognition based on style is rare. Murray et al. [2012] classify photographs according to a few photographic styles. There are more examples in other domains; style similarity functions are used to recommend music [Aucouturier and Pachet 2002] and films [Bell and Koren 2007] based on examples of preferences. Shamir et al. [2010] describe style recognition for paintings. Doersch et al. [2012] recognize the style of street scenes that visually distinguish different cities. Our method instead focuses on styles of vector art.

The most related work to ours is a method for retrieval of sketches of similar style from an art dataset [Hurtut et al. 2011]. They propose features computed from stroke contours which are first extracted from the image to describe the style of line drawings. Their method only applies to black and white line drawings, whereas our method can also measure differences in color, shading, and texture. Also, our technical approach is different in that we collect data on the human perception of style, and fit our style similarity function to this data.

Finally, our mashup application is similar in motivation to recent work that supports search of photo databases for visually consistent content that can be combined into composites. Photo Clip Art [Lalonde et al. 2007] find objects in photos whose lighting and perspective are consistent with a target scene, while Sketch2Photo [Chen et al. 2009] generates photo composites from sketches while ensuring that the photo elements are visually consistent.

3 Clip Art Style Features

The first step in building a style similarity function is to define the numerical features that identify and distinguish clip art style. Although it is difficult to specify the exact characteristics that define style, there are a series of pictorial cues that can be used to differentiate one style from another. These cues include basic visual attributes like color, shading, and texture, as well as the actual marks such as lines, strokes, and regions [Durand 2002; Willats and Durand 2005]. Note that we do not need to decide a priori how these features differentiate style, or their relative weights; our goal is to create an overcomplete set of features that can be used by the learning algorithm to fit our similarity function to data.

Our features are computed on bitmaps rather than vector descriptions (e.g., SVG) of clip art for two reasons. First, computing on bitmaps gives us the flexibility to include clip art whether or not a vector version is available. Second, we observed a surprising variety of vector descriptions of similar content. For example, a simple black stroke could be defined with line, path, or polygon primitives, or even worse, be the result of adding a smaller foreground region to a black background region. Converting these representations into a consistent vector format is a research challenge by itself.

We identify four main aspects which we believe best characterize styles in clip art: **color**, **shading**, **texture**, and **strokes**. Together these form a 169-dimensional feature vector $\mathbf{x}$ for any individual piece of clip art. Figure 2 shows some representative examples of clip art whose styles are very different along each of the identified aspects. We now describe at a high level the list of features that form our feature vector; details are given in the Appendix.

Color. These features distinguish between different styles of color use; some styles we observe include black-and-white, monochrome, colorful, muted, and bright/saturated colors. Note that these statistics reflect styles of color usage rather than the individual colors used. The color features are implemented as statistics on the distribution of colors, including means and standard deviations of saturation, luminance, and hue, several measures of colorfulness, and the number of dominant colors. Finally, the percentage of gray and black pixels is useful to differentiate black and white from color.

Shading. These features distinguish types of shading. Some styles have a very cartoonish look, with sharp, simple color transitions; others have more realistic materials with smooth gradients. We describe shading with histograms of both color and grayscale gradients. The former captures the overall appearance and materials of colored images, while the latter captures transitions between shading and stroke lines, if any.

Texture. These features capture the presence of repeated patterns over the image. Texture defines the look of a depicted object at a small scale, and gives an intuition of how an object in an image would feel when touched. Texture features have a long history in computer vision; we use Local Binary Patterns (LBP) [Ojala et al. 2002; Zhu et al. 2010] and Haralick features [1973].

Strokes. The types of strokes used are a significant element of clip art style [Hurtut et al. 2011]. Clip art strokes typically vary in texture, thickness, and weight. Some of these characteristics are already captured by the LBP features, like thickness or uniformity of the stroke lines. We add additional stroke features using the Stroke Width Transform (SWT) [Epshtein et al. 2010], which was originally developed to recognize text in natural images. The SWT is a

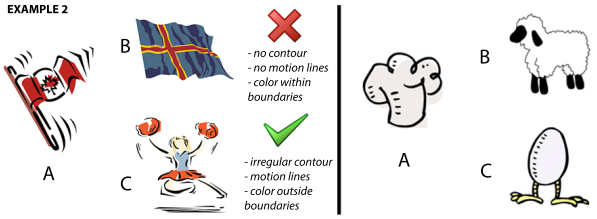

Figure 3: *Screenshots of our MTurk similarity collection interface. Left: An example given to users at the beginning of the test to make sure they understood the question asked. Right: An actual comparison triplet.*

local operator which approximates the width of the stroke at each pixel.

4 Collecting Similarity Information

We use two sources of clip art to train our models: clip art from Art Explosion[1], a commercial collection of over 200,000 pieces of clip art, and a collection of 3,600 clip art pieces that is included with Microsoft Office. For the former collection we used crowdsourcing to collect data on stylistic similarity. In contrast, the latter, smaller collection is already organized into groups of stylistic similarity.

