Building Labeling under Various Imaging Conditions

Qi Cao, Ruishan Liu, Tian Tang, Yue Zhang
Electrical Engineering Department
<qcao><rliu2><tangtian><yzhang91> @stanford.edu

***Abstract*—In this paper, we present a method to tackle building recognition and labeling problem. This task is challenging as there are huge intra-class variations due to different illumination conditions and perspectives of camera. In our proposed method, first we do background elimination based on obtaining the multi-image saliency map for each training image. For image representation, both global features, such as color and texture histograms, and local features, such as Dense SIFT, are implemented with their results compared and discussed in chapter IV. Finally, test image with cropped target region is retrieved from the known dataset based on assigned score rule. The name of the building with highest score is returned to user interface. In conclusion, the training and test accuracies of using local features are satisfactory and higher than those of using global features in sacrifice of running time.**

***Index Terms*—Multi-image saliency map (MISM), color histogram, texture histogram, Dense SIFT**

I. Introduction

Have you ever had this need: you saw a building image from the website, a friend's social media or just a poster on the street; you were attracted to it and eager to know more, about its location and/or history; you even planned to visit it? However, you didn't even know its name and had no clue where to start to know more about it. Actually, there has already existed some phone Apps dealing with your requests like this. With Google Goggles, you can do a Google search by taking a picture with your phone. If Goggles recognizes the building in the picture, it labels it and does a Google search with the label as the search keyword.

Building labeling or building recognition is an important task for a wide range of computer vision applications, not just for people's need to know the building's name. It can also be used in surveillance, automatic target detection and tracking, architecture design and so on [1].

However, building labeling is also very challenging since each building can be viewed from different angles, in different scales or under very different lighting conditions, resulting in a large variability among building images. This gives rise to the need for invariant features, i.e. image features should have minimal differences under these various conditions [2].

Our project aims to meet people's need for labeling buildings fast and accurately. Our dataset consists of famous building images collected from Flickr by searching with particular text tag. The dataset contains images taken from different viewpoints, with different scales and under different lighting conditions.

II. Related Work

A. Review of previous work

Building recognition problem is often regarded as an image retrieval problem. There is already a great amount of literature studying content based image retrieval. In early times (before 2002), many methods were based on global color and texture features [3]. For example, [4] presented a method for image region representation based on segmentation using the Expectation-Maximization algorithm on combined color and texture features; [5] discussed a method that matched the sketches of object shapes with some templates.

Besides building descriptors represented by global features, another commonly used strategy for building recognition is to extract local features from images. [3] introduced a new mid-level feature, the consistent line clusters. The color, orientation, and spatial features of line segments are exploited to group them into line clusters. [6] [7] [8] used SIFT [9] and other SIFT-based features, such as Informative SIFT (i-SIFT), to build descriptors for building recognition.

In addition, many methods have been studied to effectively dealing with lighting and viewpoints variations in building recognition [10] [11]. [12] [13] proposed some methods for large scale systems based on online learning and unsupervised clustering algorithms.

B. Our methods

Our methods are to extract the global features (Method 1) and local features (Method 2) of our building image database by constructing the multi-image saliency map (MISM) for each iamge. For a selected building region supplied by the user, we will also extracte the global and local features using same technique and do image retrieval (comparing with each building's descriptor), and then label it with the name of the most similar building in our database.

The advantages of our methods mainly come from background elimination using MISM. When training the dataset, we only extract features in co-salient regions based

on MISM, so our methods can avoid the effect of background variations. For different images of a same building, the backgrounds may be highly different, which will bring great variations in features (such as color histogram) and reduce accuracy. This will be discussed in detail in Technical Part. Besides, our two methods can be applied in the situation where we have very limited number of images, since the features have high similarities in co-salient regions.

III. Technical part

A. Background Elimination based on Co-saliency Detection

Since the various imaging conditions, or the complex background of the building images can make the extraction of invariant features difficult, we begin by eliminating the background of our building image database using Co-saliency model. Based on this model, we can build a multi-image saliency map (MISM) [14], which can then be applied to extract global color and texture features and local scale invariant features.

The goal of MISM is to extract the saliency information among multiple images. Given multiple images for a building, the multi-image saliency is defined as the inter-image correspondence, which can be obtained by feature matches. If two images contain a similar object, the object in each image should be assigned with high visual saliency values. For two images I_i and I_j, the multi-image saliency map of image I_i is defined as

$$S_g(I_i(p)) = \max_{q \in I_j} s^*(I_i(p), I_j(q)) \tag{1}$$

where p and q denote regions in images I_i and I_j, respectively. $s^*()$ represents a function that measures the similarity between two regions.

