# Scalable Visual Instance Mining with Instance Graph

Wei Li[1]
liwei2658@gmail.com
Changhu Wang[2]
chw@microsoft.com
Lei Zhang[3]
leizhang@microsoft.com
Yong Rui[2]
yongrui@microsoft.com
Bo Zhang[1]
dcszb@mail.tsinghua.edu.cn

[1] State Key Lab of Intelligent Technology and Systems, TNList, Department of Computer Science and Technology, Tsinghua University
Beijing 100084, China

[2] Microsoft Research
No. 5 Danling Street, Haidian District,
Beijing 100080, China

[3] Microsoft Research
One Microsoft Way,
Redmond WA 98052, USA

**Abstract**

In this paper we address the problem of visual instance mining, which is to automatically discover frequently appearing visual instances from a large collection of images. We propose a scalable mining method by leveraging the graph structure with images as vertices. Different from most existing work that focused on either instance-level similarities or image-level context properties, our graph captures both information. The instance-level information is integrated during the construction of a weighted and undirected instance graph based on the similarity between augmented local features, while the image-level context is explored with a greedy breadth-first search algorithm to discover clusters of visual instances from the graph. This method is capable of mining challenging small visual instances with diverse variations. We evaluated our method on two fully annotated datasets and outperformed the state of the arts on both datasets with higher recalls. We also applied our method on a one-million Flickr dataset and proved its scalability.

## 1 Introduction

Visual instances are the basic re-occurring information in the visual world. In this paper we address the problem of visual instance mining. Specifically, the goal is to automatically discover frequent visual instances from a collection of images. We are interested in mining specific instances, such as Air Force One and Monarch Butterfly, which are different from high-level visual categories like plane and butterfly.

Visual instance mining plays a fundamental role in many tasks in multimedia analysis, such as multimedia summarization [13], image archaeology [6] and trend prediction. Another potential application of this work is automatic image annotation [12]. The visual instances mined from a very large image database are likely to cover many representative instances.

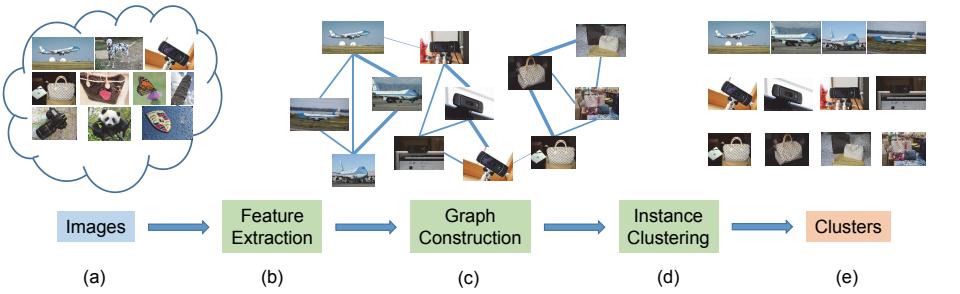



Figure 1: The framework of the proposed method for visual instance mining.

We can build a knowledge base with the semantics of discovered visual instances. Then an unseen image can be matched to some indexed visual instances, whose semantic information can be propagated. Some vertical applications, such as logo recognition [10], are also based on visual instance mining.

The problem of visual instance mining brings two challenges. The first one is the mining of small instances that have large variations. In many cases, visual instances may only cover very limited image areas, which sometimes can hardly be noticed even by humans. Different types of variations, such as scale, rotation, and occlusion, especially the variations resulted from out-of-plane rotation and non-rigid objects, require highly robust algorithms. The second challenge is the scale of the dataset. A large-scale database is essential for discovering practically useful visual instances. However, large-scale databases usually contain more noises that significantly affect the performance of the methods designed on small databases. Also, the need of high efficiency rules out complex and non-parallelizable methods.