We manually selected 1000 images from the Art Explosion dataset that cover a wide range of styles and subjects. We used Mechanical Turk (MTurk) raters to collect style information. Each test (a HIT in Mechanical Turk terminology) consisted of fifty questions. We gathered data in the form of relative comparisons [McFee and Lanckriet 2011; Schultz and Joachims 2003] since they are much easier for human raters to provide than numerical distances. Each question showed three pieces of clip art A, B, and C, and the MTurk rater is asked: "Is A more similar to B or to C?" (Figure 3.) For the Microsoft data, we automatically generated relative comparisons through random sampling constrained so that two of the three samples in each relative comparison come from the same style group.

Each HIT was preceded by a short training session that included a few trial relative comparisons with obvious answers; the users could only access the real test once they correctly answered all the trial questions. A total of 313 users took part, 51.4% female. 56.5% declared some artistic experience, and an additional 7.3% claimed some professional design experience. The duration of each HIT was approximately ten minutes, for which we paid $0.30. Five control questions were included in each HIT; HITs with two errors in the control questions were rejected, with a rejection rate of 21.5%.

5 Learning Similarity

This section describes our approach for learning stylistic similarity based on the feature vector defined in Section 3 and the training data from Section 4. Our learning approach uses a combination of previous techniques that works well for our application. Let $\mathbf{x}$ and $\mathbf{y}$ be the feature vectors for two pieces of clip art. We aim to learn a Euclidean distance metric

$$d(\mathbf{x},\mathbf{y}) = \| \mathbf{x}-\mathbf{y} \|_{\mathbf{W}} = \sqrt{(\mathbf{x}-\mathbf{y})^T \mathbf{W} (\mathbf{x}-\mathbf{y})} \qquad (1)$$

parameterized by a diagonal matrix $\mathbf{W}$. This problem is well-studied and known as *metric learning* [Kulis 2013; Schultz and Joachims 2003]. Our problem is further complicated by the fact that crowdsourced relative comparisons are not always reliable; there are several approaches to modeling worker reliability both in classification [Welinder et al. 2010] and search ranking [Chen et al. 2013]. We minimize this reliability problem in two ways. First, we use a number of control and training questions to reject bad workers. Second, we use a logistic formulation of the probability of each rating [Tamuz et al. 2011] that expects more noise for relative comparisons with less clear answers.

Specifically, we use the metric learning approach of Donovan et al. [2014], who adapt the logistic formulation of Tamuz et al. [2011] to the scenario of learning from features: Given clip arts A, B, and C, we define $q = 1$ if the rater states that A and B are more similar, and $q = 0$ if the rater states A and C are more similar. We parameterize the model by the diagonal of the weight matrix: $\mathbf{w} = \mathrm{diag}(\mathbf{W})$. We model the probability that a user rates $q = 1$ given the tuple as:

$$P^A_{BC}(q=1) = \sigma(d(\mathbf{x}_A,\mathbf{x}_C) - d(\mathbf{x}_A,\mathbf{x}_B)) \qquad (2)$$

$$\sigma(x) = 1/(1+\exp(-x)) \qquad (3)$$

In addition to this model, we aim to regularize and sparsify the weight vector. We thus assume a Laplacian prior on the weights:

$$p(\mathbf{w}) \propto \exp(-\lambda ||\mathbf{w}||_1) \qquad (4)$$

where λ is a regularization weight. Given a set of Turker ratings $\mathcal{D} = \{(A_i, B_i, C_i, q_i)\}$, we learn the weights $\mathbf{w}$ by Maximum A Posteriori estimation, which entails minimizing the following objective function:

$$E(\mathbf{w}) = -\sum_{i=1}^{|\mathcal{D}|} \log\left(P^{A_i}_{B_i C_i}(q_i)\right) + \lambda ||\mathbf{w}||_1 \qquad (5)$$

where i indexes over the training tuples. We perform optimization using L-BFGS [Zhu et al. 1997].

We set the regularization weight λ by five-fold cross-validation on the training set. After training, all weights with $w < 0.02$ are set to zero.

6 Similarity Function Evaluation

We now evaluate the influence of the regularization term, the performance of the learning process, and the quality of the training data. The training set includes 25,540 tuples gathered via Amazon Mechanical Turk and 25,000 tuples generated from the labeled data from Microsoft.

6.1 Feature Selection

The use of L1 regularization encourages a sparse set of weights. Through cross-validation, the regularization weight was set to $\lambda = 1.3$. After the thresholding step, we reach a final weight vector with 78 non-zero weights; the 91 zero-weight features can be ignored. The weights are given in the Supplemental Material. Some features are deemed irrelevant (e.g., the entropy of both luminance and color), while others can be covered with a smaller set of features (e.g., by using a few bins of LBP instead of all). The highest weighted feature is the number of unique hues; the number of RGB colors is the second highest. Several bins of the LBP, Haralick, Color Gradient, and Stroke width features are also highly weighted. The weights show that these high-dimensional features can be simplified to lower-dimensional combinations for our application.

