An Unsupervised Approach for Comparing Styles of Illustrations

Takahiko Furuya
Graduate School of Medicine and Engineering
University of Yamanashi
Kofu, Japan
g13dm003@yamanashi.ac.jp

Shigeru Kuriyama
Department of Computer Science and Engineering
Toyohashi University of Technology
Toyohashi, Japan
sk@tut.jp

Ryutarou Ohbuchi
Graduate School of Medicine and Engineering
University of Yamanashi
Kofu, Japan
ohbuchi@yamanashi.ac.jp

***Abstract*— In creating web pages, books, or presentation slides, consistent use of tasteful visual style(s) is quite important. In this paper, we consider the problem of style-based comparison and retrieval of illustrations. In their pioneering work, Garces et al. [2] proposed an algorithm for comparing illustrative style. The algorithm uses supervised learning that relied on stylistic labels present in a training dataset. In reality, obtaining such labels is quite difficult. In this paper, we propose an unsupervised approach to achieve accurate and efficient stylistic comparison among illustrations. The proposed algorithm combines heterogeneous local visual features extracted densely. These features are aggregated into a feature vector per illustration prior to be treated with distance metric learning based on unsupervised dimension reduction for saliency and compactness. Experimental evaluation of the proposed method by using multiple benchmark datasets indicates that the proposed method outperforms existing approaches.**

Keywords- Local image feature, illustration style tag, illustration style feature, unsupervised distance metric learning.

I. INTRODUCTION

Stock illustrations (or clipart) are used daily in web pages, presentation slides, advertisement leaflets, and other media. Some of these illustrations are provided for a fee, while others are available for free, for example, as a part of a productivity software license, or as a public-domain repository of illustrations. Increasing number of such illustrations has created a demand for easy-to-use and effective methods to find illustrations users want.

Typically, a clipart collection is organized hierarchically, by occasion, season, characteristic object included, etc. Or, it may be searched. Search by keywords, however, requires careful, consistent labeling of a large number of illustrations. Certain clipart repository do offer search based on similarity of illustrations. For example, Microsoft's repository allows you to find clip arts similar to the one you present.

A search for "similar" visual content may employ techniques from well-studied content-based image search algorithms (e.g., [1]). These algorithms compare images mostly based on shape of objects contained in the image. Other features, such as color and texture may also be employed. For example, given an illustration of a human figure, these algorithms would try to find illustrations including human figure. But such a search may be less useful if one wishes to find illustrations having the same artistic style (e.g., black and white, silhouette, art-nouveau), but containing different objects (e.g., an iris, instead of a woman).

Relatively small number of work exist that deal with retrieval or recognition of illustrations or drawings. Most of previous work in the area of illustration and/or drawing retrieval focused on objects contained in, not style of, the illustrations. Simple geometrical features and their topological relationship are used for sketch-based retrieval of art work [3][4] and technical drawings [5]. Features extracted from vector representations of line drawings are used in these work. In a more recent work on vector graphics clip art retrieval, [1] used both vector graphic and image representations of an illustration for feature extraction; color and texture features are extracted from image representation, while topology and geometry are extracted from vector graphic representation.

Garces et al. [2] introduced the algorithm for illustrative style comparison that relies on supervised learning of labels. The algorithm extracts multiple heterogeneous global image features from each illustration. Using a set of illustrations labelled with styles as the corpus, the algorithm then applies supervised learning to produce refined similarities among illustrations. Similar supervised approach has been taken by Karayev, et al. [6] in an effort to recognize styles in photographs. In reality, however, such supervised approaches may have limited applicability. Only a very small subset of illustrations is labeled with style tags. Furthermore, style tags are not shared among existing collections. It is thus important to develop an illustrative style descriptor that does not rely on supervised learning employing stylistic labels.

In this paper, we propose a novel algorithm to derive style descriptors for illustrations without relying on supervised learning. The algorithm assumes, as its input, illustrations in array-of-pixels representation. If the illustration is in vector graphic representation (e.g., Scalable Vector Graphic (SVG) format), it is converted into array-of-pixels format prior to feature extraction. The algorithm then densely samples local image features from a multi-resolution image pyramid representation of the illustration. The local image features are then aggregated into a feature vector per illustration, prior to be treated with a dimension reduction algorithm that achieves *unsupervised distance metric learning* of the illustration feature space. An experimental evaluation using a pair of

benchmarks showed that our descriptor significantly outperforms the one proposed in [2].