Most existing work focused on large instance mining, or image clustering. A number of methods based on min-hash [1, 2, 7] are developed for this purpose. Typically, min-hash values on BoW feature sets are computed and sketch collisions indicate matched image pairs. These methods can find landmarks in large image collections, but they do not work well on small instances, because the Jaccard similarity between two images sharing a small instance is low, leading to a low min-hash collision probability. Furthermore, when the dataset is large, the number of random collisions that lead to false positives will dramatically increase, as analyzed in [16]. Philbin and Zisserman [9] constructed a matching graph by searching with every image as a query in the dataset and then finds its dense sub-graphs. Their matching graph is a powerful way to represent image-level context properties, but image search with spatial verification makes the graph construction slow, and it also suffers from a low recall due to the whole-image matching. Among a few methods that target for scalable small instance mining, geometric min-hashing (GmH) [3] and thread of features (ToF) [16] are two representative works. GmH extends min-hash by considering the geometric relationships between local features, thus being more tolerant with the size of instances. ToF first discovers feature threads each of which consists of a set of local features with similar descriptors and neighborhoods, then leverages min-hash collisions to find clusters of feature threads, and generates final instance clusters by voting within each thread cluster. These methods are capable of discovering small instances via instance-level local matches, but they do not consider the structure of connections between images.

To tackle these two challenges, we propose a novel method that is robust and scalable from the graph perspective for mining both large and small instances. We argue that a graph

structure constructed by local matches not only helps to capture the similarities between instances, but also reflects image-level context properties via indirect paths, thus combining the advantages of [9] and [16]. The framework is shown in Figure 1. First of all we extract local SIFT [8] features, quantize them [11], and augment each local feature with the Hamming Embedding [4] binary code of its feature vector and neighbor features. Then we build a weighted and undirected instance graph with images as vertices. The similarity scores between the augmented local features, which combine the HE weighting scheme [5] and the Jaccard similarity of neighboring visual word sets, contribute to the weights of edges between images. From the sparse instance graph we are able to efficiently discover instance clusters by the proposed greedy breadth-first search algorithm. Experiments show that our method outperforms the state-of-the-art ToF method on both clustering performance and running speed.

The rest of the paper is organized as follows. We introduce the proposed method in Section 2. Section 3 discusses some implementation details. Section 4 presents the experimental results, followed by conclusions and future work in Section 5.

# 2 The Proposed Framework

As shown in Figure 1, our method has three steps: feature extraction, graph construction, and instance clustering. Next, we discuss each step in detail.

## 2.1 Feature Extraction

Given a collection of images, the first step of our method is to extract local SIFT [8] features from each image and quantize each feature to a visual word [11] using a one-million codebook trained on a separate dataset. Each image is then represented by a set of features $F = \{f_1, f_2, \ldots, f_n\}$, where $n$ is the number of features. Each feature $f_i$ $(1 \leq i \leq n)$ is a tuple $< v, g, x, y, s >$, where $v$ is its visual word ID, $g$ is the image ID, $(x, y)$ is its location, and $s$ is its scale.

Due to the information loss of quantization and the limited discriminability of single local features, we augment each local feature using the information of both the raw feature vector and its neighbors. First, we compress the 128-dimensional floating-point SIFT feature vector into a compact 32-bit binary code representation by Hamming Embedding (HE) [4]. HE projects feature vectors to a 32-dimensional space by a random orthogonal matrix, and then binarizes them with thresholds learned independently on each visual word. Conceptually, it further partitions the Voronoi cell of each visual word to encode more information. Second, we integrate the visual word IDs of its neighbor features, where the neighboring relationship is measured by the Euclidean distance between feature locations. We only consider neighbor features that have similar scale (scale ratio between 0.5 and 2) to the augmented feature. Finally each augmented local feature is represented by the tuple $< v, g, c, V >$, where $c$ is the 32-bit HE binary code and $V$ is the set of neighboring visual words.

## 2.2 Graph Construction

After feature extraction and augmentation, we have a large set of augmented local features. In this step we match the augmented local features to construct the instance graph and thus

connect images. In this way the graph captures both instance-level similarities and image-level context properties.