[1] www.novadevelopment.com

	MTurk		MS
	Raw	Majority	
Learned Weights	0.72	0.81	0.95
Uniform Weights	0.68	0.75	0.94
Humans	0.68	0.74	N/A
Oracle	0.83	1	N/A

Table 1: *Accuracy of our method (with and without training) and two baselines, on both the MTurk and Microsoft testing data. Higher values are better (see text for details).*

	MTurk		MS
	Raw	Majority	
Learned Weights (MTurk + MS)	1.75	1.57	1.18
Learned Weights (MTurk only)	1.76	1.64	1.30
Learned Weights (MS only)	2.52	1.77	1.11
Uniform Weights	1.83	1.73	1.39

Table 2: *Perplexity of our method on both the MTurk and Microsoft testing data. Lower values are better (see text for details).*

6.2 Evaluation on a Testing Set

We gathered a new set of relative comparisons to form a separate testing set to evaluate our model. We sampled 1,000 new tuples from the Art Explosion collection, and 10,000 tuples from the Microsoft labeled data. We used MTurk to obtain 10 ratings per tuple for the Art Explosion data; this redundancy helps us to understand which tuples have clear answers. We removed tuples with high disagreement, i.e., tuples with MTurkers split 5-5 or 6-4 in their judgment of which pair is more similar. This left 633 reliable comparisons, each with 70% or more agreement. Disjoint training tuples were used between the training and test sets, though both sets of tuples used clip art pieces drawn randomly from the same clip art collection.

We evaluate performance by two metrics on the test set: *accuracy* and *perplexity*. Accuracy is the percentage of testing tuples correctly predicted by our method. For MTurk tuples, accuracy can be computed in two ways: *raw* and *majority*. Raw accuracy counts each of the 10 opinions per tuple separately; majority assumes the majority opinion is correct and assigns all votes to the winner. So, an ideal predictor which always chooses the majority opinion would have a majority accuracy of 1, but a raw accuracy of less than 1 assuming there is disagreement between human raters. In either case, a completely random predictor would have an accuracy of 0.5.

Perplexity Q is a standard measure of how well a probability model predicts a sample; it takes into account the uncertainty in the model, giving higher weight to predictions where the model outputs higher confidence (i.e., P^A_{BC} close to 1 or 0). It is given by

$$Q = 2^{-(\ln P(\mathcal{T}))/|\mathcal{T}|} \tag{6}$$

where $P(\mathcal{T})$ is the probability of the test set according to a given model, computed by Equation 3 over all test tuples. The perplexity is 1 for a model that makes perfect, confident predictions at all times. The perpexity is 2 for a model that outputs 0.5 (total uncertainty) for all evaluations. The perplexity is worse for a model that makes highly confident but wrong predictions. Perplexity can also be computed with raw and majority data.

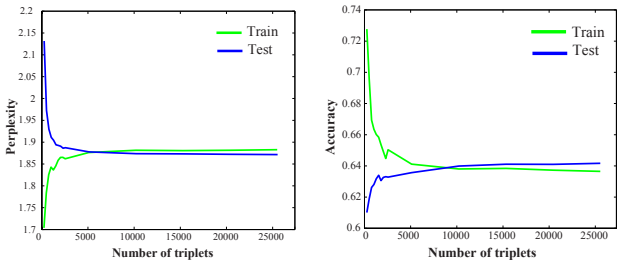

Figure 4: *Perplexity (left) and accuracy (right) on the test data as a function of the number of MTurk tuples used during training.*

We show accuracy data in Table 1 and perplexity data in Table 2, both for MTurk relative comparisons and Microsoft data. We show the results of our model both with uniform weights $\mathbf{w} = [1...1]^T$, and with weights learned from training data. Note that there is no need to separate the Microsoft data into raw and majority, since there is no disagreement. We compare our results with two baselines in addition to a random baseline. The *oracle* predictor has access to all the human ratings and always gives the majority opinion; note that its raw accuracy is not 1 due to human rater disagreement. The *human* accuracy is a measure of how well the average individual human performs relative to the majority; it is computed as an average over each human rater's percentage of agreement with other raters on the same tuples. Note that perplexity cannot be computed for the oracle and human models since they are not probability distributions.

For the MTurk data, the accuracy of our model is roughly equal to average human accuracy without training; with trained weights, our model performs better than human accuracy. Our model is able to predict the majority opinion 81% of the time. Not surprisingly, the oracle predictor is still significantly better than our model. Our model is 95% accurate on the Microsoft data. This data is easier, since in each tuple two of the clip art pieces have the same style; tuples with three different styles (as is common in the MTurk data) are more subjective.

We also experimented with only training on the Microsoft or MTurk datasets. Training on only the Microsoft data performed poorly on the MTurk data, while training on only the MTurk data performed reasonably on the Microsoft data. The combination of both datasets during training performs the best or nearly the best on both testing sets (Table 2).