Now, taking a folder of images for one building, we apply the principle described above for every pair of images (take 3 images as an example, we build the MISM through calculating the co-saliency regions for image pair 1/2, 1/3 and 2/3). And finally construct the MISM for each building image through averaging all the S_g's.

The multi-image saliency detection method mainly consists of 5 stages:

Pyramid Decomposition of an Image Pair: This stage is used to obtain a pyramid of images with decreasing resolutions, see Fig.1. We divide each image into a sequence of increasingly finer spatial regions by repeatedly applying Normalized Cuts segmentation algorithmth [15] to the regions at each level. It's a pyramid decomposition because each region at one level may be divided into several subregions at the next level.

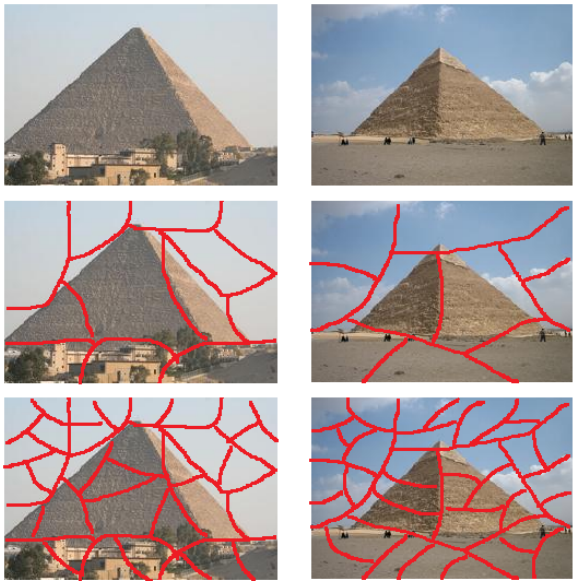

Fig. 1: Pyramid Decomposition of Images

Region Feature Extraction: In the background elimination phase, two global properties are used as descriptors of regions, i.e., color and texture descriptors.

For color descriptor, RGB, L*a*b* and YCbCr color spaces are used together to represent the color feature, so every pixel is represented as a 9-dimensional color vector. The other desciptor, the texture descriptor is created only from RGB space. All pixels in the image pair are quantized into clusters by using the k-means clustering algorithm. Each cluster center is called a codeword. For each region, we compute the histogram by counting the number of codewords at each bin (i.e., cluster).

The Co-Multilayer Graph Representation: After feature extraction, it is time to measure the similarity so as to infer the co-salient object from a pair of images. To measure the similarity, we denote the co-multilayer graph $G = (V, E)$ with nodes $v \in V$ and edges $e \in E$ [14]. Two nodes v_i and v_j are connected by the directed links e_{ij} and e_{ji}, which have weights $w(e_{ij})$ and $w(e_{ji})$ respectively. A three-layer example of the graph model is showed as Fig.2. Each region is represented as a node, which connects with other nodes by the directed edges. Each node not only has links with the neighboring layers within an image, but also connects with nodes in the other image. But note it won't connects nodes in the other image at the same level.

Here is the funtion to assign a weight to each edge of the graph.

$$w_{ij} = \begin{cases} exp(-\theta_f d(f_i, f_j)) & \text{if } l_i - l_j = -1 \text{ or } l_i - l_j = 0, \\ 0 & \text{if } \|l_i - l_j > 1\| \text{ or } l_i > l_j, \end{cases} \tag{2}$$

with

$$d(f_i, f_j) = \chi^2(f_i, f_j) = \sum_{z=1}^{Z_f} \frac{(f_i(z) - f_j(z))^2}{f_i(z) + f_j(z)} \tag{3}$$

