Towards Automated Large Scale Discovery of Image Families

Mohamed Aly[1], Peter Welinder[1], Mario Munich[2] and Pietro Perona[1]
[1] Computational Vision Lab, Caltech, Pasadena, CA *vision.caltech.edu*
[2] Evolution Robotics, Pasadena, CA *www.evolution.com*
`{malaa, welinder, perona}@caltech.edu` `mario@evolution.com`

Abstract

Gathering large collections of images is quite easy nowadays with the advent of image sharing websites, such as flickr.com. However, such collections inevitably contain duplicates and highly similar images, what we refer to as image families. Automatic discovery and cataloguing of such similar images in large collections is important for many applications, e.g. image search, image collection visualization, and research purposes among others. In this work, we investigate this problem by thoroughly comparing two broad approaches for measuring image similarity: global vs. local features. We assess their performance as the image collection scales up to over 11,000 images with over 6,300 families. We present our results on three datasets with different statistics, including two new challenging datasets. Moreover, we present a new algorithm to automatically determine the number of families in the collection with promising results.

1. Introduction

The advent of new image sharing and social networking website, e.g. *flickr.com* and *facebook.com*, has made it quite easy to gather huge image collections of millions of images [19]. However, such collections will inevitably contain duplicates and highly similar images, i.e images with significant visual content overlap, with possibly different color, scale, contrast, positions, and viewpoints. We refer to such similar sets of images as image families. The automatic organization and cataloguing of such collections by discovering these families has many applications: 1) it is desirable for next-generation image search engines that utilize visual content for searching large corpora of images, where organizing images into related families can greatly improve search speed and accuracy [8]; 2) it is useful for automatic categorization of large personal image collections, e.g. grouping all vacation images having *Eiffel Tower*; and 3) it is crucial for large scale visual object/category/scene recognition research which relies on collections of annotated images, and automating this annotation process is indispensable specially for millions of images.

This work focuses on the problem of automatically identifying image families in unprocessed image collections. 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 [14, 15]. We compare their performance as the number of images scale up to over 11,000 with over 6,300 families.

We investigate two scenarios for family discovery after computing image similarities: 1) a semi-supervised scenario, in which we assume we know the number of families beforehand. We compare two graph partitioning algorithms for clustering: Normalized Cuts [18] and Agglomerative Clustering [11]. 2) an unsupervised scenario, in which we do not know the number of families in advance. We present a new algorithm based on Connected Component Labeling [4], that automatically clusters and estimates the number of families.

We present our results on three datasets: 1) a CD/DVD game covers dataset consisting of 11,431 images and 6,361 families, 2) Caltech buildings dataset with 250 images and 50 families, and 3) Oxford buildings dataset [17] with 272 images and 11 families.

Image family discovery is related to image clustering [9, 8] and near-duplicate image detection [2, 12, 6], however it is different in three respects: 1) We use the term "family" to indicate groups of images having high visual similarity with possible change in color, viewpoint, scale, .. etc. In that sense it is a special case of an image "cluster", which might refer to a visual category or a type of natural scenes [5], and more general than "near-duplicates" as defined in [12]. 2) Near-duplicate detection is mostly applied to image retrieval systems or near-duplicate shot detection in movies [2]. 3) Clustering into hundreds or thousands of clusters was largely not studied. Large collections with millions of images will have hundreds of thousands of families, so it is extremely important to scale up automatic image clustering and family discovery to handle such collections.

Our contributions are: 1) we systematically compare two approaches for measuring image similarity and assess their performance on datasets of increasing complexity, scaling up to over 11,000 images and 6,300 families, 2) we present a new algorithm for automatically clustering and estimating the number of families, 3) we present two new challenging annotated datasets that can be used for benchmarking performance of different algorithms.

2. Datasets

We present our results on ten subsets from three distinct datasets of different statistics and complexities.

The first dataset, which we call the **games** set, is a collection of CD/DVD covers for video games[1] on different consoles (e.g. Xbox, PlayStation, ... etc). We consider as image family the set of all images of the same game on different consoles and in different languages. Discovering image families in this collection is challenging and more general than near-identical images, see fig. 1. The dataset has a total of 11,431 medium resolution (600×400) images. We manually sifted through the images and identified 6,361 families. We divided the dataset into 8 subsets of increasing difficulty, see Table 1. Games 16 is the easiest subset having families with at least 16 members, and contains 210 images and 10 families. Games 01 is the hardest subset having families with at least 1 members (i.e. including unique images), and contains 11,431 images and 6,361 families.

