AUTOMATIC DISCOVERY OF IMAGE FAMILIES: GLOBAL VS. LOCAL FEATURES

Mohamed Aly[1] *Peter Welinder*[1] *Mario Munich*[2] *Pietro Perona*[1]

[1] Computational Vision Lab, Caltech
Pasadena, CA, USA
vision.caltech.edu
{malaa, welinder, perona}@caltech.edu

[2] Evolution Robotics
Pasadena, CA, USA
www.evolution.com
mario@evolution.com

ABSTRACT

Gathering a large collection of images has been made quite easy by social and image sharing websites, e.g. *flickr.com*. However, using such collections faces the problem that they contain a large number of duplicates and highly similar images. This work tackles the problem of how to automatically organize image collections into sets of similar images, called *image families* hereinafter. We thoroughly compare the performance of two approaches to measure image similarity: global descriptors vs. a set of local descriptors. We assess the performance of these approaches as the problem scales up to thousands of images and hundreds of families. We present our results on a new dataset of CD/DVD game covers.

1. INTRODUCTION

The internet has made it quite easy to collect large image collections using websites like *flickr.com* or *facebook.com*. However, such collections are more likely to contain lots of duplicates and highly "similar" images i.e. images that have significant content overlap and share some regions with possibly different color, scale, contrast, and position. We call such similar images **image families**. The automatic organization and cataloguing of such collections is quite important for both research and consumer applications e.g. object recognition and image retrieval applications among others.

This work focuses on the problem of automatic family discovery in an unprocessed collection of images i.e. automatically identifying images that are "similar" enough to belong to the same family. We compare two broad approaches for measuring similarity between images: global descriptors vs. a set of local descriptors. The global approach represents each image by one feature descriptor computed from the whole image. The local approach represents each image by a set of local feature descriptors computed at some interesting points in the image [5, 6]. These similarity measures are then input to two graph partitioning algorithms: Normalized Cuts [9] and Agglomerative Clustering to automatically extract the image families. We also test the effect of the dataset size on the performance of these approaches.

Near-duplicate image detection has been studied extensively [4, 1], where the goal is to retrieve all images in a database that are similar to a query image. In contrast, there has been very little work on automatically clustering near-duplicate images, e.g. [3]. Our work is different in two respects. First, most work focused on image retrieval and not on automatic clustering. Second, image families as defined here is much broader and harder than near-duplicate images as defined in [3]. A near-duplicate does not include introducing new content in the image or shuffling different regions of the image to different locations, see Fig. 1.

2. DATASET

We collected a set of CD/DVD covers for games on PC and on different kinds of game consoles (e.g. Xbox, PlayStation, ... etc.) [1]. The problem of image families in this dataset is very clear, see Fig. 1. We define an **image family** as the set of all images of the same game on different consoles and in different languages. Notice how there can be considerable variability within each family, compared to near-duplicate images. In row 1 the two images have different color, front and back cover undergo different scaling, and have different languages. In row 2 the images have different colors, different front covers, different parts of the back cover, and different locations of logos (the Aeon Flux white logo). In row 3 the images have different scales, and the right one lacks the back cover. In row 4 the images have different colors and languages.

The dataset has a total of 11,443 images. To get a ground truth for comparisons, we implemented a simple annotation tool to manually identify **families** of images [2]. We divided the dataset into 6 subsets of increasing difficulty, see Table 1. Subset #1 is the easiest having families with at least 20 members, and contains 102 images and 4 families. Subset #6 is the hardest having families with at least 4 members, and contains 3958 images and 648 families.

[1] Collected from www.freecovers.net

[2] Dataset and annotations are available at vision.caltech.edu/malaa/datasets/caltech-games

Fig. 1: Example image families. First row shows two images from a 007 game on Xbox (English) and PS2 (German). Second row shows Aeon Flux game also on PS2 and Xbox. Third row shows two versions of Armored Core on PS2, one without back cover. Fourth row shows a German and an English version of Atari Anthology on PS2.

3. FEATURES AND CLUSTERING

3.1 Global Features

We define the approach of global features as that in which each image is represented by a single feature vector, capturing information from the whole image. No attention is paid to the constituents of the image, such as individual regions or objects. Once each image's feature is computed, we can measure the similarity between any pair of images using some distance metric e.g. L_2 distance. We compare several popular feature descriptors:

Subset	Min # members	# images	# families
1	20	102	4
2	16	210	10
3	12	644	43
4	8	1360	125
5	6	2292	271
6	4	3958	648

Table 1: Subsets used in the experiments.

