Realtime Storefront Logo Recognition

Frank Liu and Yanlin Chen
{liuf, yanlinc}@stanford.edu
Stanford University

Abstract

Storefront brand image (logo) recognition has many useful applications such as location recognition and advertisement. Various methods have been proposed using local and global feature matching. However, it continues to be a challenging area; unlike nature scene images, which are rich in texture details, brand images often lack texture variation, and therefore provide fewer feature points for matching. In addition, reference and test images may be acquired with different rotation, resolution, size, quality, and illumination parameters. In this paper, we propose a highly efficient and accurate approach for recognizing logos in storefront images. We separate our algorithm into two primary phases - representation and recognition, each of which deals with a separate portion of the logo recognition classification algorithm. After training our classifier with 257 pre-segmented storefront logos, we were able to get an 86.0% classification accuracy on 100 test images taken from a wide variety of locations.

1. Introduction

Extensive research has been dedicated to the field of object recognition and various sub-problems within that field have been explored as well. In this paper, we focus on logo recognition, with particular interest in storefront logos. Storefront brand image recognition has many useful applications, such as indoor localization in malls, incentivized shopping (coupon delivery), and mobile advertising. Logo recognition can also greatly improve any modern-day visual search engine, such as Google Goggles. [1]

Although noiseless logo recognition on plain backgrounds has been a well-studied subject for decades, logos that appear in natural scenes are much harder to detect. [13] Despite the ease with which humans can recognize storefront logos, logo detection in computers is incredibly difficult, especially when the input training logos do not contain large pixel gradients or changes in texture. Furthermore, camera-related difficulties often arise in storefront logo classification, including, but not limited to: changes in rotation and zoom, movement of the global camera position, blurring of output image, and differences in illumination.

We therefore formulate our problem as follows: given a series of training images annotated with brand names and an query image, how do we detect all the logos (one or more of the brands) in the query image? During the course of our project, we experimented with various methods for feature detection and descriptors on our storefront image database and propose a complete framework for both logo representation and detection.

The remainder of this paper is organized as follows. Section 2 describes related work relevant in the context of logo recognition and summarized the key main contributions of this paper. In section 3 it is discussed with details how the proposed framework can be used to detect all the logos in the query image. We present experimental results in Section 4, before concluding our work in Section 5.

2. Overview

2.1. Review of Previous Work

Most successful algorithms for correlating a series of test images with training images rely on image matching. Image matching is a long-standing problem in computer vision that has implications in a wide variety of sub-fields, including automated navigation and panorama recognition. [6, 5] A popular technique for comparing and matching images is to acquire a series of preferably invariant feature keypoints and descriptors. There is a huge pool of existing research for keypoint detection, including the Harris corner detector, scale-invariant feature transform (SIFT), and speeded-up robust features (SURF). [2, 3, 4] Due to its invariance under rotation, and zoom, SIFT has developed a reputation as the state-of-the-art feature descriptor for object recognition and image matching. Furthermore, while SIFT features are not invariant under all affine distortions, they are relatively robust under changes in camera zoom and twist, which makes SIFT an ideal candidate for our system.

There is a plethora of recent work involving SIFT features. Brown and Lowe utilize SIFT features for automatic panorama recognition and stitching by jointly optimizing

the camera parameters of different input images using the coordinates of matched sift features. [5] Due to the invariant nature of SIFT features, their system has found lots of success in academic settings, and is often used as the base technique for image stitching in many commercial software suites. Jin et al. found success using a modified version of SIFT (SSIFT) to recognize Chinese characters in complex documents. [8]

Another approach for acquiring invariant features relies on histogram of oriented gradients (HOG) to acquire a set of feature descriptors used for object recognition. [7] HOG has found lots of success in the sub-field of object detection. The intuition for HOG is that local object appearance and shape can often be well-characterized by the distribution of local intensity gradients or edge directions. [7] In practice, the HOG algorithm divides the image window into small spatial regions, or *cells*, and computes a histogram of oriented gradient for each cell, before re-normalizing the cells by looking at adjacent blocks. After tiling the detection window of normalized descriptor blocks, HOG uses the combined feature vector in a SVM classifier (e.g. linear SVM for simplicity and speed). HOG is typically used in select local image patches with normalized histogram of gradient orientations features, whereas SIFT concentrates on generating a descriptor for each interest point found in the image.