We can check whether we have collected enough tuples by holding back some of the training data and observing accuracy and perplexity (Figure 4). We can see that we have collected more than enough randomly sampled data; improvements stop at around 10,000 triplets relative to the 25,000 we collected. However, our triples are randomly sampled; it may be that sampling triples closer in style could add further discriminative power [Tamuz et al. 2011].

6.3 Failure Cases

Our similarity measure disagrees with the MTurk majority 19% of the time; we include the 25 worst examples where our probability is most inconsistent with the MTurk opinions in the Supplemental Materials. We show two typical examples in Figure 5. In the left example, the style of all three clip arts is very different, and it is surprising the MTurk opinion is so consistent either way. The right example shows another common error; our metric generally believes two clip arts with color are more similar with each other than to a black and white clip art. However, in this case, the iconic

nature of the examples overwhelms other differences for humans.

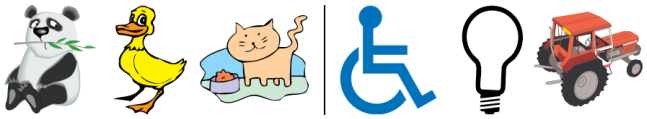

Figure 5: *Two tuples incorrectly labeled by our similarity function. In both cases, Turkers agreed (by a 9-1 margin) that the first clip art is more similar to the second than to the third, whereas our algorithm scores the first and third as more similar.*

7 Applications

We apply our style similarity metric in three ways: to cluster and visualize the space of clip art styles, to perform search, and to support a mashup application for creating compositions of clip art.

7.1 Clip Art Style Visualization

We use our style distance function as a basis for visualizing the diverse styles of clip art in our dataset. In particular, we use the popular t-SNE [van der Maaten and Hinton 2008]; this technique maps a high dimensional feature vector to a 2D space where similar styles are located close to each other. To create the visualization in Figure 6, we reduce the entire dataset to 100 examples by k-means on the $\mathbf{Wx}$ values, select only dogs, and then perform t-SNE. We can observe a clear separation of style; colorfulness increases from top to bottom, while stroke complexity varies left to right.

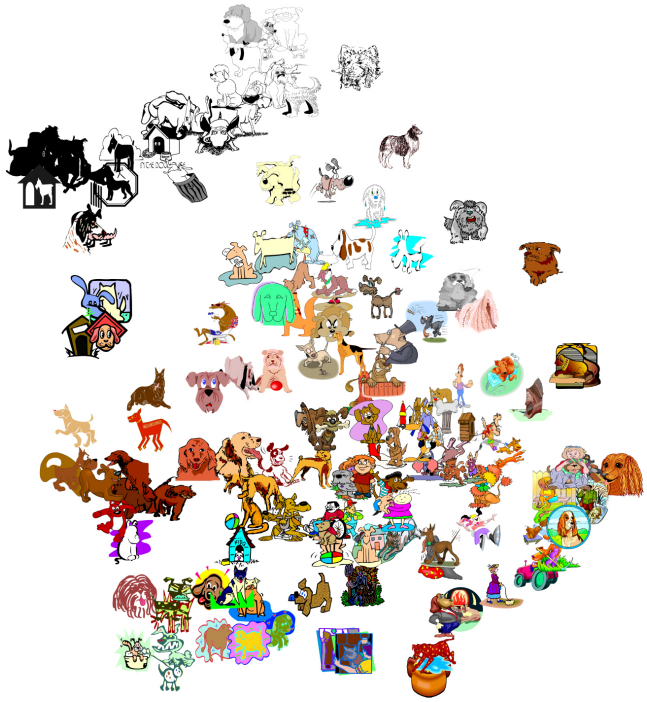

Figure 6: *A 2D embedding of clip art styles, computed using t-SNE, shown with "dog" examples.*

7.2 Search

Figure 10 shows typical results of search queries using our method; we show an additional 500 examples in the Supplemental Material. Each image in the left column shows a query image. The next column shows the results from the dataset that our method judges to be most similar. In each case, the algorithm appears to have recovered at least some artwork from the exact same artist and style, after searching in the entire dataset of 194,663 pieces. The other six columns show the top 3 results each for keyword queries, i.e., the three most similar *cat* images, the three most similar *fish* images, and so on. The amount of clip art in each category ranges from 598 (*sky*) to 1836 (*man*). In many cases, the algorithm does not find matches by the same artist, but finds excellent matches nonetheless, with similar stroke styles, similar fill styles, and so on. For example, the dinosaur query is drawn in a woodcut style with solid color fills. The first two *cat* results appear to be from the same artist, whereas the third is a tiger that is not a woodcut but is similar nonetheless. The other examples show cases where woodcut styles are found, or, when no more woodcuts can be found, similar non-woodcuts are returned.

Search is typically evaluated with precision-recall, where the goal is to return search results that are in the same category as the query. In our case, most of our data is not cleanly separable into relevant and irrelevant categories. The Microsoft data is separated into groups; however, this represents only 1% of our overall data. Also, we found that many of the groups had similar styles to each other, which means that clip art from one group is often relevant to queries from another group.