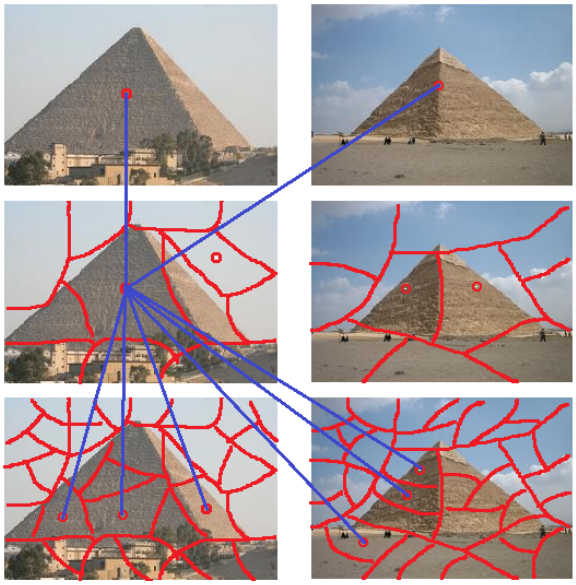

Fig. 2: Co-multilayer Graph Model

Where i and j denotes two nodes and l_i and l_j represent the level number; f_i and f_j denote the color or texture descriptor for regions i and j respectively. Z_f denotes the dimensional number of the descriptor. θ_f is a constant that controls the strength of the weight. $\chi^2()$ denotes the chi-square distance.

Normalized Simrank Similarity Computation: Based on the co-multilayer graph, we next measure the similarity between two region nodes using a link-based method called SimRank. And finally use (1) to calculate multi-image saliency map.

The basic intuition behind SimRank is that two objects are similar if they are referenced by similar objects [16]. Define $s(a,b)$ as the similarity score between objects a and b. $s(a,b)$ can be calculated by:

$$s(a,b) = \frac{C}{|In(a)||In(b)|} \sum_{i=1}^{|In(a)|} \sum_{j=1}^{|In(b)|} s(In_i(a), In_j(b)) \quad (4)$$

where C is a decay factor lies in [0, 1], $|In(a)|$ denotes the number of in-neighbors $In(a)$ and $In(a)$ for node a. To eliminate the fluctuation of in-neighbor node number, we normalize the similarity score by

$$s^*(a,b) = \frac{s(a,b)}{max(s(a,a), s(b,b))} \quad (5)$$

and use this result to calculate multi-image saliency map,

$$S_g(I_i(p)) = max_{q \in I_j}(s^*(I_i(p), I_j(q))) \quad (6)$$

where p and q are the region nodes in image pair (I_i, I_j). Here, in consistency with [14], we choose node pair (p,q) from the same level. By computing the similarity score between node pairs iteratively, we get the similarity between two images.

Co-saliency map: For a training set with N images, we extract N-1 co-saliency maps for each image based on Multi-Image Saliency Maps (MISMs) obtained by color descriptors and texture descriptors. The co-saliency map is defined as a linear combination of color and texture saliency maps where the weight represents the contribution of each map. Let SS_i denote the co-saliency map for image I_i and $R(I_i)$ is a set of regions in image I_i.

$$SS(I_i(p)) = \beta_1 S_g^c(I_i(p)) + \beta_2 S_g^t(I_i(p)) \quad (7)$$

for all $p \in R(I_i)$, where $\beta_1 + \beta_2 = 1$. The choice of parameters will affect the effectiveness of the algorithm. The experiments with various combinations of β_1 and β_2 will be presented and discussed in later chapter. Upon getting N-1 co-saliency maps for each image, we calculate the mean of them and end up with N co-saliency maps for each building. With the histogram of each image, we can proceed to image retrieval and building recognition for test images.

B. Global Feature extraction

Color Descriptor: We build color histogram in HSV color space using only the pixels in co-salient regions. HSV color space is a more intuitive representation of color and closer to the way human being perceive color.

First, we use a nonlinear transformation to change RGB channels into HSV channels.

$$R' = R/255$$
$$G' = G/255$$
$$B' = B/255$$
$$C_{max} = \max(R', G', B')$$
$$C_{min} = \min(R', G', B')$$
$$\Delta = C_{max} - C_{min}$$

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times (\frac{G'-B'}{\Delta} mod 6) & C_{max} = R' \\ 60^\circ \times (\frac{B'-R'}{\Delta} + 2) & C_{max} = G' \\ 60^\circ \times (\frac{R'-G'}{\Delta} + 4) & C_{max} = B' \end{cases}$$

$$S = \begin{cases} 0 & C_{max} = 0 \\ \frac{\Delta}{C_{max}} & C_{max} \neq 0 \end{cases}$$

$$V = C_{max}$$

Then the HSV space is uniformly quantized into a total of 256 bins, which includes 16 levels in H, 4 levels in S, and 4 levels in V. By calculating the different types of colors appeared and the number of pixels in each type of the colors appeared, we get the color histogram in HSV space. The color histogram of an image is relatively invariant with translation and rotation and varies only slowly with the angle of view.