The rest of this paper is organized as follows. Section II presents the proposed algorithm. Experiments and their results are presented in Section III, followed by summary and conclusion in Section IV.

II. Algorithm

A. Overview

Our proposed illustration style comparison algorithm has two major attributes; use of *local visual features* and *unsupervised distance metric learning*. Fig. 1 illustrates our style similarity comparison algorithm.

The proposed algorithm characterizes a style of an illustration by using a set of densely sampled, local, low-level, statistical image features. This is different from an approach by Garces et al. [2], which used heterogeneous set of global image features. We combine two local features; *Local Binary Patterns* (*LBP*) [7] to describe texture and *HSV* histogram computed in Hue-Saturation-Value (HSV) color space. A set of features, either of LBP or HSV, is densely extracted from an image of illustration. The set of features is then aggregated into a feature vector per image via *Fisher Vector* (*FV*) aggregation [8]. FV is one of the best performing local feature aggregation methods for image recognition [9]. As the FV aggregation ignores position of each local feature, FV-aggregated features have certain invariance to shape deformation, a property desirable for comparison of illustration styles (but not necessarily for certain type of object recognition). The FV-aggregated LBP features, *FV-LBP*, and FV-aggregated HSV features, *FV-HSV*, are then fused into a single, high-dimensional (~18K dimensions) feature per image by simple concatenation of the two feature vectors.

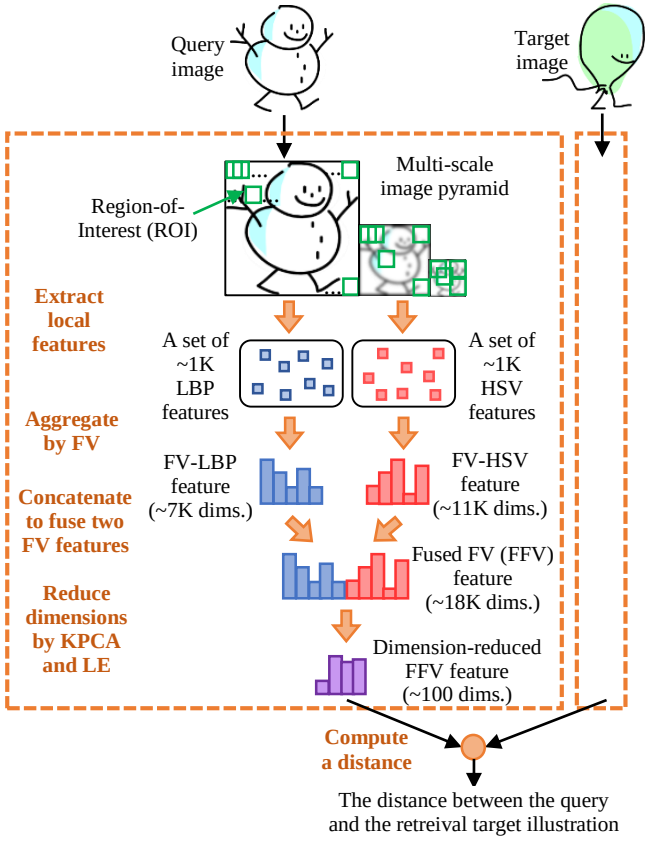

Fig. 1. Overview of our algorithm.

We then perform unsupervised dimensionality reduction on the concatenated vector by using a combination of *Kernel PCA* (*KPCA*) [10] and *Laplacian Eigenmaps* (*LE*) [11]. The dimension reduction has two purposes; improving saliency of the feature for better accuracy in style similarity comparison, and improving computational efficiency of image-to-image style comparison. To further reduce memory footprint for large-scale retrieval, we quantize each element of the dimension-reduced feature vector by using small dynamic range integer representation.

