MatchMiner: Efficient Spanning Structure Mining in Large Image Collections

Yin Lou Noah Snavely Johannes Gehrke

Department of Computer Science, Cornell University
{yinlou, snavely, johannes}@cs.cornell.edu

Abstract. Many new computer vision applications are utilizing large-scale datasets of places derived from the many billions of photos on the Web. Such applications often require knowledge of the visual connectivity structure of these image collections—describing which images overlap or are otherwise related—and an important step in understanding this structure is to identify *connected components* of this underlying image graph. As the structure of this graph is often initially unknown, this problem can be posed as one of exploring the connectivity between images as quickly as possible, by intelligently selecting a subset of image pairs for feature matching and geometric verification, without having to test all $O(n^2)$ possible pairs. We propose a novel, scalable algorithm called MatchMiner that efficiently explores visual relations between images, incorporating ideas from relevance feedback to improve decision making over time, as well as a simple yet effective *rank distance* measure for detecting outlier images. Using these ideas, our algorithm automatically prioritizes image pairs that can potentially connect or contribute to large connected components, using an information-theoretic algorithm to decide which image pairs to test next. Our experimental results show that MatchMiner can efficiently find connected components in large image collections, significantly outperforming state-of-the-art image matching methods.

1 Introduction

The last decade has witnessed explosive growth in the availability of image data. Thousands of photos are added every minute to online repositories, such as Flickr and Facebook, with views covering large parts of the Earth. A number of new geometric computer vision applications have been built on top of such large-scale photo collections of places [4, 5]. A key requirement of these systems is to identify the connectivity of an image collection in the form of an *image graph* where each image is a node, and where an edge connects each pair of images that visually overlap. For unstructured image collections, the structure of this graph is initially unknown—i.e., we do not know which pairs of images match, and accordingly what edges exist—and thus needs to be discovered, usually using the tools of feature matching and RANSAC-based geometric verification [7] to test the existence of edges (i.e., overlapping images). However, this matching and verification process is relatively expensive, so it is desirable to obtain an image graph that is as complete as possible while rigorously matching and verifying a minimal number of edges.

In addition, given an image collection, it is often unnecessary to discover a complete description of the underlying image graph, as a much sparser graph contains sufficient

Fig. 1. A visual path from Il Vittoriano to the inside of the Colosseum in our Forum dataset. Each consecutive image pair exhibits spatial overlap. In an image graph, links between densely-connected subgraphs are important for applications such as 3D reconstruction and visualization as they contribute to a more complete scene and a more accurate 3D model.

information for many applications. For example, if the graph underlying a set of n images is complete (i.e., each image overlaps every other image), then a star graph (one with $n-1$ edges, each connecting one node to a central node) might be a sufficient description of the graph for certain applications, as compared to exhaustively matching all n choose 2 pairs of images. In the context of structure from motion (SfM), a sparse graph is sometimes even more desirable than a complete description; for instance, Agarwal et al. take an image graph and apply a sparsification technique to derive a simpler "skeletal" graph for efficiently reconstruction [1, 8]. On the other hand, breaking the graph into two separate connected components (CCs) is undesirable, as it can lead to separate, disconnected models for SfM methods. As a motivating example, Figure 1 shows a "visual path" through a set of images of Rome (out of a collection of nearly 75K images) connecting two landmarks, the Il Vittoriano Monument and the Colosseum. While both of these landmarks are densely photographed, photos linking the two are much more difficult to find, hence finding such connections is critical to connecting these two monuments in a 3D model. This motivates our goal of discovering the large CCs of an image collection as completely, and as efficiently, as possible.

We do this through a method that proposes edges to verify through feature matching and spatial verification. If each edge verification step on an image pair successfully finds that the pair indeed matches, then problem would be much easier—we would need to test at most $n-1$ edges to find a spanning forest for the image collection. In practice, however, it is difficult to know in advance which pairs to test, as many pairs of images do not match. Many state-of-the-art image matching systems [4, 8, 9] use techniques based on image retrieval techniques (such as bag-of-words and inverted files [10]) to determine likely matches, but these metrics are not always good predictors of whether two images match. While several techniques have been proposed to improve retrieval performance [11, 12] based on the idea of refining the query and retrieve images repeatedly, many of them introduce overhead at the retrieval stage since the similarities have to be recomputed, which substantially increases the total running time.

Another source of difficulty comes from the fact that large-scale image datasets often contain many "singleton" images, i.e., images that match no or very few other images, such as a closeup of a pigeon, and verifications on such images are wasted. Even worse, some near-singleton images contain confusing features that result in them being

similar to other images under bag-of-words methods. Detecting such "bad" singleton images is expensive because in the worst case we have to verify a query image against all other images in the dataset to conclude whether the query image is indeed a singleton. Currently, most large-scale matching systems do not explicitly model such outlier images, and hence waste computation time trying to match them.

