Scalable Visual Instance Mining with Threads of Features [*]

Wei Zhang[†], Hongzhi Li[‡], Chong-Wah Ngo[†], Shih-Fu Chang[‡]
[†] Dept. of Computer Science, City University of Hong Kong
[‡] Dept. of Computer Science, Columbia University
wzhang34-c@my.cityu.edu.hk, hongzhili@cs.columbia.edu
cscwngo@cityu.edu.hk, sfchang@cs.columbia.edu

ABSTRACT

We address the problem of visual instance mining, which is to extract frequently appearing visual instances automatically from a multimedia collection. We propose a scalable mining method by exploiting Thread of Features (ToF). Specifically, ToF, a compact representation that links consistent features across images, is extracted to reduce noises, discover patterns, and speed up processing. Various instances, especially small ones, can be discovered by exploiting correlated ToFs. Our approach is significantly more effective than other methods in mining small instances. At the same time, it is also more efficient by requiring much fewer hash tables. We compared with several state-of-the-art methods on two fully annotated datasets: MQA and Oxford, showing large performance gain in mining (especially small) visual instances. We also run our method on another Flickr dataset with one million images for scalability test. Two applications, instance search and multimedia summarization, are developed from the novel perspective of instance mining, showing great potential of our method in multimedia analysis.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Instance Mining; Thread of Features; min-Hash; Clustering; Summarization

1. INTRODUCTION

In the past decade, multimedia researchers mainly focused on visual search (given a query, retrieve similar images), while visual mining has not yet been fully studied. This paper addresses the problem of *visual instance mining*, which is to automatically discover and extract frequent visual instances out of a data collection. The term "*instance*" in this paper is referred as a specific "visual entity" (e.g., an object/location/logo/person). We highlight this problem by comparing with its relatives: (1) different from visual search, instance mining does not have any query as entry point; (2) compared to general "object class" discovery (e.g., car, dog) [2, 8, 18, 23], specific instances (e.g., this car, this dog) are expected as output; (3) compared to image-level clustering [15], instance mining also extracts objects covering small image areas.

[*]Area chair: Xian-Sheng Hua

MM'14, November 3–7, 2014, Orlando, Florida, USA.
Copyright 2014 ACM 978-1-4503-3063-3/14/11 ...$15.00.
http://dx.doi.org/10.1145/2647868.2654942.

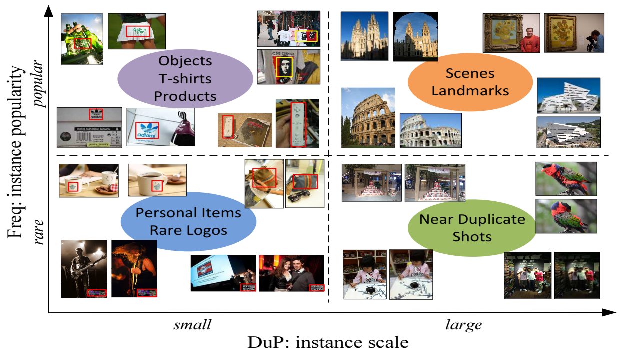


Figure 1: A wide range of visual instances, plotted on a 2D plane of "Freq-DuP". Note *Freq* is the number of images sharing an instance, and *DuP* indicates the instance scale. We propose a robust instance mining method, particularly effective for small scale instances.

As shown in Fig. 1, there are various instances in a real-world data collection. Roughly, they can be characterized by two dimensions: "Frequency" (Freq) and "Duplicate Proportion" (DuP). "Freq" of an instance is the number of images sharing that instance, and "DuP" denotes the duplicate proportion among images with the instance. Roughly speaking, "Freq" depicts the popularity (rare or popular) of an instance, and "DuP" indicates the instance scale (small or large). Small and rare instances (Fig. 1 bottom-left) correspond to personal belongings (e.g., a personal mug) or rare logos. Large and popular instances (top-right) are mostly famous landmarks or scenes, while large and rare ones (bottom-right) are often near duplicates shots (probably taken by the same user within a short time of period). Small and popular instances (top-left) are usually consistent patterns featuring popular objects (e.g., famous prod-

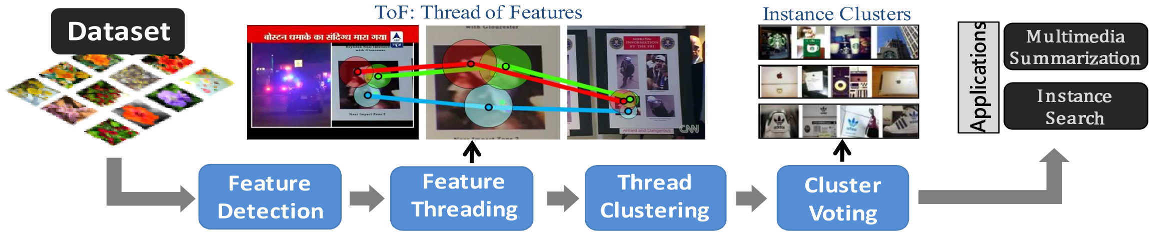


Figure 2: The framework of our proposed instance mining approach.

ucts, visual memes). For a real application, Freq is usually positively correlated with the popularity/significance of an instance, and DuP is negatively correlated with the mining difficulty.

Mining frequent instances from large multimedia collection is essential for many tasks in multimedia analysis, such as multimedia summarization, trend predication, product recommendation and instance search. For example, given the photos uploaded in a certain time and location (e.g., Flickr photos uploaded during last year in New York City), instance mining could be applied to (1) organize the collection for browsing; (2) discover popular items (e.g., restaurant, tee) or events (e.g., Macy's parade); (3) analyze trends for fashion/product/tourism. As shown in Fig. 8, 9 and 10, popular instances mined from a dataset could be treated as a "high-level" feature for understanding the dataset. People can grasp the key ideas of the data collection with a quick glance at the mining results, saving a lot of time and effort. Another interesting aspect to note is the serendipity in mining results. Some instances in the dataset could be very difficult to be noticed by a human. However, automatic mining could reveal some of these underlying features.

Instance mining is still challenging in terms of efficiency and effectiveness. Most existing methods suffer from either scalability or quality issue. Methods designed for small instances [11, 22, 24, 20] can only handle few thousands of images. On the other hand, scalable methods [14, 4, 5] are mostly less compatible with small instances. Scalable instance-level mining remains to be a problem. In this work, we address both problems at the same time, targeting for mining all frequent instances, especially small ones, at million scale dataset. Specifically, a two-step framework is adopted. As shown in Fig. 2, we first thread features that potentially share an instance via efficient pruning, and then cluster these threads to vote for instance clusters. The first step is motivated by the fact that only a very small fraction of local features, which are likely to repeat among images, are helpful in instance mining. We highlight these features by threading them together. The second step is featured by threads clustering rather than image clustering, which significantly boosts the chance of mining small instances.

