Partition Min-Hash for Partial Duplicate Image Discovery

David C. Lee*, Qifa Ke†, and Michael Isard†

Carnegie Mellon University*, Microsoft Research Silicon Valley†
dclee@cs.cmu.edu, {qke,misard}@microsoft.com

Abstract. In this paper, we propose Partition min-Hash (PmH), a novel hashing scheme for discovering partial duplicate images from a large database. Unlike the standard min-Hash algorithm that assumes a bag of words image representation, our approach utilizes the fact that duplicate regions among images are often localized. By theoretical analysis, simulation, and empirical study, we show that PmH outperforms standard min-Hash in terms of precision and recall, while being orders of magnitude faster. When combined with the start-of-the-art Geometric min-Hash algorithm, our approach speeds up hashing by 10 times without losing precision or recall. When given a fixed time budget, our method achieves much higher recall than the state-of-the-art.

Key words: partition min-hash, min-hash, partial duplicate image discovery

1 Introduction

In this paper, we introduce a new method for partial duplicate image discovery in a large set of images. The goal of partial duplicate image discovery is to find groups of images in a large dataset that contain the same object, which may not necessarily occupy the entire image. Figure 1 shows examples of such groups of partial duplicate images. Partial duplicate image *discovery* differs from partial duplicate image *retrieval* in that there is no particular query image, but instead the task is to search for all groups of duplicate images in a dataset. Such a task is useful for identifying popular images on the web so that images can be ranked by their importance, for grouping similar images returned by a web search so that users can navigate the returned results more easily and intuitively, or for unsupervised discovery of objects.

Min-hash is a standard hashing scheme for discovering near-duplicate text documents or web pages [1]. Recently min-hash and its variants have been successfully applied to discovering near duplicate images [2, 3], image clustering, image retrieval and object discovery [4]. In the min-hash algorithm, a hash function is applied to *all* visual words in an image ignoring the location of visual words, and the visual word with minimum hash value is selected as a global descriptor of the given image.

Unlike text documents which are usually represented by bag of words, images are strongly characterized by their 2D structure—objects are often spatially *localized* in the image and there exist strong *geometric constraints* among the visual words in an object.

* This work was done during an internship at Microsoft Research Silicon Valley.

Fig. 1. Examples of partial duplicate images. The duplicate region occupies a small portion of each image.

Figure 1 shows some examples where the objects, and therefore the duplicate regions, are localized in the images. However, standard min-hash treats all visual words independently. A straightforward application of min-Hash to images ignores both locality and geometric constraints.

Geometric min-hash (GmH) [4] improves upon standard min-hash by considering the dependency among visual words. It first computes a min-hash value in a way similar to standard min-hash. The rest of the hash values in a sketch are then chosen only within a certain proximity of the first visual word. However, the locality property is still ignored in Geometric min-hash (GmH) when computing the first min-hash. If the first min-hash does not repeat between two matched images, the containing sketch is not repeatable and becomes useless.

Our ultimate goal is to detect partial-duplicate images from a web scale image database, where both precision/recall and computational cost are critical for scalability. In this paper, we aim to exploit locality and geometric constraints to improve precision/recall and to reduce computational cost. We propose Partition min-Hash (PmH), a novel hashing scheme to exploit locality, i.e. the fact that duplicate regions are usually localized in an image. In PmH, an image is first divided into overlapping partitions. Hashing is then applied independently to the visual words within each partition to compute a min-hash value.

Since the duplicate regions are localized, it is likely that one of the overlapping partition contains the common region among partial-duplicate images. By hashing within the partition instead of over all of the image, the min-hash is more repeatable among partial-duplicate images. By theoretical analysis, simulation, and experiments on real images, we show that PmH not only outperforms standard min-Hash in both precision and recall, but is also more than ten times faster for hashing, and more than two times faster overall including image preprocessing (at 1000 sketches/image). We also show that, when alloted the same amount of time, the proposed method achieves much higher recall than previous methods.

Partition min-hash and geometric min-hash can be used in conjunction by first partitioning images and then applying GmH to each partition. This improves the precision/recall of GmH, while speeding-up hashing by an order of magnitude.

To further utilize geometric constraints among visual words, we augment PmH by encoding the geometric structure in the sketches. Specifically, the geometric relation-

ship among visual words in a sketch is quantized into an ID. This ID is then concatenated to the sketch to form the final representation of the image region.