Fugire 3-4 show the importance of background elimination for builidng color histogram. As shown in the figures, background color can have a great impact on the color histogram, which results in the dissimilarity between color

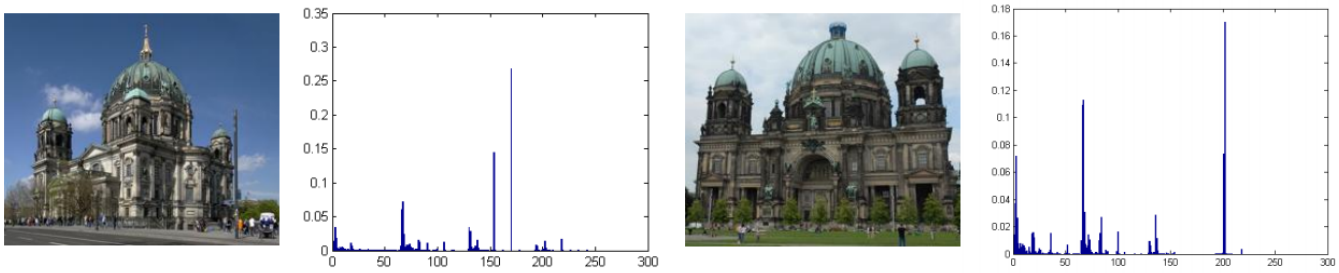

Fig. 3: Color Histograms Before Background Elimination

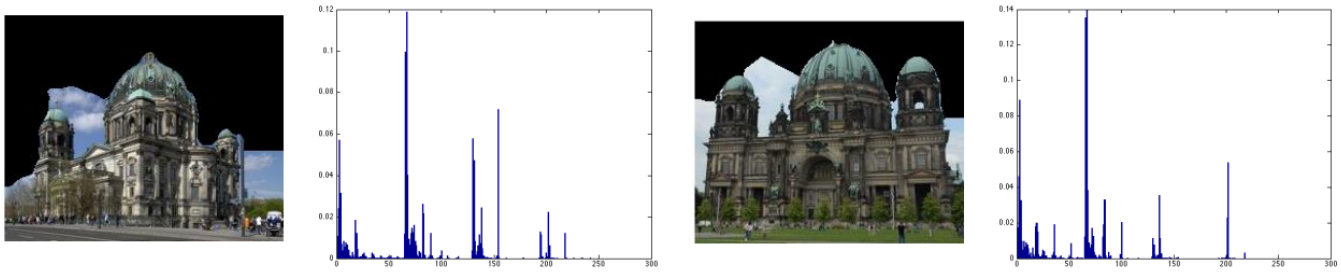

Fig. 4: Color Histograms After Background Elimination

histograms of images for the same building.

Texture Descriptor: Co-occurrence matrix is used to build a 8-dimensional texture histogram for each image with background eliminated. Co-occurrence matrix captures numerical features of a texture using spatial relations of similar gray tones. From the normalized co-occurrence matrix, we derive the mean and standard deviation of four features: power, contrast, correlation and entropy to form a texture descriptor.

$$Power = \sum_i \sum_j p[i,j]^2$$

$$Contrast = \sum_{n=0}^{N-1} n^2 \left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} p[i,j] \right\}, \text{ where} |i-j| = n$$

$$Correlation = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (i,j) p[i,j] - \mu_x \mu_y}{\sigma_x \sigma_y}$$

$$Entropy = -\sum_i \sum_j p[i,j] log(p[i,j])$$

where $p[i,j]$ is the $[i,j]$th entry in a gray-tone spatial dependence matrix, and N is the number of distinct gray-levels in the image.

Histogram Concatenation: For each image, color and texture histograms are concatenated to form an overall histogram as the global feature. It can be represented as

$$H = \begin{bmatrix} C & \mu T \end{bmatrix} \tag{8}$$

where C and T are color and texture histograms, and μ is a nonnegative weight.

C. Local Feature extraction

SIFT feature: Compared with globla features, local features are considered to be more robust and more suitable to describe objects like buildings in the images. Now, we can apply the MISM of each building to substract the complex background of every image in our dataset, with only co-saliency regions (regions of the buildings) left. Then we extract the SIFT of each processed image, taking it as the descriptor of the building in this image.

