Logo Retrieval in Document Images

Rajiv Jain and David Doermann
University of Maryland, College Park, MD, USA
rajivj@cs.umd.edu and doermann@umiacs.umd.edu

***Abstract*—This paper presents a scalable algorithm for segmentation free logo retrieval in document images. The contributions include the use of the SURF feature for logo retrieval, a novel indexing algorithm for efficient retrieval and a method to filter results using the orientation of local features and geometric constraints. Results demonstrate that logo retrieval can be performed with high accuracy and efficiently scaled to a large datasets.**

Key Words: Logo retrieval; SURF; Indexing; Document Images;

I. INTRODUCTION

The most common information retrieval framework used for document images continues to be based on indexing text obtained from optical character recognition (OCR). However, two major drawbacks of OCR are that it cannot process graphical objects and is unable to accurately handle unique or unusual fonts. One of the most important and common graphical objects present in document image datasets are logos. They are prevalent on a number of document image genres including memos, letters, and official documents and can be used to identify organizations or symbols of interest on the page.

There are many challenges associated with performing efficient and accurate logo retrieval in document images. First, documents are often binary images that preclude many texture based features. Second, the binarization of the images adds noise that can distort the original logo. Third, scanned document images are typically high resolution images ranging from 2 to 5 megapixels and logos can be comprised of less than 1% of the document's surface areas. Fourth, all of the current approaches rely heavily on training on the logos of interest. In a typical retrieval scenario, however, the logos being queried for are not known ahead of time.

In recent years, there have been a number of papers about the related topics of logo detection, recognition, and retrieval for document images. Given a document image, logo detection can be defined as the problem of finding a logo's boundary on the page without regard to class. Logo recognition (or matching) on the other hand is the problem of determining which class a given logo belongs to. Logo retrieval can be viewed as a combination of the two problems where one wants to simultaneously detect and recognize a logo across a dataset given some query image.

Logo detection was recently explored by [16], [17] and [18]. In [16], Zhu detects logos on a page using connected component features and a Fisher classifier. Wang uses a decision tree to grow rectangle boundaries around candidate logos in [17]. In [18], Li uses local descriptors found using difference of Gaussians and described using connected component features to detect logos. A comparison of their work is shown in Table 1 on the Tobacco 800 dataset.

Approach	Training Images	Recall	Precision	Speed
Zhu [16]	50	84.2%	73.5%	680 ms
Wang [17]	100	80.4%	93.3%	-
Li [18]	50	86.5%	99.4%	340 ms

Table 1: Logo detection scores

In [20], Rusinol explores efficient logo retrieval on logos by indexing shape context descriptors. He achieves 82.6 mean average precision (MAP) on the Tobacco 800 dataset. Zhu extends his detection work in [8] to build a retrieval system and performs recognition by matching local shape context descriptors. He reports a MAP score of 82.6%. The closest work to ours has been done by Rusinol [2], which performs logo retrieval using a bag of SIFT features. He reports a true positive rate of 90.2 % and a false positive rate of 1%, but the experiments are done on a different dataset that is not publically available making direct comparison difficult.

The main goal of this work is to design a retrieval algorithm that scales to a million image corpora without training on logos of interest. An ideal solution would be to index features extracted from a document as we do with words in text retrieval [21]. Unfortunately image features are often high dimensional and exact indexing remains an open research challenge due to the "Curse of Dimensionality", which causes traditional indexing approaches to perform worse than linear search on high dimensional data [22]. A number of researchers have used approximate nearest neighbor techniques such as KD-trees [4] and locality sensitivity hashing (LSH) [5] to try and address this problem, but they do not scale to large corpora where each local feature is indexed independently because they work best when stored in memory. Others such as [2] use a bag of features framework by assigning each feature vector a codeword. This method also begins to degrade as the vocabulary size grows because of the hard quantization when assigning code words with high dimensional data [15]. We instead chose an approach that indexes feature vectors that are distinct along the same dimensions together.