1.1 Related Work

Large scale partial duplicate image discovery is closely related to image retrieval, which includes two popular themes. One theme represents an image as a bag of visual words, and then applies approaches from the text domain for efficient image indexing and retrieval [5–8]. Another theme uses hashing schemes to efficiently find similar images [3, 9–12].

Naive application of image retrieval methods to partial duplicate image discovery can be done by using every image in the set as a query image. This has the computational complexity of the retrieval method multiplied by the number of images, which becomes prohibitive if the computational complexity of the retrieval method is more than $O(1)$. Hashing based methods are more suitable for partial duplicate image discovery, because all images can be hashed into a hash table and hash collisions can be retrieved as similar images, which can then be further expanded into more complete image clusters by image retrieval [4].

In this paper, we focus on designing efficient hashing schemes for scalable partial duplicate image discovery. Like previous works, we represent an image as a set of visual words [5], which are obtained by quantizing local SIFT feature descriptors [13, 14]. Min-hash [1] and its variants can then be applied to finding similar sets and therefore similar images [3, 4]. In particular, we are inspired by geometric min-hash [4].

2 Min-Hash Algorithm

In this section we present some background on the min-hash algorithm. Min-hash is a Locality Sensitive Hashing scheme [15] that approximates similarity between sets. When an image is represented as a set of visual words, the similarity between two images can be defined as the Jaccard similarity between the two corresponding sets of visual words I_1 and I_2:

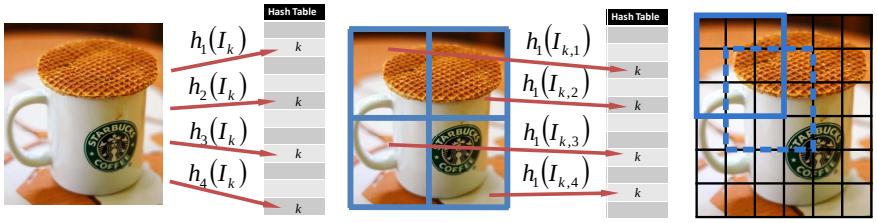
$$sim(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|},$$

which is simply the ratio of the intersection to the union of the two sets.

Min-hash is a hash function $h : I \mapsto v$, which maps a set I to some value v. More specifically, a hash function is applied to each visual word in the set I, and the visual word that has minimum hashed value is returned as the min-hash $h(I)$. One way to implement the hash function is by a look-up table, with a random floating-point value assigned for each visual word in the vocabulary, followed by a min operator. The computation of the min-hash of a set I involves computing a hash of every element in the set and the time taken is therefore linear in the size of the set $|I|$. More details on min-hash can be found in [1, 3].

Min-hash has the property that the probability of hashing collision of two sets is equal to their Jaccard similarity:

$$P(h(I_1) = h(I_2)) = sim(I_1, I_2).$$



(a) (b) (c)

Fig. 2. (a) In standard min-hash, min-hash sketches are extracted from the entire image. (b) In Partition min-hash, the image is divided into partitions and a min-hash sketch is extracted for each partition. (c) Overlapping partitions (in thick blue solid/broken line) are more likely to capture the entire duplicate region and lead to better performance. Sketches can be pre-computed for each grid element (thin black line) to avoid most of the redundant computation for overlapping partitions.

Since the output of a min-hash function v is actually a visual word, it carries the position and scale information of the underlying local feature for geometric verification.

In image retrieval or partial duplicate image discovery, we are interested in finding images which have similarity greater than some threshold θ. In other words, we would like the probability of collision to be a step function:

$$P(h(I_1) = h(I_2)) = \begin{cases} 1 & \text{if } sim(I_1, I_2) \geq \theta; \\ 0 & \text{otherwise.} \end{cases}$$

This step function can be approximated by applying k min-hashes to a set and concatenating them into a *sketch*. Then n sketches can be computed for an image and all of them can be added to the hash table. Under this setting, two images will collide if they share one common identical sketch. The probability for two images to collide in the hash table becomes:

$$P(h(I_1) = h(I_2)) = 1 - \left(1 - sim(I_1, I_2)^k\right)^n, \tag{1}$$

which approximates the step function. The sharpness of the "step" and threshold θ can be controlled by varying the sketch size k and the number of sketches n.

This scheme of computing n min-hash sketches of size k will be the baseline for our method, and we denote it as "standard min-hash".

3 Partition Min-Hash

Based on the observation that duplicate regions among partial-duplicate images are usually localized, we propose a new method called Partition Min-Hash. It has better precision and recall and runs orders of magnitude faster than standard min-hash. It is also very easy to implement and only has partition size and overlap as tuning parameters. The rest of this section introduces the method and discusses its performance.