Scale Invariant Feature Transform (SIFT) is a common approach to transform an image into a large collection of local feature vectors [17]. Each of these feature is invariant to image translation, scaling, and rotation, and partially invariant to illumination changes and affine or 3D projection. When we apply the MISM to our dataset and get the images basically with the buildings, we can expect to extract more robust and more accurate local features using SIFT.

The first step is to extract the scale-invariant features to identify the keypoints from the building objects in our images, and to generate descriptors based on those keypoints. The keypoints are chosen on locations at maxima and minima of a difference of Gaussian function applied in scale space, to achieve rotation invariance and a high level of efficiency.

For the efficiency, we can separate the 2D Gaussian function into 1D Gaussian functions in the horizontal and vertical directions:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-x^2/2\sigma^2) \tag{9}$$

Usually for the keypoint localization, all smoothing operations are done with $\sigma = \sqrt{2}$, which could be approximated with

sufficient accuracy using 7 sample points.

At the training phase, every building image in our dataset is first convolved with the Gaussian function using $\sigma = \sqrt{2}$ to get an image A. It is then repeated a second time with a further incremental smoothing of $\sigma = \sqrt{2}$ to get a new image, B, which now has an effective smoothing of $\sigma = 2$. The difference of Gaussian function for this building image is then obtained by subtracting image B from A.

The second step is to characterize the keypoints we localized in the previous step. Usually each key location is assigned with the coordinates of the location, a scale and an orientation, so the descriptors are invariant to scale and orientation.

For each pixel in our building images with background eliminated, the orientation R_{ij} for each pixel can be computed using pixel differences:

$$M_{ij} = \sqrt{(A_{ij} - A_{i+1,j}) + (A_{ij} - A_{i,j+1})} \tag{10}$$

$$R_{ij} = atan2(A_{ij} - A_{i+1,j}, A_{i,j+1} - A_{i,j}) \tag{11}$$

Where A_{ij} represents a pixel, and M_{ij} represents the image gradient magnitude.

Given a stable location, scale, and orientation for each pixel in our building images, we can then generate the descriptors to describe the local image region. Basically each descriptor represents the coordinates, scale and orientation of the keypoints.

Dense SIFT: In this paper, we implement Dense SIFT as the local feature (function dsift.m in VLFeat library).

DSIFT is a dense version of SIFT, which computes a large number of SIFT descriptors for densely sampled keypoints with identical size and orientation in a short time. This function is called iteratively for calculating local SIFT features for all the images in the same training set.

The single parameter of DSIFT is size, which controls the size of a SIFT spatial bin in pixels. A DSIFT descriptor with bin size equal to m is equivalent to a SIFT keypoint of scale $m/MAGNIF$ as the bin size multiplier is defaulted to $MAGNIF = 3$ in SIFT descriptor. And the salient advantage of DSIFT is that it runs much faster than SIFT [18].

Figure 5 shows DSIFT keypoints matching result.

D. Image retrieval

Upon getting the color-texture histogram and DSIFT descriptor of the test image, we need to search the training dataset and retrieve a match for it.

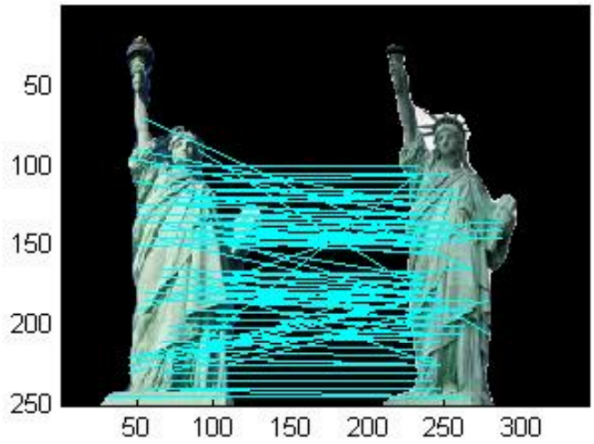

Fig. 5: Dense-SIFT Keypoints Matching

For both methods, we use the same weight parameter

$$scoreRule = [15\ 12\ 10\ 8\ 6\ 5\ 4\ 3\ 2\ 1]$$

.

For each test image in Method 1 (global features), we call library function $vl\_alldist$ to calculate the difference between test image histogram and all the histograms in training dataset and sort the results according to their relevance. Assign corresponding scores to the top 10 ranking training images given by vector scoreRule. The building with maximum total score is the winner and its name is returned to user interface output.

