Large Scale Image Deduplication

Tzay-Yeu Wen
Stanford University
tywen@stanford.edu

Abstract

With the rise of the Internet and personal digital camera, it becomes easy for researchers to get image data in mass quantity. With these large amount of image data coming from different sources on the Internet, quality of the image is becoming a major problem. Many Images from the Internet are not original but are produced by cropping or transforming from another image. Those images add little information to the dataset and should be removed, but the large amount of data make it infeasible for humans to examine each image and find the duplicate counterpart. Therefore it is crucial to develop algorithms that is both efficient and scalable to process large amount of data.

In this paper, we use an efficient and scalable method that can find near-duplicate images in an image dataset. In order to take advantage of the traditional information retrieval methods, that are designed for large scale retrieval, we represent each image using visual word. However, the visual word approach lost all the information about the geometry structure of the image, which generate many false positive when the dataset become larger. We use a method proposed by [13], that groups visual words using local image feature to increase the discriminability.

1. Introduction

Near-duplicate image detection is a special kind of image retrieval problem, which is relatively easy and well studied compared to other computer vision problems. Several image features, for example [7] and [3], have been proposed to calculate the similarity of two image or image parts. Those features are robust to noise and many image transforms; Therefore are more than enough for duplicate image detection.

Image representation aggregated features into Bag-of-Words [2] can further increase the accuracy of large dataset image retrieval. Using SIFT BoW, [9] has proposed a method which will find similar images in the image dataset. Their experiments showed that their method maintains high accuracy for up to 1M of images.

[1] try to use min-Hash, a method borrowed from text retrieval, to determined the similarity of two bags that contain similar words. [13] proposed another features aggregate algorithm that can take advantage from both local and global features. It use multiple bags per image and calculate the similarity as the sum of matching bags. On top of the bag-of-word representation, it also consider the geometry relation between words inside a bag.

The challenge however, is to be able to handle massive amount of images using reasonable computation resources. The amount of data an image retrieval algorithms, like K-means or KNN, can processed are constrained by the amount of available memory. Therefore many methods that can reduce the memory footprint or scale to multiple machines had been proposed.

[4] proposed a method that reduce the feature representation of each image into less than 100 bytes. This increase the limit on a single machine but the result is less accurate.

Another approach proposed by [6] is to compute the approximate nearest neighbor on features using Map-Reduce, which can process large amount of data with less accuracy.

2. Feature Extraction

2.1. Bundle Feature

For each incoming image we will build a discriminative feature representation of it. We want the feature to be invariant to some image transformations including rotation, illumination, scale and crop, because those are the most common methods used by people when processing images.

SIFT, which is invariant to rotation and scale, is one of the robustest point feature in computer vision. It can achieve very high precision on small dataset. However, when the dataset become larger, false positive increase rapidly. To counter this problem, an intuitive method is to increase the feature space by combing multiple SIFT features into a feature bundle. Two bundles are consider matched if matched SIFT features exceed certain threshold.

As one can imagine, how the features are grouped can greatly affect the performance of the algorithm. Commonly used clustering algorithms like K-means, which group fea-

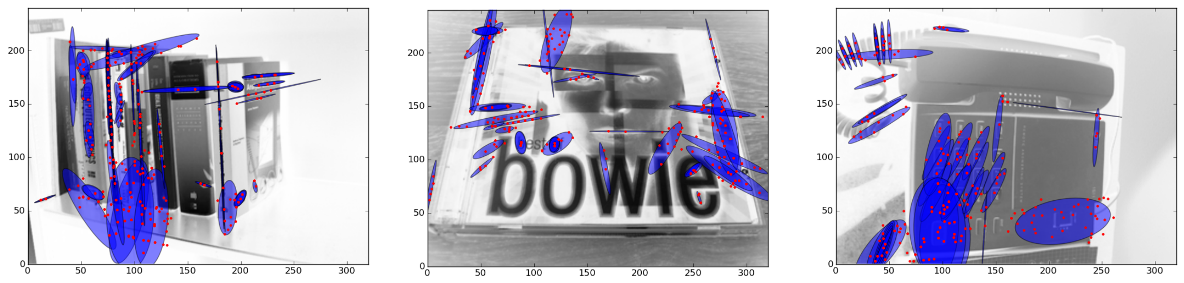


Figure 1. Extracted Features. Blue regions are the MSER features and red points are the corresponding SIFT features