- SIFT: we compute a standard SIFT [5] descriptor for the whole image, which is then normalized to have unit norm. We use our Matlab implementation.
- Gist: we compute a Gist [7] descriptor for the whole image, which is further normalized to unit length. We used the code available from the authors' website.
- HOG: we compute a Histogram of Oriented Gradients [2] descriptor for the whole image. We use our Matlab implementation.
- Bag-of-words: The idea is inspired from natural language processing applications, where each text document is represented by a histogram of word occurrences in the document. To make the jump from words to "visual" words, local features are extracted from the images, and then they are vector quantized using K-means, to create a codebook of "visual words". Then, each image is represented by the histogram of visual words present in the image. We used the affine covariant feature detector [6] together with SIFT descriptors, and used the code available from the authors' website. We compare different sizes of codebooks: 1K, 5K, 10K, and 25K visual words. We also compare two variants of bag-of-words:

1. **Raw**: where we use raw histogram counts of visual words, and normalize it to unit norm.
2. **Tf-idf**: where the histogram counts are weighted according to the popularity of the word in the database [**?**].

3.2 Local Features

In this approach, we represent each image by a set of local feature descriptors computed from different points in the image. There are different types of interest point detectors that can be used, like affine covariant features [6], difference of Gaussian [5] ...etc. To be consistent with the experiments above, we use the affine covariant feature detector together with SIFT descriptors. Each image i is represented by a collection of local SIFT feature descriptors $\mathbf{f}_{i_k}$ where $k = 1, \ldots, n_i$ and n_i is the number of features in image i. Each descriptor has an associated label $l_{i_k} = i$ and location in the image $\mathbf{x}_{i_k}$. In the experiments, each image had on average 1000 features.

In order to measure the similarity between a pair of images, we need to match features in image i to features in image j. A naive way to do the matching by exhaustive search blows up quickly, as it scales with $O(n^2)$ where n is the total number of features. To keep the computation time under control, we use a set of Randomized Kd-trees [8], called Kd-forest, to do an approximate nearest neighbor search. First, we add all the features from all images into the Kd-forest. Then, for each feature $\mathbf{f}_k$ we get the nearest neighbor $\mathbf{g}_k$ with label l_{g_k} such that it is not in the same image. Define $\mathbf{1}\{\cdot\}$ as the indicator function that returns a value of 1 when the expression in parentheses is true and zero otherwise. We then compare 4 approaches to measure similarity, with increasing complexity:

- Simple: here the similarity between images i and j, s_{ij}, is defined as $s_{ij} = \sum_{k \in \text{image}\, i} \mathbf{1}\{l_{g_k} = j\}$ i.e. we simply

count the number of neighbors that belong to image j.

- NN-ratio: here we do further processing of images with at least 5 nearest neighbors, which are images with the potential of being similar. We start by performing an exhaustive nearest neighbor search between features of images i and j. This step is also repeated in the next two methods. The similarity is computed as $s_{ij} = \sum_k \mathbf{1}\{d(\mathbf{f}_{i_k}, \mathbf{g}_{i_k 1}) \times \delta_1 \leq d(\mathbf{f}_{i_k}, \mathbf{g}_{i_k 2})\}$ where $d(\cdot,\cdot)$ is the distance between two feature vectors, $\mathbf{g}_{i_k 1}$ is the nearest neighbor to $\mathbf{f}_{i_k}$, $\mathbf{g}_{i_k 2}$ is the second nearest neighbor, and $\delta_1 > 1$ is a constant. This formula counts a feature $\mathbf{f}_k$ only when the distance to its nearest neighbor is significantly smaller than the distance to its second neighbor [5]. We set $\delta_1 = 1.1$.

- Image-aff: after getting the neighbors between images i and j, we check spatial consistency of matched features. We use a RANSAC algorithm to fit an affine transformation, H_{ij}, that maps locations of features in image i to the matching features in image j [**?**]. The similarity is defined as $s_{ij} = \sum_k \mathbf{1}\{d(H_{ij}(\mathbf{x}_{i_k}), \mathbf{x}_{j_k}) < \delta_2\}$ where $\mathbf{x}_{j_k}$ is the location of the matching feature in image j. This simply counts the number of features that are consistent with the computed affine transform H_{ij}. We use $\delta_2 = 25$ pixels.