3.1 Method Details

An image is divided into p rectangular regions of equal area, which we call partitions (Figure 2(b)). Then, instead of extracting min-hash sketches from the entire image, min-

Algorithm 1 Partition min-hash

Initialize N_s independent min-hash sketch functions h_i, where $i = 1, \cdots, N_s$.
Initialize N_s hash tables T_i, which map a min-hash sketch s to image ID k.
for all Images I_k in database **do**
  Divide image I_k into a grid, $I_{k,j}$, where $j = 1, \cdots, N_g$
  For each partition, determine the grid elements that are associated with the partition.
  for $i = 1, \cdots, N_s$ **do**
    for $j = 1, \cdots, N_g$ **do**
      Extract sketch $s_i \leftarrow h_i(I_{k,j})$
    for all Partitions **do**
      Look up sketches s_j extracted from grids that belong to current partition, and select true min-hash sketch s^*
      Add (s^*, k) to hash table T_i

hash sketches are extracted for each partition independently. The p min-hash sketches, one extracted from each partition, are inserted into the hash table.

As will be analyzed in the following section, partitions that fully and tightly capture a duplicate region between images lead to better precision and recall, compared to cases in which a duplicate region spans several partitions, or where the partitions are much larger than the duplicate region. With evenly divided partitions, the duplicate is often split into two or more partitions. To alleviate this, we design partitions to be overlapping (Figure 2(c)) and multi-scale. This gives us a better likelihood of having at least one partition that captures the duplicate region completely. This may remind the reader of the sliding window technique, widely used for object detection. The spirit is, in fact, similar: we are hoping for one of the subwindows to hit a region of interest, which, in our case, is an unknown duplicate region.

We can avoid redundant computation on overlapping partitions by precomputing and reusing min-hashes. An image is divided into a grid, where the grid elements are the greatest common regions among partitions that cover that region (Figure 2(c)). Min-hash sketches are precomputed for each grid element w_i. Then the min-hash sketch for a partition P is computed by looking up elements $\{w_i\}$ that are associated with that partition P and picking the true min-hash sketch among the precomputed min-hash sketches on elements:

$$h(P) = \min_i \{h(w_i) | w_i \in P\}$$

The entire algorithm is summarized in Algorithm 1.

3.2 Theoretical Analysis on Performance

In this section, we will analyze the speed and performance of partition min-hash and show that it achieves higher precision and recall in less time than standard min-hash.

For the sake of comparison, we will keep the number of sketches per image equal for both cases. With the same number of sketches stored in the hash table, the two methods will use the same amount of memory. For example, if n sketches per image were computed for standard min-hash, we will compute n/p sketches for each of the p partitions for partitition min-hash, so that the total number of sketches per image equals $p \cdot n/p = n$.

Speed Processing time can be divided into two components: the time to compute sketches and the time for other overhead operations, such as reading the image from disk, extracting visual words, adding sketches to hash tables, etc. The overhead will be the same for both methods, with a fixed number of images and sketches per image for both methods, so the difference in time will be in computing sketches.

As mentioned in Section 2, the time to compute a min-hash sketch is linear in the size of the set. In partition min-hash, each sketch is the result of applying min-hash to a partition instead of the entire image, so the time it takes to compute each sketch is reduced by a factor of M_i/M, where M_i is the number of features in partition i and M is the number of features in image. On average, M_i/M will be roughly equal to the ratio of the area of the partition and the area of the image, which, in the case of non-overlapping partitions, is $1/p$, where p is the number of partitions. So the overall time to compute sketches reduces by $1/p$. In the case of overlapping partitions, additional time is required to look up grid elements associated with each partition, compared to the non-overlapping case, however this is small compared with the time to hash features. The key to this speedup is that in standard min-hash, each feature participates in all n sketches, so must be hashed n times. In PmH, each feature only participates in n/p sketches, so the number of computed hashes is reduced by a factor of p.

Precision and Recall We now show that the sketches created by partition min-hash have better discriminative power than the sketches created by standard min-hash, despite the fact that they take less time to compute. We will study precision and recall by analyzing the collision probability of true matching image pairs and the collision probability of false matching image pairs. The collision probability of true matching pairs is equal to recall, defined as the number of retrieved positives divided by the number of true positives. The collision probability of false matching pairs is related to (but not equal to) precision, defined as the number of retrieved positives divided by the number of retrieved images. We have derived in Section 2, Equation (1) that the collision probability of standard min-hash is equal to $P(h(I_1) = h(I_2)) = 1 - \left(1 - sim(I_1, I_2)^k\right)^n$. We will show that partition min-hash achieves higher collision probability for true matching pairs and lower probability for false matching pairs, thus achieving higher precision and recall.