The main contribution of this paper is the proposal of a visual mining approach that addresses the scalability and small issues at the same time. Compared to other methods, our approach is more effective in mining small instances, while being scalable for million scale dataset. Furthermore, through two applications, we show novel solutions to the problem of instance search and multimedia summarization. Both of our applications are the first attempts to solve the original problems (search/summarization) from a new perspective of data mining.

2. RELATED WORKS

In this section, we review related works based on their targeting instance types.

Small Instances Mining. Methods designed for *small* instances usually work on small scale dataset.

Early studies on Common Pattern Discovery [11, 22, 24] model this as an optimization problem. These methods are computational expensive, thus can only operate on small scale dataset (up to few hundreds images). It is reported in [11] that two hours of running time is required for a dataset with 600 images.

Another approach is Frequent Itemset Mining (FIM), which are initially used to find sets of products bought together by customers (e.g., beer and diaper). Sophisticated algorithms, i.e., APriori [1], Eclat [25], and FP-growth [7], have been developed for this purpose. Quack introduces FIM to visual mining in [17], by treating local image patches as transactions and visual words as items. This method is effective object discovery, but slow in support-counting. In practice, it can only deal with thousands of images.

Sivic [20] extracts key objects and characters from a featured movie by directly clustering grouped local features. Each feature is first grouped with its neighbours. Then frequent objects are extracted as clusters of the grouped features. This is a direct and effective approach for instance mining. However, the scalability remains a big issue, since pairwise similarity evaluation is essentially quadratic. Moreover, only instances with fixed size can be mined, since features must be grouped before clustering.

Large Instances Mining. Methods designed for *large* instances are much scalable.

Philbin [15] constructs a matching graph by searching each image in turn, and then partitions dense sub-graph as clusters. They managed to discover large instances (mostly landmarks). However, small instances could be extremely difficult to mine, since the graph is constructed by full-image search. As for speed, it takes two hours for clustering a 37k dataset, since searching every image against the whole dataset is costly.

Chum [4] mines similar images efficiently by min-Hash [3] with the representation of BoW. Correspondingly, key collisions are extracted as clusters. This method is capable of finding landmarks in large dataset. However, it does not apply to small instance discovery, since Jaccard similarity between images sharing an small instance can be extremely low, and the probability of collision for such images decays quickly as the similarity drops.

Mining Both Small and Large Instances. There only exist a few methods that are both efficient for large dataset and compatible for small instances.

Letessier [10] use Random Maximum Margin Hashing to generate a prior distribution and then adaptively sample and verify frequent objects. A fundamental difference from our method is that [10] mines small instances by clustering local points (versus ToFs in our approach). Our method is potentially more effective and robust, since we uniquely consider the co-occurrence of multiple local points in instances through clustering feature threads.

Pineda [16] extracts instances by clustering visual words, where each visual word is represented as a set of images that have features quantized to it. This method is effective in mining both small and large instances, but can only work on small scale dataset. The performance decays quickly as more images are involved, since more noises are introduced to the representation of visual words.

Geometric min-Hash (GmH) [5] extends min-Hash by considering the dependency among visual words. For a sketch, it computes the first hash key as standard min-Hash does. Secondary hash values are then chosen within a local proximity of the first key. This method improves the collision probability for small instances. However, this performance boosting only exists when the first key repeats, which is still difficult for small instances.

This paper targets for instance-level mining in large dataset (up to million scale). We build our method upon min-Hash for efficient processing, and address the small problem by threading features.

3. INSTANCE MINING

As in Fig. 2, our method consists of two major steps: Feature Threading and Thread Clustering. We start by introducing how to thread features efficiently (Section 3.1), and then show how to mine both small and large instances effectively with these threads (Section 3.2).

3.1 Feature Threading

Our mining algorithm adopts a bottom-up approach, building up visual instances with the elementary components: Thread of Features (ToF). As illustrated in Fig. 2, a ToF corresponds to a set of consistent local features across multiple images. It keeps potential features that link instances, and discards noisy features that are unlikely located on frequent instances. This is motivated by the fact that only a small fraction of features are helpful in instance mining. This phenomena is different from image search, where every feature must be indexed since the query stays unknown.

For mining, ToF serves as an important cue for identifying frequent instances. Whenever there is an instance among a set of images, there must exist several ToFs connecting these images. On the other hand, we can hardly extract any ToFs among random images sharing no common instance. In principle, ToFs should be (1) compact to only link potential features from instances; (2) complete to cover as many instances as possible; (3) efficient to extract and thus be scalable on large dataset. Next, we will discuss our solution to these challenges.

In our work, local SIFT features are quantized to visual words before processing, since comparing raw features is too costly. In image search [19], features quantized to the same visual word are considered to be matched. However, such matching is too noisy. In other words, most of the features quantized to the same visual word are not necessarily consistent enough for threading. Therefore, we also consider

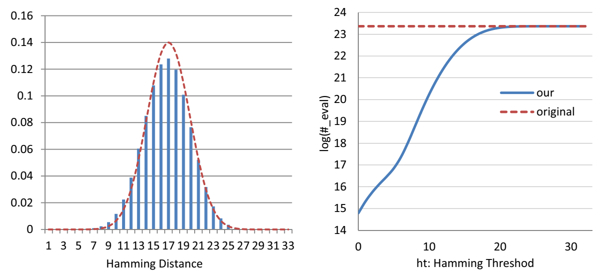


Figure 3: Left: The distribution of Hamming distances among features extracted from a 10k subset of Flickr images. The curve $Bin(32, 0.5)$ is overlaid as red dashed line. Right: the number of evaluations with respect to different Hamming signature threshold, ht. The red dashed line indicates the case without using Hamming code.

the neighbouring points around each feature F for robust matching. In particular, l nearest features are considered to augment the central feature F as a small local patch. These neighbouring features are selected within a small region centered at F, and with the similar scale[1] as F. In our implementation, a local patch is created around each local point by concatenating its $l = 10$ neighboring visual words. The term "central feature" indicates the SIFT feature of the local point.

Threading among these augmented features involves lots of similarity evaluations, i.e., intersection of two sets: counting the number of common visual words between a pair of patches. Two features are threaded if their local patches share at least $t = 2$ common visual words. For a dataset with N images and each with n local features on average, the direct way requires $(Nn)^2$ evaluations. By restricting the threading among features quantized to the visual word [20], the evaluation is reduced to $(\frac{Nn}{w})^2 \times w = (Nn)^2/w$, where w is the size of the visual vocabulary. However, this number is still too large for large dataset.

