Image Recognition and Indexing

Chat Fai Geoffrey Mak
Department of Electrical Engineering
Stanford University
cfmak@stanford.edu

***Abstract*—Lately a lot of attention is focused on image search. This work aims at indexing images to build an inverted file system for image retrieval. Each image is indexed by the sift descriptors [1] at its Censure keypoints [2].**

At the client side, a phone camera is used to scan one of the indexed images, extract its sift descriptors, and query the server to retrieve the nearest image (in terms of descriptor space) from the database.

Keywords-component; image search, Censure detector, sift, image indexing, inverted file system

I. INTRODUCTION

Lately a lot of attention is focused on image search. This work aims at indexing images to build an inverted file system for image retrieval. Each image is indexed by the sift descriptors [1] at its Censure keypoints [2].

At the client side, an phone camera is used to scan one of the indexed images, extract its sift descriptors, and query the server to retrieve the nearest image (in terms of descriptor space) from the database.

This paper will first describe the inverted file system structure and its query method. Then the paper will present a review of the Censure feature detection algorithm and Sift descriptor calculation. Lastly, some experimental setup and results will be discussed.

II. INVERTED FILE SYSTEM

A. Definition

A usual (forward) file system is a system that maps a key, i.e. filename, to a value, i.e. the data stored in the file. In contrast, an inverted file system is a system such that it uses the data to map to the filename.

In the case of image search, when the client scans an image, the image is a large chunk of data that is to be mapped to its corresponding filename on the server. If the server recognizes the query image, it returns the filename of the query image to the client.

Because the scanned image is prone to error, for example, white noise, luminance, perspective transform, etc., one should not expect the scanned image to be exactly the same as one of the known images. In fact, this problem is a code word matching problem – the query image is matched to the closest known image (as the code word) based on a distance criterion.

B. Distance Criterion

A set of Sift descriptors (in the dozens) is expected to be associated with each image (both query and known). Each descriptor is a 128-dimensional vector with its norm normalized to 1.

The similarity score between the query image and a known image is based on the number of Sift descriptors that is present in both images. Because of the errors of the query image, the Sift descriptors from the query image are generally not equivalent to those of the known image. Instead, the descriptor from the query and the descriptor from the known image are said to match if their L2 distance is within some epsilon.

If the number of known images is large, for example in the millions, and as a result the number of known Sift descriptors is also large, a K-mean algorithm is usually used to cluster the descriptors to form code words. Each code word will then be associated to multiple images. Any query descriptor that falls into the Voronoi cell is decoded to the code word.

In this work, because the number of images (40) is small, so clustering is not used.

The algorithms to calculate the Censure keypoints and Sift descriptors will be discussed in section III and IV respectively.

C. Inverted File System Structure

An inverted file system is an array of key-value pairs, with the key being the Sift descriptor of all known images and the value being an array of image entry. An image entry consists of an image id and possibly other refinement data, e.g. hamming embedding[3], for improved matching accuracy. If no clustering is done, then each codeword only associate to one image only, and so the array of image id will have only one element; otherwise, one codeword may associate to multiple images, so the array of image id will contain all identifiers of images that have descriptors that is clustered into the same Voronoi cell.

To match a query descriptor to a codeword, a brute force method is used to find the closest codeword. Otherwise, an approximate method can also be used for speed optimization.

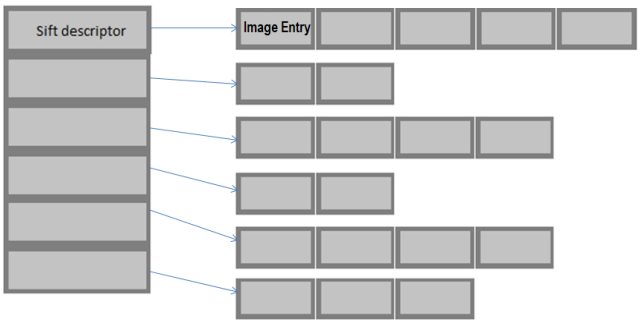

Fig. 1 Inverted file system structure. Each sift descriptor is a 128-dimensional vector. Each image entry consists of a 4-byte image id plus an 8-byte hamming embedding.

III. Censure Keypoint Detector

A keypoint in an image is usually an extrema point of convolving the image with a finite length filter. Some common filters are the Difference of Gaussian (DoG) and Laplacian of Gaussian (LoG). However, the number of arithmetic operations to perform a convolution is expensive. Also, several downsampling is often required to calculate the filter response extrema over several image scales.

Motilal Agrawal et al. suggested using three integral images (one square integral image, two slanted integral images) and an octagon bi-level filter to approximate LoG calculation [2]. Keypionts are found by applying octagon filters of different sizes to an image and finding the locations of extrema. If the extrema is above a certain threshold, the location is considered as a keypoint. An octagon shape is chosen as the filter's shape because it is a close approximation to a circular LoG filter. Seven scales of the filter are applied to the image for scale invariance, with sizes set according to the original paper.

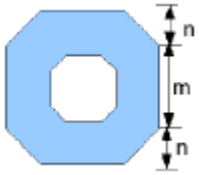

Fig. 2 A bi-level octagon filter used to approximate the LoG. The blue area is positive, while the center white area is negative. The DC response is normalized to 0.