The second set, which we call the **caltech** set, consists of 250 images of 50 different buildings around the Caltech campus. We took 5 photographs of each building, with different scale and camera viewpoint. We consider as family the 5 images of each building, i.e. we have 50 families of 5 images each, see fig. 2. Images were down sampled to 800×600 pixels. This set is challenging because we have considerable change in viewpoints and scales for each building. We will make these two datasets and annotations available online for benchmarking purposes.

The third set is the **oxford** buildings set[2] used in [17]. The set originally has 5,062 images obtained from *flickr.com* by searching for 11 Oxford landmarks. We only used a small subset, those labeled as "*good*" i.e. having a nice clear picture of the building. The *good* set has 272 images with 11 families (one per landmark). Images were used in their original resolution, which is about 1024×768. This set is even more challenging, as it contains extreme differences in lighting conditions, scales, contrasts, and viewpoints, see fig. 3.

[1] Collected from *www.freecovers.net*
[2] Available at *tinyurl.com/dg32em*

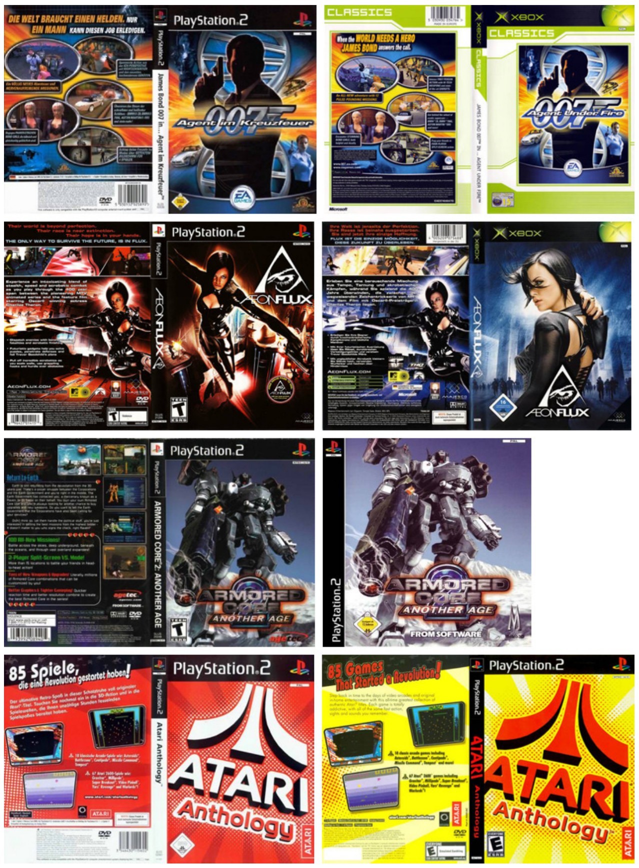
PlayStation 2
XBOX
CLASSICS
007 Agent im Kreuzfeuer
007 Agent Under Fire
AEON FLUX
ARMORED CORE 2 ANOTHER AGE
85 Spiele, die eine Revolution gestartet haben!
85 Games That Started a Revolution!
ATARI Anthology

Figure 1. Sample images from **games** set. Row 1 shows two images from 007 game on Xbox (Eng.) and PS2 (German), which have different colors, front and back cover undergo different scaling, and have different languages. Row 2 shows images from Aeon Flux on Xbox and PS2, which have different colors, different front covers, different parts of the back cover, and different locations of logos (the Aeon Flux white logo). Row 3 shows images from Armored Core, which have different scales, and the right one lacks the back cover. Row 4 shows an English and German version of Atari on PS2, which have different colors and languages.

3. Similarity Measures

We compare two broad approaches for measuring similarity between pairs of images:

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 dissimilarity between any pair of images using some distance metric, such as L_2 distance used in this work. We compare several popular feature descriptors:

- SIFT: we compute a standard SIFT [14] descriptor for the whole image, which is then normalized to have unit norm. We use our Matlab implementation.

- Gist: we compute a Gist[3] [16] descriptor for the whole image, which is further normalized to unit length.