We first discuss the similarity measure between two augmented features, denoted by $f_i$ and $f_j$, where $f_i =< v_i, g_i, c_i, V_i >$ and $f_j =< v_j, g_j, c_j, V_j >$. For the purpose of building the instance graph, we are only interested in highly similar features. In order to efficiently compute the similarity, we partition the feature space by the visual word $v$. In other words, if $v_i \neq v_j$ their similarity $s_{ij}$ will be 0. This reduces the time cost to $1/w$ of the brute-force method, where $w$ is the codebook size which is one million in this work.

For $f_i$ and $f_j$ having $v_i = v_j$, the idea is to combine the similarity of their HE binary codes and neighbor feature sets, as they are complementary to each other in terms of feature discrimination. As in [5], we define the similarity score of HE codes as

$$s_{HE} = -\log_2(\frac{1}{2^l}\sum_{k=0}^{h} C(l,k)), \tag{1}$$

where $l$ is the length of HE code (32 in our work), $h$ is the Hamming distance between $c_i$ and $c_j$, and the combinatorial function $C(l,k)$ is computed by

$$C(l,k) = \frac{l!}{k!(l-k)!}. \tag{2}$$

All the possible $l+1$ values of $s_{HE}$ are computed in advance and stored in a look-up table. The similarity score of neighbor feature sets is their Jaccard similarity:

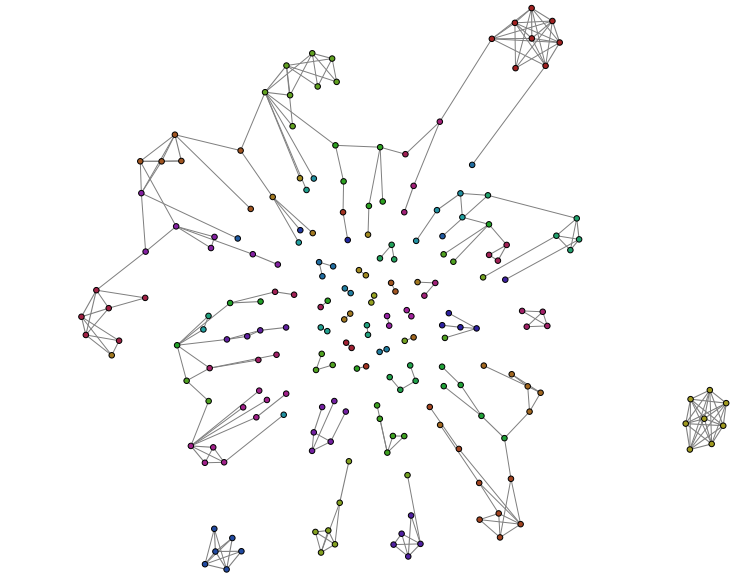
$$s_{NR} = \frac{|V_i \cap V_j|}{|V_i \cup V_j|}. \tag{3}$$

As $s_{HE}$ is in the range $[0,l]$ and $s_{NR}$ is in the range $[0,1]$, we normalize $s_{HE}$ to $[0,1]$ and the final similarity between two augmented local features is