The paper is laid out as follows: Section II discusses our use of SURF features, section III explains indexing of these features, section IV covers geometric filtering used to improve retrieval performance, section V describes the experiments and section VI analyzes the results.

II. FEATURE EXTRACTION

Our use of local descriptors was motivated by the work of Ke, Sukthankar, and Huston [1], which showed excellent results for the near duplicate image retrieval problem when using the SIFT descriptor. One can imagine logo retrieval in document images as an extension of the near duplicate image retrieval problem in computer vision, where one wishes to

978-0-7695-4661-2 2012

DOI 10.1109/DAS.2012.54

find all similar images that could have been created from simple image transformations such as cropping, scaling, or rotating. Thus local features that are scale and rotation invariant are desirable for logo retrieval because of their ability to match sub sections of images with these transformations. Large affine translations are not a concern for logo retrieval in this work since document images are generally created by scanning on a flat surface. We were further encouraged by the work of Rusinol and Llados [2], which used SIFT features in a bag of features framework to spot logos in binary scanned document images.

We chose the Fast Hessian interest point detector for this paper because the work in [3] shows that it outperforms the Difference of Gaussian and Harris-Laplace interest point detectors while being three times faster to compute per image. The SURF descriptor was chosen instead of the SIFT descriptor because [3] showed that it is more resilient to noise, which often occurs from the binarization process.

The SURF descriptor for a given patch is calculated by first equally subdividing a given patch into a 4x4 grid. For each subsection, the Haar wavelet response Dx and Dy are computed in the x and y directions respectively. The original SURF descriptor calculates the following 4 attributes ($\sum$Dx, $\sum$Dy, $\sum$ |Dx|, $\sum$|Dy|) per interest point. However, the first 2 features $\sum$Dx and $\sum$Dy contain very little information in binary images. Hence they are excluded to form a more compact 32 dimensional feature vector, which is ¼ the size of the SIFT descriptor, without any loss in accuracy.

To build a naïve retrieval system using these descriptors one would first extract and store SURF features from each document image offline. Then at query time one would extract SURF features from the logo, perform a pairwise comparison between all the features extracted from the document and the logo, and then choose the document with the most matches. We will refer to this approach as the brute force method. Given that on average 7000 features are extracted from each document, 500 features are extracted from each logo, and 32 calculations are required for each feature comparison, it becomes quickly apparent that this approach will not scale to large datasets due to its computational requirements.

III. Feature Indexing

A. Generating Index Keys

In this paper we explore a method motivated by a recently proposed indexing technique for near duplicate images [6], which attempts to groups feature vectors that are distinct along the same dimensions together. They define the *distinctiveness* D for a given feature vector *v* as:

$$D(i) = |v_i - \mu_i| * \sigma_i \quad (2)$$

Where μ_i and σ_i are the mean and standard deviation for the distribution of position *i* over the feature vector. We found that the method proposed in that paper did not perform well because the equation they use to quantify the distinctiveness was rewarding, instead of penalizing, dimensions with high variance and the direction of the distinctiveness is lost by taking the absolute value. We propose an alternative distinctiveness measure using the Z-score from statistics as follows:

$$D(i) = \frac{v_i - \mu_i}{\sigma_i} \quad (3)$$

Both μ_i and σ_i are computed offline for each of the 32 dimensions in the SURF feature vector using features extracted from randomly selected documents in the CDIP collection. We then create 2 index keys for each feature vector by taking the 6 positions with the highest distinctiveness and the 6 positions with the lowest distinctiveness score and sorting the values numerically. Note there are 3 times fewer index keys than the 6 required for the algorithm presented in [6]. The index is further expanded by using one bit to represent the sign of the Laplacian in the fast Hessian detector and another bit to represent whether the hash came from the highest or lowest distinctness scores. This indexing scheme provides a hash space of 3,624,768 possible hashes.

An example for a 10 dimensional feature vector and an index made of 3 positions is given in Table 2 to help clarify the indexing procedure. Here the index keys become the positions with the 3 highest distinctiveness scores (highlighted in blue) and the positions with the 3 lowest distinctiveness scores (highlighted in yellow) sorted numerically. The first (or high) key is (6, 7, 10) and the second (or low) key is (4, 5, 8).