Similarly, for Method 2 (local features), we call library function $sift\_match$ to calculate the similarity between test image SIFT descriptor and all the SIFT features in dataset. The top 10 candidates are assigned with a score and the building with highest score is the match.

IV. Experiment

A. Dataset

Our training and test dataset contains 30 buildings. For each building, we use 5 images for training, and 3 for test. All images are downloaded from Flickr, with sizes between 150×250 and 350×500. Figure 6 shows some examples of training and test images.

B. Compute MISM for training images

For each training image, we compute the similarities between it and other training images of the same building, to get its MISM. The co-saliency map is represented as a grayscale image that has the same size with original image. Higher intensity represents higher similarity and saliency, that is, the region with higher intensity is more common in each image, and more likely to be a part of the building. Figure 7 shows some examples of training images and the corresponding MISMs.

Fig. 6: Some Examples of Dataset Images. (a) Training Images. (b) Test Images.

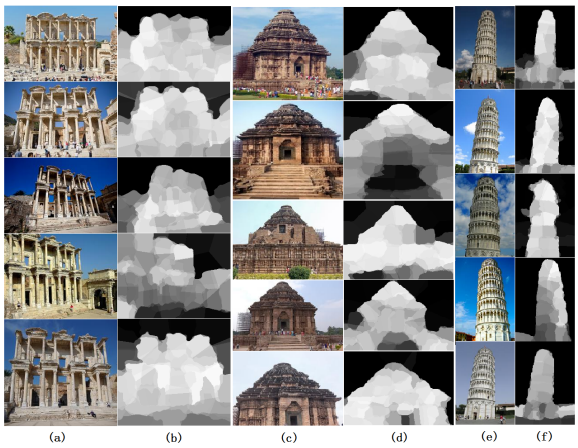

Fig. 7: Images and Their MISMs. (a)(c)(e) Original Image Groups. (b)(d)(f) Co-saliency Maps.

C. Training

Upon getting the MISMs, we extract global (color and texture histograms) and local features (DSIFT) from salient regions in each training image, and save them as .mat files for later computation.

Method 1: For the five MISMs of a building, we choose a threshold a to decide co-salient regions, and only extract global features from these regions in corresponding training images. Since the performance of MISM algorithm is different for different buildings, the thresholds should also be chosen differently.

We use a 150 by 364 matrix H to represent color and texture histograms. Each row is the histogram of one image, and the first 256 columns represent color histograms, with the last 8 columns representing texture histograms.

Method 2: For deciding co-salient regions, we choose one threshold for all the 150 training images. It is smaller than the thresholds in Method 1, since SIFT feature is a local feature, and is relatively robust to backgrounds. Compute Dense SIFT features of co-salient regions in each image.

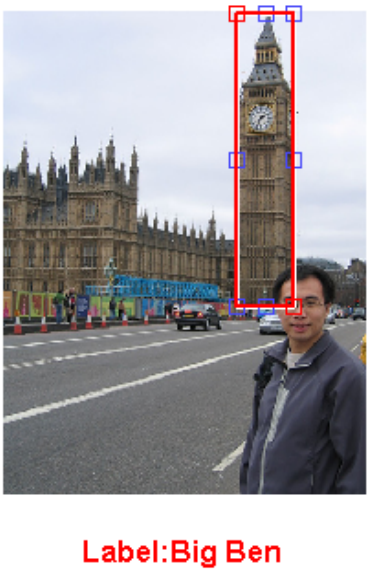

Fig. 8: User Interface of the System

We use a 150 by 1 cell array S to represent Dense SIFT features, with each cell representing the features of one image.

D. User Interface

This is how our building labeling system works: when users input an image, it first asks them to crop a region that they want to label, to get a smaller query image. Then, it computes features of the query image, and outputs labeling result. Figure 8 shows the user interface of our system.

E. Results of Method 1

We adjust the weight μ in (8), and get the training and test accuracies as μ varies, shown in Figure 9.

The training and test accuracies almost have same changing trends. As μ increases from 0 to 1, the accuracies first increase slightly, remain maximum and almost constant at $0.35 \leq \mu \leq 0.5$, and decrease fast when $\mu \geq 0.5$. The best training accuracy is 77.8% at $\mu = 0.4$, and the best test accuracy is 46.7% at $\mu = 0.35$.