2.2. Summary of leading novelties

Most logo recognition work in the past has relied on the standard SIFT descriptor. After trying several feature detection algorithms, we settled on PCA-SIFT, a feature detection algorithm by Ke and Sukthankar. [11] Like standard SIFT, PCA-SIFT descriptors encode the salient aspects of the image gradient in the feature point's neighborhood; however, instead of using SIFT's smoothed weighted histograms, PCA-SIFT applies principal component analysis (PCA) to the normalized gradient patch. In the context of logo recognition, not only are PCA-based descriptors more robust to image deformations, but they are also more compact than standard SIFT descriptors. Using PCA-SIFT results in significant storage space benefits as well - the dimensionality of feature descriptors drops from 128 in standard SIFT to at most 20 for PCA-SIFT. Furthermore, it improves greatly improves the runtime for the feature matching process.

Since the SIFT descriptors are only rotation invariant around z-axis of the image plane [3], we apply several empirical rotations to the x and y axis on the training images in order to increase robustness. During the matching process, a sliding window approach is applied to reduce the number false positives. We further discuss this technique in Section 3.2.4.

Figure 1 provides an example of our systems final set of matches for logo recognition. Note how our algorithms use of the sliding window technique removes all false positives from the logo matching process.

Figure 1: An example of our system's ability to match a logo in a natural image. The bottom image is the actual input, while the top is the database match.

3. Storefront Logo Recognition

3.1. Technical Summary

We separate our algorithm into two primary phases - representation and recognition. The representation phase takes a series of segmented storefront logo's and shrinks them into a large KD-forest database of PCA-SIFT features. Conversely, the recognition phase takes a natural scene image and attempts to find all storefront logos present in the image. A complete overview of our approach is shown in Figure 2.

During the representation phase, we manually segment each individual training image such that portions which do not contain logos are cropped out. The pre-processing stage then takes each input logo and applies a series of transformations in order to make the output PCA-SIFT descriptors as invariant to affine transformations as possible. For simplicity, we will refer to this representation stage as extraction of *affine-invariant PCA-SIFT descriptors*. Once that is complete, our system then builds a KD-forest using the extracted features.

During the matching process, we are given an input query image. In order to improve our recognition accuracy, we first enhance each image by running it through a high-pass filter. We then extract PCA-SIFT features for the enhanced image. During classification, we employ a sliding window technique to reduce the number of false positive cases - if done correctly, the window containing the logo will have the highest number of true positive nearest-neighbor matches.