Let us first analyze the collision probability of true matching image pairs. To simplify the analysis, we will consider the case of non-overlapping partitions (Figure 2(b)). The arrangement of partitions with respect to the region with duplicate content can then be categorized into three cases, illustrated in Figure 3(a),(b),(c). For the purposes of analysis, we have assumed that features are spread across the image and each partition within an image contains the same number of features (including matching and non-matching background features). Once the simplified analysis on non-overlapping cases is done, it will be easy to infer that overlapping partitions are more likely to generate "preferred" partitions.

Case (a): The duplicate region is contained within a single partition in each image. Since duplicate features are contained in only 1 of the p partitions, the similarity between the two images $sim(I_1, I_2)$ must be less than $1/p$. Now, the similarity between the partitions containing duplicate region is $p \cdot sim(I_1, I_2)$, where p is the number of partitions. Since n/p sketches are extracted from those partitions, the overall collision

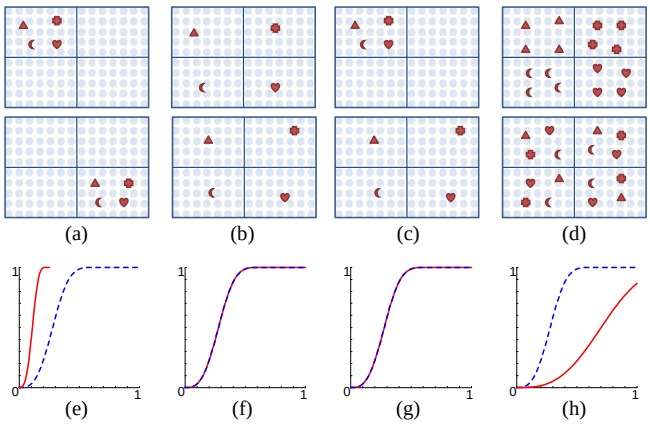


Fig. 3. (a)(b)(c): Illustration of true matching image pairs. Various symbols in red represent matching features across images. Lightly colored circles in the background represent non-matching features and are assumed to be spread out uniformly across partitions. (a): The duplicate region is captured in a single partition in each image. (b): The duplicate region is split across all partitions. (c): Mix of (a) and (b). (d): Illustration of a false matching image pair. Features are randomly distributed. (e)(f)(g)(h): Collision probabilities plotted against image similarity for cases (a)(b)(c)(d), respectively. Red solid curve: Collision probability of partition min-hash. Blue broken curve: Collision probability of standard min-hash. X axis: Similarity. Y axis: Collision probability.

probability is equal to

$$P(h(I_1) = h(I_2)) = 1 - \left(1 - (p \cdot sim(I_1, I_2))^k\right)^{n/p},$$

and is plotted in Figure 3(e). Partition min-hash achieves a higher collision probability than standard min-hash in this case.

It is possible for duplicate regions to lie in between partitions. In such case, it can be shown that the collision probability is less than what was derived above, but is still greater than standard min-hash. Moreover, overlapping partitions increases the chance of having a partition that covers a duplicate region, thus increasing the collision probability even further. We obtained the collision probability for overlapping partitions through simulation and it is reported at the end of this section.

Case (b): The duplicate regions are split up among partitions. The illustration shows the most extreme case where the duplicate region is split across all p partitions. Since we are considering an actual duplicate image, each partition from one image will have a corresponding matching partition in the other image, e.g. partition 1 from image 1 matches with partition 1 from image 2, partition 2 from image 1 matches with partition 2 image 2, and so on. Now for each pair of matching partitions, the similarity between the pair will be the same as the original similarity $sim(I_1, I_2)$. For each partition, n/p sketches are extracted, but the image will be considered as colliding if any one of the p

pairs collide. So the overall collision probability is equal to

$$\begin{aligned} P(h(I_1) = h(I_2)) &= 1 - \left(\left(1 - (sim(I_1, I_2))^k\right)^{n/p}\right)^p \\ &= 1 - \left(1 - sim(I_1, I_2)^k\right)^n, \end{aligned}$$

which is the same as the collision probability for standard min-hash, and it is plotted in Figure 3(f). In practice, the splitting of duplicate regions will typically not be as extreme as a split across all p partitions and the collision probability will be somewhere in between case (a) and case (b).