Contributions. To address these problems, we develop an efficient and effective algorithm called **MatchMiner** that applies to large-scale collections of images of scenes to discover large CCs. MatchMiner works by intelligently maintaining a shortlist of image pairs to match, and re-ranking this shortlist over time based on feedback from successful and unsuccessful prior matches. We show that a novel algorithm based on relevance feedback can be employed in MatchMiner to effectively re-rank image pairs while introducing little overhead in the image retrieval process. This algorithm significantly improves the success rate of finding true matching pairs. To prevent singleton images from appearing in the shortlists of other images, we also propose a simple yet effective measure called *rank distance* to prune out false positives and to increase the probability of success. Finally, we use an information-theoretic model of the problem to choose edges that minimize a objective based on expected entropy given estimated prior probabilities of matches based on visual similarities.

We demonstrate the effectiveness of MatchMiner on several image collections with ground truth obtained by exhaustive matching, as well as larger collections with tens or hundreds of thousands of images. Our experiments show that MatchMiner can effectively identify large CCs in an image graph with a relatively small number of matching operations, and that for large problems it can produce significantly better results than current techniques, such as Image Webs [9], given the same budget of matches performed.

2 Related Work

Several techniques have been proposed to match large-scale image collections. Probably the most related method is Image Webs [9], whose matching system has two phases. The first phase proposes image pairs based on visual similarity using a standard bag-of-visual-words model. Potential image pairs that straddle different connected components (CCs) are given priority for verification. This phase results in a (hopefully large) set of CCs. In the second phase, spectral graph theory is employed to increase the connectivity within each CC. Our work differs in that we are mainly interested in finding as complete a set of CCs as possible, as this is critical to many applications including 3D reconstruction. To the best of our knowledge, our work is the first to incorporate practical relevance feedback methods in the domain of large-scale image matching.

Chum and Matas formulated the problem as one of clustering and presented a method based on min-wise hashing to quickly generate cluster seeds (image pairs) whose similarity is above a user-defined threshold [13]. These seeds are then spatially verified to filter out false positives. To increase recall, they then grow these seeds using query expansion [12]. One problem with their approach is that the quality of their final graph structure highly depends on those initial seeds, and their method can easily miss important connections between components of the image set.

Structure within CCs is also important in many problems. Techniques such as skeletal graphs [1] and dominating sets [3] attempt to sparsify the graph while retaining important structures. However, these techniques often require a (near) complete set of edges within CCs given in advance, while our formulation of the problem aims at discovering those CCs and therefore can be viewed as an initial step in such systems. Other work takes a probabilistic approach to identifying important structures in a graph [2].

Large-scale matching is a key component in many city-scale 3D reconstruction systems [4, 8]. For instance, Frahm *et al.* build a reconstruction on a single PC using GPUs [4], leveraging GIST features and compact binary codes in a clustering algorithm for very fast image matching. Iconic images in these clusters are then chosen and connected between clusters. However, they sacrifice graph quality for efficiency, and thus they may lose important connections between landmarks, breaking a scene up into several separate connected components; our goal is to find not just "easy" clusters, but to connect even weakly connected subsets of images, such as those shown in Figure 1.

Our technique is also related to recent image retrieval techniques based on bag-of-visual-words models [11, 12]. As in our work, some of these methods modify the image query to either reduce the effect of confusing visual words or enhance the power of relevant words. *Recursive average query expansion* [12] constructs a new query by averaging verified results of the original query, i.e., a new query vector is formed using positive feedback. *Incremental spatial re-ranking* [11] automatically detects tf-idf failure (too few features landing in the same visual words are geometrically consistent), incrementally builds a statistical model of the query object, and learns relevant spatial context to boost retrieval performance. Despite their effectiveness, most of these methods introduce significant overhead in the retrieval stage of large-scale image matching since the similarities between the query and database images must be recomputed using the updated query. This is especially problematic in our setting, where *all* images are potential queries. We address this with a query modification technique that is extremely efficient and can take into account negative information.

3 MatchMiner Algorithm

As mentioned in the introduction, we formulate the process of finding structure in a large image collection as that of finding all of the large connected components (CCs) of the underlying image graph. While this is not the only possible way to formulate the matching problem, it fits with our goal of connecting up the images as best as possible, so that later processing stages (e.g., of an SfM pipeline) can obtain as complete models as possible. To that end, imagine that by some process we can obtain the "ground truth" image graph (e.g., by exhaustively matching all pairs of images),[1] and hence the ground truth CCs. Our task is then to quickly discover these ground truth components (without prior knowledge), by testing as few image pairs as possible. While we may only be able to find an approximation of the ground truth CCs in a reasonable time, e.g., due to weak edges that are difficult to discover, we want the approximate set of CCs we find to be as complete as possible. More formally,

[1] Note that this $O(n^2)$ process is intractable for very large image collections, which is why we require approximate methods.

Definition 1 (Spanning Structure Mining). *Given an image set $\mathcal{I} = \{I_1, ..., I_n\}$ and a geometric verification predicate $GV(I_i, I_j)$, the ground truth image graph G_g is formed by creating a node for each image and by creating an edge between images I_i and I_j if $GV(I_i, I_j)$ holds for $i \neq j$. We want to efficiently find all large connected components of G_g, which we refer to as the* spanning structure *of G_g.*

We define GV as a standard feature matching and geometric verification procedure, involving SIFT matching [6] with approximate nearest neighbor search and a ratio test for classifying true and false feature matches. These feature matches are then verified using a RANSAC-based estimation of the fundamental or essential matrix, depending on the availability of camera calibration information [7].