We noticed that, even for features quantized to the same visual word, most of the similarity evaluations are still unnecessary and lead to no intersection. In this paper, we further reduce the computation by using a short binary Hamming signature [9] for early pruning. A short binary code, extracted from the central feature, is embedded to each central feature as its "signature". This signature is computed by randomly projecting the feature vector to lower dimensional space and thresholding to binary codes. Hamming distance is checked prior to the actual similarity evaluation. Only patches with low Hamming distance are subject to actual similarity evaluation. Note that Hamming distance is much efficient ("XOR") to compute compared to full similarity evaluation. In this way, the set intersection is evaluated only if two patches are with the same central visual word, and their signatures are with a distance less than the Hamming Threshold (ht). Consequently, the total number of evaluations is further reduced to:

$$(\frac{Nn}{w} \times \text{CDF}(ht))^2 \times w = \frac{(Nn)^2}{w} \times \text{CDF}^2(ht), \qquad (1)$$

[1]The scale of local SIFT feature.

where CDF(ht) is the Cumulative Distribution Function of Hamming distance, which could be approximated as a Binomial distribution $Bin(K, 0.5)$ for K-bits Hamming signature [9]. We use $K = 32$-bits Hamming codes throughout this paper. Fig. 3 (left) plots the distribution of Hamming distances for a 10k image subset of Flickr dataset (see Section 4.1 for details). As shown, the probability mass function of $Bin(32, 0.5)$ roughly fits the actual distribution. According to Eq. 1, our method only involves a fraction of $\text{CDF}^2(ht)$ evaluations compared with [20]. For commonly used Hamming threshold $ht = 10{\sim}12$, this fraction is around $(\sum_{i=0}^{10\sim12} \text{PMF}(i))^2 = 0.06\%{\sim}1.16\%$. Fig. 3 (right) plots the actual number of evaluations when mining on this 10k dataset. Compared with [20], the computation is significantly reduced. With the help of Hamming signature, we efficiently extract several ToFs from each visual word, by traversing the inverted file structure.

In summary, one ToF corresponds to a set of reliably matched local patches with (1) same central visual word; (2) small Hamming distances for their signatures; and (3) roughly consistent neighbourhood. Thus, ToF serves as a reliable link connecting images sharing consistent features. In practice, only a small fraction of features are linked as threads, while large portion features are discarded as noises. Compared to traditional BoW, ToFs are much smaller in size and cleaner in quality, which enables efficient and reliable instance mining in large dataset.

3.2 Instances Mining with ToF

After feature threading, each ToF is represented as a set of image IDs. In this section, we show that this neat representation is efficient for processing, while being effective in mining both small and large instances. Since our method is also based on min-Hash for efficient clustering, we first analyze the main problems of min-Hash in instance mining, then show how our strategy avoids these problems.

3.2.1 *Min-Hash Revisited*

Here we only briefly summarize min-Hash, and more details can be found in [3, 6]. Min-Hash (mH) is a randomized algorithm for efficient sets-similarity estimation. It hashes a set to the minimal index according to a random permutation, where the collision probability of two sets equals to their Jaccard similarity. Thus a large number k of min-Hash functions approximates the actual similarity. In practice, multiple (s) hashing keys are grouped together as a s-tuple called sketch. In the scenario of mining, images with at least one sketch collision are treated as similar images. With images represented as BoW vector, the probability [6] that two images I_1 and I_2 having at least one sketch collision is given by:

$$P_C(\text{I}_1, \text{I}_2) = 1 - (1 - sim(\text{I}_1, \text{I}_2)^s)^k. \tag{2}$$

Small Instances: The first problem is on small instances. The probability of collision decays rapidly as the similarity drops, especially when a large sketch s is used. In our experiments, $sim(\text{I}_1, \text{I}_2)$ is around 0.02 (see Section 4.1 for details), thus $P_C \approx 0.3\%$ even with $k = 500$ hash tables. To have a decent probability of observing at least one collision, usually we have to use very large number of hash tables, which requires more computation. As a result, most of the mining results [4, 15] in previous works are about large landmarks.
Random Collisions: Next, we analyze the other problem: random collisions (false positives), which is an important aspect but often ignored by previous methods. That is, besides true patterns, how many random image pairs are extracted. Let us consider a dataset with N random images, each of which has n local features on average, and a visual vocabulary $\mathcal{V} = \{v_1, v_2, ..., v_w\}$ of size w for quantization. Assume local features in this dataset are quantized to each visual word with equal chance. For a pair of random images I_A and I_B, let $X_1, X_2, ..., X_w$ be a list of indicator random variables with

$$X_i = \begin{cases} 1, & \text{if } v_i \in \text{I}_A \ \& \ v_i \in \text{I}_B, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

That is, $X_i = 1$ only if both images have the i-th visual word. Since each X_i is identical and independent to each other, the expected number of common visual words m is given by:

$$m = \mathbf{E}[\sum_{i=1}^{w} X_i] = w \times \mathbf{E}[X_1] = w(1 - (\frac{w-1}{w})^n)^2. \tag{4}$$

Let $x = 1/w$, the term $(\frac{w-1}{w})^n = (1-x)^n$ can be expanded with Taylor expansion near $x = 0$ as $1 - nx + \mathcal{O}(x)$. This linear approximation for $m = n^2/w$ is already accurate enough, since x is approaching 0 for a large vocabulary. Then the Jaccard similarity between a random pair of images can be written as:

$$\epsilon = \frac{|\text{I}_A \cap \text{I}_B|}{|\text{I}_A \cup \text{I}_B|} = \frac{m}{2n - m} \approx \frac{n}{2w - n}. \tag{5}$$

This ϵ is important as it estimates the average similarity for random image pairs, which can be used to threshold large number of false positives. For each $\binom{N}{2}$ pairs of images, the number of image pairs found in k hash tables follows the binomial $Bin(k, \epsilon^s)$. As a result, the expected number of random image pairs mined from the whole dataset is:

$$RC = \binom{N}{2} \times k \times \epsilon^s. \tag{6}$$

Take Oxford105k used in [4] for example, where $N = 104,844$, $n = 2,805$, $w = 2^{17}$, $k = 512$, and $s = 3$. According to Eq. 6, $RC = 3.34 \times 10^6$, which roughly matches the number reported in [4]: $38.4 \times N = 4.02 \times 10^6$, where both *random* and *true* collisions are counted. That is to say, besides true image pairs, more than 3 million random pairs are expected to be extracted from a 100k dataset. Note RC grows quadratically in N, which becomes a big problem for large scale mining. These random pairs need to be verified with expensive post-operations, such as spatial verification and image retrieval.