B. Multi-Scale Local Feature Extraction

1) Texture feature

We employ LBP to describe texture of illustration. An LBP is a histogram computed locally at a square *Region-Of-Interest* (*ROI*) of image having size $N_R \times N_R$. Experiments below use N_R=25 whose value is chosen based on preliminary set of experiments. The histogram is generated from N_R^2 integer values computed at every pixel in the ROI. The integer value LBP_p for a pixel p is computed as an 8 bit string, whose range is [0, 255] for decimal notation, from a 3×3 pixel region centered around p. N_R^2 (e.g., 25^2=525) integer values are accumulated into a histogram having 2^8=256 bins, and the histogram is normalized by its L1 norm.

To make the original LBP feature more compact and more robust against background and scale change of illustrations, we add the following three modifications to the original LBP. First, we exclude 8 bit strings whose number of bit changes exceeds 5 from constructing the LBP histogram. This modification reduces the number of bins for LBP histogram from 256 to 28. Secondly, a pixel p is omitted from voting for LBP histogram if differences of pixel values among p and its neighbor pixels are less than 5. By this modification, image regions having almost no textures, e.g., background of an illustration, are excluded from LBP computation. Thirdly, we extract multi-scale LBP for robustness against scale change of illustration. Specifically, an input illustration is first resized so that its longer side becomes 320 pixels. The resized image is down-sampled twice to generate an image pyramid having three layers, whose longer sides are 320, 160, and 80 pixels respectively. For each layer, ROIs having size 25×25 pixels are densely sampled at an interval of 8 pixels. We utilize integral histogram [12] to efficiently compute LBP histograms for ROIs.

2) Color feature

Our local HSV histogram has an overall dimensionality of 36, which is divided into 8, 10, and 18 bins, respectively, for Hue (H), Saturation (S), and Value (V) components. ROIs for HSV histograms are sampled in the same manner as for LBPs. That is, ROIs with 25×25 pixels are densely sampled at every 8 pixels on three layers of an image pyramid. In HSV color space, V component corresponds to gray-level intensity of an image. As the V indirectly captures thickness of lines, shade,

or shadows, and density of lines etc., we give it more quantization levels, i.e., 18, than the other two color components. The HSV histogram is normalized by its L1 norm.

When we examine pixel values of illustration images represented in HSV color space, we notice several characteristics. Among illustration images, pixel values for S and V components tend to fall in either lowest (0.0) or highest (1.0) extremes of the range. This is not like images of natural scenes in which pixel values in HSV color space is distributed more evenly. We thus add two dedicated bins at the lowest (0.0) and highest (1.0) values of S and V components, and divide the rest of bins having uniform intervals. Also, value of H component is unreliable if values of S or V are very small. We thus omit pixels from histogram computation when $S < 0.02$ or $V < 0.02$. As with LBP, we use integral histogram for efficiently computing HSV histograms for ROIs.

C. Feature Aggregation

For each local feature (i.e., LBP or HSV), a codebook for FV aggregation is learned by using Gaussian Mixture Model (GMM) clustering performed on 250,000 local features randomly selected from all the local features extracted from the training set of images. We learn N_v=128 clusters, or codewords. Dimensionality of an FV-aggregated feature becomes $2\times N_v\times N_d$ for each image where N_d is the dimensionality of local feature (N_d=28 for LBP and N_d=36 for HSV). Therefore, FV-LBP vector has the dimension 7,168 and FV-HSV has the dimension 9,216.

After the aggregation, the FV-aggregated feature goes through power-normalization followed by *L2* normalization, as in [8]. Power normalization defined in (1) is applied to the FV-aggregated feature **f**;

$$\tilde{\mathbf{f}} = \mathbf{sign}(\mathbf{f})\,|\,\mathbf{f}\,|^{\alpha} \qquad (1)$$

We use α=0.3 for FV-LBP and α=0.1 for FV-HSV.

Finally, the FV-LBP and FV-HSV vectors, both of which are power normalized and then L2-normalized, are concatenated. The concatenated vector is then normalized by its L2-norm to become raw *Fused FV* (*FFV*) illustration style feature vector. The raw FFV feature vector has dimensionality of N_D=7,168+9,216=18,432 before the unsupervised similarity metric learning involving dimension reduction described in the next section.