Our approach works by proposing sets of image pairs to match, based on visual similarities and rank information accumulated as the algorithm proceeds. Initially, we use a standard bag-of-visual-words approach to model visual similarity. We train a vocabulary tree offline on 50,000 images of a single city to yield 1 million visual words. For each image in the input collection $\mathcal{I}$, its SIFT descriptors are assigned to the approximately closest visual word. The image is then represented as a histogram of visual words (a vector of length 1M), weighted using standard *tf-idf* weighting [10]. In this paper, we use the same vocabulary tree for all experiments, and we use the dot product on normalized vectors as our image similarity metric.

3.1 Algorithm Overview

Our image matching method, called **MatchMiner**, consists of two steps. In Step 1, each image is issued as a query image, and highly ranked images are given priority for verification. This step employs Rocchio's relevance feedback [14] to improve the quality of retrieval results. In Step 2, we re-rank image pairs for testing with a preference for merging large CCs. We propose a *rank distance* to eliminate spurious image pairs, and use visual similarity to estimate the probability that a match will succeed. Note that in both steps, we skip verifying image pairs that are already in the same CC, as well as image pairs that have already been tested. We follow the standard practice [8, 9] of setting a verification budget, i.e., the maximum number of image pairs that may be tested; we use a budget of $K \times |\mathcal{I}|$ (i.e., on average each image is matched to at most K other images), and we terminate after the budget is exhausted, returning whatever CCs are found. We now describe these two steps in detail.

3.2 Step 1: Finding Initial Connected Components

Initially each image forms its own connected component, and each image is issued as a query to retrieve visually similar images. Over time, to maintain a high probability of success in linking CCs, we update each query using a modified version of Rocchio's relevance feedback [14]. In Rocchio's relevance feedback, a query vector Q_t is modified based on a shortlist of retrieved images. In our case, geometric verification (GV) is performed on the top k most similar images to a query, which partitions the k images into a positive set $\mathcal{P}$ and a negative set $\mathcal{N}$. These sets are used as feedback to create a

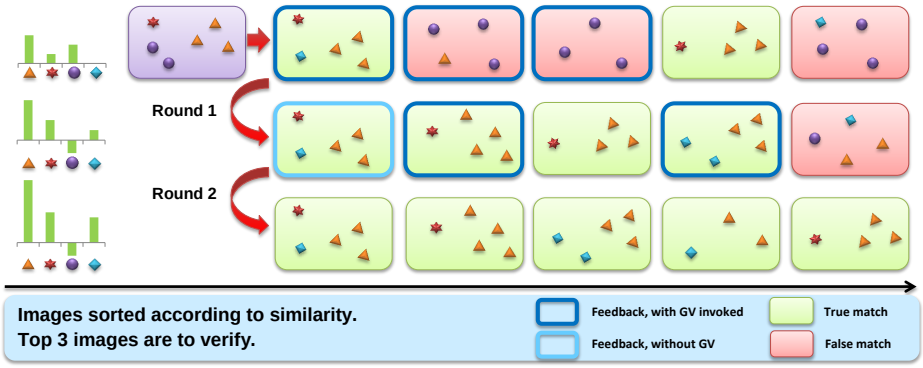


Fig. 2. Illustration of Step 1. The visual word distribution of the initial and modified query vectors are shown on the left. Row 1 uses the original query vector, and rows 2 and 3 use queries with feedback incorporated. The algorithm skips images already in the same CC of the query image, such as the third image in row 2.

new virtual query vector Q_{t+1} (before normalization), based on the following formula,

$$Q_{t+1} = Q_t + \frac{\alpha^{t+1}}{|\mathcal{P}|} \sum_{I \in \mathcal{P}} I - \frac{\beta^{t+1}}{|\mathcal{N}|} \sum_{I \in \mathcal{N}} I \tag{1}$$

where α and β are dampening factors to prevent topic drift, and I denotes original bag-of-words vector for an image. The new query vector is then normalized and used to retrieve a new set of relevant images. In our experience, setting $\alpha = \beta = 0.8$ gave good retrieval results. Unlike in most settings for Rocchio's relevance feedback, we allow negative weights on features, as this produced better results in our experiments.

Figure 2 illustrates an example of Step 1. The query image (purple) is first used to retrieve a shortlist of images (Row 1). We verify the top 3 images in the shortlist (in dark blue boxes) to get feedback. Here, the feedback indicates that purple circles are "noisy" and therefore down-weighted in the next query. The new query vector is normalized, and images are retrieved again (Row 2). We do not match images that are already in the same CC; for example, we skip the third image in row 2. The query is again refined according to feedback to retrieve more true matches in the shortlist. Note that as we keep refining the query, the shortlist generally becomes cleaner, and in addition the algorithm is able to find important features that are not in original query (the blue diamond in this example). It is easy to see that in Step 1, the algorithm invokes GV at most $Tk \times |\mathcal{I}|$ times, where T is the number of rounds of relevance feedback.