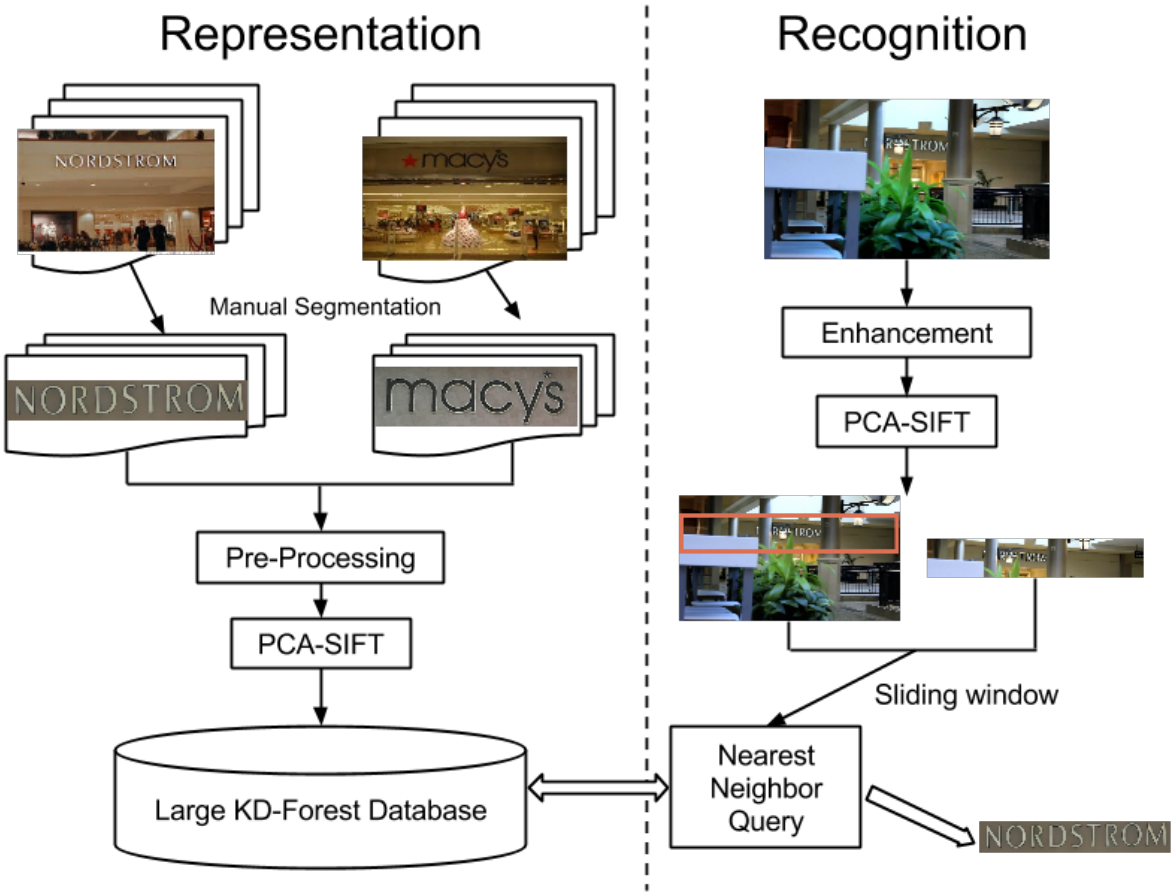

Figure 2: Pipeline of our approach.

3.2. System Details

3.2.1 Training Image Pre-Processing

Although SIFT is often touted as invariant to a wide variety of image transformations, all SIFT (and SIFT-based) features are actually invariant to only four of the six parameters in an affine transformation. [14] Because of this, we decided to first manipulate each input logo to create a larger set of new training images. Morel and Yu did this in their Affine-SIFT (ASIFT) algorithm, which tilts the viewpoint of each input image using a predetermined set of longitudinal and latitudinal rotation matrices. [14] After generating the skewed images, they acquired SIFT features for all of the new images and mapped the coordinates back to the viewpoint of the original logo using the inverse of each of the original homography matrices. Using this enhanced set of SIFT features for each image, they were able to successfully match images of the same scene from a wide variety of different camera angles.

We apply a very similar technique in our system to make each input logo as affine-invariant as possible. During the representation phase, our algorithm pre-processes each training image by adding different levels of x-axis and y-axis (longitudinal and latitudinal) rotation using a set of pre-determined rotation values along the two vertical and horizontal center axes of each image. Our system then acquires features for each processed training image and maps the features back to coordinates of the original image to create a set of super-features for matching.

3.2.2 Extraction of Affine-Invariant PCA-SIFT Features

As mentioned in Section 2.2, we use a PCA-SIFT, a SIFT variant with a smaller descriptor dimension than standard SIFT (128 values per descriptor as opposed to 20 or fewer values per descriptor).

Once all input image PCA-SIFT features have been extracted, our system builds a single KD tree of feature descriptors for each training logo. [12] The KD tree data structure is used to quickly solve nearest-neighbor queries. By

building a KD tree and properly setting the parameters for the feature matching algorithm, we can quickly match features from test images to those in the training image.

3.2.3 Query Image Enhancement

In our initial experiments, we found PCA-SIFT to be somewhat intolerant to image blur. Given that image blur is a common occurrence in images taken on mobile devices, we decided apply an enhancement step in the recognition phase of our algorithm - in this step, each input query image is run through a high-pass filter to reduce any blur that may be present in the image. For sharp images, this step keeps the original image relatively unchanged. For blurry images, this step enhances corners and gradients in the image, both of which are essential for our application.

3.2.4 Sliding Window

Use of a sliding window for logo recognition is a novel idea. Here, we make the assumption that input images taken with a camera are rarely every sideways. In other words, if a storefront logo appears in a query image, it is likely to be oriented in the upright direction. This allows us to apply a series of rectangular masks over the input image with dimensions which approximately correspond to that of the logo in interest. In other words, using pre-determined dimensions of the training logo, our system divides each test image into regions and matches each region to find possible logos. In Figure 3, we show an example of our sliding window technique on a query image.