As with all approximate nearest neighbor algorithms there is no guarantee that two points indexed to the same key truly match. To solve this problem we store a low dimensional representation of the feature vector along with the index key and verify an indexed feature vector falls within a given distance threshold of the query at runtime. To minimize the storage cost and computational requirements of this matching, the SURF feature vector is reduced to 8 dimensions using PCA. This indexing scheme is used to create an inverted index as follows:

Key 1 -> Doc ID -> X, Y coordinates, Orientation, Feature Vector
Key 2 -> Doc ID -> X, Y coordinates, Orientation, Feature Vector

Each index key points to the unique ID for the document it was computed from and its associated feature vector. The X and Y coordinates as well as the orientation of the interest point are stored for geometric filtering discussed in the next section.

X	1	2	3	4	5	6	7	8	9	10
v_i	5	7	3	2	1	9	8	0	6	10
μ_i	5	5	5	5	5	5	5	5	5	5
σ_i	1	1	1	1	1	1	1	1	1	1
$D(i)$	0	2	-2	-3	-4	4	3	-5	1	5

Table 2: Distinctiveness scores for an example feature vector

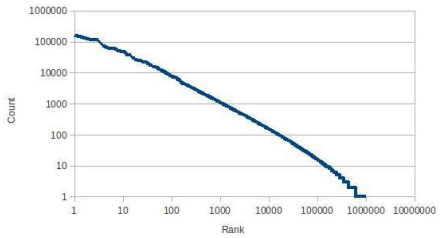

Figure 1: Graph of the index key frequencies sorted by their rank.

B. Properties of the Index

Figure 1 shows the document frequency of a given hash for a set of 1000 scanned documents and approximately 7 million interest points. The hashes clearly follow a power law distribution and this phenomenon has been noticed in several previous papers using local descriptors [14]. In this case, the most frequent hashes appear to be associated with straight lines, which occur very frequently throughout the dataset. The 1000 indexes with the high frequency are put on a stop wordlist because these points are not discriminative and occur several times in most documents. This removes approximately 20% of the interest points from the index as well as significantly speeds up query performance since indexes with the largest number of entries take the longest time to load from disk.

This indexing scheme is designed to reside on disk. Each entry in the index is 19 bytes (6 bytes for the document ID, 4 bytes for the X, Y coordinates, 1 byte for the Direction, and 8 bytes for the Feature Vector). Thus an average image with 7000 features, each with 2 entries, requires approximately 266Kb of disk space. Once the high frequency hashes are removed, this is reduced to 212KB of disk space per image.

IV. Filtering using Geometric Consistency

Image retrieval systems built on indexing local descriptors have traditionally used RANSAC [12] to perform geometric verification. Others such as [13] have used Hough transforms for the same purpose because RANSAC performance degrades if a significant portion of matching features are outliers. Both of these approaches are designed to identify a valid 3D pose for object recognition, but are not necessarily the best fit for document images where transformations in a 2D plane are of concern.

Since affine transformations are not a concern, a much simpler 2-step approach is used. The first step takes advantage of the orientation information provided by interest points found using the fast hessian detector. The orientation difference of valid matching points between a logo query and document image should be relatively constant and equal to

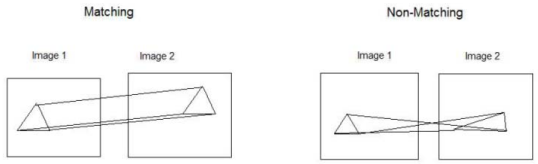

Figure 2: An illustration of the triangle filter.