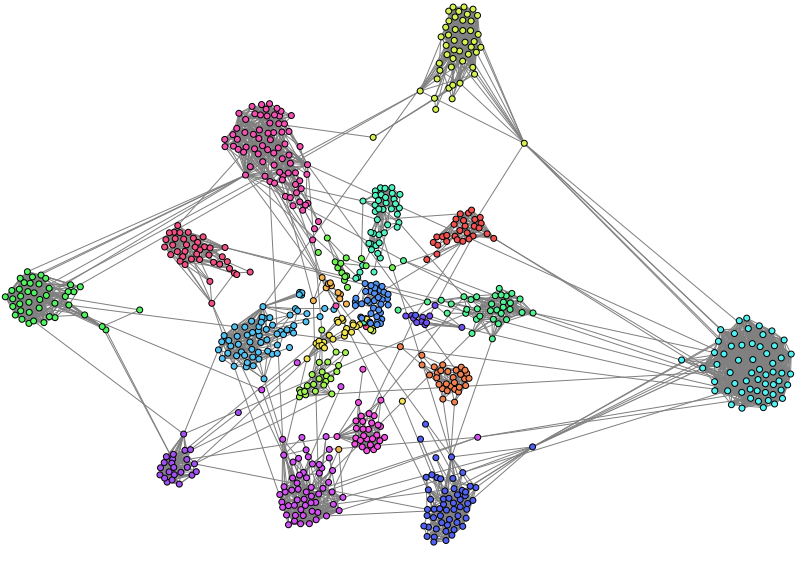
$$s_{ij} = \frac{1}{l}s_{HE} + s_{NR}. \tag{4}$$

Note that $s_{HE}$ can be computed efficiently by $XOR$ operation and table look-up, while the computation of $s_{NR}$ is relatively slow. Therefore, to further speed up the computation, we prune the computation process and set the similarity directly to 0, if the Hamming distance $h$ is larger than a certain Hamming threshold $h_t$. As $h_t$ is usually small and $h$ follows a binary distribution $B(l,0.5)$, this pruning technique can save remarkably much computational effort. To increase the precision, we also set similarity to 0 if there are less than two common neighbors in $V_i$ and $V_j$.

Based on the defined similarity measure, we can construct the instance graph. The graph $G =< V,E >$ is a sparse weighted undirected graph. Each vertex $v \in V$ represents an image in the database. For each feature pair of $f_i$ and $f_j$ that have non-zero similarity score, they contribute the score $s_{ij}$ to the weight of the edge $e \in E$ connecting the two images $g_i$ and $g_j$. Multiple contributions to the same edge are combined by summing up all the scores, resulting in an edge with a higher weight. The weight between two images reflects the confidence that they share the same visual instances. High confidence can be gained either by a single strong local feature match or multiple weak matches. Figure 2 illustrates two instance graphs constructed from two datasets MQA [15] and PartialDup [14].



(a) MQA



(b) PartialDup

Figure 2: Visualization of the instance graphs of two datasets (a) MQA and (b) PartialDup (nodes without any connections are not shown), constructed by our method. Colors of vertices illustrate the ground truth clusters. Edge width reflects the confidence of two images sharing the same visual instance. The layout of each graph places together images which are connected by high confidence edges. Clear clustering patterns can be observed.

### 2.3 Instance Clustering

The instance graph gives a good insight of the structure of the image database. We propose a greedy breadth-first search (GBFS) algorithm to discover clusters by exploring the graph structure, given a specific image as entry point. The main idea is to add images in the connected components in a layer-by-layer greedy way, while the rules become stricter as more distant vertices are explored. The detailed algorithm is summarized in Algorithm 1.

**Algorithm 1** GBFS Clustering on the Instance Graph

**Input.** The instance graph $G$ and an image $I$.
**Step 1.** Initialize an instance cluster $C$ that contains only one image $I$. Initialize an empty queue $Q$ and enqueue $I$ to $Q$.
**Step 2.** Dequeue an image from $Q$ and denote it as $I^*$. Find all neighbors of $I^*$ in the graph $G$ and order them by corresponding edge weights in a descending order. The neighbor set is denoted as $N(I^*)$. Remove from $N(I^*)$ all elements that are already in cluster $C$.
**Step 3.** For each image in $N(I^*)$, use a test (average weight test) to judge if it should be added to $C$. If it is added to $C$, enqueue it to $Q$.
**Step 4.** Loop step 2 and step 3 until $Q$ is empty or the search in the maximum search depth $d_{max}$ is completed.
**Output.** The instance cluster $C$.