tures by some distance metrics, are not suitable for this problem due to the following reason. First, because of different image depths and object occlusion, nearby feature may not belong to the same objects. The resulting bundle will depend on multiple objects and will be harder to match when the image is cropped. Second, it is hard to determined how many clusters each image should have, the number may vary greatly between complex images and simple images. Finally, even the image are similar, depending on different initial condition feature might not be grouped consistently. Therefore, a better clustering algorithm that is consistent across different images and can consider image context will be preferable.

We use Maximally Stable Extremal Regions (MSER), which detects affine-covariant stable regions in an image, as our clustering algorithm. This method insure that the consistent regions can be detected when cropping, rotating, or translating is applied. It also can handle illumination change provide that the variance of contrast remain small.

2.2. Construction

In this section we will describe how to build the bundle image features using SIFT and MSER. For each image I_i, extract the SIFT features $S_i = \{s_{ij}\}$, where $s_{ij} = (x, y, \vec{f}),\ \ x, y \in \mathbb{R},\ \ \vec{f} \in \mathbb{N}^{128}$, and the MSER features $M_i = \{m_{ij}\}$, where $m_{ij} = (x, y, c_{xx}, c_{xy}, c_{yy})$ and c_{xx}, c_{xy}, x_{yy} are the covariance of the region. Finally, define the bundles for image I_i as $B_i = \{b_{ij}\}$.

We define the bundle feature

$$b_{ij} = \{s | s \in S_i \text{ and } s \text{ is inside region } m_{ij}\} \qquad (1)$$

The inside of m_{ij} is define by the ellipse, which is an approximation of the actual region, calculated from the covariance terms. We discard those bundles that has it's ellipse width or height larger than half of the image width or height because it is very hard to reproduce the same region in two different images. We also discard those bundle that has no or only one SIFT feature because this indicates that the region is too small and can potentially match to many other regions. In order to save storage and computing power, we also discard bundles that has it's SIFT features overlap with another bundles by more than 97%.

3. Image Matching

3.1. Visual Word

To retrieve large amount of image from the database, we need an efficient image representation that can be easily processed and indexed, compare to SIFT feature. The state of the art in image retrieval is to model image as document and image feature as visual word. We use the method propose by [10] to convert SIFT feature into visual word. It calculates the center of each visual word using hierarchical K-means [8]. After the center is calculated, we assign each feature to the first k nearest visual words to reduce the quantization error.

3.2. Hamming Embedding

To further reduce the quantization error, we use Hamming Embedding(HE) to calculate the similarity between features that are assigned the same visual word. This method assign an extra Xbits signature $\vec{he}$, which represent the relative position of the feature in the cluster, for each SIFT feature.

$\vec{he}$ is calculated using the following steps.

1. Build a random matrix $G \in \mathcal{R}^{128\times128}$ and apply QR factorization on it to get Q, an orthonormal matrix.

2. Build the transform matrix M using the first X rows in Q. Apply M on SIFT feature $\vec{f}$ will project it from the original 128 dimensions space to a X dimensions space.
$$\vec{h}_i = M \times \vec{f}_i,\quad M \in \mathcal{N}^{X\times128}$$

3. Let V be the visual words set and v_i be the visual word assigned to $\vec{f}_i$. Calculate the median value $\vec{\tau}_s$ of each cluster.
$$\vec{\tau}_s = \text{Median}(\ \{\vec{h}_i | v_i = v_s\}\ ),\quad v_s \in V$$

4. Finally, calculate $\vec{he}$, the signature, using $\vec{\tau}_s$ where v_s is the visual word assigned to $\vec{f}_i$

$$\vec{he}_i[j] = \vec{\tau}_s[j] < \vec{h}_i[j], \quad \text{for } j \text{ in 1 to } X$$

Given the signature $\vec{he}$ the similarity between two features is calculated by the hamming distance of the signature.

$$\text{Sim}(\vec{f}_i, \vec{f}_j) = \text{HammingDistance}(\vec{he}_i, \vec{he}_j)$$