There are several reasons why we use Rocchio's relevance feedback to modify the query. First, we find that it significantly improves accuracy even after one or two rounds of relevance feedback, as new query vectors are created with a bias towards features in $\mathcal{P}$, and against features in $\mathcal{N}$. Second, Rocchio's relevance feedback is very efficient. Note that the new query vector is a linear combination of the old query vector and image vectors in the shortlist; moreover, the dot product is linear in the first argument: $\langle ax + by, z\rangle = a\langle x, z\rangle + b\langle y, z\rangle$, which means that we do not need to recompute the similarity between the new query and image vectors from scratch. Instead, we simply reuse precomputed dot products between images to compute the similarity between the

new query and other image vectors, a very fast operation. Although geometric verification is the bottleneck of the whole system, we still desire an efficient re-ranking scheme since it is a heavily used subroutine.

In summary, Step 1 performs T rounds of relevance feedback, where each round considers the top k images in each image's shortlist. T and k are chosen so that the overall budget $K \times |\mathcal{I}|$ is not exhausted. Step 1 returns an initial set of CCs and the number of geometric verifications that have already been spent.

3.3 Step 2: Merging Connected Components

After Step 1, we have discovered a set of initial connected components, and have used up some portion of the verification budget. As others have observed, we find that the probability of geometric consistency of neighbors for a given image decreases quickly with the rank of the neighbor in the shortlist, and so verification on image pairs proposed solely based on visual similarity can frequently fail after a point; the verification budget would not be used fruitfully if we continue to match such pairs. Therefore in Step 2, we use a weighting function $w(\cdot)$ to re-rank the shortlists of each image, aiming to increase the probability of success. Each image retrieves its top neighbor with highest weight, and geometric verification is performed on such pairs until the verification budget is exhausted. Again, we avoid matching images already in the same CC.

As our objective is to create large CCs, we prefer a weighting function that proposes image pairs that can potentially merge two CCs into an even larger CC. We formalize this intuition using the concept of entropy.

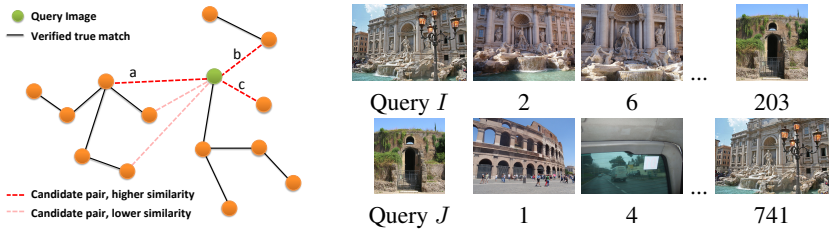
Minimizing Entropy. Let $\mathcal{C}$ denote a set of connected components, where each component $c \in \mathcal{C}$ is a subset of images. The *entropy* of $\mathcal{C}$, denoted $H(\mathcal{C})$, is defined as:

$$H(\mathcal{C}) = -\sum_{c \in \mathcal{C}} p(c) \cdot \log p(c), \tag{2}$$

where $p(c)$ is the probability that an image arbitrarily selected from the dataset belongs to connected component c, i.e., $p(c) = \frac{|c|}{|\mathcal{I}|}$.

When initially every image forms its own CC, the entropy of this set of components is at its maximum, $\log |\mathcal{I}|$. If we are able to connect all images into one CC, then the entropy is 0. As we merge CCs, the entropy strictly decreases monotonically until and unless we find exactly all CCs in G_g (i.e., we find the ground truth CCs). Thus our task can be re-formulated as minimizing $H(\mathcal{C})$ given the verification budget.

Intuitively, if we merge two small CCs, the entropy $H(\mathcal{C})$ will not decrease by much. However, when two CCs are merged into a large one, $H(\mathcal{C})$ will quickly decrease. We now show this more formally. Consider two images that straddle two CCs and are geometrically consistent; we can thus merge these CCs, c_x and c_y. Let $x = |c_x|$ and $y = |c_y|$. The entropy $H(\mathcal{C})$ upon merging c_x and c_y will decrease as $\Delta H(\mathcal{C}) = \frac{x}{n} \log \frac{n}{x} + \frac{y}{n} \log \frac{n}{y} - \frac{x+y}{n} \log \frac{n}{x+y}$. Let $z = x + y$. It can be easily shown that $\frac{\partial \Delta H(\mathcal{C})}{\partial z} \propto \frac{1}{n} \log \frac{z}{z-y}$. Thus, as z goes to infinity, $\frac{\partial \Delta H(\mathcal{C})}{\partial z}$ converges to 0, which means we reach a



(a) Illustration of entropy-descent strategy of choosing edges to verify. The algorithm will pick edge a to verify, as it is strong and spans large CCs.

(b) Motivation for the rank distance. The rank after two rounds of relevance feedback is shown below the image. Image J is proposed by similarity to image I, but pruned by the rank distance.