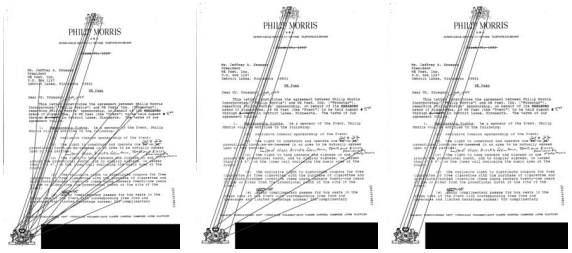

Figure 3: (a) no filters, (b) orientation filter, (c) triangle filter

the skew between the images. Thus, the orientation of each query interest point is subtracted from all matching interest points in the database and normalized to fall within 0 and 360 degrees. For a given image with matching interest points, a sliding scale of 6 degrees is used. Interest points that fall within the window with the largest number of matches are kept and the rest are discarded. In cases of images with erroneously matched interest points this can significantly reduce error rate. The sliding window is trivial in cost and can be programmed on the order of O(n), where n is the number of matching points. Note how the number of false matches is significantly reduced in Figure 3b.

The second step uses a stricter filter, but with the tradeoff that its computational cost is $O(n^3)$. Triangles are computed from all combinations of 3 matching points between the query and document image. Given paired triangles in the query and document image, the difference between the corresponding angles is computed. If the angles differ by 3 degrees, the triangle is ignored. Features that are a part of at least 2 valid triangles are retained and the final score returned by this step for ranking results is the number of matching triangles. Similar approaches have been taken by [23] and [24]. Figure 2 illustrates this triangle filter. Figure 3 shows how these two filters remove false positives.

To limit the effect of a large number of matches on the computation of the triangle filter, the 25 matches with the smallest distances are stored per image before applying the second filter. To reduce the cost of the triangle filter in a large scale implementation, one could randomly sample the set of all triangles. However, in practice we found this filtering to be nominal in cost because there were rarely more than 10 erroneous matching points after the first filter was applied, so all triangles were sampled in our implementation. While efficiency is always a concern, the filtering can afford to be more expensive than the feature matching because only the top results need to be verified and these calculations can be offloaded onto the client's machine in an actual system.

Figure 4: 15 sample pages from the Tobacco 800 dataset

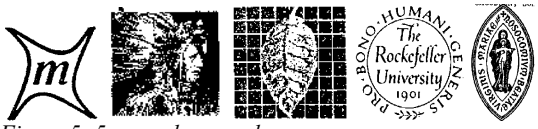

Figure 5: 5 example query logos

V. EXPERIMENTS

A. Tobacco 800 Dataset

The UMD Tobacco 800 dataset ([9], [10], [11]) is an 800 document/1290 page subset of a CDIP 7 million document/42 million page dataset received from the tobacco company lawsuits. All images have been scanned in binary format and range in resolution from 150 DPI to 300 DPI. Figure 4 shows how noisy the images can be as a result of the binarization. It has become the standard public dataset for work on logos in document images. Ground truth labels of the logos were created in ([7], [8]) and only consist of the graphical portion of the logo. The dataset contains 35 unique logos classes across 435 pages. Only logos classes with 2 or more occurrences are used as query images in our experiments. In our experiments we resize each image be have a greatest dimension of 2000 pixels or 180 DPI to reduce the number of features generated for images with higher resolution.

B. Evaluation Metrics

Average precision is a standard metric in information retrieval that combines precision and recall into one score. It can be calculated by Equation 3 where n is the number of results, P(k) is the precision at rank k, rel(k) is 1 if result k is relevant and 0 otherwise, and R is the number of relevant documents in the dataset:

$$Avg\ Precision = \frac{\sum_{k=1}^{n}(\mathrm{P(k)} * \mathrm{rel(k)}}{\mathrm{R}} \quad (3)$$

The score reported in the results for a given system is the mean average precision (MAP) which is the mean of the average precision scores across all queries. A few logos are disproportionately represented in this dataset so the MAP score is also computed by taking the average across all classes as well as all queries. Queries are submitted for each of the ground truth logos provided by [7] against all 1290 pages of the Tobacco 800 dataset. Examples of these logos can be seen in Figure 5.

VI. RESULTS

A. Results on the Tobacco 800 dataset

The following 3 configurations of the system were tested and results are in Table 3: Brute force searching, indexed search with geometric verification, indexed search with without geometric verification.