3.3. Bundle Matching

In this retrieval framework, what we actually do is matching bundles rather than images because the similarity between two images is simply calculated by summing the similarity between each pair of bundles in these two images.

$$S(I_i, I_j) = \sum_{b_1 \in I_i} \sum_{b_2 \in I_j} S(b_1, b_2) \tag{2}$$

Where b_1, b_2 are bundle from the first and second image, respectively. The bundle similarity is calculated by the multiplying the standard information retrieval ranking function by the correlation between bundles. We have tried different ranking functions, amount them BM25 gave the best result.

$$S(b_1, b_2) = Corr(b_1, b_2) \sum_{s \in b_1 \cap b_2} \text{RankingFunction}(s) \tag{3}$$

The correlation term is composed by similarity and locality.

$$Corr(b_1, b_2) = Corr_s(b_1, b_2) + \lambda Corr_l(b_1, b_2) \tag{4}$$

Where $Corr_s(b_1, b_2)$ is simply the number of visual word these two bundles have in common.

$$Corr_s(b_1, b_2) = |b_1 \cap b_2| \tag{5}$$

Currently this number is not normalized but we are experimenting with different normalization schemes, such as $max(|b_1|, |b_2|)$ and $max(area(b_1), area(b_2))$, to see if there are improvement to make.

We assume that the extracted regions are the invariant part of an image that are stable to image translation. Therefore, the feature location inside two matching bundles should be consistent, which is taken into account by the second term $Coor_l(b_1, b_2)$. A high score means that the feature inside these two bundles are positioned on similar location. The term is defined by

$$Corr_l(b_1, b_2) = \sum_{i=1}^{|b_2|} O^D_{b_1}(s_{2,i}) < O^D_{b_1}(s_{2,i+1}) \tag{6}$$

Where $O^D_{b_1}(x)$ is the order of feature x in bundle b_1 with respect to a defined geometric order D. To make the locality term invariant to rotation, we ordered the feature by θ, which is it's position (θ, r) on the polar coordinate.

4. Experiment

4.1. Dataset

We will evaluate our method using two different image dataset, one for accuracy and one for performance.

For accuracy measurement we will use a dataset produced by [9], which contains 2550 distinct images. For each image we will generated 3 images by randomly combine translation, cropping and rotation, which will result in a total 10200 images dataset. We will query the database using all the 10200 images and calculate the accuracy accordingly. Following [13] we will use mAP as our evaluation metric.

For performance measurement we use a dataset provided by ILSVRC2010, which contains 100 categories, 1.2M images, and 126GBytes of data. We indexed the whole dataset with 100K visual words and produce the final representation file with size around 26GBytes.

We use three different query image sets, which are produced by randomly select 5 images from each category and randomly cropping out 10%, 30%, 50% of area respectively, to evaluate our system performance on different level of occlusion. We also record the CPU and memory usage.

4.2. Result

As Figure 4 shows the locality term plays a important role on overall performance. Compare to the original paper [13], which define the geometric order on X-Y coordinate, our method have a much higher optimal λ. This suggests that in our experiments the local term can more accurately represent the similarity between images, which confirms that rotation invariance can improve accuracy.

We also experiments with different visual words size. The original image data contain about 2.6M SIFT features, which are quantized into visual words. The size of visual word affect the performance greatly. If there are too few visual words, false match will increase. On the other hand, too many visual words will make matching impossible because most of the visual words will only map into one SIFT feature.

4.3. Implementation, Runtime and Memory Usage

The experiments were conducted on a Intel i5-2500K machine with 16GBytes of memory. Although I use multiple core to speedup to experiments, all the numbers reported are ran on a single core to reduce other factor that may affect the run time and make it easier to compare with others work.