Fig. 3. Illustration of Step 2. (a) Image pairs will be chosen to maximize the reduction in expected entropy, except for (b) pairs with large rank distance that are pruned.

local maxima of $\Delta H(\mathcal{C})$. This shows that if we fix a component c_y, we should merge it with the largest CC possible so as to obtain the largest decrease in $H(\mathcal{C})$.

We can use this idea to define a weighting function $w(\cdot)$ as the *expected reduction in entropy*, $E_I(J)$, for matching an image J to a query image I—the decrease in entropy weighted by the probability of a successful merge:

$$w_I(J) = E_I(J) = p_I(J) \cdot \Delta H(\mathcal{C}) \tag{3}$$

where $p_I(J)$ is an estimated probability of success of the merge. We define this probability later in this section. Note that after Step 1, we need to choose pairs (I, J) very carefully, as the probability of success for an arbitrary merge is low.

Figure 3(a) illustrates how a query image (the green node) chooses its next image to verify. Black lines are already verified true matches and red dashed lines are candidate pairs to verify; darker red indicates higher similarity. Although there are three strong candidates, our method chooses edge a since its expected decrease in entropy is highest.

Eliminating Singleton Images. Large-scale, unstructured image datasets often contain a significant number of singleton images ("distractor" or "noise" images) that match no other image, such as a closeup of a random cobblestone or of water. Worse, we have observed that these often contain confusing visual words that cause those images to be relatively highly ranked in the shortlists of other images. We propose a simple yet effective method to eliminate these images.

Consider two images I and J, and assume I is a "good" image, while J is an outlier image. In Step 1, when we use J as a query, all of the verifications will fail. However, each round of relevance feedback adds negative feedback pushing J away from these initially similar (according to the bag-of-words model), yet non-matching images. After T rounds, the refined query will be far away from those non-matching images. Let $Rank_I(J)$ denote the rank for image I using image J as a query in Step 1. Although $Rank_I(J)$ might be small because of noisy features present in J, $Rank_J(I)$ will often be large since this is not a "true" match. Figure 3(b) illustrates this effect with a real example from one of our datasets. Based on this observation, we propose as a distance

function the *rank distance* $R(I, J)$ for a pair of images I and J, defined as the harmonic mean of $Rank_J(I)$ and $Rank_I(J)$:

$$R(I, J) = 2Rank_I(J)Rank_J(I)/(Rank_I(J) + Rank_J(I)) \quad (4)$$

If either of $Rank_J(I)$ or $Rank_I(J)$ is large, $R(I, J)$ is large, indicating that the pair (I, J) may not be promising. Our experiments show that we can use this ranking to prune many (near) singleton images effectively and increase the probability of performing successful matches.

Probability of successful match. We now define the probability of geometric consistency $p_I(J)$ using similarity and rank information. Let $\mathrm{SimNeighbor}_{N_s}(I)$ denote the set of the top N_s most visually similar neighbors to a query image I. We model the likelihood of geometric consistency as a normal distribution $\mathcal{N}(0, \sigma_s)$. Similarly, let $\mathrm{RankNeighbor}_{N_r}(I_i)$ denote a set of closest N_r neighbors to the query image I_i, based on rank distance $R(I, J)$. We wish to highly score images that are both visually similar to the query image, and highly ranked in each other's rank list; accordingly, we define the estimated probability of geometric consistency as follows:

$$p_I(J) \propto \begin{cases} \frac{1}{\sqrt{2\pi}\sigma_s} e^{-\frac{(s-1)^2}{2{\sigma_s}^2}}, & J \in \mathrm{SimNeighbor}_{N_s}(I) \cap \mathrm{RankNeighbor}_{N_r}(I) \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where s is the original similarity of image I and J.

We use the original queries (rather than modified queries) because we observe that true matches can sometimes be pushed down by relevance feedback in the ranking. However, since the original similarity function is very noisy, we need to filter out false matches and have found the rank distance an effective way to do so.

4 Experiments

To evaluate the effectiveness of MatchMiner, we performed experiments on several medium- and large-scale datasets. We downloaded five datasets from Flickr[2]: **Acropolis**, **Pantheon**, **St. Paul's**, **Forum**, and **Washington DC**, using keyword search. Images in the Acropolis, Forum, and Washington DC datasets come with geotags, while the other images do not. We sample two small datasets according to their geotags, forming smaller datasets Forum 1 and Forum 2. The ground truth image graphs for Forum 1, Forum 2, Acropolis, Pantheon, and St. Paul's are computed by exhaustive geometric verification on all image pairs, as these are small enough so that exhaustive matching is tractable. Table 1 summarizes these datasets.