3.2.2 *ToF Mining*

After Feature Threading, the rest part of our method can be briefly summarized as follows. ToFs are first clustered using min-Hash, where clusters are extracted as key collisions in hash tables. Then instance clusters are discovered with a simple voting.
Thread Clustering. Our method solves the Small-Instance problem by clustering ToFs instead of images. A ToF may only give a single "link" over a set of images Ω, while multiple ToFs would indicate a strong evidence of an instance over Ω. Although min-Hash is not good for mining images sharing a small instance, we show that it is quite suitable for mining correlated ToFs.

Suppose a set of images that share a small instance, where $m \ll n$ and ϵ goes to 0. Thus hashing images can hardly find the instance. In this work, we turn to exploit correlated ToFs (Co-ToF), instead of clustering images. Correlated ToFs are the set of threads that consistently show up (or disappear) together on images. Clusters of correlated ToFs lead to instance hypotheses. Intuitively, it suggests that these images are sharing several consistent local regions. This strategy is especially good for mining small instances. Luckily, ToFs consisting the pattern can still show high similarities, though the images are with low ϵ between each other. Note that the probability of ToFs collision is totally independent on the instance scale, which makes it suitable for instance mining. The only factor that matters is the similarity among ToFs composing an instance, which is usually high after feature threading.

Note that our strategy also addresses the Random Collision problem. With compact and clean ToF, similarities among ToFs composing an instance are much higher than similarities among images sharing the instance. The reason is that Feature Threading has discarded large portion of noisy features. According to Eq. 2, to ensure a high probability of collision $P_C > \alpha$, we need at least $k > \log_{1-sim^s}(1-\alpha)$ hash tables. That is, much fewer k hash tables are needed with a slightly higher sim, since $\log_{1-sim^s}(1-\alpha)$ is steep when $sim \approx 0.02$ and $\alpha \approx 0.8$. This property is very useful, since large computation can be saved with fewer hash tables.

Cluster Voting. After Threads Clustering, we adopt a voting step to get instance clusters from threads clusters. For each cluster of ToFs, all images that are linked by these threads are considered as potential instance holders. To further reduce noises, only the candidate images linked by most (80%) ToFs are voted as instance clusters.

Our method helps small instances mining. There may only a few threads extracted for small instances. But the probability of collision (these threads are hashed to the same key) only depends on the similarity of these threads. It has nothing to do with the number of threads. Small objects would just lead to a small cluster with only a few threads. In summary, the main advantages of our mining strategy can be summarized as follows: (1) fewer hashing tables (k) are needed, since the noises are significantly reduced with ToFs; (2) we have better probability to extract small instances.

4. QUANTITATIVE EVALUATION

We compare our method (Co-ToF) with three baselines: (1) mH: image discovery via min-Hash [4]; (2) Co-VW: object mining via co-occurring visual words [16]; (3) GmH: Geometric min-Hash for objects discovery [5]. The first one (mH) is included in our evaluation as a baseline using standard min-Hash. The other two methods are selected as representatives for scalable instance-level mining methods, as discussed in Section 2.

4.1 Datasets

MQA: A total number of 438 images are crawled from Flickr and Google Image, by querying 52 instances names (e.g., Wall Street Bull). This dataset[2] is originally used for visual instance naming [27], such that a wide range of real-life instances are covered, including fashion, vehicle, flower, pet, food, product, logo, landmark and art. Each visual instance has 5∼15 (8.5 on average) image examples with different background (i.e., small duplicate proportion). We use this dataset to test the performance on small instances.

[2]http://vireo.cs.cityu.edu.hk/mqa/

Table 1: Summary for the dataset used in our experiment.

Dataset	# image	# cluster	cluster size	ϵ for GT
MQA	438	52	8.5 ± 2.2	0.0185
Oxford5k	5,062	364	6.5 ± 21.6	0.0251
Oxford100k	99,782	-	-	-
Flickr	1,000,000	-	-	-

Oxford5k: This dataset[3] has 5,062 Flickr images by crawling landmarks in Oxford. As for the ground-truth, the 11 landmark buildings are labeled manually. Different with image search, our evaluation takes all images with labels {"Good", "OK", "Junk"} as true results, since we are evaluating instance level mining. However, we still found this dataset heavily under-annotated: besides 11 labeled landmarks, there still exist large number of unlabeled instance clusters. Such annotation is reasonable for image-search evaluation, but is inadequate for instance mining. In this case, we re-annotate this dataset by (1) searching each image in turn with traditional image search system, and (2) pooling the top results for manual annotation. With our fine annotation[4], it ended up with 364 clusters (including the original 11 landmarks) involving 2,369 images in total.

Oxford100k: This dataset [13] has 99,782 Flickr images by crawling popular tags. Note images in this dataset are not annotated, and are treated as negative images in our quantitative evaluation, although there might be some frequent instances inevitably.

Flickr: This dataset has one million images downloaded from Flickr by crawling "recent uploaded photos". No restrictions on tags, users, or locations are enforced at the time of dataset construction. Though we expect this dataset to be random, it still has several patterns inside, since there might be some popular objects/landmarks shared by different users and consistent images uploaded in same location. Note that we do not have the ground-truth for this dataset, and it serves as a "blind" dataset for scalability test.

General Settings. A hierarchical vocabulary with one million leaf nodes is constructed [12]. With this vocabulary, each image is representated as a set of visual words. As shown in Table 1, the average Jaccard similarity (ϵ) of ground-truth (GT) images for MQA (Oxford5k) is 0.0185 (0.0251), while ϵ for random image pairs is around 0.0013. For hashing, k sketches of size 3 ($s = 3$) are used for all methods.

4.2 Evaluation Metric

In this paper, we adopt F-measure as the evaluation metric, which is frequently used in clustering. Instance mining is essentially a clustering problem, where both false positive and false negative decisions should be penalized. Specifically, *recall* is computed as the number of detected true image pairs divided by total number of true pairs, and *precision* is calculated as the percentage of true image pairs among all

[3]http://www.robots.ox.ac.uk/∼vgg/data/oxbuildings/

[4]This annotation has already been made available at: http://vireo.cs.cityu.edu.hk/gt_clusters.oxford5k

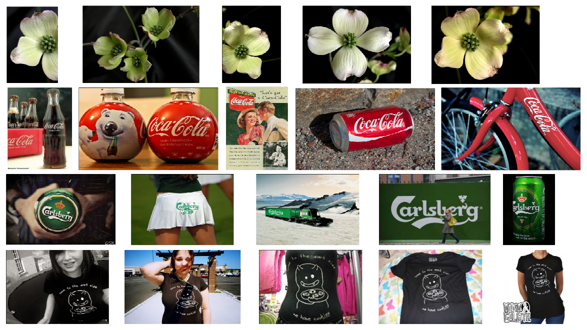


Figure 5: Example instances mined from MQA dataset. Each row corresponds to a sample instance cluster.

the detected pairs. Then F-measure is the harmonic mean of precision and recall.