Case (c): The duplicate region is contained in one partition in one image and split up into p partitions in the other image. The partition which contains the entire duplicate region has non empty intersection with all p partitions from the other image and has the probability to have the same min-hash value proportional to their similarity, which is equal to $sim(I_1, I_2)$, as on average the number of duplicate features and the number of non-duplicate features are reduced by the same ratio from the entire image. Again in this case, the collision probability is equal to

$$P(h(I_1) = h(I_2)) = 1 - \left(1 - sim(I_1, I_2)^k\right)^n,$$

plotted in Figure 3(g), but in practice most images will lie somewhere between cases (a) and (c). The simulation at the end of the section confirms the above analytic result.

Case (d): The collision probability of false matching image pairs. We assume that in a false match, duplicate features among two images are scattered randomly across the image, as opposed to being localized in some partition. This is illustrated in Figure 3(d). Partitioning randomly scattered features can be considered as random sampling, and the expected similarity between one randomly sampled partition and another randomly sampled partition reduces by a factor of p, i.e., the expected similarity between any partitions in the two images is equal to $sim(I_1, I_2)/p$. As there are p partitions for each image, there are a total of p^2 combinations leading two partitions to collide. Therefore, the collision probability is equal to

$$\begin{aligned} P(h(I_1) = h(I_2)) &= 1 - \left(\left(1 - (sim(I_1, I_2/p)^k\right)^{n/p}\right)^{p^2} \\ &= 1 - \left(1 - (sim(I_1, I_2)/p)^k\right)^{np}, \end{aligned}$$

which is lower than standard min-hash, and is plotted in Figure 3(h). In practice, some partitions will have higher number of duplicate features and have higher similarity than $sim(I_1, I_2)/p$, which leads to an overall collision probability that is higher than what is derived above. But the chances of having a partition with significantly high number of duplicate features will be low, and the true collision probability will be close to what we derived.

Overlapping partitions and simulation verification. So far, the analysis has been done for non-overlapping partitions where the duplicate region is also within some partition. In practice, the duplicate region may stride over partition boundaries, so we use overlapping partitions to achieve higher chance of capturing the duplicate region in one

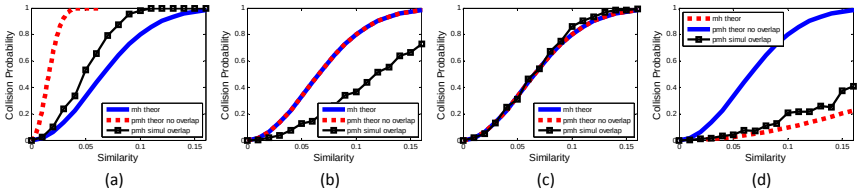


Fig. 4. Simulated collision probability of partition min-hash with overlapping partitions for four cases. 'mh theor': Theoretical rate of standard min-hash. 'pmh theor no overlap': Theoretical rate of partition min-hash with non-overlapping partitions. 'pmh simul overlap': Simulated rate of partition min-hash with overlapping partitions.

partition. The theoretical collision probability for overlapping partitions is complicated due to dependence among sketches from overlapping partitions. Instead, we use synthetic examples to simulate the case where the duplicate regions are not aligned with partitions for the four cases in Fig. 3, and apply partition min-hash with overlapping partitions. The simulation was done by synthesizing images with visual words distributed as described in the four cases and applying partition min-hash to the images. The simulation is repeated 1000 times and the collision probabilities are reported in Figure 4.

Compared to standard min-hash, we see that the collision probability of partition min-hash with overlapping partitions is higher in case (a) and similar in case (c). For case (d), which is false matching, the probability of false collision is much lower than standard min-hash. Compared to the ideal non-overlapping case, both (c) and (d) have similar performance, while (a) is not as good as the ideal case. This is expected as we are using the same number of sketches for both overlapping and non-overlapping cases. As a result, the number of sketches per partition for overlapping case is lower. This affects case (b) the most where its collision probability is lower than the standard min-hash. In practice, most duplicates will occupy a portion of the image and will be in between case (a) and case (b). Moreover, since we pre-computed min-hash for each grid, we can get more sketches for each overlapping partition almost for free.

4 Evaluation of Partition Min-Hash

In this section, we present quantitative evaluations of our method using two datasets, our own dataset collected from the web and the Oxford buildings dataset [6].