For each sliding window, we acquire matches to each KD-tree of features in the logo database. The sliding window with the maximum number of matches to each logo is labelled as the *primary window*:

$$s_i = \max_j(m_{ij}), \forall \text{i}, \tag{1}$$

where s_i is the score of the query image with respect to the ith training logo in the database, and m_{ij} is the number of matches between the jth sliding window in the query image and the ith training logo.

If the score of the query image exceeds a certain percentage of the the total number of features in image i, the desired database match, our system labels that query image with a flag that indicates the logos presence. Because our system is designed to be realtime, we do not apply RANSAC to the image matches, as the complexity of RANSAC is more than enough to negatively impact the runtime of our system.

4. Results

To test our algorithm, we gathered a large dataset of 307 images, each of which contained one of six different storefront logos located in the Stanford Shopping Center. We limited the storefront logos to well-known brands, i.e. Bloomingdales, Macys, Nike, Nordstorm, Starbucks, and Urban Outfitters. Out of these 307 images, we selected 250 of them for our training set, and manually segmented them so that only the logo was visible. These 250 images represented our set of *ideal* logos, i.e. ones that we believed to encompass a wide range of illuminations and viewpoints without being too blurry. We then fed these 250 training images through the representation phase of our system. Using a mobile phone, we proceeded to take 100 arbitrary, unsegmented images, each containing one of these six storefront logos, and ran them through the recognition phase of our system. Table 1 provides the recognition results for our system. Tables 2 and 3 compare the performance of our system with and without sliding windows.

We implemented our system mostly in C, making use of OpenCV and VLFeat, two popular open source libraries for general computer vision tasks and feature extraction, respectively. All matrix operations and perspective transforms were performed in OpenCV, while VLFeat was used for extracting PCA-SIFT features. The results of our experiment on an 2.5GHz Intel Core i5, 512KB L2 cache are shown below. We compiled our code with all optimizations enabled (-O3). On average, our algorithm can process a single query image in approximately 935ms, or just under one second. For PCA-SIFT, we used a descriptor dimension of 16.

5. Conclusion and Future Work

In this paper, we have presented an algorithm for storefront logo recognition. We first discuss the challenges and motivation for storefront logo recognition, before discussing existing related work. We then present our recognition algorithm for storefront logos. Despite the difficulty of the task, our proposed system is able to reliably detect multiple logos in input images at an extremely rapid rate. To increase the potential of our algorithm, we employ a database of affine-invariant PCA-SIFT features and make use of sliding windows to improve recognition accuracy. By building a feature database from 257 different training images of known storefront logos, our algorithm was able to achieve an impressive 86.0% accuracy on a test set of 100 images, each containing a single logo. With a larger database of storefront logos and more computing power, our algorithm would be able to recognize a wider range of logos in a shorter amount of time.

Our proposed algorithm is comparable in performance to previous work on pre-segmented vehicle logo recognition by Psyllos etal., which, in comparison to unsegmented storefront logo recognition, is a much easier problem. [5] However, there are still many areas with which we can improve our system, including, but not limited to:

Figure 3: Diagram of our sliding window technique.

Storefront logo	Detected	Not detected	Total	Percent recognized
Bloomingdale's	18	3	21	85.7%
Macy's	11	3	14	78.6%
Nike	17	2	19	89.5%
Nordstorm	9	3	12	75.0%
Starbucks	23	1	24	95.8%
Urban Outfitters	8	2	10	80.0%
Total	86	14	100	86.0%

Table 1: Results for 100 total test images of six different storefront logos. Note the high recognition rate for our algorithm.

Storefront logo	True positive logo matches	False positive logo matches
Bloomingdale's	18	0
Macy's	11	1
Nike	17	0
Nordstorm	9	0
Starbucks	23	0
Urban Outfitters	8	0
Total	86	1

Table 2: Logo recognition results with our sliding window technique enabled. Only 1 false positive logo match occurred (the false positive is factored as "not detected" in Table 1).