D. Unsupervised Similarity Metric Learning

To improve accuracy and efficiency of image-to-image style comparison, we perform unsupervised dimensionality reduction on the FFV features by using KPCA and LE. For both KPCA and LE, we use all the illustrations (N_I=4,591) in the database for learning improved similarity metric. We use KPCA with dot kernel, which linearly project a high dimensional (i.e., N_D=~18K dims.) FFV vectors onto a lower dimensional (e.g., N_K=512 dims.) subspace.

In an attempt to further improve retrieval accuracy, the KPCA-projected FFV features are projected onto a lower dimensional (e.g., N_L=128 dims.) by using LE. LE finds non-linear projection onto a subspace where diffusion distance in the original feature space is preserved as L2 (or Cosine) distance. LE-projected FFV features are compared by using Cosine similarity to generate retrieval results.

E. Quantization

Dimensionality reduction described above produces more compact FFV features (e.g., 128 dims.) than raw FFV features (e.g., 18,432 dims.). Memory footprint of dimension-reduced FFV feature is still significant, as it is real-valued; assuming 4 Byte floating point representation, a 128-dimensional FFV descriptor occupies 4×128=512 Byte per illustration. We try to reduce its memory footprint further by using quantized, small dynamic range integer representation for each element of the vector. To efficiently handle a large scale dataset, small memory footprint is essential.

We try a simple approach, which is to represent each element of dimension-reduced FFV descriptor by using small dynamic range integer. A set of feature vectors processed by KPCA and LE tend to distribute around the origin of its feature space [13]. This "sphered" distribution of the dimension-reduced features is discretized by using the following method. The floating point value $\mathbf{f}_d$ of *d*-th dimension of the FFV feature vector is converted into an integer value i_d having 2^n quantization levels whose range is $R_n = [-2^{n-1}, 2^{n-1}-1]$ by using the following equation,

$$i_d = \mathrm{round}\left(2^{n-1} \times \frac{\mathbf{f}_d}{3\sigma} \right) \qquad (2)$$

where round() denotes a rounding function and the standard deviation σ is computed with respect to all dimensions of all the FFV features. If i_d produced by (2) is out of the range R_n, it is clipped to its boundary values. Using the standard deviation σ makes the discretization somewhat robust against outliers. If we use n=8, that is, 2^8=256 quantization levels, a feature vector would occupy 128 Byte per illustration.

III. Experimental Evaluation

A. Benchmark database

To evaluate effectiveness of the proposed algorithm, we use the benchmark dataset supplied by Garces et al. [2]. This dataset contains 4,591 illustrations with a wide range of styles. Fig. 2 shows examples of illustrations in the dataset. Out of the 4,591 illustrations, a set of 1,000 samples, which we call "MTurk", were collected from the Art Explosion dataset [15] and a set of 3,591 samples, which we call "MS", were collected from a set of clipart included in the Microsoft Office. The set of 4,591 samples is divided into two subsets, i.e., a train set for supervised learning and a test set for evaluation. Since our algorithm is fully unsupervised, i.e., no labels are required, we use all the images in the benchmark to learn a codebook for FV aggregation and to learn a subspace for dimensionality reduction.

We use two evaluation protocols. The first protocol is called triplet testing which was used in [2]. Given a triplet of three illustrations $\mathbf{I}_a$, $\mathbf{I}_b$ and $\mathbf{I}_c$, where $\mathbf{I}_a$ has more similar style to $\mathbf{I}_b$ than to $\mathbf{I}_c$, a triplet test is considered as success if $\mathbf{I}_b$ is ranked higher than $\mathbf{I}_c$ in the retrieval result for the query

$\mathbf{I}_a$. Accuracy of the triplet testing is a success rate obtained through 10,633 triplet tests.

As the second evaluation protocol, we use mean Average Precision (mAP). To compute mAP, we use the set of 3,591 images in the MS dataset that has label information. The set of 1,000 images in the MTurk dataset is used as distracter, that is, they are considered as incorrect in the retrieval results.