F. Results of Method 2

The accuracies and average runtime (s/image) for Method 1 and 2 are shown in Table I. For Method 2 (Dense SIFT features), the training accuracy is 95.3%, and the test accuracy is 81.1%, which are both better than Method 1. However, the average runtime for Method 2 is 12.7 s/image, while that of Method 1 is just 0.24 s/image.

Method	Training Accuracy	Test Accuracy	runtime (s/image)
1	77.8%	46.7%	0.24
2	95.3%	81.1%	12.7

TABLE I: Accuracies and Runtime of Two Methods

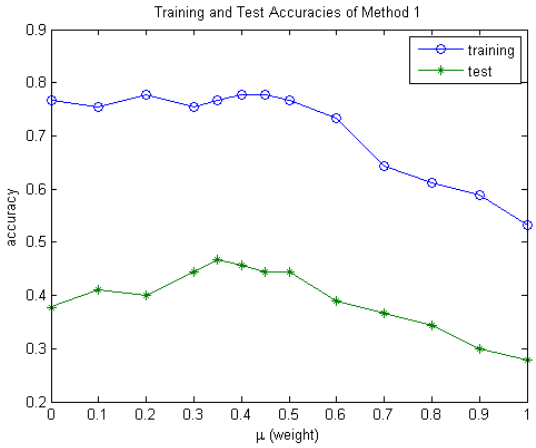

Fig. 9: Training and Test Accuracies of Method 1

G. Result Analysis

From Table I, the training and test accuracies for Method 2 are much higher than Method 1. This is because SIFT is a description for small patches containing only several pixels, while color and texture histograms describe the overall characteristics of a large region. Besides, for Method 1, the dimensions of color and texture histograms are 256 and 8, respectively; and accuracies are highest when $\mu \approx 0.4$. Thus, texture histogram has only a slight effect on overall histogram, compared to color histogram, which has leading effect. In fact, many buildings in our dataset have very similar colors, such as Arch of Triumph and Trevi Fountain. In this case, Method 1 is not as effective as Method 2. Figure 10 shows the case, where Method 2 can correctly label these two buildings while Method 1 cannot.

However, for some buildings with salient colors, but without complicated local structures, Method 1 may be more effective than Method 2. For example, figure 11 shows the labeling results for Forbidden City and Khufu Pyramid by two methods. Method 1 labels the two buildings correctly, while Method 2 labels them wrong.

Method 2 requires longer average runtime, because for each image, SIFT feature is stored as a large matrix, about 128 by 1000. However, histogram is just stored as a 1 by 264 vector. Thus, in test, Method 2 requires much more computations.

V. Conclusion

Background elimination by building MISM is important in feature extraction, since image background variations can have great effect on features, especially global features such as color histograms.

Generally, Method 2 using local features has higher training and test accuracies, with longer average runtime. However, in some cases where local structures of buildings are not salient, Method 1 using global features may perform better than Method 2.

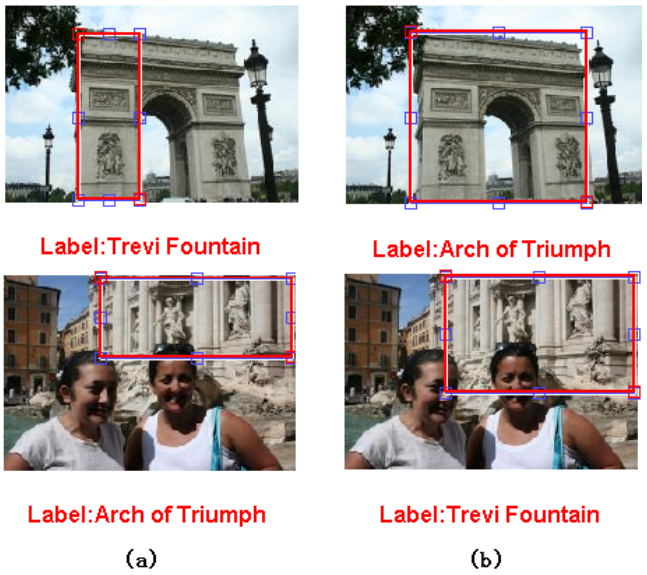

Fig. 10: Labeling Results by Method 1 and 2. (a) Method 1. (b) Method 2.