In Algorithm 1, the average weight test computes the average weight between the test image and each image already in $C$. It passes the test if and only if the average weight is larger than a threshold $w_t$. To punish images more distant from the input image $I$, $w_t$ becomes larger as the search goes deeper. In our experiments, considering the efficiency and the balance between precision and recall, we set the maximum search depth $d_{max} = 2$ and $w_t = 0.5$ when search depth is 1, and $w_t = 5$ when search depth is 2.

The algorithm runs with each image as an entry point in turn. Note that there may be duplicate and highly similar clusters obtained from different entry points. The duplicate clusters can be simply removed without affecting the evaluations. Like our baseline ToF [16] method, we do not further post-process similar clusters.

## 3 Implementation

The whole framework is implemented in parallel. Feature extraction is easy to parallelize, as the computation on each image is independent. Instance clustering is also a parallel step, since clusters obtained from different entry points can be computed at the same time. Graph construction partitions the feature space to parallelize the input, but the output needs more consideration. As different partitions may contribute to the same edge, we maintain a partial instance graph within each feature space partition, and in the end we merge them together to form the complete instance graph. The merging is much faster than the computation of the similarity score, so it will not become a bottleneck of our method.

## 4 Experiments

In this section we first conduct experiments on two benchmark datasets, i.e., MQA [15] and PartialDup [14] to evaluate the proposed method, and then apply it on a large scale dataset

to show its scalability.

## 4.1 Benchmark Datasets

The MQA dataset [15] and the PartialDup dataset [14] are leveraged to quantitatively evaluate the performance of our method. The first dataset contains 438 images with 52 instance clusters, while the second has 20 clusters and totally 782 images. We select these two datasets because they have a variety of visual instance sizes and different types of intra-class variations exist. MQA mainly emphasizes out-of-plane rotation, non-rigid objects, various lighting conditions, and occlusion. On the other hand, PartialDup is closer to the definition of web duplicates, having variations such as scale changes, in-plane rotation, different image quality, and cropping.

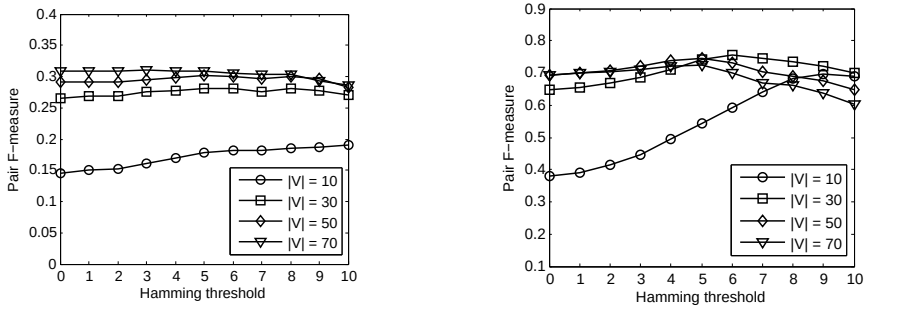
## 4.2 Evaluation Measures

Let us use $C_{GT}$ to denote the set of ground truth clusters, and $C_{DR}$ the discovered cluster set. The same evaluation measures as in [16] are adopted, which are based on image pairs and called Pair Measures in this paper. They include Pair Precision, Pair Recall, and Pair F-measure. Pair Precision ($P_{pair}$) is defined as the number of image pairs in both $C_{GT}$ and $C_{DR}$ divided by the number of image pairs in $C_{DR}$. Pair Recall ($R_{pair}$) is defined as the number of image pairs in both $C_{GT}$ and $C_{DR}$ divided by the number of image pairs in $C_{GT}$. Pair F-measure ($F_{pair}$) is the harmonic mean of $P_{pair}$ and $R_{pair}$.