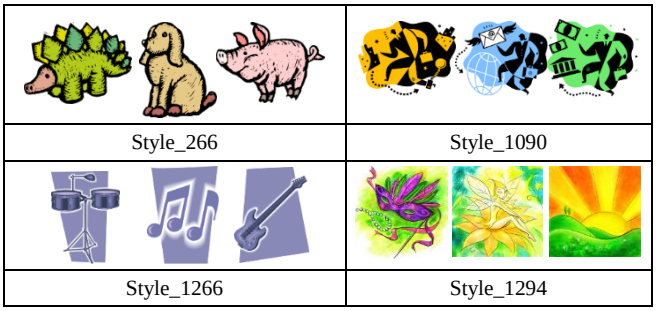

Fig. 2. Examples of illustrations from a benchmark dataset used in [2].

B. *Effectivness of Similarity Metric Learning*

Fig. 3 shows effectiveness of dimension reduction of FFV feature, where "FFV" plots the accuracy of raw FFV feature without reduction. "FFV (KPCA)" and "FFV (KPCA+LE)" plot the accuracy of FFV feature projected onto a lower dimensional space, where the latter method first project a raw feature vector onto 512-dimensional sub-space with KPCA and the dimension is further reduced by LE. We can observe that these dimensionality reductions significantly improve retrieval accuracy. "FFV (KPCA)" with 256 dimensions increases mAP score by about 0.04 compared to "FFV". "FFV (KPCA+LE)", which captures non-linearity of the feature distribution, achieves higher accuracy by mAP=0.63 with 128 dimensions.

Dimensionality reduction significantly accelerates distance computation among FFV features. The 128-dimensional FFV feature compressed by KPCA takes only about 0.002 seconds for computing distances between the query and the 4,591 images in the dataset. On the other hand, the raw 18,432-dimensional FFV feature is more time-consuming; it takes about 0.057 seconds for computing distances. For a larger-scale dataset, e.g., that having 100K illustrations, the compact (e.g., 128-dimensional) feature would be essential for interactive retrieval. The computation times above were measured by using a single thread code run on a PC having two Intel Xeon E5-2650V2 CPUs and 256 GByte of DRAM.

C. *Training Set Size and Retrieval Accuracy*

In the experiments above, we used all the 4,591 images contained in the benchmark dataset for unsupervised learning, i.e., codebook learning for FV aggregation and subspace learning for dimensionality reduction. In this section, we evaluate the influence of the number of training images on the retrieval accuracy. We randomly select N_t samples from the set of 4,591 images for learning. We run the experiment 5 times and the average of mAP scores is used for evaluation.

Fig. 4 plots retrieval accuracy against the number of training samples N_t. "FV-LBP", "FV-HSV", and "FFV" use N_t images only for learning the codebook for FV aggregation. Meanwhile, "FFV (KPCA)" and "FFV (KPCA+LE)" use N_t images for both codebook learning and subspace learning. In Fig. 4, for "FV-LBP", "FV-HSV", and "FFV", mAP scores nearly unchanged for all the N_t we have experimented. We can confirm that the codebook learning for FV aggregation is insensitive to the number of training samples, whereas KPCA and LE are more sensitive to. Accuracies for "FFV (KPCA)" and "FFV (KPCA+LE)" significantly decrease at lower N_t, especially for the latter. It is noteworthy that dimensionality reduction such as KPCA and LE require sufficient number of training samples for finding appropriate subspaces.

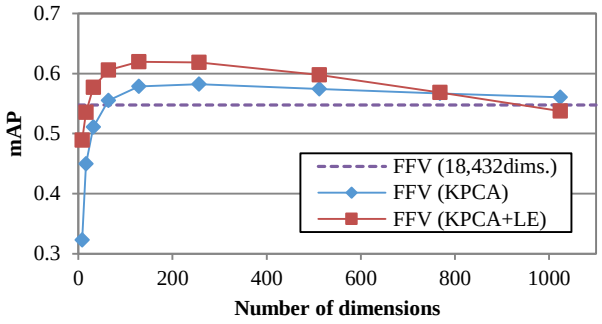

Fig.3. Reduced feature dimension and retrieval accuracy (mAP).