4.1 Experimental setup

In our own dataset, we have collected 778 images from the web and manually categorized them into 19 categories, where the images within each category are partial duplicates. There are no exact duplicates among this set of 778 images. The set contains 17595 true matching image pairs (belonging to the same category) out of $778 \times 777/2 = 302253$ total pairs. Such a large set of image pairs is adequate for evaluating hashing schemes. The average Jaccard similarity for true pairs is 0.019. The number of features per image ranges from 200 to 1000, and we have quantized them using a visual word vocabulary with one million visual words. It takes about 100ms per image to extract

visual words. Min-hash functions are implemented as lookup tables of random floating point values assigned per each visual word, followed by a min operation.

The Oxford buildings dataset [6] consists of 5062 images collected from Flickr by searching for particular Oxford landmarks. The annotation provides whether an image in the database is a partial duplicate of a query image for 55 queries. As the time that was taken to extract visual words were not reported in the Oxford buildings set, we assumed it took 100ms per image in our reported graphs.

We tested the quality of collision pairs that are retrieved by counting the number of true pairs and false pairs that were retrieved using our proposed method. Recall was computed as the number of retrieved true *pairs* divided by the total number of true pairs. Precision was computed as the percentage of true *pairs* among all retrieved pairs. This measure differs from the number of *images* retrieved from a set of partial duplicate images. F-measure was computed as the harmonic mean of precision and recall.

In order to get the final clustered set of dupilcate images, post-operations, such as connected components and query expansion, should be performed, since min-hash based methods only retrieve a subset of pairs of images in a group of duplicate images probabilistically and does not complete the clustering. We did not include the post-operations in our evaluation and evaluated only the pairs that it retrieved, as it allows a more direct comparison of the various min-hash methods themselves, which is the focus of our study.

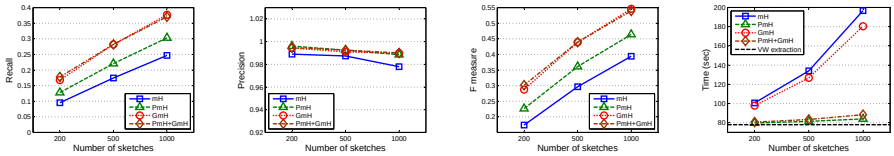
Experiments were run on a single 32-bit machine with a 3.2 GHz processor and 4GB memory.

4.2 Results on Our Dataset

We have empirically tested the effect the number of partitions per image and the area of overlap of neighboring partitions, by varying them from 16 to 144 and 0% to 80%, respectively. An overlap area of 50% gave the best recall. Recall tends to increase as the number of partitions increase. The time taken was the least between 64 and 121 partitions. We have chosen 100 partitions per image and 50% overlap for the following experiments. We have used two hashes per sketch in our experiments.

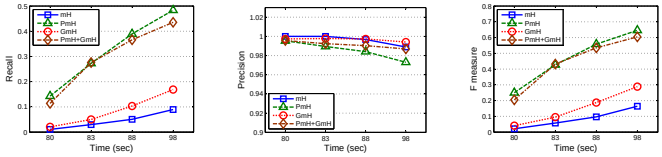
We have tested and compared four methods: standard min-hash (mH), partition min-hash (PmH), geometric min-hash (GmH) [4], and the combination of min-hash and geometric min-hash (PmH+GmH). They were compared under two scenarios: constant number of sketches per image (Figure 5) and constant runtime (Figure 6). The first scenario allows us to evaluate how discriminative the sketches are for each method, and how long it takes to compute a single sketch. The second scenario allows us to evaluate how our proposed method compares given the same computational resource.

In the first scenario (Figure 5), all hashing schemes have high precision, with PmH being slightly better than mH. In the mean time, PmH improves the recall by more than 20% when compared to mH. The speed of the hashing process of PmH is 16 times faster than mH. When our partition scheme is applied to GmH, speed improves by 9 times. When the time for extracting visual words is added, PmH is 2.5 times faster than mH, and PmH+GmH is 2 times faster than GmH, at 1000 sketches/image. At 1000 sketches/image and 2 min-hash/sketch and assuming 1000 features per image, min-hash requires about two million table look up operations, which is a significant amount of



(a) Recall (b) Precision (c) F-measure (d) Time

Fig. 5. Performance on our dataset with fixed number of sketches per image. F-measure is the harmonic mean of precision and recall. Time scale starts around the time it took for extracting visual words from images, denoted by "VW extraction."



(a) Recall (b) Precision (c) F-measure

Fig. 6. Performance on our dataset with fixed time budget.