[3]Code available at *tinyurl.com/ch537q*

Subset	Min. size	# images	# families	#features
Games 16	16	210	10	207,089
Games 12	12	645	43	586,047
Games 08	8	1,380	127	1,231,072
Games 06	6	2,312	273	2,038,712
Games 04	4	3,961	646	3,459,909
Games 03	3	5,212	1,063	4,476,982
Games 02	2	7,054	1,984	5,882,444
Games 01	1	11,431	6,361	8,524,514
Caltech	5	250	50	246,356
Oxford	5	272	11	423,907

Table 1. Subsets used in the experiments. Top ten subsets are from the games dataset, then the caltech and oxford sets. Second column shows min. size of families

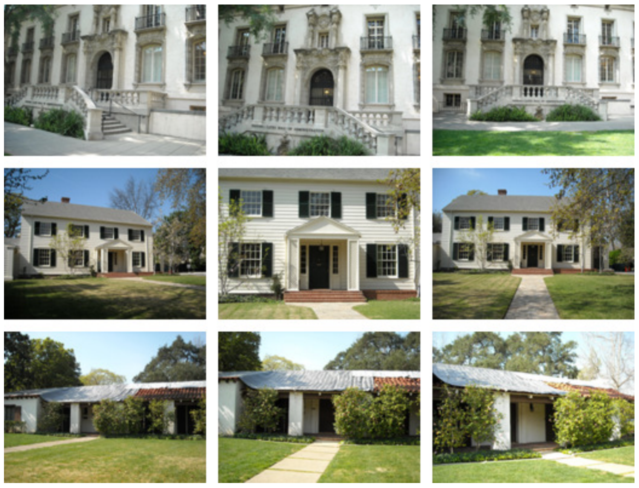
Figure 2. Sample images from **caltech** set. Each row shows three images for a different building taken from different angles and distances.

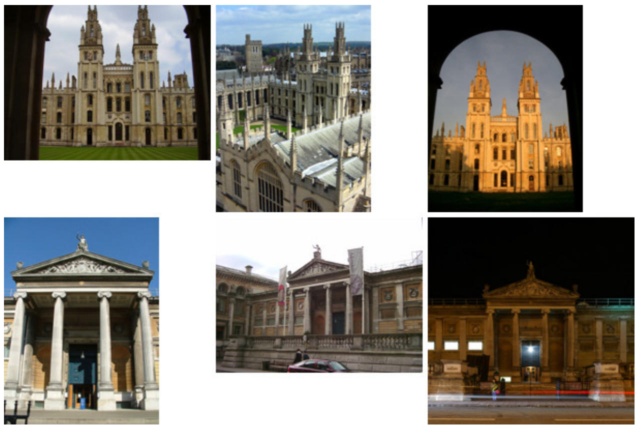
Figure 3. Sample images from **oxford** set. Each row shows three images for a different building taken from the *good* set.

- HOG: we compute a Histogram of Oriented Gradients [3] descriptor for the whole image. We use our Matlab implementation.

- Bag-of-words (BoW): 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 get "visual" words, local features are extracted from the images and vector quantized using K-means to create a codebook of visual words. Each image is represented by a histogram of visual words present in that image. We used the affine covariant feature detector [15] together with SIFT descriptor[4]. We compare different sizes of codebooks: 1K, 5K, 10K, 25K, and 50K visual words. We also compare two variants of BoW:

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

3.2. Local Features

In this approach, each image is represented 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 [15], difference of Gaussian [14] ...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 $k = 1, \ldots, n_i$

[4]Code available at *tinyurl.com/detvd2*

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 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 [13], 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 $l_{g_k} \neq l_k$ i.e. 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 3 methods to measure similarity, with increasing complexity:

1. **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 common features between images i & j.

2. **Image-aff**: first, we perform another processing step. For every image i we process all images that have at least t_c common features, and compute exhaustive nearest neighbors between image i and such images. We set $t_c = 5$ features in the experiments. Next, we check spatial consistency of those 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 [7]. 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.

3. **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. After computing exhaustive nearest neighbors between potential matching image pairs as in **image-aff**, 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.