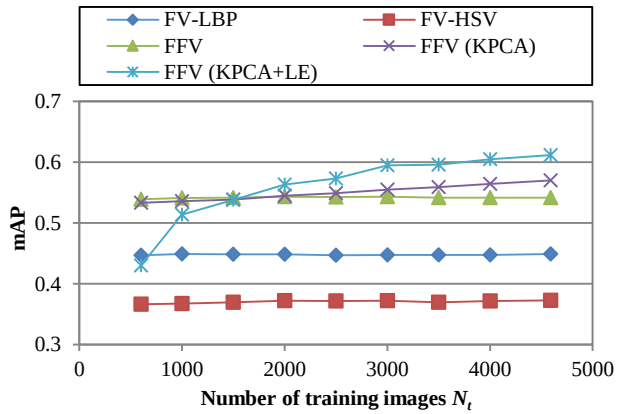

Fig. 4. Training dataset size and retrieval accuracy (mAP).

D. *Quantization Levels and Retrieval Accuracy*

Here we evaluate the impact FFV feature quantization has on retrieval accuracy. Fig. 5 plots retrieval accuracies against the number of levels for quantization, where we set the dimensionalities of FFV(KPCA) and FFV(KPCA+LE) at 128. For both FFV(KPCA) and FFV(KPCA+LE), quantization with 64 levels retains retrieval accuracy of FFV features without quantization. The feature vector quantized by using 64 levels (i.e., 6 bits) occupies 96 Bytes $(= 6\text{bits} \times 128 / 8)$ per image. About 10K such descriptor would easily fit in a 2nd (or 3rd) level cache of a contemporary CPU for an efficient on-cache processing.

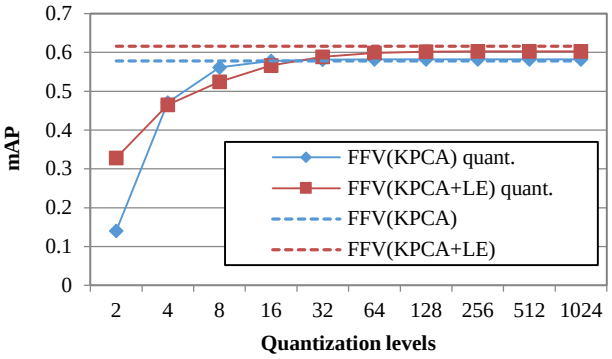

Fig. 5. Quantization levels and retrieval accuracy (mAP).

E. *Comparison with the other Features*

In this section, we compare retrieval accuracy of our FFV feature with two other algorithms. First algorithm to be compared against is the one by Garces et al. [2]. It uses four global feature vectors, i.e., color, shading, texture, and stroke, extracted from each illustration. Distance between a pair of features is computed by using supervised distance metric learning. It uses a "weak" form of supervision based on human annotation of style similarity presented with a triplet of illustrations. The second algorithm, which we call FV-DSIFT, uses densely sampled SIFT feature [14] aggregated by using FV aggregation [8]. FV-DSIFT is used widely in image object recognition task. For the FV-DSIFT, we densely extracts 1,200 SIFT features having random scales and random positions from an illustration. The set of SIFT features is aggregated by FV into a per-image feature vector for comparison. We use the codebook of size N_v=64 for FV-DSIFT. Note that both proposed FFV feature and the algorithm by Garces (Baseline) employ color information extracted from HSV set of images as well as shape/texture feature extracted from an intensity image. FV-DSIFT, on the other hand, only uses shape/texture feature extracted from intensity image.

Table I compares the retrieval accuracies of our FFV feature against the other ones. The performance of our FFV (mAP=0.58) significantly outperforms the baseline (mAP=0.36). Non-linear dimensionality reduction by LE, denoted by FFV (KPCA+LE), achieves further improves (mAP=0.63), which is the highest score we have obtained through the experiments. Lowest accuracy of the FV-DSIFT suggests that the lack of color information negatively impacts retrieval accuracy.

The baseline algorithm shows high accuracy if evaluated by using triplet testing, especially for the MTurk dataset. However, its mAP score is lower than that of the FFV. We speculate that this is a result of overfitting of the baseline algorithm to supervision provided by triplets of the MTurk dataset. The triplets of the MTurk dataset, which were obtained via crowd-sourcing, is influenced by subjective criteria of multiple individuals. These sets of criteria are inherently different from the set of tags used in the MS dataset. Overfitting the MTurk dataset would result in a higher accuracy for the MTruk dataset but a lower accuracy for the MS dataset.