We observed that Pair Measures cannot tell the difference between fragmented clusters and a single big cluster, as long as they cover the same set of image pairs. Therefore, to have a better evaluation, we also propose another type of measurements called Cluster Measures that favor big clusters. For each cluster $C_A$ in $C_{GT}$ and every cluster $C_B$ in $C_{DR}$, we compute the precision as $|C_A \cap C_B|/|C_B|$, the recall as $|C_A \cap C_B|/|C_A|$, and the F-measure as the harmonic mean of precision and recall. The $C_B$ with the best F-measure is selected, and its precision and recall are defined as the measures for the ground truth cluster $C_A$. Averaging precision and recall over all the clusters in $C_{GT}$ results in the final Cluster Precision ($P_{cluster}$) and Cluster Recall ($R_{cluster}$). Cluster F-measure ($F_{cluster}$) is the harmonic mean of $P_{cluster}$ and $R_{cluster}$.

## 4.3 Parameter Selection

There are two parameters that have large impacts on the performance, i.e., the number of neighbor features $|V|$ and the Hamming threshold $h_t$. We tried different combinations of the two parameters and plot the curves of Pair F-measure in Figure 3. Small $|V|$ and $h_t$ will reduce the memory cost and increase running speed, but we observed from Figure 3 that at least one of them needs to be large to achieve a good performance, especially $|V|$. A small $|V|$ fails to provide enough neighborhood context information and reduces the discriminative power of augmented features, while a small $h_t$ leads to low tolerance with variations. For the MQA dataset, although using a very large $|V|$ improved the F-measure, it resulted in an apparent drop of precision and cost more memory space. Therefore, we set $|V| = 50$ and $h_t = 5$ for a balance. For the PartialDup dataset, we choose $|V| = 30$ and $h_t = 6$.




(a) MQA (b) PartialDup

Figure 3: Performance of our method with different number of neighbor features $|V|$ and Hamming threshold $h_t$ on (a) MQA and (b) PartialDup.

## 4.4 Algorithm Comparison

As ToF outperforms other state-of-the-art visual instance mining methods [16], we adopt ToF as our baseline. We implemented it by ourselves and achieved similar performance on MQA dataset as in their paper with similar parameter configurations. Table 1 shows the comparison results of ToF and our instance graph (IG) method on the two datasets.

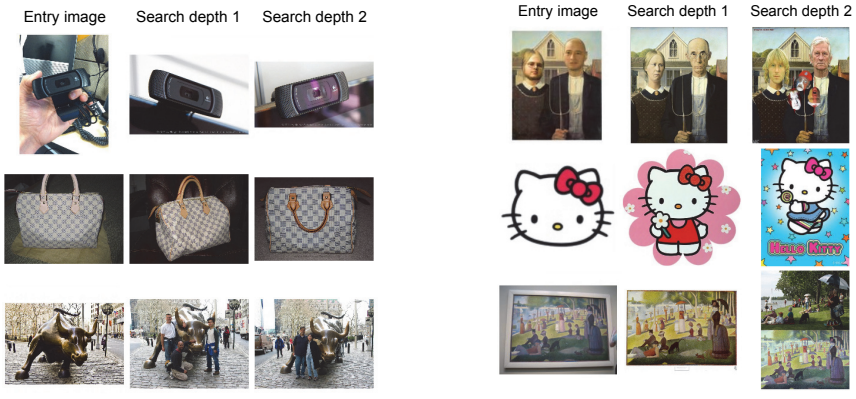
From Table 1 we can see that, both Pair F-measure and Cluster F-measure of IG outperform those of ToF on both datasets, which resulted from a large recall improvement although with a slight loss of precision. Figure 4 illustrates some clustering examples on the instance graph, from which we can see the effectiveness of the graph structure in image relationship representation over simple feature threads. Furthermore, IG performs much better than ToF when using Cluster Measures. This indicates that ToF suffered from cluster fragmentation while IG solved this issue by fully exploring the graph with the GBFS algorithm.