4. Clustering & Performance Measures

After computing the similarity/dissimilarity between pairs of images as explained above, we get an affinity matrix S with elements s_{ij} defining the similarity/distance between images i & j. We investigate two scenarios for processing this affinity matrix to cluster images into families:

4.1. Semi-supervised Clustering

Where we assume we know the number of families beforehand. Here we compare two graph-theoretic algorithms for clustering a weighted graph represented by an affinity matrix S:

1. **Normalized Cuts** (NC): which tries to infer a k-way partition of S such that the mean normalized cut is maximized [18, 20]. Define $links(A, B) = \sum_{i \in A, j \in B} s_{ij}$ which is the total weighted connections between subsets A & B, and $degree(A) = links(A, S)$ which is the total weight of A. Given a k-way partioning of S into K subsets $V_1, \ldots, V_k$, the mean normalized cut is defined as: $mncut = \frac{1}{K} \sum_{i=1}^{K} \left(links(V_i, V_i^C)/degree(V_i) \right)$ where V_i^C is the complement of subset V_i. $mncut$ is maximized by relaxing the problem, converting it into an eigen-value problem, solving the relaxed one and then searching for a sub-optimal solution[5].

2. **Agglomerative Clustering** (Ag): which builds clusters recursively bottom-up. First, each image belongs to its own cluster [11]. 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 $al = \left(\sum_{i \in A, j \in B} s_{ij} \right) / |A|\,|B|$ where $|A|$ & $|B|$ are the sizes of clusters A & B.

4.2. Unsupervised Clustering

Here we assume we do not know the number of families in advance. We present a new algorithm, which we call **Crancle** (Clustering with Ranked Connected Component Labeling), to automatically cluster the images and estimate the number of families. The algorithm proceeds in three steps:

1. Given the affinity matrix S, we compute a binary connectivity matrix C such that $c_{ij} = 1$ iff images i & j are connected. For each image i, we rank the images in order of decreasing similarity, by sorting row i of S. Then, we set $c_{ij} = 1$ for $j \leq r$ for the top r ranked images i.e. we connect image i with its r most similar images.

[5]Code available at *tinyurl.com/d6ynz9*

```
function labels = crancle(S, r)
  %compute Connectivity matrix C
  C = zeros(m,m);
  for i=1:m
    [s, ids] = sort(S(i,:),'descend');
    C(i,ids(1:r)) = 1;
  end
  %make C symmetric
  C = min(C, C');
  %get connected components
  labels = concom(C);
```

Figure 4. Matlab code for Crancle algorithm

2. Update C such that $c_{ij} = c_{ji} = \min(c_{ij}, c_{ji})$. This makes sure C is symmetric, in addition to eliminating spurious matches by marking i & j as connected only if j is among the top r most similar to i and vice versa.

3. Given C, we perform a two-pass Connected Component Labeling [4] to identify isolated clusters in C.

Fig. 4 shows Matlab code for the algorithm, where we assume S is a similarity matrix i.e. larger values mean more similarity. The intuition behind step 2 above is that images belonging to the same family should be ranked higher in each others' list. Step 3 discovers families by identifying the connected components, and discovers images that are not directly connected in C but are connected through some other images, the so-called transitive connectivity. For example, if image i is connected to j, and image j is connected to k, but there is no connection in C between i & k, step 3 will identify i & k as belonging to the same family.

The algorithm has some good properties: 1) It does not depend on the scaling of S, only the relative values are important; 2) it has only one parameter, r, the number of top ranked images to consider. Small values for r result in a lot of clusters, while large values result in a few clusters. Since the value of r will depend on the dataset, and we want an automated process, we use a simple heuristic to estimate r. We use 10% of the data as a validation set, and check values of r from 1 to 20. The value that returns the best performance is used for clustering.

4.3. Performance Measure

We report results using two performance metrics:

1. Mean Confusion Matrix Performance (MCMP): which is used in the semi-supervised scenario, when the number of clusters is known in advance. The confusion matrix U has entries u_{fk} in row f and column k such that u_{fk} is the number of images that belong to ground truth family f but were classified with cluster k. The MCMP is defined as MCMP $= \frac{1}{N_f} \sum_f u_{ff} / \left(\sum_l u_{fl}\right) \times 100\%$ where N_f is the number of families.

2. F-Measure (FM): which is used in the unsupervised scenario, when the number of inferred clusters is not necessarily equal to the number of ground truth families [10]. Define ground truth families as F, and the inferred families as K. Define precision and recall of cluster K_k with respect to family F_f as: $prec(F_f, K_k) = L_{fk}/|K_k|$ and $rec(F_f, K_k) = L_{fk}/|F_f|$ where L_{fk} is the number of images in cluster k that belong to ground truth family f. Then define $FM(F_f) = \max_k 2 \times prec(F_f, K_k) \times rec(F_f, K_k) / (prec(F_f, K_k) + rec(F_f, K_k))$ which assigns to each ground truth family f the cluster that best matches it. Finally, the F-Measure is the weighted average, defined as FM $= \frac{1}{N} \sum_f FM(F_f) \times |F_f| \times 100\%$, where N is the total number of images.