7.3 Clip-Art Mash-ups

We also demonstrate the usefulness of our similarity metric with a simple clip-art mashup application (Figure 7). The application allows users to search for and combine multiple pieces of clip art into a composition. Our clip art library is organized into 13 common categories (e.g., *dog, tree, house*); the user can also search by keyword to find objects not in common categories, or to add a modifier (e.g., *running dog*). We provide 11 pre-made backgrounds that the user can select from, including a blank white background. Clip art is added to the composition by dragging and dropping from the search results; it can be resized or rotated, and the layer order can be modified. The app is implemented in HTML5. We use Apache Lucene to extract keywords from clip art file names and tags. We show several examples of mash-ups created with our app in Figure 1 and the Supplemental Video.

During search, clip art that matches the keyword and category query are sorted by their style similarity to all currently-used clip art (averaged over the existing clip art). The sort order automatically updates whenever clip art is added or removed from the canvas.

When there is no clip art in the current canvas, a naïve approach would be to order the search results randomly. However, some styles are more common in the dataset than others, leading to some styles not being represented in the results. As in other search problems [Radlinski et al. 2009], it is important to produce results with diverse styles. We produce diversity as follows: In advance, we produce a two-level hierarchical clustering; the entire dataset is clustered into 100 clusters by k-means on $\mathbf{Wx}$ values, and then these cluster centers are clustered again to produce seven top-level clusters. Then, at search time, the first seven search results are sampled sequentially from the seven top-level clusters. This process repeats to generate the next seven results, and so on. Within a top-level cluster, the second-level clusters are also sampled sequentially, to avoid repetition within the cluster.

Figure 7: *Our mash-up interface. The user searches for clip art in the left panel by typing keywords and/or by selecting categories from the drop-down menu. Results are shown below the search button, sorted by stylistic similarity to already-selected clip art.*

Figure 8: *Typical mash-ups created by Turkers using our similarity.*

Evaluation. To evaluate the impact of our similarity measure on mash-up creation, we performed two different MTurk studies. In each study, users were asked to create high-quality compositions. Some users were provided with a version of the application in which search results are sorted by similarity to the existing composition, and others received an interface which sorts search results randomly. Users were not told of this difference.

In the first study, users were given an open-ended task: they were free to choose the topic of the composition, the background, and the number of images to use. In the second study, we asked for a specific story, fixed the background, and required the users to include at least four different pieces of clip art and perform at least four different searches. Study details are in the Supplemental Material.

Figure 9: *Typical mash-ups created by Turkers without our similarity metric. All results are included in the Supplemental Material.*

We gathered 38 compositions for the first study (19 with our metric on), and 95 compositions for the second (47 with our metric on). We show several typical examples of compositions both with our metric and without in Figures 8 and 9, respectively (we include all compositions in the Supplemental Material). We then asked a separate pool of MTurk workers to evaluate the compositions on both style coherence and general preference. Specifically, we performed a 2AFC test between randomly sampled compositions with and without our metric. We asked the questions: *which composition has a more coherent style?*, and *which compositions do you like better?* For both tests, we perform a one-sample, one-sided t-test comparing the mean of users preferences against the null hypothesis (people have no preference, $\mu_0 = 0.5$). Compositions created with our metric were perceived as having a more coherent style (67% of agreement for the first and 69% for the second, with $p < 0.01$), and participants liked them more (63% for the first and 66% for the second, with $p < 0.01$). The effect sizes for style coherence were 1.4 and 2.4 for the first and second experiments, and 1.4 for general preference for both experiments; these effects are large.

When using the similarity-based interface, the clip art pieces used in the mash-ups had a mean position of 24 in the search results (confidence interval: ± 4.4), whereas without our metric, the mean position was 40 ± 7.1. This indicates that, although users need to look through many results in order to find content that matches their goals, our interface cuts the length of the search nearly in half.

8 Conclusion

We have presented a similarity function for illustration style that is trained from crowdsourced data on the human perception of similarity. We also demonstrate a mash-up application for combining clip art of a consistent style into a scene. There are a number of ways we could improve our system. We could collect data on how humans name illustration styles, so that a user could ask for a "sketchy dog." We could learn relative attributes of style [Parikh and Grauman 2011], so that a user could ask for a dog similar to the current one but "more colorful." Our metric learning technique is fairly simple and there may be other, possibly non-linear methods that work as well or better; we have made our data public for further experimentation. Finally, a more fundamental problem is to understand and parse the elements that form an illustration, such as outlines, fills, object identity, and so on. (Even identifying the outline strokes in the vector art in our libraries is non-trivial.) This analysis could lead to richer and more accurate analysis of illustration style and similarity.

Beyond our application of clip art style, we believe that recognizing the style of visual content will become increasingly important as the amount of online content increases and remixing becomes more and more common. Design-focused social networks such as Pinterest can also benefit from the ability to search by style, without relying on manual tagging. While visual recognition of semantics is a mature field, recognition of style is less well explored and perhaps more challenging, since style perception is more subjective. Given the appropriate domain-specific features, our approach could easily generalize to other kinds of style similarity, e.g., pictures, fonts, graphic designs, architectural elements, etc. We believe that training from data on the human perception of style is a promising and general approach to this problem.