The results using the graphical logo alone were lower than expected. A close inspection of the results showed that the system was operating with high precision for all of the logos, but noisy logos that were heavily impacted by the binarization or small logos that comprised of a small portion of the entire page for which very few features were extracted had very low recall. One positive result from this data was that there was only a 10-14% drop in the MAP score between the brute force query and the indexed query. Most of this loss was due to a loss in recall from fewer matching points. Another positive result from this data was that the geometric verification significantly improved the results by increasing the MAP by 17-22%. This was largely due to the increase in precision.

System	MAP per logo	MAP per query
Brute force	.67	.59
Index with geometric verification	.57	.45
Index without geometric verification	.35	.28

Table 3: Results on the tobacco 800 dataset for graphical logos

B. What is a logo?

The logo queries chosen from the ground truth of [7] omit contextual text from the logo when possible to limit the test set to graphical objects. However, in reality for each logo there is almost always uniquely identifying titles or text blocks adjacent to the logo that could be used as part of a query image to boost performance. In many cases the text is more consistent, prominent and distinct then the logo. Three more image queries are run on the Tobacco 800 dataset using the indexed search with geometric verification to compare how the contextual text surrounding the logo affects performance: Logo alone, Text alone, Logo + Text. The logos are reused from the prior experiment and the Text and Logo + Text images are manually extracted for each page containing logo. The MAP per query and MAP per logo class are again used as metrics for performance. Examples of the Logo, Text, and Logo + Text images can be found in Figure 6.

The results in Table 4 show a significant improvement gained by combining the textual and logo information and indicate that graphical objects should not be isolated from its surrounding context when performing logo retrieval on document images. For some documents, logos contain the most distinctive features and for others, the text surrounding the logos is more distinctive. By combining the two, the image query algorithm benefits from have more information and more descriptors. Since this the algorithm operates with high precision the additional text descriptors do not result in many more false positives. One exception was the "Philip Morris" text image, which found several other document images that contained the words but not the logos.

Logo	Text	Logo + text
	THE AMERICAN TOBACCO COMPANY	 THE AMERICAN TOBACCO COMPANY
	HARVARD MEDICAL SCHOOL	HARVARD MEDICAL SCHOOL 

Figure 6: Examples of the text context found with logos

System	MAP Score per logo class	MAP Score per query
Logo only	.57	.45
Text only	.56	.63
Logo + text	.87	.88

Table 4: Results for graphical and text logos

C. *Efficiency tests*

20 logo queries were run on datasets comprised of 10, 100, 1290, 10875,108993 images to measure the impact of an increasing database size on query performance. The features for each of these datasets were indexed and geometric verification was performed for the top 100 results. The query images had about 500 features and required about 1000 database lookups per query. The query performance was tested on a single 2.67 GHZ CPU with the index residing in memory, on a solid state drive (SSD), and on a traditional 5400 RPM hard drive to measure how different architectures effected performance using. The Linux disk cache was cleared prior to each query to ensure that the results were not skewed by the operating system. Code optimizations and system configurations could possibly further improve these results. The results are shown in Table 5 and do not include a constant time of 500 ms required to start the program and extract features from the query image. The brute force search took approximately 45 minutes for 1 query to run on 1290 images and clearly could not scale to a large dataset so it was not rerun for each of these datasets.

As expected, the results show that storing the index in memory is the most efficient. Given the high cost of memory, however, it is unlikely to be used in a large scale implementation. The hard disk was expected to have the worst performance because of the 10 ms random seek time. The index was laid out on disk sequentially and accessed in sorted order to minimize the number of disk rotation and head movements to achieve better then random seek times. The SSD used in this experiment had a random seek time of .2 ms and was thus expected to significantly outperform the hard disk. However, it appears to only be about 4X faster on the largest dataset, making it hard to justify its higher cost. Given these results it appears that the best approach to scaling would be to distribute the database across many traditional hard drives to reduce the cost of disk seek time in order to allow the database to scale to millions of images.