4.1 Effectiveness of Relevance Feedback

Step 1 of MatchMiner uses relevance feedback; to understand how much relevance feedback improves the accuracy, we first compare relevance feedback with dot product and

[2] `http://www.flickr.com`

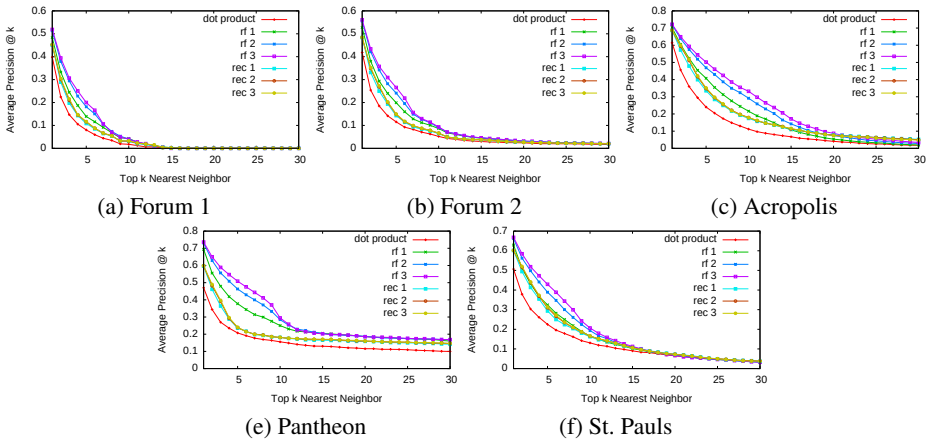
Table 1. Datasets. Each row lists: *Name*, the name of the dataset; *# Images*, the number of images in the dataset; CC_1, the size of the largest connected component in the ground truth image graph (if known); $|V_g|$, the number of images that are not singletons (i.e., in their own CC) in the ground truth image graph; μ_I, the average number of features per image; and σ_I, the standard deviation of number of features per image.

Name	# Images	CC_1	$\|V_g\|$	μ_I	σ_I
Forum 1	1069	320	692	7741	6379
Forum 2	2911	1213	1937	6528	4957
Acropolis	2961	2105	2451	3969	2242
Pantheon	1123	855	947	10784	13309
St. Pauls	3000	2045	2372	11388	10191
Forum	74391	-	-	7748	5752
DC	375531	-	-	5379	5161

recursive average query expansion [12], using Average Precision (AP) @k. Each image in a dataset is issued as a query and all retrieved images are ranked solely based on their similarity to the query. Precision@k of a query Q is computed as $|\{I \mid GV(Q, I)\}|/k$, and AP@k is obtained by averaging Precision@k for all queries. In these experiments, each round of relevance feedback and recursive average query expansion examines the top five images in the shortlist. We compare with average query expansion because it is a state-of-the-art retrieval method with little overhead in the retrieval process. Figure 4 shows the AP@k on five datasets. Relevance feedback achieves significantly better AP@k than dot product and recursive average query expansion for the top images in the ranked lists. As we are mainly focused on low Ks, i.e., a relatively small verification budget, we conclude that relevance feedback is quite suitable for Step 1. Note that two rounds of relevance feedback produce excellent retrieval performance while consuming a relatively small amount of the verification budget (at most 10 verifications per query), and therefore we use two rounds for all further experiments.

4.2 Effectiveness of Rank Distance

Step 2 in MatchMiner is a combination of favoring merging large CCs and avoiding matching singleton images. We first evaluate the effectiveness of rank distance in pruning singleton images. Recall that in Step 2, for each image I in the dataset, we verify its next k images after relevance feedback. In this experiment, we propose those k images according to $p_I(J)$ and plot the average number of false edges in those k images that are pruned by using rank distance, where $N_s = N_r = 200$. Figure 5 shows the results on five datasets. By using rank distance, we can prune almost half of the images among k images without invoking actual verification. This means at least half of the top k images in N_s of Step 2 are false matches to the query, but they will be pruned by looking at N_r, which can give a large savings and increase the probability that we encounter an important edge between two CCs. On all of our datasets, the rate of pruning true edges stays consistently below 0.1%, i.e., the rank distance rarely makes mistakes.



(a) Forum 1 (b) Forum 2 (c) Acropolis

(e) Pantheon (f) St. Pauls

Fig. 4. Average precision @ k for dot product, one round of relevance feedback (rf 1), two rounds of relevance feedback (rf 2), three rounds of relevance feedback (rf 3), one round of recursive average query expansion (rec 1), two rounds of recursive average query expansion (rec 2) and three rounds of recursive average query expansion (rec 3).

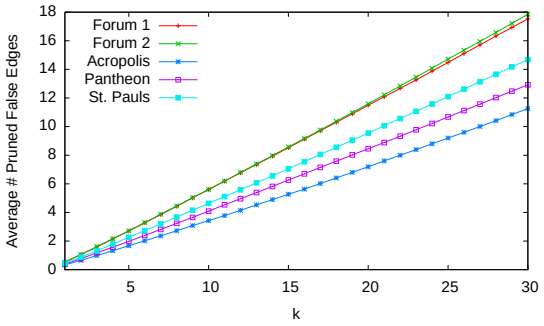


Fig. 5. Avg. number of false edges pruned by rank distance in the k images considered in Step 2.

4.3 Evaluation Metric

In order to evaluate our results, we need some measure of how similar two sets of connected components are. For this we use *normalized mutual information* [15], a popular similarity measurement for clustering algorithms; we adopt this to measure similarity of two connected component sets.