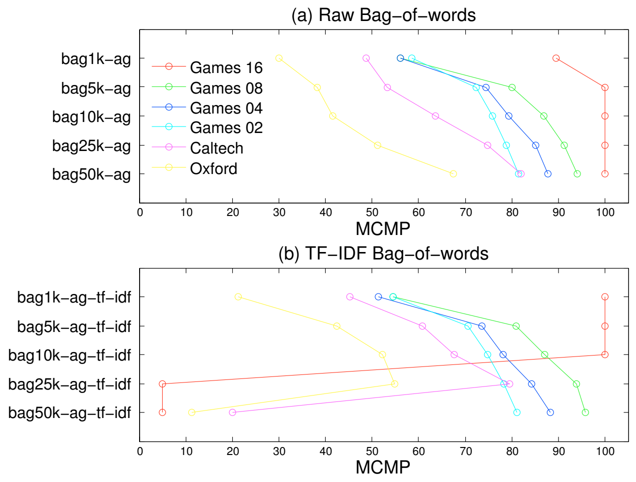

Figure 7. Mean Confusion Matrix Performance for variations of bag-of-words for semi-supervised scenario . (a) shows results for increasing number of words with raw histograms; (b) shows results for increasing number of words with tf-idf weighting. Results are shown for sizes of 1K, 5K, 10K, 25K, and 50K words, using Agglomerative Clustering.

5. Experiments and Discussion

We performed thorough comparisons of the two approaches in §3 on the two clustering scenarios in §4.

Semi-supervised Clustering

- Fig. 5 shows results for this scenario on the subsets in table 1. The subsets are sorted in increasing complexity, and we notice that performance follows suit and degrades with increasing complexity. The oxford set yields the worst result, followed by

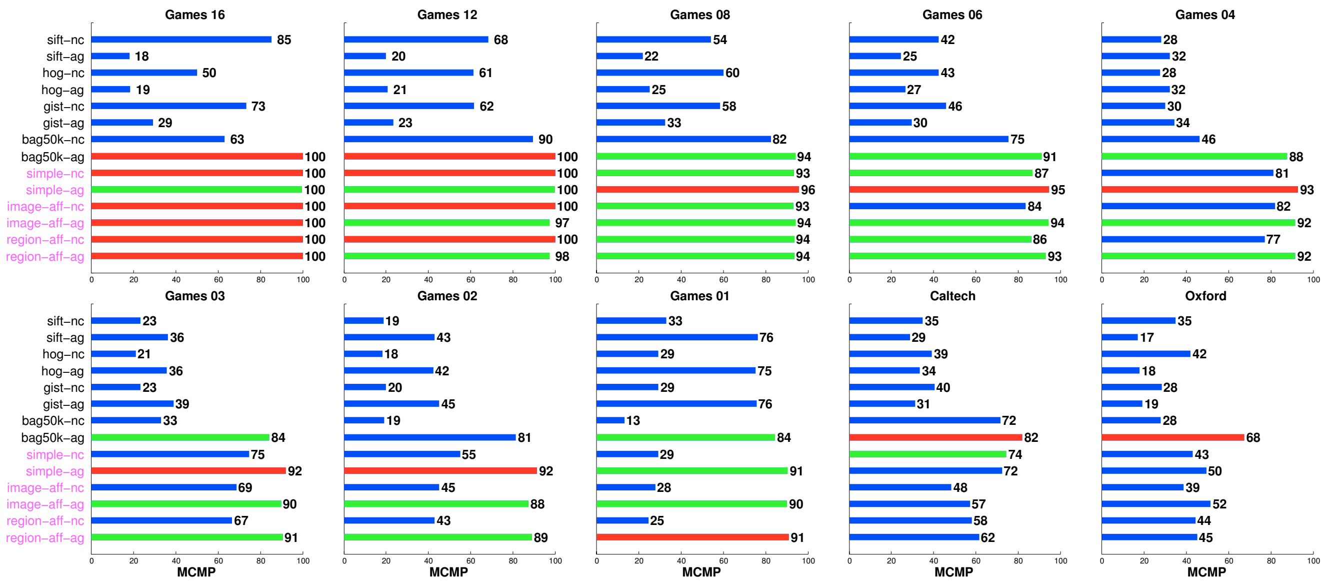

Figure 5. Mean Confusion Matrix Performance for Semi-Supervised Scenario. Red bars are the maximum value, green bars are values within 10% of the max, and blue bars are the rest. Suffix *-nc* signifies Normalized Cuts and *-ag* signifies Agglomerative Clustering.