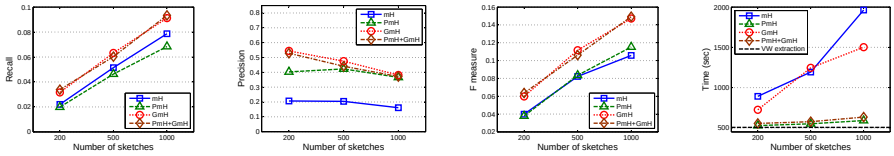
computation. Partition min-hash reduces this number of look up operations by a factor proportional to the number of partitions. Furthermore, the overall improvement in speed becomes more significant when more sketches per image (n) and more min-hashes per sketch (k) are used and min-hash operation contributes more to the overall execution time. It is beneficial to use a greater number of sketches and min-hashes, because it approximates the ideal step function better, which was discussed in Section 2, and leads to better performance. The constraint that limits the number of sketches and min-hashes is the computation time, and partition min-hash alleviates this constraint.

In the second scenario (Figure 6), PmH and PmH+GmH have much higher recall than mH and GmH. When allowed to run for 5 seconds, our PmH has 9.1 times higher recall than mH, and PmH+GmH has 9.2 times higher recall than mH, while GmH has 1.7 times higher recall than mH (mH: 3.0%, PmH: 27.2%, GmH: 5.0%, PmH+GmH: 27.6%). All of the hashing schemes have high precision (mH: 100%, PmH: 98.9%, GmH: 99.7%, PmH+GmH: 99.2%). As we can see, given a fixed time budget in real applications, the speed up of our partition min-hash leads to significant improvement in recall.

Figure 9 shows a sample set of images having min-hash sketch collisions using PmH+ GmH with 500 sketches and 100 partitions per image with 50% overlap. In a typical application these collisions are used as seeds to complete the image clusters using query expansion [4].

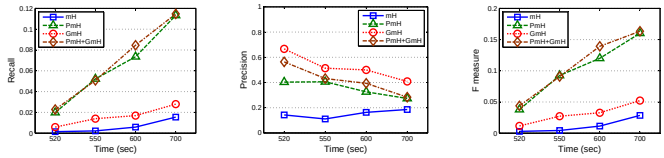
4.3 Results on Oxford Buildings Dataset

We have performed the same experiments on the Oxford building dataset. Figure 7 shows results with fixed number of sketches per image. On this dataset, the performance of min-hash is low, with a particularly low precision of about 20%. With such low precision, the improvement made by Partition min-hash is more pronounced—the precision improvement over mH is 200%. The speed improvement is also significant, consistent with our own dataset. For hashing, PmH runs 17 times faster than mH, and



(a) Recall (b) Precision (c) F-measure (d) Time

Fig. 7. Performance on Oxford buildings dataset with fixed number of sketches per image. Time scale starts around the time it took for extracting visual words from images, denoted by "VW extraction." (100ms per image was assumed)



(a) Recall (b) Precision (c) F-measure

Fig. 8. Performance on Oxford buildings dataset with fixed time budget.

PmH+GmH runs 11 times faster than mH. Our approach also speeds up GmH by 7 times for hashing, without losing precision or recall. When time for extracting visual words is added, PmH is 3.3 times faster than mH, and PmH+GmH is 2.4 times faster than GmH, at 1000 sketches/image.

Figure 8 shows results with fixed runtime. With the same amount of computational resource, PmH and PmH+GmH achieves significant improvement in recall (mH:0.6%, PmH: 7.4%, GmH: 2.3%, PmH+GmH: 8.5%, Time: 100sec).

4.4 Scalability On Six Million Images

To demonstrate the scalability of our method, we applied PmH+GmH to search for all partial-duplicate matches in a dataset of six million images collected from the web. The method took 131 minutes to run on a single 3.2GHz machine with 4GB memory, with the following parameters: 16 partitions per image with 50% overlap and 16 sketches per image. Our method was able to retrieve many partial duplicate images, however, since we have no ground-truth available for this image corpus, we do not present quantitative results other than timing information.

5 Geometry Sensitive Hash

A typical application of minHash is to use the collisions as cluster seeds that will be expanded (by image retrieval) into complete clusters of partial-duplicate images [4]. In doing so, it is important to reduce false positives in these seeds before they are verified by full geometric verification [4], especially for large scale data set where the number of false positives tends to increase.