- Region-aff: since some regions of the image can undergo different transforms (see row 1 in Fig. 1), we can enhance the similarity measure by considering different affine transforms for different regions in the image. We divide the image into 200×200 pixels overlapping regions with a stride of 100 pixels, and fit a separate affine transform H_{ijl} for each such region. We then count the total number of features consistent with these individual transformations i.e. $s_{ij} = \sum_{k,l} \mathbf{1}\{d(H_{ijl}(\mathbf{x}_{i_k}), \mathbf{x}_{j_k}) < \delta_3\}$, where $\delta_3 = 10$ pixels.

3.3 Clustering

We compare two parametric unsupervised clustering methods in order to automatically discover image families. In all the experiments we assume that we know the number of families beforehand without knowing any family labels.

- Normalized Cuts (NC): Given an affinity matrix S with elements s_{ij} defining the similarity between image i and j, the Normalized Cuts [9] algorithm tries to recursively divide the matrix into two disjoint sets, A and B, such that the normalized cut is maximized, which is defined by $\frac{cut(A,B)}{assoc(A,S)} + \frac{cut(A,B)}{assoc(B,S)}$ where $cut(A,B) = \sum_{ij} s_{ij}\ \forall i \in A$ and $j \in B$, and $assoc(A,S) = \sum_{ik} s_{ik}\ \forall i \in A$. We use the Matlab code available from the authors' website.

- Agglomerative Clustering (Ag): which builds clusters recursively bottom-up. First, each image belongs to its own cluster. Then at every iteration, two clusters A and B that maximize an objective function are combined into one cluster. The objective function used is the Average Linkage, defined as $al = \frac{\sum_i \sum_j s_{ij}}{|A|\,|B|}\ \forall i \in A$, $j \in B$, where $|A|$ is the size of cluster A.

4. RESULTS AND DISCUSSION

We ran experiments on the subsets defined in Table 1. The results are reported in Fig. 2. The different approaches are listed vertically, while the horizontal axis plots the mean confusion matrix performance defined as $p = \left(\sum_f \frac{c_{ff}}{\sum_k c_{fk}} \right) / F$ where F is the number of families in the subset and c_{fk} is the confusion matrix entry in row f and column k, and represents the number of images that belong to family f in the ground truth but were classified as family k. Looking at the results, we make the following comments:

- For simpler subsets, global features perform almost as well as local features, while with harder subsets local features are at least 10% better, see subset #6. This is because as the number of families increases and family size decreases, it becomes a much harder clustering problem for global features where there is not enough information to accurately measure similarity between images.

- For local features, it is surprising that using the simple approach does not degrade performance, and in fact it gives best performance overall of 93% on subset #6. We think the reason for this is that even in the hardest subset which have around 4 million features, the number of features is not big enough to degrade the nearest neighbor performance of the Kd-Forest.

- For simpler subsets, NC gives better performance than Ag, while with more difficult subsets, Ag is better than NC. This might be because with increasing number of families, the spectral clustering with NC finds harder time to accurately compute eigenvalues/eigenvectors which are used to cluster the data.

- Increasing the codebook size for bag-of-words generally increases performance, specially for harder subsets. For example in subset #6, using 25k words gets performance within 10% of the best. This is because using more words makes the histogram sparser, which in turn makes it more discriminative between different image families.

- For local features, bag-of-words performs the best. SIFT, HOG, and Gist perform significantly worse specially in harder subsets.

- Using tf-idf weighting does not provide any performance gain than using the raw histograms. We think this is because it is not discriminative enough to differentiate between different images.

5. CONCLUSION AND FUTURE WORK

This work focuses on the important problem of **automatic family discovery**. We present a new dataset of CD/DVD covers that clearly demonstrates the problem, and provides a harder benchmark than near-duplicate image datasets used before. We compar two broad approaches for tackling this problem: global vs. local features. We test the performance