Fig. 6 shows an example of a query and its retrieval result using the baseline (supervised), FV-DSIFT, and FFV, which demonstrates the superiority of our feature over the other ones. We can observe that the proposed FFV feature retrieves illustrations having different objects and colors but having similar style to the query.

Table I. Style comparison algorithms and retrieval accuracy (mAP).

Algorithms	Accuracy on triplet test		mAP
	MTurk	MS	
Baseline (supervised)[2]	0.82	0.95	0.36
Baseline (unsupervised)[2]	0.75	0.94	0.39
FV-DSIFT [19]	0.65	0.89	0.29
FFV (KPCA)	0.77	0.98	0.58
FFV (KPCA+LE)	0.64	0.94	0.62

F. *Scalability*

To evaluate the scalability of our proposed algorithm, we performed the experiments by using datasets with larger number of images than the MTurk+MS dataset [2]. We created three larger-scale datasets by adding distractor images to the MTurk+MS dataset from other clipart repositories. The first dataset is "MTurk+MS+AE(10K)" having 10,000 images. It is a union of the set of 4,591 images of the MTurk+MS dataset and the set of 5,409 distractor images randomly selected from the Art Explosion dataset [15]. The second and the third datasets are "MS+OC(5K)" having 5,000 images and "MS+OC(10K)" having 10,000 images. They were created by adding 1,409 or 6,409 distractor images randomly selected from the openclipart repository [16] to 3,591 images of the MS dataset. As with evaluation using the MTruk+MS dataset, retrieval accuracies for these three "inflated" datasets are evaluated by using labeled images of the MS dataset and the other distractor images are considered as incorrect in the retrieval results.

Table II summarizes retrieval accuracies for the three larger-scale datasets. In the table, elements of both FFV(KPCA) and FFV(KPCA+LE) feature vectors are quantized to 64 levels after dimension reduction . Retrieval accuracies for the larger-scale dataset are lower than that for the (smaller) MTurk+MS dataset. This is expected, since these three larger-scale datasets contains more distractor images than the MTurk+MS dataset. Nevertheless, retrieval accuracies in Table II for larger scale datasets are still reasonably good as mAP scores for both FFV(KPCA) and FFV(KPCA+LE) are over 0.5.

Table II. Retrieval accuracies (mAP) for larger-scale datasets.

Features	MTurk +MS (5K)	MTurk+MS +AE (10K)	MS +OC (5K)	MS +OC (10K)
FFV(KPCA)	0.58	0.53	0.56	0.53
FFV(KPCA+LE)	0.62	0.58	0.59	0.58

IV. Conclusion and Discussion

This paper proposed and evaluated a novel style-based image retrieval algorithm for illustrations. The image descriptor proposed, *Fused Fisher Vector* (*FFV*), has two

major attributes; densely sampled local visual features and unsupervised distance metric learning. An illustration is represented by a set of texture-based local features and a set of color-based local features. Each set is aggregated into a feature vector per image by using Fisher Vector (FV) [8].

A style feature vector per illustrations, called (raw) Fused Fisher Vector (FFV), is produced by concatenating the two aggregated features. For efficacy and efficiency in comparison, unsupervised distance metric learning via a combination of Kernel PCA and Laplacian Eigenmaps is performed on raw FFV, followed by small dynamic range (e.g., 6 bit) quantization of each element of the vector.

An experimental evaluation using multiple benchmark datasets showed that our proposed algorithm is more accurate in comparing illustration styles than the algorithm proposed in [2]. Memory footprint of our illustration style descriptor is small enough (e.g., 96 Bytes per illustration) so that a large scale style-based illustration retrieval system is practical.

Multiple avenues for future exploration remain. Obviously, we want to further improve accuracy and efficiency of descriptor for illustration style comparison. For example, deep neural network trained on a set of illustrations, instead of photographic images [6], might yield a better descriptor tuned to illustrations. It would also be interesting to develop a set of style tags or classifications that transcends particular illustration database and reflects human perception and cognition.