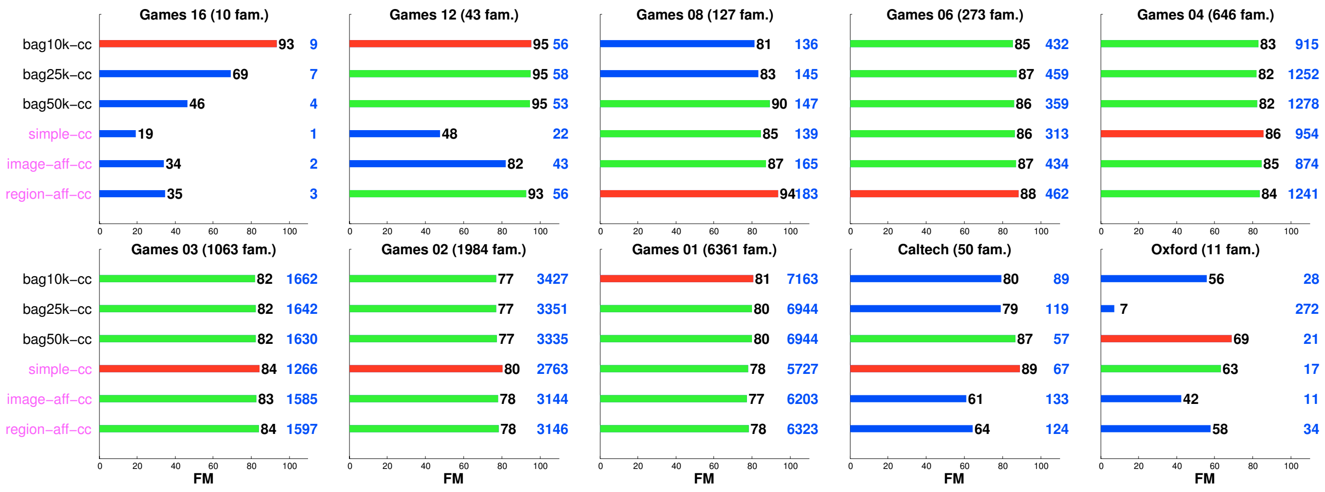

Figure 6. F-Measure Performance for Unsupervised Scenario. Red bars are the maximum value, green bars are values within 10% of the max, and blue bars are the rest. Suffix *-cc* signifies Crancle clustering algorithm. Ground truth number of families in each subset is in the title of each subplot, and number of families discovered from Crancle is in blue.

the caltech set.

- Sift, HOG, and Gist perform very poorly with increasing subset complexity. This suggests they are not useful in this application.

- Agglomerative clustering performs much better than normalized cuts with increasing number of families. The reason is that clustering into k families with NC requires computing k eigenvectors, and this becomes increasingly prone to round-off errors and the scaling of the affinity matrix as k exceeds hundreds of families.

- Local features tend to fare better on the games subsets. BoW performs best on caltech and oxford sets, while remaining within 10% on the games subsets. This is because the latter two subsets contain much more variability within the family, specially viewpoint and lighting changes, and BoW seems more tolerant to such variability, specially with larger codebooks.