		Average query time (ms)		
# of Images	# of features	Disk	SSD	Memory
10	58501	53.9	27.5	5.3
100	577019	277.6	117.1	7.1
1290	7382701	2059.4	190.7	16.4
10887	66165019	3775.8	496.5	77.5
108993	663512013	8661.4	2894.2	751.9

Table 5: Query performance

VII. CONCLUSION

The results clearly demonstrate the effectiveness of SURF features for logo retrieval in document images. The indexing scheme used for this paper is shown to greatly improve efficiency, while only moderately impacting the accuracy of the SURF features. The retrieval results when combining the logos with its textual context performs at the state of the art for the tobacco 800 dataset. For future research we hope to expand this algorithm to the full 7 million tobacco image corpus and compare the performance of image retrieval to OCR.

REFERENCES

[1] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In ACM Multimedia, 2004

[2] M. Rusinol and J. Llados. Logo spotting by a bag-of-words approach for document categorization. ICDAR, 111–115. 2009.

[3] H. Bay, A. Ess, Tinne Tuytelaars, Luc J. Van Gool, SURF: Speeded up robust features, CVIU, pp 346-359,

[4] J. Beis, and D. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces" CVPR, pp. 1000–1006, 1997,

[5] A. Andoni and P. Indyk. "Near-Optimal Hashing Algorithms for Near Neighbor Problem in High Dimensions". FOCS, 2006.

[6] Auclair, A. Vincent, N. Cohen, L.D. Hash functions for near duplicate image retrieval. WACV 2009. Pages 1-6.

[7] G. Zhu, Y, Zheng, D. Doermann,and S. Jaeger. Multi-scale Structural Saliency for Signature Detection. CVPR, pp. 1-8, 2007.

[8] Guangyu Zhu and David Doermann. Automatic Document Logo Detection. ICDAR, pp. 864-868, 2007.

[9] D. Lewis, G. Agam, S. Argamon, O. Frieder, D. Grossman, and J. Heard, "Building a test collection for complex document information processing," SIGIR, pp. 665–666, 2006.

[10] G. Agam, S. Argamon, O. Frieder, D. Grossman, and D. Lewis, The Complex Document Image Processing (CDIP) test collection project, IIT 2006. http://ir.iit.edu/projects/CDIP.html.

[11] The Legacy Tobacco Document Library (LTDL), University of California, San Francisco, 2007.http://legacy.library.ucsf.edu/.

[12] M. Fischler and R. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, 24(6), June 1981.

[13] D. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2) (2004) 91–110.

[14] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In Proc. BMVC., 2008.

[15] J. Philbin et al.. Lost in quanti-zation: Improving particular object retrieval in large scale image databases.In CVPR, 2008.

[16] G. Zhu and D. Doermann. Logo matching for document image retrieval. In ICDAR '09, pages 606–610, 2009.

[17] H. Wang and Y. Chen. Logo detection in document images based on boundary extension of feature rectangles. ICDAR, p. 1335–39, 2009.

[18] Z. Li, M. Schulte-Austum, and M. Neschen. Fast Logo Detection and Recognition in Document Images. ICPR , pages 2716 – 19, 2010

[19] M. Rusiñol, D. Aldavert, R. Toledo and J. Lladós Browsing Heterogeneous Document Collections by a Segmentation-free Word Spotting Method. ICDAR, pages 63-67. 2011.

[20] M. Rusiñol and J. Lladós. Efficient Logo Retrieval Through Hashing Shape Context Descriptors. DAS 2010, pages 215-222.

[21] Baeza-Yates and R. Ribiero-Neto. *Modern Information Retrieval*. Addison-Wesley, Longman, Boston, Mass, 1999.

[22] R. Bellman, *Dynamic Programming*, Princeton University. Press, Princeton, NJ, 1957.

[23] T. Nakai, K. Kise, and M. Iwamura, "Use of affine invariants in locally likely arrangement hashingfor camera-based document image retrieval", DAS, vol. 3872, pp.541–552, 2006.

[24] X. Liu and D. Doermann. Mobile retriever - finding documentwith a snapshot. CBDAR. , pages 29–34, 2007.