ACKNOWLEDGEMENT

REFERENCES

[1] P. Martins, R. Jesus, M. Fonseca, and N. Correia, "Clip art retrieval combining raster and vector methods," in *Content- Based Multimedia Indexing (CBMI)*, June 2013, pp. 35–40.

[2] E. Garces, A. Agarwala, D. Gutierrez, and A. Hertzmann, "A similarity measure for illustration style," *ACM Transactions on Graphics (SIGGRAPH 2014)*, vol. 33, no. 4, 2014.

[3] M. F. Barroso, M. J. Fonseca, B. Barroso, P. Ribeiro, and J. A. Jorge, "Retrieving clipart images by content," in *Proceedings of the 3rd International Conference on Image and Video Retrieval* (*CIVR* `04), LNCS, 2004.

[4] P. Sousa and M. J. Fonseca, "Geometric matching for clip-art drawing retrieval," *J. Vis. Commun. Image Represent.*, vol. 20, no. 2, pp. 71–83, Feb. 2009.

[5] M. J. Fonseca, A. Ferreira, and J. A. Jorge, "Sketch-based retrieval of complex drawings using hierarchical topology and geometry," *Computer-Aided Design*, Volume 41, Issue 12, December 2009, Pages 1067–1081

[6] S. Karayev, A. Hertzmann, M. Trentacoste, H. Han, H. Winnemoeller, A. Agarwala, and T. Darrell, "Recognizing image style," in *BMVC2014*, 2014.

[7] T. Ojala, M. Pietikäinen, and T. Mäenpää, (2002), "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. PAMI*, **24**(7), 971–987.

[8] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV'10, 2010, pp. 143–156.

[9] A. V. Ken Chatfield, Victor Lempitsky and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proceedings of the British Machine Vision Conference*, 2011, pp. 76.1–76.12.

[10] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[11] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[12] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, June 2005, pp. 829–836 vol. 1.

[13] L. Saul, K. Weinberger, J. Ham, F. Sha, and D. Lee, *Spectral Methods for Dimensionality Reduction*. MIT Press, 2006.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[15] Nova Development, http://www.novadevelopment.com

[16] Openclipart, https://openclipart.org

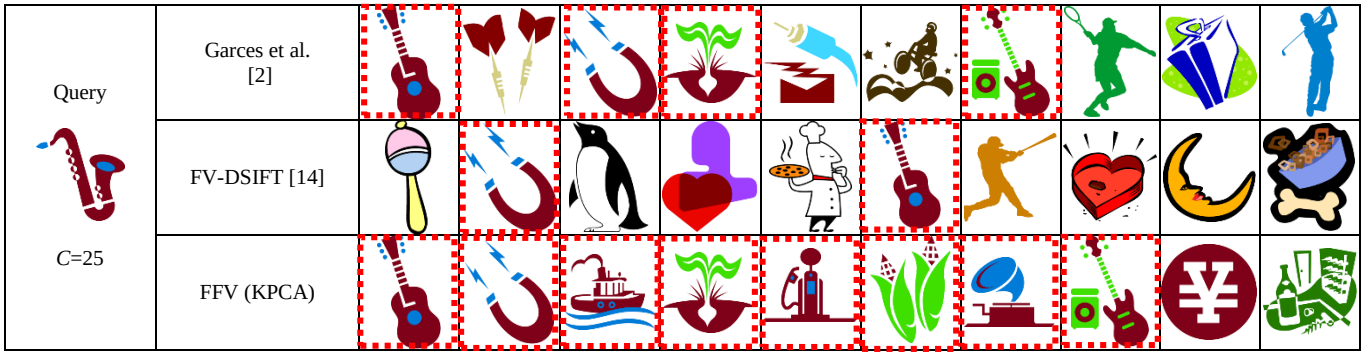

Fig. 6. Example of a query and its retrieval result using the benchmark by [2]. Illustrations with red dots indicate correct results for the query. *C* indicates the number of illustrations which belong to the same style category as the query. Our proposed FFV feature compressed down to 512 dimensions by KPCA produces better results than the other features.