4.3 Mining MQA dataset

Fig. 4 compares different methods on MQA, which is a small dataset with small instances. In terms of F-measure, all methods, except mH, perform strongly on this small scale dataset. As discussed in Section 3.2, mH is not suitable for mining small instances, where similarities between true image pairs are low. In such case, mH can hardly distinguish true images out of random pairs, resulting in poor precision and recall. On the other hand, other methods are much compatible to small instances. Our method Co-ToF converges quickly and leads the performance at different k. It only needs very few tables to produce a decent result. Co-VW also performs well on this dataset, and reaches its peak around $k = 20$. For this small dataset (438 images) indexed with a large vocabulary (1M), the sets of visual words for different instance patterns are unlikely to overlap. Thus there are not too much noises in the posting list of each visual words. This situation is ideal for Co-VW, but does not hold, as more and more images (e.g., 100k$\sim$1M) are added to the dataset, e.g., Oxford dataset in Section 4.4. GmH also performs well on this dataset by choosing secondary hashing keys from the proximity of the first key. This strategy prevents random collisions effectively, which significantly improves precision over mH. However, it is still difficult for the first key to collide for small instances, since the first key is generated randomly as mH does. The chance for these randomly selected features to collide is low, and it is even lower when the images share a small instance (as in MQA). Therefore, low recall is still a problem for GmH.

Fig. 5 shows several example instances mined from MQA dataset. Based on our observation, (1) instances with high-Freq and large-DuP are easy to mine, e.g., buildings, scenes; (2) our method is effective in mining small instances, e.g., Coca-Cola, Wii-mote; (3) some instances, e.g., animals, flowers, are still difficult to mine, since features extracted on these smooth, non-rigid instances are not stable, which corrupts feature threading.

4.4 Mining Oxford Dataset

Fig. 6 compares different methods on Oxford dataset. We first evaluate the performance on the fully annotated Oxford5k (top), and then report the performance on the complete Oxford105k (bottom). Note that this 105k dataset is a combination of Oxford5k and Oxford100k, where Oxford100k is added as distracting images.

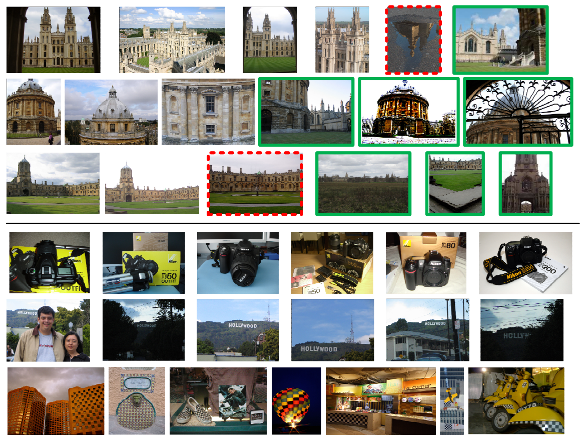


Figure 7: Example instances mined from Oxford105k dataset, using our method. Top 3 rows: example instances in ground-truth set. Images with green-solid border are in the "junk" set of Oxford dataset, while red-dashed border indicates false positives. Bottom 3 rows: example instances outside ground-truth set. The last row is a false cluster.

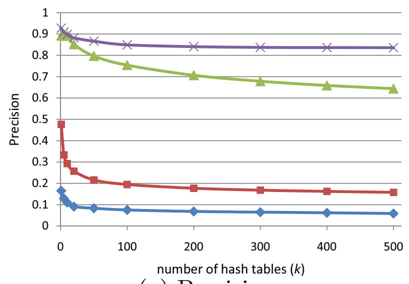
The performance on Oxford dataset is roughly consistent as MQA. The main differences between Oxford and MQA are two folds: larger instances and more distracting images.

First, the average Jaccard similarity between true image pairs is higher on Oxford dataset[5]. In this case, mH suits better for this dataset by effectively discovering large instances. However, it is still not comparable with GmH and Co-ToF, which work for both small and large instances. Second, more and more distracting images, which do not belong to any cluster, are included in our evaluation. Generally, more distracting images bring more irrelevant/noisy features, which makes instance mining even harder. For Co-VW, more noisy features are attached to the posting list of each visual word, which brings down the similarity of visual words composing an instance. As a result, the performance of Co-VW degenerates quickly as dataset grows. For mH, the number of random collisions grows quadratically in the dataset size (Eq. 6). Thus it gives lower precision on larger dataset. For GmH, more random collisions are observed as more distracting images are added, but it is less sensitive due to the way of choosing secondary hash keys. As expected, our method performs consistently better than other methods, and the performance gap is even larger as the dataset grows. We noticed in our experiments that the size[6] of threads does not grow as fast as dataset size, since our method only threads potential features for mining.

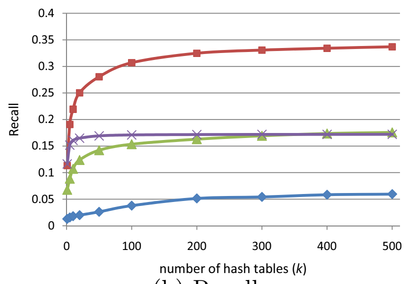
Performance on Small Instances. As shown, Oxford dataset is with both small and large instances. Since our method is tailored for small instances, we are also interested in the performance on particular small instances. To separate small instances from large ones, we first calculate the average Jaccard similarity for pairwise images in each ground-truth cluster, and then cut the whole ground-truth

[5]From 0.0185 to 0.0251 is adding 35% more common local features.

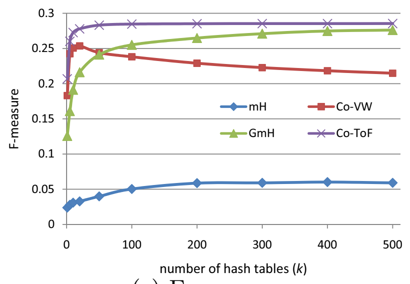
[6]Number of threaded features.



(a) Precision

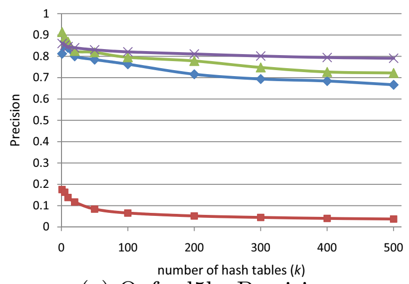


(b) Recall

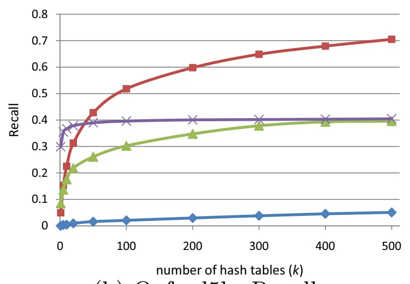


(c) F-measure

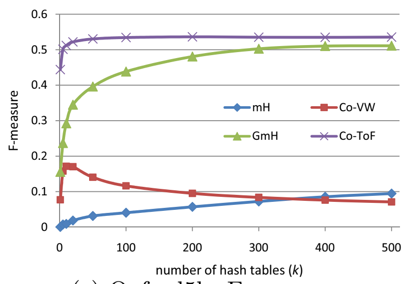
Figure 4: Performance comparison on MQA dataset.