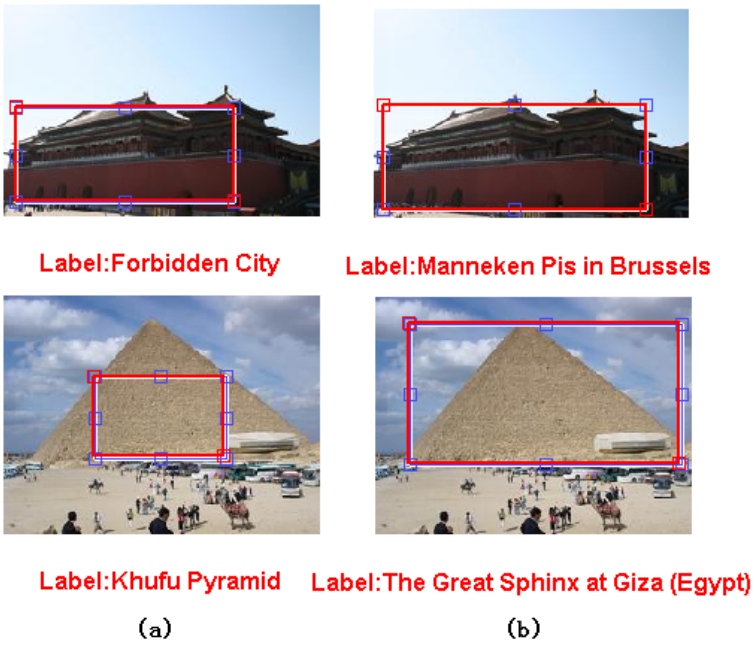

Fig. 11: Labeling Results by Method 1 and 2. (a) Method 1. (b) Method 2.

References

[1] S. Lee, N. Allinson, Building Recognition Using Local Oriented Features, IEEE TRANS- ACTIONS ON INDUSTRIAL INFORMATICS. 9, 3, 2013, pp. 1697-1704.

[2] Nguyen, PG & Andersen, H.J. (2011) Fast and efficient local features detection for building recognition', Studies in Computational Intelligence, vol 339, s. 87-104.

[3] Li, Yi, and Linda G. Shapiro. "Consistent line clusters for building recognition in CBIR." Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 3. IEEE, 2002.

[4] Carson, Chad, et al. ”Region-based image querying.” Content-Based Access of Image and Video Libraries, 1997. Proceedings. IEEE Workshop on. IEEE, 1997.

[5] Del Bimbo, Alberto, Pietro Pala, and Simone Santini. ”Visual image retrieval by elastic deformation of object sketches.” Visual Languages, 1994. Proceedings., IEEE Symposium on. IEEE, 1994.

[6] Zhang, Wei, and Jana Kosecka. Experiments in building recognition. Technical Report GMU-CS-TR-2004-3, George Mason University, 2004.

[7] Fritz, Gerald, et al. ”Building detection from mobile imagery using informative SIFT descriptors.” Image Analysis. Springer Berlin Heidelberg, 2005. 629-638.

[8] Zhang, Wei, and Jana Kosecka. ”Localization based on building recognition.”Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on. IEEE, 2005.

[9] Lowe, David G. ”Distinctive image features from scale-invariant keypoints.”International journal of computer vision 60.2 (2004): 91-110.

[10] Nguyen, Giang P., Hans Jrgen Andersen, and Morten Friesgaard Christensen. ”Urban building recognition during significant temporal variations.” Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on. IEEE, 2008.

[11] Chung, Yu-Chia, Tony X. Han, and Zhihai He. ”Building recognition using sketch-based representations and spectral graph matching.” Computer Vision, 2009 IEEE 12th International Conference on. IEEE, 2009.

[12] Cao, Jiuwen, Tao Chen, and Jiayuan Fan. ”Fast online learning algorithm for landmark recognition based on BoW framework.” Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conference on. IEEE, 2014.

[13] Zheng, Yan-Tao, et al. ”Tour the world: building a web-scale landmark recognition engine.” Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[14] Li, Hongliang, and King Ngi Ngan. “A co-saliency model of image pairs.” Image Processing, IEEE Transactions on 20.12 (2011): 3365-3375.

[15] J. Shi and J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pp. 888905, Aug. 2000.

[16] G. Jeh and J. Widom, Simrank: A measure of structural- context similarity in KDD, 2002, pp. 538543.

[17] Lowe, David G. “Object recognition from local scale-invariant features.” Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, 1999.

[18] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. 2008. Web. 17 March 2015.