Table 1 also shows the results of two variations of our method. IG-HE defines the similarity of augmented features simply as $s_{ij} = \frac{1}{l} s_{HE}$ without the neighbor information. Compared to IG, the performance of IG-HE dropped on MQA, but stayed at the same level on PartialDup. This shows that neighbor information is more useful when viewpoints are changed and non-rigid objects exist. IG-EW uses equal weights for different search depths in the GBFS algorithm. The Pair Measures dropped dramatically as more noises were introduced. However, the Cluster Measures became even better, which shows that indirect connections contain context information for better clustering.

With a 16-core machine and the same level of code optimization, the core steps of ToF (feature threading + thread clustering + cluster voting) cost 6.7s and 8.9s on MQA and PartialDup, respectively, while the core steps of IG (graph construction + instance clustering) only take 1.5s and 0.9s. This proves the efficiency of our IG method over ToF.

## 4.5 Instance Graph Visualization

Figure 2 visualizes the instance graphs of the two datasets, constructed with the best parameters. We can see that the graphs successfully captured the structure of the datasets, as clear clustering patterns can be observed. The sparsity of the graphs is also confirmed, as their average degrees of vertices are 1.26 and 11.17, respectively.




(a) MQA (b) PartialDup

Figure 4: Illustration of the high recall of the IG method over the ToF method on (a) MQA and (b) PartialDup. The three images in each row demonstrate a search path in the instance graph, where the left and the right images are connected via the middle one. ToF method is not capable of discovering such relationships due to its thread structure and voting scheme.

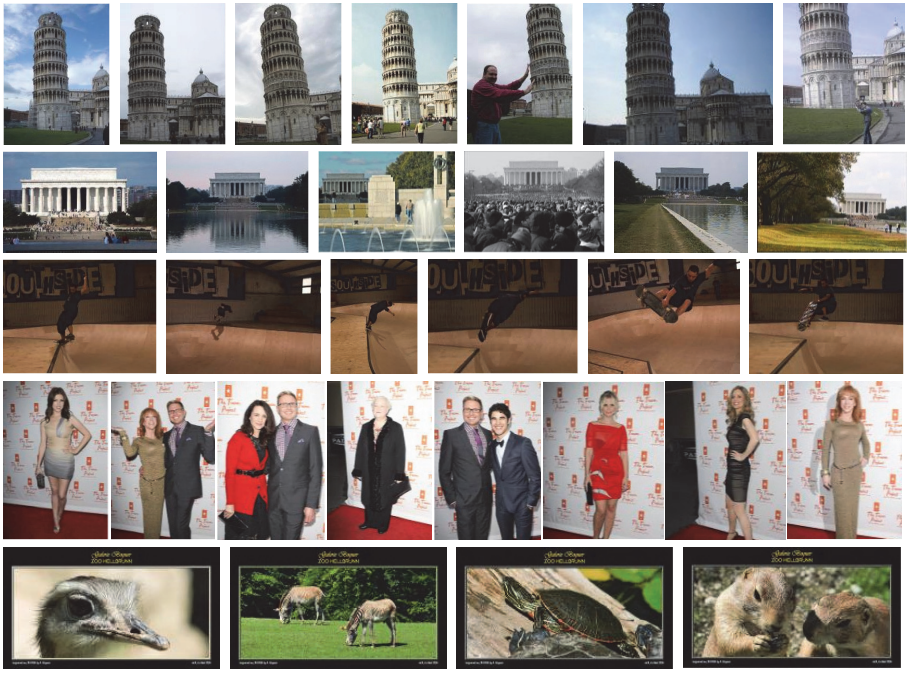


Figure 5: Example clusters mined from the Flickr dataset. The discovered instances vary from large-size buildings (rows 1 and 2) and medium-size sign (row 3) to small-size logo and watermark (rows 4 and 5).