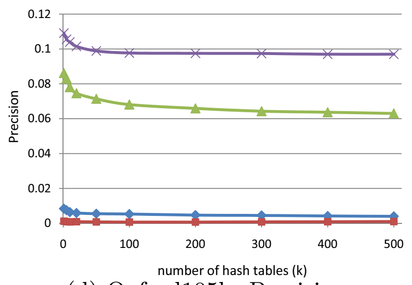
(a) Oxford5k: Precision



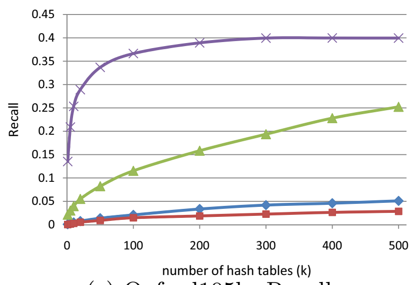
(b) Oxford5k: Recall



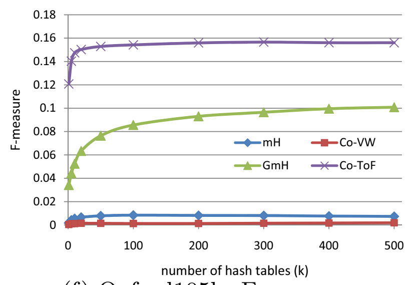
(c) Oxford5k: F-measure



(d) Oxford105k: Precision



(e) Oxford105k: Recall



(f) Oxford105k: F-measure

Figure 6: Performance comparison on Oxford dataset. Top: Oxford5k, Bottom: Oxford105k.

Table 2: The performance (F-measure) on the *small* subset of instances in Oxford dataset. A mid-level value of $k = 300$ ($1 \sim 500$ evaluated in Fig. 6) is adopted for all methods.

	mH	Co-VW	GmH	Co-ToF
Oxford5k	0.0006	0.1370	0.1670	0.1836
Oxford105k	0.0006	0.0005	0.0347	0.0994

instances into halves according to this similarity. Instance clusters with lower half similarities are regarded as *small*. Table 2 presents the detailed performance on this *small* subset. On Oxford5k, Co-ToF, GmH, and Co-VW show significant advantage over mH, which does not work on small instances. Co-VW works on Oxford5k, but the performance decays badly with more distracting images added. Both GmH and Co-ToF work for small instances, and give decent performance on this subset. GmH extracts small instances effectively by improving the collision probability when the first hash key repeats. Co-ToF outperforms GmH by removing noisy background features, such that both first and secondary hash keys are with high chance to collide.

Table 3: Precision of the mining results for Flickr dataset, estimated on 1000 random image pairs.

	mH	Co-VW	GmH	Co-ToF
Precision	0.233	0.011	0.363	0.423

Fig. 7 shows several examples mined on Oxford105k. In addition to the patterns in the ground-truth (top 3 rows), we also show some patterns mined on Oxford100k (bottom 3 rows). As observed, some of the patterns labeled as “junk” (less than 25% region is visible) in the original paper [13] are also discovered in our method, demonstrating the ability for small instance discovery.

Failure cases. Some instances, e.g., animals, statues, are missed by our method, since local features are not always repeatable across these images and the ToFs cannot be established. Some false-alarm instances (e.g., last row of Fig. 7) are mined, because some locally consistent local patches could appear frequently in irrelevant images. However, this problem can be alleviated by post-operations as spatial verification.

4.5 Mining Flickr Dataset

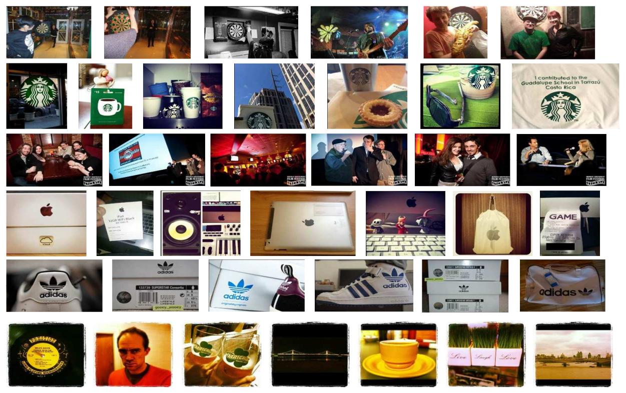


Figure 8: Example instances mined from Flickr dataset. Due to space limitation, this figure only shows example instances with 5∼10 ToFs. Instances from top to bottom: the dart board, Starbucks, logo in the corner, Apple logo, Adidas logo, a border style.

Since it is impossible for us to annotate such a large dataset, Flickr dataset is mainly used for scalability test in this paper. We do not evaluate recall on this dataset, and the precision is roughly estimated by sampling 1000 random image pairs from the mining results. As shown in Table. 3, Co-VW gives most noisy result, due to the large scale of this dataset. Min-Hash is good at finding highly similar images, but it also returns large number of false positives. GmH gives better precision by carefully selecting secondary hash keys. Co-ToF achieves the best precision by removing noises through feature threading. Note that the precision alone may not be able to tell the full story on performance, but it gives a gist on noise levels of different method.

The mined instances on this dataset are dominated by near duplicates probably uploaded by the same user, which are less-interesting. Thus in Fig. 8, we only show example instances with small-DuP and high-Freq. As shown, some of the small patterns corresponds to popular objects (the dart board) or product logos (Apple, and Adidas), since they are heavily shared by a wide range of users. The mining results could indicate the instance popularity. Since these objects only occupy a small area on the images with different background, most of them are missed in other mining methods. There is also a small logo found as in the 3rd row of Fig. 8, which is small and embedded in a corner of the images. Perhaps these photos are uploaded by a TV station or an organization. The last row gives a serendipity. These set of photos seem to be irrelevant, if judged by human. In fact, by careful investigation, we notice that the instance is the common style along the borders. A possible explanation is probably a photo editing tool (e.g., Instagram), with this special border-style, is getting popular among Flickr users.