4.3.1 Query Runtime

The complexity of our query algorithm is linearly proportional to the size of image dataset, but as Table 2 shows, the

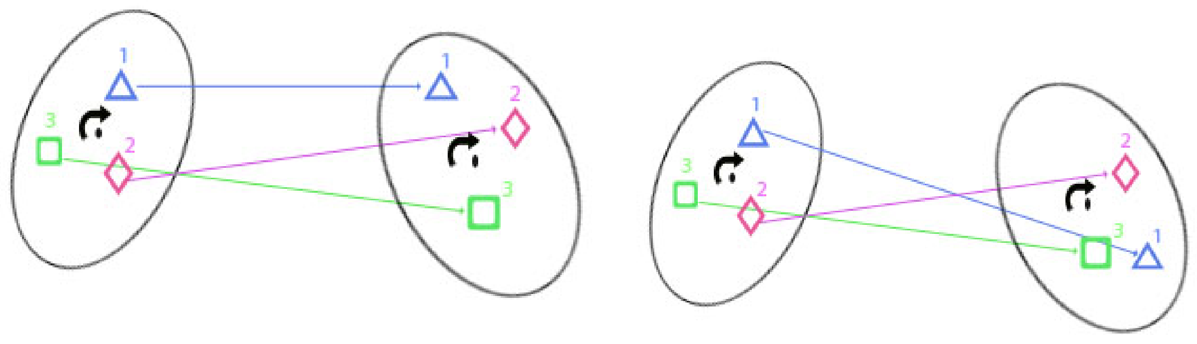


Figure 2. The order of the features are marked with number. Left: These two bundle have the same feature order; therefore get full score on $Coor_i$. Right: The triangle and the diamond are out of order, but the triangle and rectangle are in order.

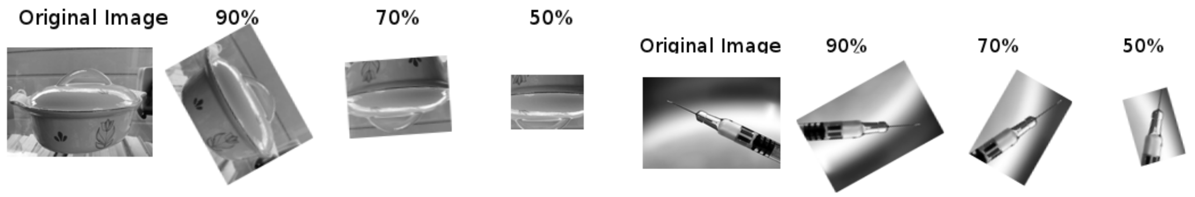


Figure 3. Some sample images and their respective query images

run time only increase about 17 times when the data size is 100 times larger.

There are two major factors that affect the run time. First, when the number of visual word increase, possible matching images for each visual word decrease; therefore, the images we need to consider do not increase linearly. Second, the Hamming Embedding will filter out more false matches when the cluster size of each visual word increase. The result of using HE are shown in Table 4.

We also breakdown the runtime into Feature Extraction, Visual Word Conversion, and Query. The SIFT feature and MSER feature are computed using VLFeat library [12], and use OpenCV [8] for the approximate nearest neighbor search. Table 2 shows the average run time of one image query.

4.3.2 Indexing Runtime

In order to convert the SIFT feature into visual word we need to compute the mapping at index time. This include computing the center of each visual word in SIFT space and the Hamming Embedding median. There are more than 10 billion of SIFT features in the dataset; therefore, we random sample 10M of them as the training data to reduce the runtime and memory usage. We use the hierarchical k-means library from OpenCV [8] and it's runtime is shown in Table 1.

10K Images With 50K VWord	19m46s
1M Images With 100K VWord	239m

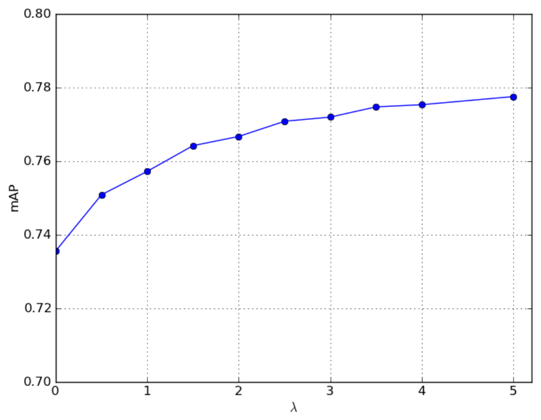
Table 1. Visual Words Clustering Time

	10K Images 50K VWord	1M Images 100K VWord
Feature Extraction	0.066362	0.066362
Visual Word Conversion	0.023341	0.013630
Query	0.033187	0.575602
Total	0.122890	0.655594