The local geometric structure of features in duplicate regions is usually preserved across images. For example, in Figure 1, the top of the Eiffel tower is always above the base, and the word "Starbucks" is always above the word "coffee". Instead of verifying

Fig. 9. Example images with sketch collisions. Left: Our dataset. Right: Oxford buildings dataset

these local geometric relationships after sketch collisions are retrieved, we encode such local geometric structure into the sketches, so that they can be checked at an earlier stage. This reduces the number of false positive collisions, and therefore reduces the number of full geometric verifications that need to be performed after expansion by image retrieval, saving computational expense.

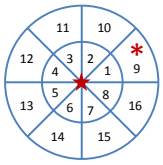
We encode geometric structure into the sketches by hashing the geometric relationship between features. This is achieved by creating an integer ID which encodes the relative geometric configuration[1] among visual words in a sketch, and concatenating it to the min-hash sketch. We call this Geometry Sensitive Hashing. There are many ways to hash the local geometric structure using the relative location and scale of features. We use a simple hash function to quantize the geometric structure into 32 IDs, 16 for the relative position of two features (2 along the radial direction (near, far), and 8 along the tangential direction), and 2 for relative scale (Fig. 10(a)). When there are more than two features in a sketch, hashes for all combination of pairs are concatenated.

Evaluation of Geometry-Sensitive Hashing Figure 10(b) shows the number of true/false positives when applying Geometry-Sensitive Hashing(GSH), given 500 sketches per image. It shows that GSH decreases the number of false positives for all 4 hashing schemes with a negligible computational overhead. We have also observed that GSH is more effective for PmH than for GmH in reducing the number of false positives.

6 Conclusion

We have proposed two novel improvements to min-hash for discovering partial duplicate images in a large set of images: partition min-hash and geometry-sensitive hash. They are improved hashing functions which make use of the geometric configuration information available in images, and take advantage of the fact that duplicate regions

[1] The scale and dominant orientation output by feature detectors can be used to normalize the coordinate system at each point to derive the relative configuration.



(a)

	Without GSH			With GSH		
	TP	FP	Time (s)	TP	FP	Time (s)
Standard	3132	40	56.0	2731	9	55.8
PmH	3963	30	3.4	3617	9	3.5
GmH	5065	46	48.8	4778	32	48.9
PmH+GmH	5066	38	5.4	4744	26	5.5

(b)

Fig. 10. (a) Geometry sensitive hashing (sketch size = 2): a grid is defined centered at the first visual word in the sketch. The second visual word * has a grid id of 9, which is used as part of the hash key. (b) Evaluation of GSH. TP/FP: number of retrieved true/false positive pairs.

are localized in an image and that geometric configurations are preserved across duplicate regions. The methods are easy to implement, with few tuning parameters. We have shown that the proposed hashing method achieves higher precision and recall and runs orders of magnitude faster than the current state-of-the-art. We have also shown that the speed-up allows us to afford a larger number of sketches, which in turn improves the hashing performance, given the same amount of computational resource. Although we have shown the effectiveness of partition min-hash in the domain of images, this method may be applicable to other domains where min-hash is used, such as duplicate document detection, if similar locality properties exist in those domains.

References

1. Broder, A.Z.: On the resemblance and containment of documents. (In: In Compression and Complexity of Sequences (SEQUENCES97)
2. Chum, O., Philbin, J., Isard, M., Zisserman, A.: Scalable near identical image and shot detection. In: Proc. of the Int. Conf. on Image and Video Retrieval. (2007)
3. Chum, O., Philbin, J., Zisserman, A.: Near duplicate image detection: min-hash and tf-idf weighting. In: Proceedings of the British Machine Vision Conference. (2008)
4. Chum, O., Perdoch, M., Matas, J.: Geometric min-hashing: Finding a (thick) needle in a haystack. In: CVPR. (2009)
5. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV. (2003)
6. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
7. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV. (2008)
8. Wu, Z., Ke, Q., Isard, M., Sun, J.: Bundling features for large scale partial-duplicateweb image search. In: CVPR. (2009)
9. Grauman, K., TrevorDarrell: Pyramid match hashing: Sub-linear time indexing over partial correspondences. In: CVPR. (2007)
10. Jain, P., Kulis, B., Grauman, K.: Fast image search for learned metrics. In: CVPR. (2008)
11. Ke, Y., Sukthankar, R., Huston, L.: Efficient near-duplicate detection and sub-image retrieval. In: in Proc. of ACM Int. Conf. on Multimedia. (2004)
12. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large databases for recognition. In: IEEE CVPR. (2008)
13. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **20** (2003) 91–110
14. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. PAMI **27**(10) (2005) 1615–1630
15. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proc. of ACM symposium on Theory of computing. (1998)