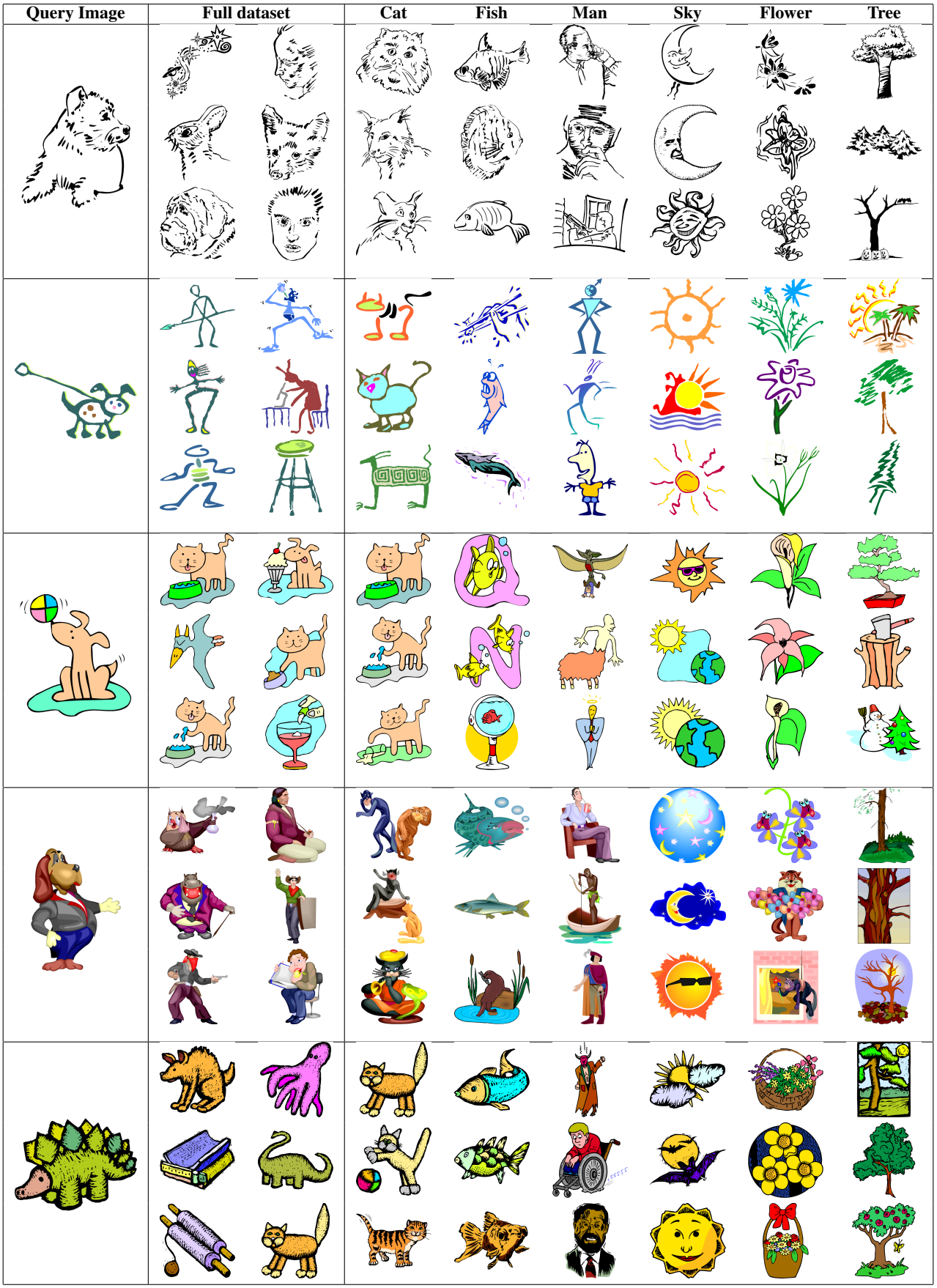

Figure 10: *Style-based search. The leftmost artwork is the query image. The next column shows the most similar images in the full dataset of 200k images. The remaining columns show the top three search results in six different categories.*

Acknowledgments We want to thank the reviewers for their insightful comments, the participants of the experiments, Carlos Bobed for his help setting up the experiments, Peter O'Donovan for sharing his regression code, and Daniel Osanz for some designs. This research has been funded by the European Commission, Seventh Framework Programme, through projects GOLEM (Marie Curie IAPP, grant: 251415) and VERVE (ICT, grant: 288914), the Spanish Ministry of Science and Technology (TIN2010-21543), and a generous gift from Adobe Systems. The Gobierno de Aragón additionally provided support through the TAMA project and a grant to Elena Garces.

References

AUCOUTURIER, J.-J., AND PACHET, F. 2002. Music Similarity Measures: What's the Use? In *Proc. SMIR*, 157–163.

BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Trans. Graphics 25*, 3, 637–645.

BELL, R. M., AND KOREN, Y. 2007. Lessons from the Netflix Prize Challenge. *SIGKDD Explor. Newsl. 9*, 2, 75–79.

CHEN, T., CHENG, M.-M., TAN, P., SHAMIR, A., AND HU, S.-M. 2009. Sketch2photo: Internet image montage. *ACM Trans. Graphics 28*, 5, 124:1–124:10.

CHEN, X., BENNETT, P. N., COLLINS-THOMPSON, K., AND HORVITZ, E. 2013. Pairwise Ranking Aggregation in a Crowd-sourced Setting. In *Proc. WSDM*, 193–202.

DATTA, R., JOSHI, D., LI, J., AND WANG, J. Z. 2008. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys 40*, 2, 5:1–5:60.

DOERSCH, C., SINGH, S., GUPTA, A., SIVIC, J., AND EFROS, A. 2012. What Makes Paris Look like Paris? *ACM Trans. Graphics 31*, 4, 101:1–101:9.

DURAND, F. 2002. An Invitation to Discuss Computer Depiction. In *Proc. NPAR*, 111–124.

EPSHTEIN, B., OFEK, E., AND WEXLER, Y. 2010. Detecting Text in Natural Scenes with Stroke Width Transform. In *Proc. CVPR*, 2963–2970.

GOOCH, B., AND GOOCH, A. 2001. *Non-Photorealistic Rendering*. AK Peters Ltd.

HARALICK, R. M., SHANMUGAM, K., AND DINSTEIN, I. 1973. Textural Features for Image Classification. *IEEE Trans. Systems, Man and Cybernetics 3*, 6, 610–621.

HASLER, D., AND SUSSTRUNK, S. 2003. Measuring Colourfulness in Natural Images. In *Proc. SPIE: Human Vision and Electronic Imaging*, vol. 5007, 87–95.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image Analogies. In *Proc. SIGGRAPH*, 327–340.

HURTUT, T., GOUSSEAU, Y., CHERIET, F., AND SCHMITT, F. 2011. Artistic line-drawings retrieval based on the pictorial content. *ACM J. Computing and Cultural Heritage 4*, 1, 1–23.

KALOGERAKIS, E., NOWROUZEZAHRAI, D., BRESLAV, S., AND HERTZMANN, A. 2012. Learning Hatching for Pen-and-Ink Illustration of Surfaces. *ACM Trans. Graphics 31*, 1:1–1:17.

KULIS, B. 2013. Metric learning: A survey. *Foundations and Trends in Machine Learning 5*, 4, 287–364.

LALONDE, J.-F., HOIEM, D., EFROS, A. A., ROTHER, C., WINN, J., AND CRIMINISI, A. 2007. Photo clip art. *ACM Trans. Graphics 26*, 3, 3:1–3:10.

LESSIG, L. 2008. *Remix: Making Art and Commerce Thrive in the Hybrid Economy*. Penguin Press.

LI, C., MEMBER, S., AND CHEN, T. 2009. Aesthetic Visual Quality Assessment of Paintings. *IEEE J. Sel. Topics in Signal Processing 3*, 2, 236–252.

LI, H., ZHANG, H., WANG, Y., CAO, J., SHAMIR, A., AND COHEN-OR, D. 2013. Curve Style Analysis in a Set of Shapes. *Computer Graphics Forum 32*, 6, 77–88.

MCFEE, B., AND LANCKRIET, G. 2011. Learning Multi-modal Similarity. *J. Machine Learning Research 12*, 491–523.

MURRAY, N., BARCELONA, D., MARCHESOTTI, L., AND PERRONNIN, F. 2012. AVA: A Large-Scale Database for Aesthetic Visual Analysis. In *Proc. CVPR*, 2408–2415.

O'DONOVAN, P., LĪBEKS, J., AGARWALA, A., AND HERTZMANN, A. 2014. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Trans. Graphics 33*.

OJALA, T., PIETIKÄINEN, M., AND MÄENPÄÄ, T. 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. PAMI 24*, 7, 971–987.

PARIKH, D., AND GRAUMAN, K. 2011. Relative attributes. *In Proc. ICCV*, 503–510.

RADLINSKI, F., BENNETT, P. N., CARTERETTE, B., AND JOACHIMS, T. 2009. Redundancy, Diversity and Interdependent Document Relevance. *SIGIR Forum 43*, 2.

SCHULTZ, M., AND JOACHIMS, T. 2003. Learning a Distance Metric from Relative Comparisons. In *Proc. NIPS*.

SHAMIR, L., MACURA, T., ORLOV, N., ECKLEY, D. M., AND GOLDBERG, I. G. 2010. Impressionism, Expressionism, Surrealism: Automated Recognition of Painters and Schools of Art. *ACM Trans. Applied Perception 7*, 2, 8:1–8:17.