- Performing extra spatial checks with local features

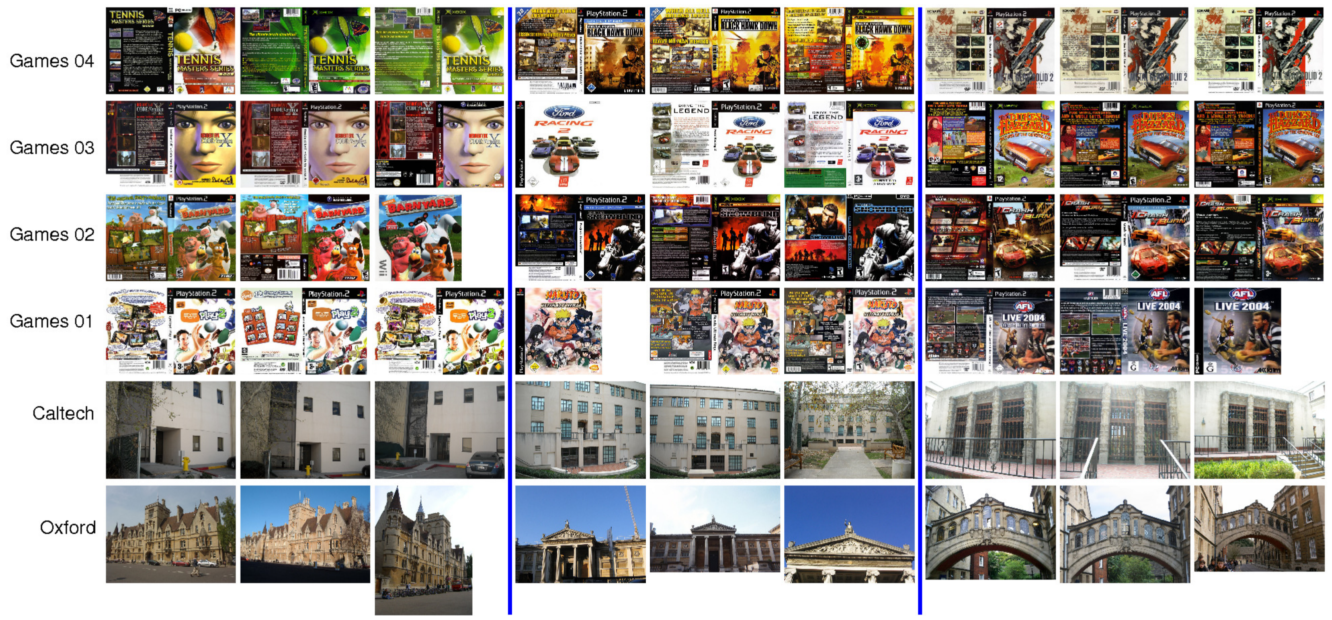

Figure 8. Example of discovered families for semi-supervised scenario using bag-of-words with 50K words and agglomerative clustering . Each row shows three example images from each of three families.

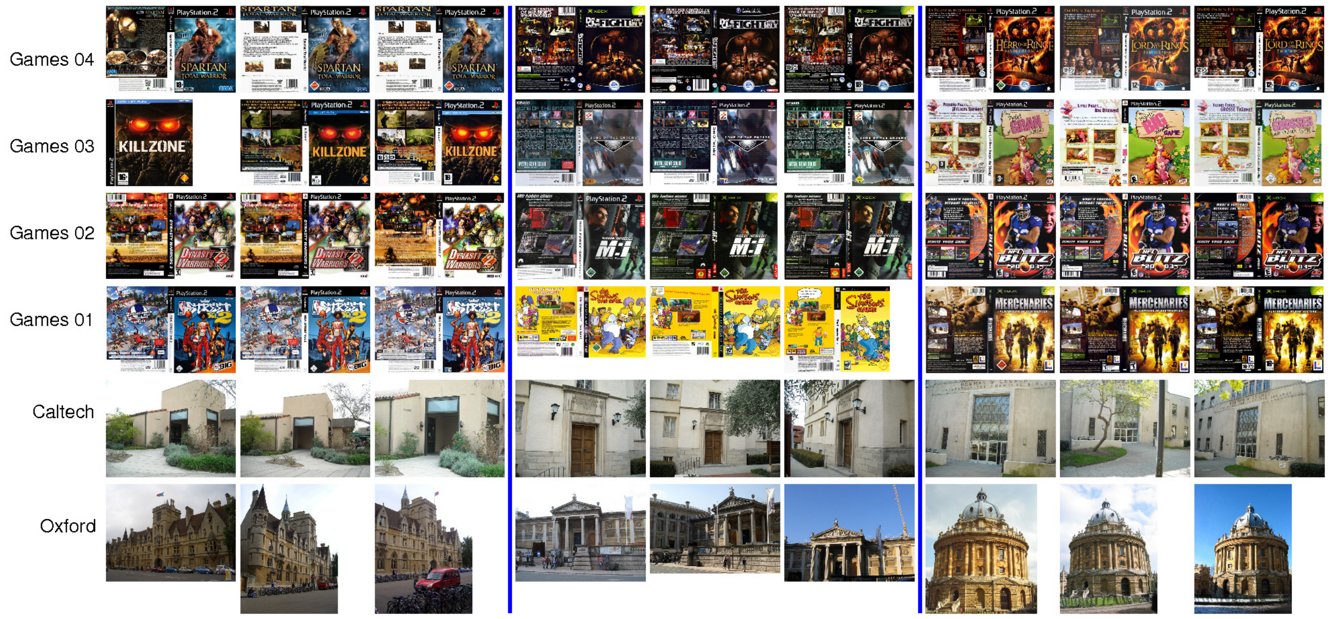

Figure 9. Example of discovered families for unsupervised scenario using bag-of-words with 50K words . Each row shows three example images from each of three families.