Table 1: Performance comparisons on the MQA and PartialDup datasets.

| *Dataset* | *Method* | $F_{pair}$ | $P_{pair}$ | $R_{pair}$ | $F_{cluster}$ | $P_{cluster}$ | $R_{cluster}$ |
|---|---|---|---|---|---|---|---|
| MQA | ToF | 0.284 | 0.639 | 0.183 | 0.440 | 0.776 | 0.307 |
| MQA | IG | 0.301 | 0.613 | 0.200 | 0.510 | 0.756 | 0.385 |
| MQA | IG-HE | 0.294 | 0.618 | 0.193 | 0.506 | 0.758 | 0.379 |
| MQA | IG-EW | 0.292 | 0.495 | 0.207 | 0.519 | 0.742 | 0.400 |
| PartialDup | ToF | 0.745 | 0.887 | 0.643 | 0.477 | 1.000 | 0.313 |
| PartialDup | IG | 0.757 | 0.835 | 0.693 | 0.717 | 0.983 | 0.565 |
| PartialDup | IG-HE | 0.757 | 0.837 | 0.691 | 0.717 | 0.983 | 0.564 |
| PartialDup | IG-EW | 0.587 | 0.467 | 0.792 | 0.839 | 1.000 | 0.723 |

### 4.6 The Scalability of the Proposed Method

We tested the scalability of our method on a dataset that contains one million images crawled from Flickr. Due to the lack of ground truth, we do not quantitatively evaluate the instance mining performance on this dataset. Figure 5 shows some example clusters automatically discovered from the images. We can see that our method is capable of mining both large and small visual instances with different types of variations. The core graph construction and instance clustering steps only cost less than an hour with a 16-core machine, which shows the scalability of our method.

## 5 Conclusions and Future Work

In this paper we have presented a scalable instance graph method to tackle the visual instance mining problem. The instance graph is constructed via local matches and reflects the global structure of an image database. We propose algorithms for efficient computation of similarity between augmented features and discovering clusters on the constructed graph. Experimental results show the superior performance of our method compared to the state of the arts and also its scalability to large-scale datasets.

One possible future work is to fully explore the structure of the instance graph. For example, weight propagation can be applied to update the similarity between images. Another future direction is to improve the instance clustering method on the graph, making it more effective and no longer require an entry point.

## 6 Acknowledgment

The work of Wei Li and Bo Zhang was partially supported by the National Basic Research Program (973 Program) of China (Grant Nos. 2013CB329403 and 2012CB316301), National Natural Science Foundation of China (Grant Nos. 91120011 and 61273023), and Tsinghua University Initiative Scientific Research Program No. 20121088071.

# References

[1] O. Chum and J. Matas. Large scale discovery of spatially related images. *IEEE TPAMI*, 2010.

[2] O. Chum and J. Matas. Fast computation of min-hash signatures for image collections. In *CVPR*, 2012.

[3] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: finding a (thick) needle in a haystack. In *CVPR*, 2009.

[4] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[5] H. Jegou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 2010.

[6] L. Kennedy and S.-F. Chang. Internet image archaeology: automatically tracing the manipulation history of photographs on the web. In *ACM MM*, 2008.

[7] D. Lee, Q. Ke, and M. Isard. Partition min-hash for partial duplicate image discovery. In *ECCV*, 2010.

[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[9] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *ICVGIP*, 2008.

[10] S. Romberg, L. Pueyo, R. Lienhart, and R. van Zwol. Scalable logo recognition in real-world images. In *ICMR*, 2011.

[11] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *ICCV*, 2003.

[12] X.-J. Wang, L. Zhang, M. Liu, Y. Li, and W.-Y. Ma. Arista - image search to annotation on billions of web photos. In *CVPR*, 2010.

[13] X.-J. Wang, Z. Xu, L. Zhang, C. Liu, and Y. Rui. Towards indexing representative images on the web. In *ACM MM*, 2012.

[14] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *CVPR*, 2009.

[15] W. Zhang, L. Pang, and C.-W. Ngo. Snap-and-ask: answering multimodal question by naming visual instance. In *ACM MM*, 2012.

[16] W. Zhang, H. Li, C.-W. Ngo, and S.-F. Chang. Scalable visual instance mining with threads of features. In *ACM MM*, 2014.