TAMUZ, O., LIU, C., BELONGIE, S., SHAMIR, O., AND KALAI, A. 2011. Adaptively Learning the Crowd Kernel. In *Proc. ICML*, 673–680.

TENENBAUM, J. B., AND FREEMAN, W. T. 2000. Separating Style and Content with Bilinear Models. *Neural Computation 12*, 6, 1247–1283.

VAN DER MAATEN, L., AND HINTON, G. E. 2008. Visualizing High-Dimensional Data Using t-SNE. *J. Machine Learning Research 9*, 2579–2605.

WELINDER, P., BRANSON, S., BELONGIE, S., AND PERONA, P. 2010. The Multidimensional Wisdom of Crowds. In *Proc. NIPS*, 2424–2432.

WILLATS, J., AND DURAND, F. 2005. Defining pictorial style: Lessons from linguistics and computer graphics. *Axiomathes 15*, 3, 319–351.

ZHU, C., BYRD, R. H., LU, P., AND NOCEDAL, J. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. on Mathematical Software 23*, 4.

ZHU, C., BICHOT, C.-E., AND CHEN, L. 2010. Multi-scale Color Local Binary Patterns for Visual Object Classes Recognition. In *Proc. ICPR*, 3065–3068.

A Feature Vector

We now give details of each feature in Section 3. Note that some of these features are more relevant to clip art style than others, and some features are completely removed by L1 regularization (Section 6.1); we include all the features we tested for completeness. To compute features, we render each clip art image to 400x400 pixels. For each image, we define a mask Ω that approximately covers the clip art. We select all non-white pixels, perform a morphological expand operation of ten pixels, and then fill the remaining holes. All the statistics are computed only on the domain of Ω.

Color. The first color features are scalar values, defined as follows: Standard deviation of hue; average saturation; standard deviation of saturation; average luminance; standard deviation of luminance; entropy of the luminance histogram, after quantizing it to 256 bins; entropy of the RGB histogram, after quantizing it to 512 bins; colorfulness, computed by the measure of Hasler and Susstrunk [2003]; colorfulness, computed as:

$$\frac{1}{|\Omega|}\sum_{p\in\Omega}|R_p - G_p| + |G_p - B_p| + |B_p - R_p|\,; \tag{7}$$

percentage of pixels that are black; and percentage of pixels that belong to the most dominant color.

Additionally, we define a few features in terms of a 20-bin histogram $C(h)$ of hue h, omitting pixels with saturation less than 0.1, similar to Li et al. [2009]. Then, we include a feature for the frequency of the most common hue ($\max_h C(h)$), and a feature for the number of dominant hues:

$$\#(h \mid C(h) > 0.05 \max_h C(h)) \tag{8}$$

We also include the same two features above applied to a quantized RGB histogram; that is, the number of pixels in the most frequent color, and the number of dominant colors.

Shading. We concatenate eight-bin gradient magnitude histograms at two resolutions of the image, 1x and 0.5x. The histograms are normalized by $|\Omega|$ for the relative figure size. The resulting bins are concatenated to form the corresponding features in $\mathbf{x}$.

The first pair of histograms measure smooth gradients, while ignoring zero gradients and sharp transitions. We define the region Φ as the region Ω minus all black pixels and pixels with zero gradients. Then, we histogram the following values for all $p \in \Phi$:

$$g(p) = \min(\max(|\nabla R_p|, |\nabla G_p|, |\nabla B_p|), 1.1) \tag{9}$$

The natural range of the above values is $[0, \sqrt{2}]$; however, we truncate any values over 1.1 to minimize the influence of strong edges.

The second pair of histograms is computed as above, but only at black pixels, and without truncation of 1.1. This histogram is meant to quantify the number of sharp edges (e.g., ink edges).

Texture. Our first texture descriptor are Local Binary Patterns (LBP) [Ojala et al. 2002; Zhu et al. 2010]. An LBP feature vector is represented as a series of patterns; each bin in our feature vector contains the number of times a certain pattern occurs in the image. The number of patterns is specified by the radius R and the sampling points P. We used rotational invariant patterns [Ojala et al. 2002] at three resolutions $\text{LBP}_{P,R}$, in particular $\text{LBP}_{8,1}$, $\text{LBP}_{12,2}$ and $\text{LBP}_{16,4}$ yielding a total of 10, 14 and 18 patterns (bins), respectively. We use two different sampling spaces: all edges of the figure, and only external contour lines. In total this descriptor yields 84 features.

Our second texture feature are Haralick texture features [1973]. We compute all 22 Haralick texture features, which are obtained from co-occurrence matrices capturing the frequency of different combinations of grayscale pixel values.

Stroke. Our stroke features are computed with the Stroke Width Transform (SWT) [Epshtein et al. 2010]. We compute SWT separately on outer edges of the figure on the silhouette, and inner edges, since their appearance is often different; the result is two arrays of likely stroke width values per pixel. We then take the mean and standard deviation of these two images. Finally, to avoid scale sensitivity we compute SWT at four different resolutions of the image: 1x, 0.5x, 0.25x and 0.12x. The result is 16 feature values.