does not increase performance that much. Indeed, using the *simple* method is usually better than the other two methods. This suggests that simple feature matching with no spatial checks is enough for this application.

- Fig. 7 shows results for bag-of-words with different codebook sizes and with/without tf-idf weighting scheme for some of the subsets. Without tf-idf weighting, the performance increases monotonically with increasing the codebook size. With tf-idf weighting, performance increases and then decreases sharply when the codebook size is comparable to the total number of features used to create the codebook. This is because in this case there are not enough features to have good statistics about word/document frequencies. However, for subsets with larger number of features, tf-idf gives a slight increase in performance over raw histograms.

Unsupervised Clustering

- Fig. 6 shows results for Crancle algorithm. The algorithm generally overestimates the number of families to within 25-40% of the ground truth number. F-measure generally decreases with increasing subset complexity as expected. Performance is in the 80-90% for the games and caltech subsets, while again it is much worse on oxford subset with maximum f-measure of 69%.

- The performance of BoW is comparable to that of local features, with a slight edge to the former. This is important as BoW is much more storage efficient than local features. With BoW, we only need to store one feature vector per image, while with local features we need to store all the local descriptors for every image. The savings become significant when we have millions of images. This makes BoW more attractive when scaling the up the size of the image collections.

Figs. 8-9 show some sample images from the families discovered by the BoW method with 50K words.

6. Conclusion

We compared two broad approaches for measuring image similarity in the context of automated discovery of image families in unorganized collections. We investigated two clustering scenarios: semi-supervised using normalized cuts and agglomerative clustering; and unsupervised clustering using a new algorithm, Crancle. We presented results on three datasets, and scaled up the problem to over 6,300 families and 11,000 images. Our main findings are:

- It is important to have different datasets with different complexities and statistics for the purpose of comparing performance of different algorithms. The games dataset has more constrained statistics as it is mostly flat artwork, while the oxford dataset has extreme lighting and viewpoint changes.

- Sift, HOG, and Gist are not suitable for this task and provide much worse results.

- Bag-of-words method is more attractive than local features as it provides comparable if not better results, while requiring significantly less storage. It is a good candidate for further study.

- The problem of automatic image family discovery has not received much attention in the vision community, specially when scaling up the problem into collections of millions of images and thousands of families.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, 1999. 3

[2] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, pages 549–556, 2007. 1, 2

[3] 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

[4] M. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 39(2):253–280, 1992. 1, 5

[5] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, volume 2, pages 524–531, 2005. 1

[6] J. Foo, J. Zobel, R. Sinha, and S. Tahaghoghi. Detection of near-duplicate images for web search. In *CIVR*, pages 557–564, 2007. 1

[7] D. Forsyth and J. Ponce. *Computer Vision: A modern approach.* Prentice Hall, 2004. 4

[8] B. Gao, T. Liu, T. Qin, X. Zheng, Q. Cheng, and W. Ma. Web image clustering by consistent utilization of visual features and surrounding texts. In *MULTIMEDIA*, pages 112–121, New York, NY, USA, 2005. ACM. 1

[9] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *ICCV*, 2003. 1

[10] K. Hammouda and M. Kamel. Collaborative document clustering. In *SDM*, pages 453–463, 2006. 5

[11] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. In *ACM Computing Surveys*, pages 264–323, 1999. 1, 4

[12] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA*, pages 869–876, 2004. 1

[13] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, pages 775–781, 2005. 4

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 1, 3

[15] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60:63–86, 2004. 1, 3

[16] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001. 3

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR*, 2007. 1, 2

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, Aug. 2000. 1, 4

[19] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. *CVPR*, 2008. 1

[20] S. Yu and J. Shi. Multiclass spectral clustering. In *ICCV*, pages 313–319, 2003. 4