Let $\mathcal{C}^*$ denote the set of CCs in the ground truth graph G_g and $\mathcal{C}$ the spanning structure to compare to the ground truth. We define the mutual information of the two sets as follows:

$$MI(\mathcal{C};\mathcal{C}^*) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{C}^*) = \sum_{c\in\mathcal{C},c^*\in\mathcal{C}^*} p(c,c^*)\cdot\log\frac{p(c,c^*)}{p(c)\cdot p(c^*)} \qquad (6)$$

where $p(c)$ and $p(c^*)$ are the same probabilities defined in Equation 2, and $p(c,c^*)$ is the joint probability that the arbitrarily selected image belongs to both c and c^*, i.e.,

Table 2. Mining results on five datasets.

(a) K = 20.

Dataset	Algo.	CC_1	$\|V_c\|$	$\overline{MI}$
Forum 1	Image Webs	105	563	0.80
	MatchMiner	266	598	0.90
Forum 2	Image Webs	728	1670	0.80
	MatchMiner	908	1744	0.85
Acropolis	Image Webs	1894	2239	0.79
	MatchMiner	1948	2305	0.83
Pantheon	Image Webs	639	847	0.59
	MatchMiner	765	869	0.74
St. Pauls	Image Webs	1816	2145	0.80
	MatchMiner	1883	2187	0.84

(b) K = 30.

Dataset	Algo.	CC_1	$\|V_c\|$	$\overline{MI}$
Forum 1	Image Webs	180	606	0.85
	MatchMiner	271	622	0.91
Forum 2	Image Webs	788	1745	0.83
	MatchMiner	937	1761	0.87
Acropolis	Image Webs	1951	2307	0.84
	MatchMiner	1978	2324	0.86
Pantheon	Image Webs	659	855	0.62
	MatchMiner	788	900	0.80
St. Pauls	Image Webs	1845	2179	0.82
	MatchMiner	1934	2248	0.89

$p(c, c^*) = \frac{|c \cap c^*|}{|\mathcal{I}|}$. In our experiments, we use the normalized mutual information $\overline{MI}$:

$$\overline{MI}(\mathcal{C}; \mathcal{C}^*) = MI(\mathcal{C}; \mathcal{C}^*) / \max(H(\mathcal{C}), H(\mathcal{C}^*)) \tag{7}$$

It is easy to verify that $\overline{MI}(\mathcal{C}; \mathcal{C}^*)$ ranges from 0 to 1. $\overline{MI}(\mathcal{C}; \mathcal{C}^*) = 1$ if we find the exact same CCs as the ground truth, and $\overline{MI}(\mathcal{C}; \mathcal{C}^*) = 0$ if two sets are independent. In general, a higher score indicates a more complete set of connected components. It is easy to verify that minimizing $H(\mathcal{C})$ (as described in Section 3.3) is equivalent to maximizing $\overline{MI}$ and therefore Step 2 of MatchMiner explicitly maximizes $\overline{MI}$.

4.4 Mining Performance Comparison

To evaluate our results compared to ground truth, we perform several sets of experiments on different Ks (defining different verification budgets). We compare our approach to Image Webs [9]; we only compare our results with the output of their Phase 1 because their Phase 2 aims at increasing connectivity within connected components, which cannot improve $\overline{MI}$ and the spanning structure of the graph.

Tables 2(a) and 2(b) show results for $K = 20$ and 30, respectively. Each table shows the size of the largest connected component (CC_1), the number of non-singleton images in the resulting spanning structure ($|V_c|$), and the normalized mutual information ($\overline{MI}$) achieved compared to the ground truth. All of the five datasets have a single large connected component that captures the majority of the scene, so CC_1 is another important measure of success. We allow two rounds of relevance feedback in Step 1 ($T = 2$), and we set $N_s = N_d = 200$.

The results suggest MatchMiner significantly outperforms Image Webs at our task. Note that when $K = 20$, MatchMiner already outperforms Image Webs when $K = 30$ on most datasets, which demonstrates the effectiveness of our approach.

4.5 Mining Results on Large-scale Datasets

We implemented a distributed version of MatchMiner on a shared cluster with 53 nodes. Each node in the cluster has 2 quad-core Xeon processors running at 2.66

Table 3. Results for large datasets. $K = 20$ for all experiments. CC_1 and CC_2 denote the size of the largest and second largest connected component, respectively. $|V_c|$ is the number of non-singleton images. *Components* shows the distribution of large connected components. $H(\mathcal{C})$ is the entropy of the mining result. *Time* is the total running time of the system (not including extracting SIFT features and learning bag-of-visual-words vectors).

Dataset	Algorithm	CC_1	CC_2	$\|V_c\|$	Components $= 2$	$= 3$	≥ 4	$H(\mathcal{C})$	Time
Forum	Image Webs	6944	6649	40689	2919	860	1011	11.92	1hr40min
	MatchMiner	13871	3088	40604	2753	812	944	11.62	1hr39min
Washington DC	Image Webs	11249	3772	146035	19526	5083	6386	16.76	6hr13min
	MatchMiner	16922	2804	140273	17276	4554	5865	16.64	6hr21min

GHz with 16 GB memory. Experiments are performed on two large datasets, Forum and Washington DC, and results are summarized in Table 3. Since the ground truth graphs for these two datasets are intractable to compute, we use $H(\mathcal{C})$ instead of $\overline{MI}$ to measure the quality of a set of CCs, where lower $H(\mathcal{C})$ indicates a higher $\overline{MI}$. Again, MatchMiner achieves significantly better results than Image Webs. These experiments demonstrate that MatchMiner is effective at finding large-scale spanning structure, and when parallelized can find such structure very efficiently.