4.6 Discussion

Performance on small instances. In our evaluation, we have separated and focused on the *small* instances in the Oxford dataset in Table 2. For MQA, the majority of the dataset is of small sizes (confirmed by the small ϵ in Table 1). Hence, results in Fig 4 have been meant to evaluate the performance focusing on small instances. The Flickr dataset is diverse and too large to manually set aside only small instances for performance comparison. However, Fig 8 only shows instances with small DuP and high Freq.

Parameter sensitivity. The parameter settings $(K; s)$ follow the recommendation of previous works [4, 6, 9, 20] with similar scenarios. Figures 4 and 6 show the performances with different values of k. The choice of parameters $(l; t)$ are selected based on a separate validation experiment ($l = 5\sim20, t = 1\sim5$), considering the tradeoff between efficiency and accuracy.

Open Issues. Since our method relies on extraction of local feature and their threading, its performance may degrade when dealing with large media quality variations introduced by significant viewpoint change, domain changes and high noises.

Table 4: The running time for each method on different datasets. For Co-ToF, the total time (C) is decomposed to Feature Threading (A) and Hashing (B) as: A + B = C.

	MQA	Oxford5k	Oxford105k	Flickr
mH	2.1 s	27.0 s	9.5 m	1.7 h
Co-VW	8.7 s	68.2 s	8.0 m	1.7 h
GmH	2.2 s	27.3 s	9.6 m	1.8 h
Co-ToF	1.0+2.8 =3.8 s	13.5+4.6 =18.1 s	6.3+1.3 =7.6 m	1.4+0.4 =1.8 h

s: second, m: minute, h: hour

4.7 Speed Efficiency

Table 4 gives a summary for the running time of different methods. These experiments were all conducted on a 8-core machine with 2.67GHz CPU, 16GB memory. The processing time excludes feature extraction and quantization, which are the same for all methods. As shown in the last row of Table 4, the total running time of our method (Co-ToF) consists of two parts: Feature Threading (Section 3.1) and Threads Hashing (Section 3.2), while other methods only have a hashing step. In general, all methods take roughly similar time on large dataset. Although Co-ToF needs an extra step for Feature Threading, the Hashing step is much efficient compared to other methods. Essentially, the hashing time is mainly affected by two factors: the total time is linear in the number of sets; computing the hash key for a set is linear in the size of the set. After feature threading, both factors are reduced significantly due to the compactness of ToF. Other methods, mH, Co-VW and GmH, need much longer time for hashing, since they take the full set of features during hashing. Note in this table, all methods adopt the same number ($k = 100$) of hash tables for the purpose of comparison. However, our method usually requires much fewer hash tables in practice, which reduces the running time significantly in real applications. As observed, the running time for our method is mainly dominated by Feature Threading. However, the extra time spent is compensated by efficient hashing step and much better results.

5. APPLICATIONS

In this section, we demonstrate two applications from the new perspective of visual instance mining. To suppress similar consective frames within a video shot, local points from the same video are not linked at the time of feature threading, which can be done conveniently and efficiently by incorporating video ID to each feature. Therefore, only inter-video threads are kept.

5.1 Multimedia Summarization

We first present a multimedia summarization application with two recent events "E1: Boston Bombing" and "E2: Malaysia Airlines Flight 370". To demonstrate, the datasets are constructed by crawling Youtube videos with keyword search "Boston Bomb" and "Malaysia Airlines Flight 370". Returned videos are then filtered with specific upload-time constraint[7]. We construct the dataset this way such that (1) there exist some repeated instances for mining; and (2) people roughly know what kind of instances should be expected, since we do not have ground-truth for these videos. Finally, we ended up with 355 videos for E1 ($\sim$92k frames) and 406 videos for E2 ($\sim$103k frames), which include official TV news as well as amateur reviews.

We apply our instance mining to summarize the video collection as shown in Fig. 9 (Left: Boston Bombing, Right: Malaysia Airlines Flight 370). The tag-cloud (top) is generated by extracting frequent terms (verbs and nouns) by parsing video metadata (title, description). Visual patterns (bottom) are extracted with the proposed method. We further select several "small (first two rows in Fig. 9 bottom), medium (3rd row), and large (last row)" size instances for visualization. As observed, most visual instances are meaningful: small instances are mostly faces of key characters, logos of TV stations; medium-size ones correspond to iconic sub-images cited in multiple videos; large instances are mostly cover photos for each event.

Compared with traditional summarization by image-level clustering, whose results are very similar as the last row in Fig. 9, our instance level summarization is much precise and complete in identifying salient items and scenes. Compared to texts summarization, which only conveys general ideas, our result gives much concrete and intuitive visual entities on what (and probably who/where) is about the data collection. One drawback of visual instance summarization would be the lack of semantic meaning for visual entities. Thus in this application, we adopt both results to give complementary summarization.

5.2 Instance Search

Instance Search (INS) [21] is a realistic problem proposed by TRECVID, which is to retrieve all occurrences of the query instance from a video collection. Compared to traditional image search, the query instance is usually specified with a small Region Of Interest on the query image, and the target instances on reference images could also be small and with different background. Apparently, this phenomenon poses small-DuP problem. However, most state-of-the-art methods [28, 26] ignore this problem and still rank images based on traditional similarity as: $sim(Q, R_i) = \frac{Q \cdot R_i}{\|Q\| \|R_i\|}$, where Q is the BoW vector extracted within the query ROI, while R_i covers full reference image. Apparently, $\|R_i\|$ *over-norms* the score for small-DuP reference images by including dense features on background, which gives lower ranking accordingly.

We apply instance mining to solve this "over-norm" problem. Our solution is straight-forward: (1) we first apply instance mining on the reference dataset to get frequent instance set $\mathcal{P}$; (2) query instance Q is then matched against $\mathcal{P}$, which is indexed with an inverted file; (3) images sharing matched instances are boosted in the initial retrieval rank-list. Though simple, this strategy solves the "over-norm" problem by leveraging small instances extracted through instance mining, which does not suffer from the small-DuP problem. We test this strategy on TRECVID'13 Instance Search task, which includes 30 query instances defined by TRECVID, and 470k reference video clips cut from 464 hours of TV series.

After mining, it ended up with 22k instance clusters ($\mathcal{P}$) that are verified with spatial consistency. Some of these instances are presented in Fig. 10 (top). As shown, there are both large instances ("the curtain") and small ones ("the Parking sign", "this pendent"). These inter-links across clips, especially on small instances, could be helpful to address the "over-norm" problem.