TABLE I. Dimension of FilterS in Pixels

scale	1	2	3	4	5	6	7
Inner (m',n')	(3,0)	(3,1)	(3,2)	(5,2)	(5,3)	(5,4)	(5,5)
Outer (m,n)	(5,2)	(5,3)	(7,3)	(9,4)	(9,7)	(13,7)	(15,10)

Suppose the dimension of the image is w*h, where w is the width and h is the height. Then the recursive calculation of an integral image is O(w*h), and the calculation of the filter response at one location is O(1). Also because no image downsampling is done, it should also give the Censure keypoint detection an edge over Sift detection in terms of speed.

IV. Sift Descriptor

Sift was presented by David Lowe [1] for both keypoint detection and description. In this work, the Censure detector is used in place of the Sift detector to find keypoints. The Sift descriptor is used to describe Censure keypoints. Also the Sift descriptors in this work are all upright, i.e. no orientation was assigned to each descriptor. Instead, for each image, four anchor points are added to the corners. When the client scans the image, the four corners will be used as reference points to do a perspective transform to correct the image to its upright position, so rotation invariance is not a necessary criterion for this project.

The calculation of the Sift descriptor involves finding the statistics, i.e. binning, of the gradient of a small image patch in the vicinity of the keypiont. The size of the image patch is determined by the size of the keypoint scale given by the Censure detector. Usually the size of the image patch is 1.5 to 3 times the size of the keypoint.

After computing the gradient magnitude of the image patch, the gradient magnitude is weighted by a Gaussian window to emphasize the importance of closer samples. Then these samples are divided into 4x4 subregions, and for each subregion the samples are binned into 8 bins according to their orientation. It gives a total of 4*4*8=128 bins. These bins are put into a 128-dimensional vector and normalized to 1.

After the normalization, if any element in the vector is still greater than 0.2, it is capped to 0.2, and the vector is renormalized again.

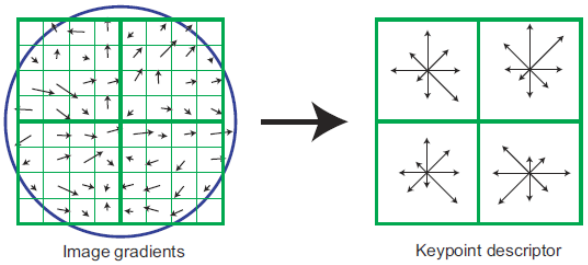

Fig. 3 The Sift descriptor weights the image gradient by a circular Gaussian (the circle), and then bins the image gradient into subregions and bin the gradient into 8 bins. This illustrative figure is taken from the original paper by David Lowe [1].

V. Experiment

A. Setup

In this experiment, 40 sample images are resized to 256x256 and then indexed by their Sift descriptors on the server machine and stored in an inverted file system (IFS) in RAM. The server machine is an old laptop computer, with 2GHz single core CPU and 1GB RAM running on Ubuntu. The

IFS is implemented in Java servlet and hosted on Apache Tomcat. The 40 sample images are taken from Nikon and Canon web sites.

The client is a iPhone4S device. The camera is operated using AVFoundation. The four anchors of the image is tried to be found using template matching per-frame. Once four anchors are found, an inverse perspective transform is done to realign the image to 256x256. Then its Censure keypoints are localized and Sift descriptors calculated. All algorithms are implemented in Objective C. No OpenCV methods are used to find Sift descriptors and Censure keypoints. All computations are done in the CPU. These descriptors are then Base64 encoded and sent to the server using HTTP Post method. The server tries to match the received descriptors to its IFS using brute force method. The id of the closest matching image (from 1 to 40) is returned to the client.

Fig. 3 Some of the sample images used in the experiment.

B. Results

The system can have retrieval rate of near 100% for every image displayed on a lab LCD monitor and well-focused.

Even when the images are displayed on a glossy iPad under indirect sunlight (i.e. Packard Atrium), and the query image is flooded with noise due to the reflection of the surrounding area, the retrieval rate still reaches a satisfactory 90%.

Error may arise due to defocusing and motion blurring.

C. Discussion and Futher Work

The high retrieval rate is expected as the number of sample images is low. For reference, in the work in [3] the authors tried to index 1M images using a similar method (no WGC, no re-ranking), and can still reach a retrieval rate (mAP) of 50%.

Further work would be to increase the number of sample images, do K-mean clustering, and include possibly weak geometric consistency and re-ranking according to Hervé Jégou [3].

Also a fuzzy weighting scheme [4] can be used to take into account of K nearest codewords rather than only the single nearest codeword. This should help reduce the decoding error.

ACKNOWLEDGMENT

C. F. Mak would like to thank Professor Bernd Girod for his excellent teaching and T.A. David Chen for his mentoring.

REFERENCES

[1] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

[2] Motilal Agrawal, Kurt Konolige, Morten Rufus Blas, “CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching”

[3] Hervé Jégou, Matthijs Douze, Cordelia Schmid, “Hamming Embedding and Weak Geometry Consistency for Large Scale Image Search,” 10th European Conference on Computer Vision (ECCV '08) 5302 (2008) 304—317

[4] Bouachir, W.; Kardouchi, M.; Belacel, N.; , "Improving Bag of Visual Words Image Retrieval: A Fuzzy Weighting Scheme for Efficient Indexation," Signal-Image Technology & Internet-Based Systems (SITIS), 2009 Fifth International Conference on , vol., no., pp.215-220, Nov. 29 2009-Dec. 4 2009