Table 2. Query Time per Image Breakdown

4.3.3 Memory Usage

To query the dataset, the only information we need is the mapping from visual word to bundles. In our design each entry in the inverted index table only takes 8Bytes. 4Bytes for the bundle id, 3Bytes for hamming embedding signature, and 1Byte for geometry order. This compact data structure allow the algorithm to store 2^{27} entries in 1GBytes of memory. Using the statistics from the 1M images dataset, which has average 91 bundles per image and average 9 feature per bundle, we can store around 164K images data in 1GBytes memory.



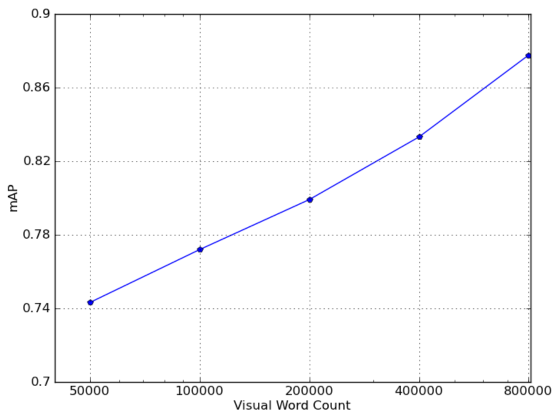


Figure 4. Left: Comparison of different λ term with visual words size 100K. $\lambda = 0$ means only the $Corr_s$ term is used. Right: Comparison of different visual words size with $\lambda = 3$. The original feature size is about 2.6M.

	10K Images 50K VWord	1M Images 100K VWord
Query	0.043012	1.331271
Query + HE	0.033187	0.575602
HE Hit Rate	0.559185	0.449681
Speed Up	22%	56%

Table 3. Query Time per Image w/ and w/o HE

	1M Images mAP
50%	0.434842
70%	0.689659
90%	0.800570
100%	0.912014

Table 4. Query accuracy for different image dataset

5. Conclusion and Future Works

5.1. Global Weak Geometric Validation

Currently, we only consider the geometric location of each feature inside one bundle. As [13] and [11] show, full geometric validation (i.e. re-ranking) can improve the retrieval performance significantly, but with a high computational cost.

In the future, it is possible to extend our weak geometric validation to consider the location of each bundle in the image. This extension have little overhead on the memory usage (around 1Bytes per bundle, or 2% for the ILSVRC2010 dataset), and only increase the computation complexity by a constant factor.

5.2. Feature Detectors

In this paper, we use the standard Difference of Gaussians(DoG) as our feature point detector. DoG has been proven to be a very effective detector, but due to characteristic of DoG and MSER, most of the detected points are near the border of the MSER feature.

Experiments on different detectors and the relation between point and area detectors may give us more insight on how to compose aggregate image representation and it's implication on retrieval performance. [5]

References

[1] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proceedings of the British Machine Vision Conference*, 2008.

[2] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[3] M. S. Extremal, J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from. In *In British Machine Vision Conference*, pages 384–393, 2002.

[4] H. Jegou, M. Douze, C. Schmid, and P. Prez. Aggregating local descriptors into a compact image representation. In *CVPR'10*, pages 3304–3311, 2010.

[5] T. Kadir, A. Zisserman, and J. M. Brady. An affine invariant salient region detector. In *European Conference on Computer Vision*. Springer-Verlag, 2004.

[6] T. Liu, C. Rosenberg, and H. Rowley. Clustering billions of images with large scale nearest neighbor search. In *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, page 28, feb. 2007.

[7] D. Lowe. Object recognition from local scale-invariant features. pages 1150–1157, 1999.

[8] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[9] D. Nistr and H. Stewnius. Scalable recognition with a vocabulary tree. In *IN CVPR*, pages 2161–2168, 2006.

[10] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[11] S. S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, R. Vedantham, R. Grzeszczuk, and B. Girod. Fast geometric re-ranking for image-based retrieval. In *ICIP'10*, pages 1029–1032, 2010.

[12] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.

[13] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 25 –32, june 2009.