By matching each Q to $\mathcal{P}$, we can get a list of relevant instances. However, we found in our experiments that not every Q can be matched against $\mathcal{P}$. In other words, our mining results only partially overlap with the 30 queries defined by TRECVID. Therefore, we only re-rank the queries which are closely matched with $\mathcal{P}$ by thresholding the matching score. After this, 10 out of 30 queries are considered as "overlap" and are further re-ranked with the mining results. As shown in Fig. 10 (bottom), most of these queries are improved by our re-ranking. The Average Precision is boosted in the range of 3.3% to 11.0%. Such examples of queries include "no smoking logo", "bust of queens" and "Parking sign". These queries originally suffer from the "over-norm" issue badly because of their small area on the reference images. To the best of our knowledge, this is the first attempt to address the search problem with data mining. This application may shed some light on new directions for instance search.

6. CONCLUSIONS

We have presented a technique and two applications on visual instance mining for multimedia collections. Our method is effective for both small and large instances, while being efficient on million scale dataset. For efficiency, ToF is proposed as a clean and robust representation, and min-Hash is adopted for fast clustering. For effectiveness, correlated ToFs are exploited for generating instance hypotheses. The quantitative evaluation shows that our method is more effective than other methods in mining instances (especially small ones) while being scalable. Two applications on instance search and multimedia summarization demonstrate the potential of our method in solving multimedia problems.

7. ACKNOWLEDGMENTS

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 118812).

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB*, pages 487–499, 1994.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

[7]E1: 04/15/2013$\sim$06/15/2013; E2: 03/08/2014$\sim$03/25/2014.

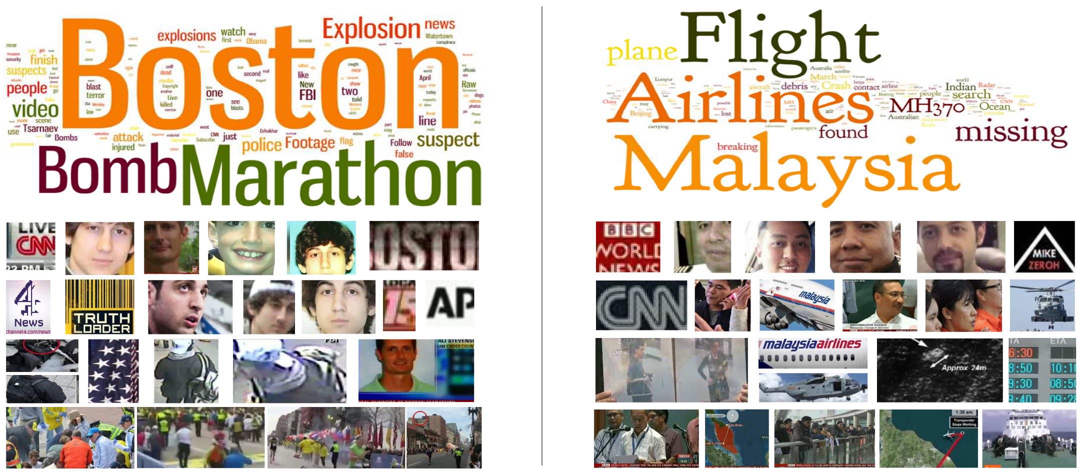


Figure 9: Youtube videos summarization for the events "Boston Bomb" (left) and "Malaysia Airlines MH370" (right), through text mining (top) and visual instance mining (bottom).

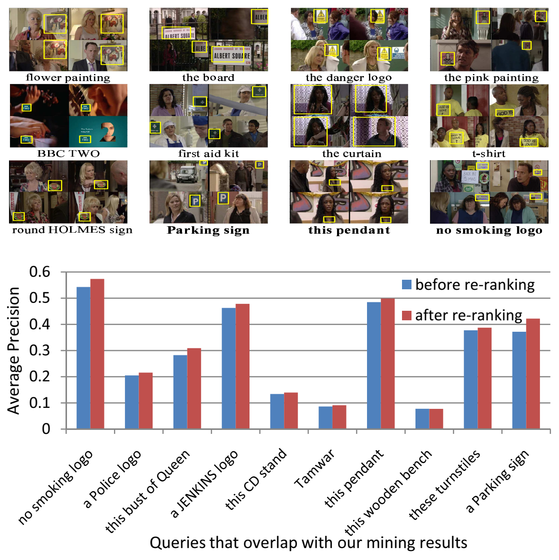


Figure 10: Top: example instances mined from TRECVID'13 Instance Search dataset. Texts below each instance cluster are labeled manually. Instances that overlap with TRECVID's 30 queries are bolded. Bottom: the performances before and after re-ranking with instance mining, for queries that are also our mined instances.

[3] A. Broder. On the resemblance and containment of documents. In *SEQUENCES*, 1997.

[4] O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE Trans. PAMI*, 32:371–377, 2010.

[5] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. *Proc. CVPR*, pages 17–24, 2009.

[6] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proc. CIVR*, 2007.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. SIGMOD*, pages 1–12, 2000.

[8] T. Hofmann. Probabilistic latent semantic indexing. *Proc. SIGIR*, pages 50–57, 1999.

[9] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87(3):192–212, May 2010.

[10] P. Letessier, O. Buisson, and A. Joly. Scalable mining of small visual objects. In *ACM Multimedia*, 2012.

[11] H. Liu and S. Yan. Common visual pattern discovery via spatially coherent correspondences. In *Proc. CVPR*, pages 1609–1616, 2010.

[12] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, pages 2161–2168, 2006.

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

[14] J. Philbin, J. Sivic, and A. Zisserman. Geometric LDA: A generative model for particular object discovery. In *Proc. BMVC*, 2008.

[15] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *Proc. ICVGIP*, pages 738–745, 2008.

[16] G. F. Pineda, H. Koga, and T. Watanabe. Scalable object discovery: A hash-based approach to clustering co-occurring visual words. *IEICE Trans. on Information and Systems*, pages 2024–2035, 2011.

[17] T. Quack, V. Ferrari, and L. V. Gool. Video mining with frequent itemset configurations. In *Proc. CIVR*, pages 360–369, 2006.

[18] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proc. CVPR*, pages 1605–1614, 2006.

[19] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003.

[20] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. In *Proc. CVPR*, 2004.

[21] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *Proc. MIR*, 2006.

[22] H.-K. Tan and C.-W. Ngo. Localized matching using earth mover's distance towards discovery of common patterns from small image samples. *Image Vision Computing*, 27(10):1470–1483, 2009.

[23] G. Xin, L. Dong, J. Brendan, Z. Mojun, C. Anni, and S.-F. Chang. Robust object co-detection. In *Proc. CVPR*, 2013.

[24] J. Yuan and Y. Wu. Spatial random partition for common visual pattern discovery. In *Proc. ICCV*, 2007.

[25] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. on KDE*, pages 372–390, 2000.

[26] W. Zhang and C.-W. Ngo. Searching visual instances with topology checking and context modeling. In *Proc. ICMR*, 2012.

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

[28] C. Zhu and S. Satoh. Large vocabulary quantization for searching instances from videos. In *Proc. ICMR*, 2012.