Storefront logo	True positive logo matches	False positive logo matches
Bloomingdale's	18	3
Macy's	11	3
Nike	17	2
Nordstorm	9	2
Starbucks	23	4
Urban Outfitters	8	2
Total	86	16

Table 3: Logo recognition results with our sliding window technique disabled. Without sliding windows, false positive matches are much more prevalent.

1. *Occlusion detection* - Occlusions, or foreground objects which partially block the view of the storefront logo (such as people or pillars), are unfortunately very common for storefront logos, and our system currently has no way of directly detecting or removing them. By integrating an occlusion detection algorithm, we can make our system a lot more robust to foreground objects.

2. *Improved recognition for multiple logos* - An input image may contain multiple storefront logos, and correctly recognizing all of them can greatly improve the potential of our system. While our algorithm can, in theory, correctly detect and label multiple logos in a single image, we have yet to rigorously test the system on a group of input images containing multiple storefront logos. By performing these tests, we may be able to better understand our methods strengths and weaknesses, and account for them appropriately. Note that, in our presentation, we had presented some results which included test images containing multiple storefront logos. However, because we did not believe that we had a large enough multiple-logo-per-image dataset, we decided to limit our test set to one-logo images in the end.

3. *Runtime improvement* - With optimized code, our current system can only process frames at a rate of just over 1 frame per second on an 2.5GHz Intel Core i5 with 512KB L2 cache. While this is already very fast, we may need to improve the efficiency of our algorithm for realtime applications with inputs from multiple devices. This can be done by reducing the PCA-SIFT descriptor dimension, or by reducing the number of feature points in our database in the representation phase of our algorithm.

Acknowledgments

We (the authors) would like to thank Hui Chao from Qualcomm for the project idea and sustained support throughout the quarter. We would also like to thank Professor Silvio Savarese and Kevin Wong (our project mentor), along with David Held, Mark Ma, and Chentai Kao for their advice and patience throughout the past three months. We really appreciate the entire teaching staff for the incredible job that they have done this quarter. Thank you for teaching CS231A!

References

[1] Google Goggles. http://www.google.com/mobile/.

[2] Harris, C. and Stephens, M., *A Combined Corner and Edge Detector*, Proceedings of the Alvey Vision Conference, 1988.

[3] Lowe, D.G., *Object recognition from local scale-invariant features.*, Proceedings of the International Conference on Computer Vision, 1999.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. *SURF: Speeded Up Robust Features.* Proceedings of the European Conference on Computer Vision, 2006.

[5] M. Brown and D.G. Lowe. *Recognising panoramas*, Proceedings of the International Conference on Computer Vision, 2003.

[6] A. P. Psyllos, C. N. E. Anagnostopoulos, and E. Kayafas. *Vehicle Logo Recognition Using a SIFT-Based Enhanced Matching Scheme*, Proceedings of Transactions on Intelligent Transportation Systems, 2010.

[7] N. Dalal and B. Triggs. *Histograms of oriented gradients for human detection*, Proceedings of the Conference on Computer Vision and Pattern Recognition, 2005.

[8] Z. Jin, K. Qi, Y. Zhou, K. Chen, J. Chen, and H. Guan. *SSIFT: An Improved SIFT Descriptor for Chinese Character Recognition in Complex Images*, Proceedings of the International Symposium on Computer Network and Multimedia Technology, 2009.

[9] B. Funt, F. Ciurea, and J. McCann. *Retinex in Matlab*, Proceedings of the IST/SID Color Imaging Conference: Color Science, Systems and Applications, 2000.

[10] E. H. Land and J. J. McCann. *Lightness and Retinex theory*, Journal of the Optical Society of America , vol. 61, no. 1, 1-11 1971

[11] Y. Ke and R. Sukthankar. *PCA-SIFT: A More Distinctive Representation for Local Image Descriptors*, Proceedings of the Conference on Computer Vision and Pattern Recognition, 2004

[12] A. Vedaldi, and B. Fulkerson. *VLFeat: An open and portable library of computer vision algorithms.* Proceedings of the International Conference on Multimedia, 2010.

[13] Y. Kalantidis, et al. *Scalable triangulation-based logo recognition*, Proceedings of ACM International Conference on Multimedia Retrieval, 2011.

[14] J. Morel, G. Yu. *ASIFT: A new framework for fully affine invariant image comparison.*, SIAM Journal on Imaging Sciences 2.2 (2009): 438-469.