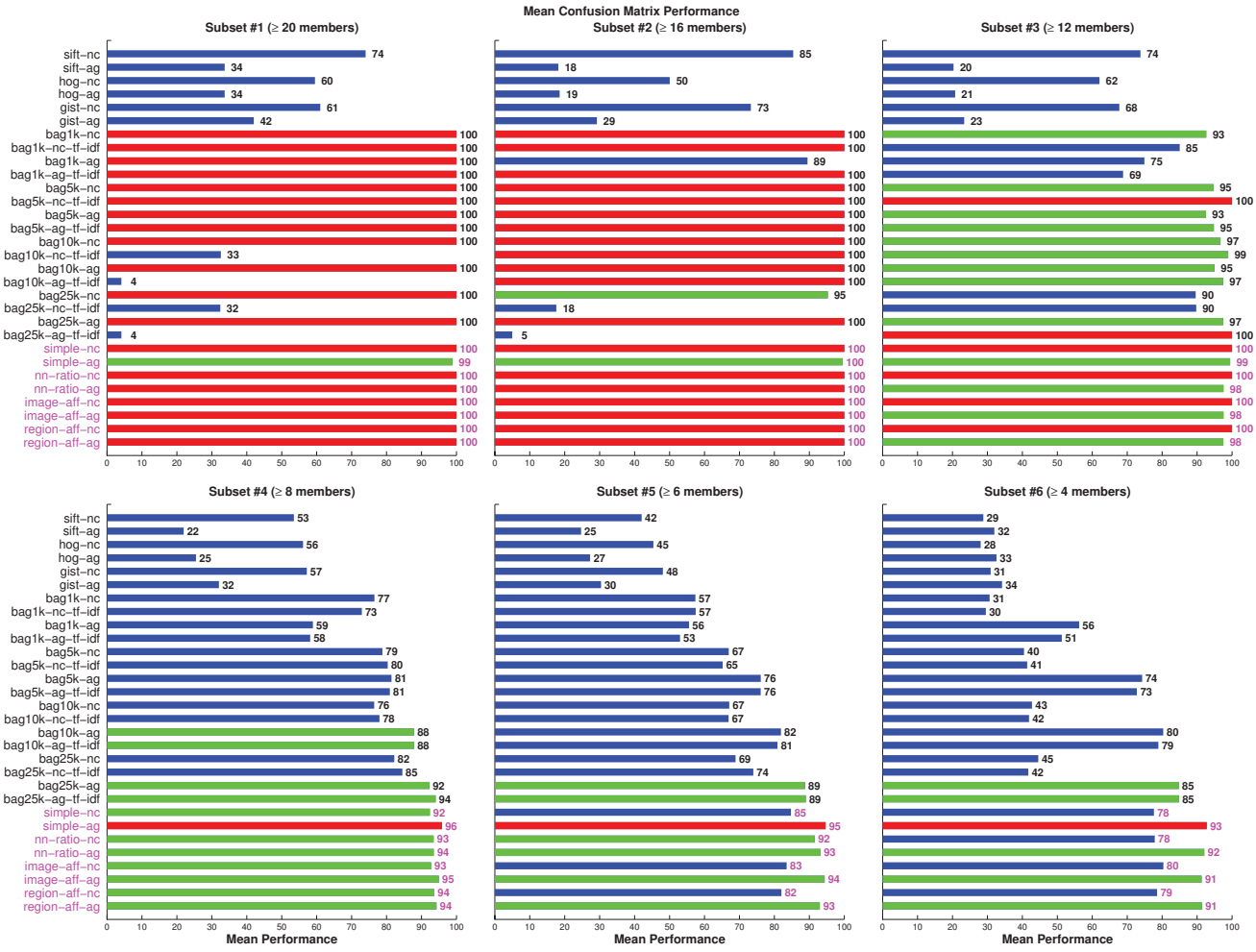

Fig. 2*:* Mean Confusion Matrix Performance. Mean performance plotted on x-axis, and different methods on y-axis. Global methods are in black, while local methods are in magenta. Bars are colored as: best value is in **red**, values within 10% of the best are in **green**, rest is in **blue**. *-nc denotes Normalized Cuts, *-ag denotes the Agg. Clustering, bag*-tf-idf denotes the tf-idf weighted version of the bag-of-words. Best performance of 91-93% on subset #6 was achieved by using local features.

of these methods on different subsets with increasing difficulty. Based on the experiments, we conclude that using local features provides a more accurate similarity measure, and we are able to achieve an average performance of 93% on the hardest subset. We plan on extending the dataset by collecting more images reaching hundreds of thousands. We also plan on extending the methods here to automatically discover the number of families in the image collection.

6. REFERENCES

[1] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, pages 549–556, 2007.

[2] N. Dalai and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893vol.1, 20-25 June 2005.

[3] J. Foo, J. Zobel, and R. Sinha. Clustering near-duplicate images in large collections. In *Proc. of Int. Workshop on MIR*, 2007.

[4] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA*, pages 869–876, 2004.

[5] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[6] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60:63–86, 2004.

[7] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001.

[8] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR*, 2007.

[9] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, Aug. 2000.