The distributions of the number of CCs of different sizes are also shown in Table 3. Image Webs tends to find more small CCs (e.g., size 2 and 3). However, those small CCs are not useful in most applications based on image graphs and we are mainly interested in large CCs. Comparing the numbers of large CCs, we can see MatchMiner merges more CCs and therefore the number of large CCs is much smaller than that of Image Webs. This shows that our information-theoretic framework can prevent the system being trapped by small uninformative CCs, and that the verification budget is used to match potential image pairs that can merge large CCs, guided by the decrease of entropy.

Figure 6 shows the largest CC of the Forum dataset produced by MatchMiner, rendered on a map based on geotags and with sample images shown; discovered edges are shown as line segments. MatchMiner successfully captures links spanning several landmarks, including Il Vittoriano, the Roman Forum, and the Colosseum. The algorithm also finds links between daytime and nighttime components of the Colosseum.

5 Conclusion

In this paper, we formulate the problem of discovering the structure of an image graph as a mining task called spanning structure mining, and we describe our novel MatchMiner algorithm that solves this problem very efficiently as compared to the state of the art. In future work, we would like to further improve our algorithms, especially focusing on finding noise images and eliminating them even earlier in the process.

Acknowledgments. We thank anonymous reviewers for their valuable comments. This research was supported by the NSF under grants IIS-0964027 and a grant from Intel.

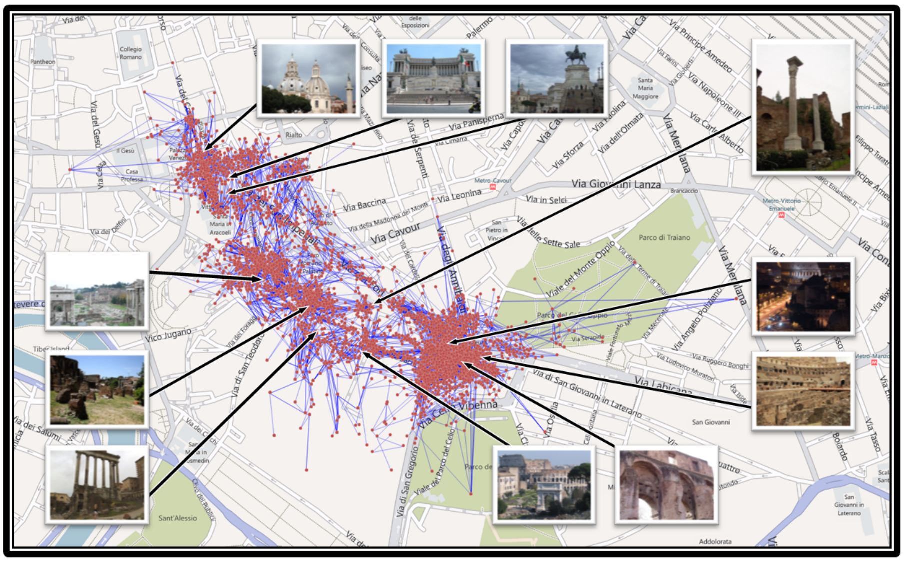


Fig. 6. Visualization of largest connected component of the image graph for Forum.

References

1. Snavely, N., Seitz, S.M., Szeliski, R.: Skeletal graphs for efficient structure from motion. In: CVPR. (2008)
2. Avidan, S., Moses, Y., Moses, M.: Probabilistic multi-view correspondence in a distributed setting with no central server. In: ECCV. (2004)
3. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica **20** (1998) 374–387
4. Frahm, J., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y., Dunn, E., Clipp, B., Lazebnik, S., et al.: Building rome on a cloudless day. In: ECCV. (2010)
5. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. ACM Trans. Graph. **25** (2006) 835–846
6. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004)
7. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. 2nd edn. Cambridge University Press (2003)
8. Agarwal, S., Snavely, N., Simon, T., Seitz, S.M., Szeliski, R.: Building rome in a day. In: ICCV. (2009)
9. Heath, K., Gelfand, N., Ovsjanikov, M., Aanjaneya, M., Guibas, L.J.: Image webs: Computing and exploiting connectivity in image collections. In: CVPR. (2010)
10. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR. (2006)
11. Chum, O., Mikulík, A., Perdoch, M., Matas, J.: Total recall ii: Query expansion revisited. In: CVPR. (2011)
12. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: ICCV. (2007)
13. Chum, O., Matas, J.: Large-scale discovery of spatially related images. IEEE Trans. Pattern Anal. Mach. Intell. **32** (2010) 371–377
14. Rocchio, J.J.: Relevance feedback in information retrieval. In: The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall Inc. (1971) 313–323
15. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: SIGIR. (2003)