

Visual Search of an Image Collection

Yi Zhou 6567008

Abstract

The goal of this report is to build a visual search system. Both feature-based and bag of visual word methods are discussed and implemented in Matlab code. The performances of different algorithms are evaluated on the MSRC-v2 dataset. The results show that the BoVW with dense feature achieves the best performance.

1. Introduction

Figure 1-1 shows the structure of a content-based image retrieval (CBIR) system. The input to the system is a query image. Then, the similar images are retrieved according to the similarity. The similarity can be computed based on some low-level visual features, such as color, texture and shapes.[3] However, these methods are proved to be performance-limited [3]. From the perspective of a human, we find the similarity according to the semantic of the images. To fill this so-called “semantic gap”, bag of visual words (BoVW)[1] method is implemented in visual search problem. Keywords are assigned to each image pattern according to the detected features.

In this report, both of the feature-based methods and the semantic-based, i.e. BoVW, methods are introduced in details. The performance is evaluated on the MSRC-v2 dataset¹.

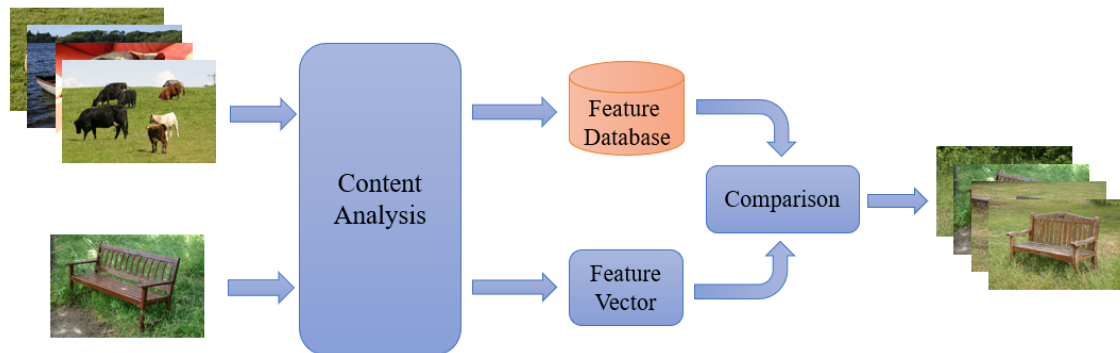


Figure 1-1 CBIR System

¹ <https://www.microsoft.com/en-us/research/project/image-understanding>

2. Feature-Based Methods

The feature is widely used in computer vision. As shown in Figure 1-1, we can define the similarity between two images based on their feature vector. There could be four steps for a feature-based visual search process.

1. Extract low-level features to form a feature vector for each image
2. Accumulate all the feature vectors and do dimension reduction through PCA
3. When a query image comes, extract its feature vector and compute the similarities score with every image in the dataset.
4. Sort out the top-N similar images and calculate the evaluation metric.

In this section, the first three steps will be introduced. Step 4 will be introduced in section 4.2.

2.1 Color-Based Methods

2.1.1 Global Color Histogram

Color is a powerful feature to describe images. The simplest color-based method is called Global Color Histogram. Firstly, we quantize the color space into q bins in each RGB channel. Then, each RGB channel can be represented by a number $bin_x \in (0, q)$. This gives us a long 4-dimensional tensor (three quantized RGB channel plus one pixel-index channel) as the image descriptor. To further reduce the dimension, we can decode the ternary array into a p-order number according to the equation

$$N = bin_R * q^2 + bin_G * q + bin_B \quad (2.1)$$

where $N \in (0, q^3)$. By this way, the size of the descriptor is reduced to two dimensions. This descriptor can be represented as a histogram.

The quantization scale q represents the resolution in color space. In other words, we subsample the color space of 256^3 into a quantized space of q^3 . The value of q should be neither too small, which makes the different colors are undistinguishable, nor too big, which gives little tolerance of the color variations and is computationally expensive.

Figure 2-1 shows an example of images of two categories with similar color histogram. We can see the white clothes and white sky are confused with the white wools. The L2 distance of these two images is $1.1476e+04$. On the other hand, the images of the same category (Figure 2-2) can also have very different color histograms due to illumination, shadow and different backgrounds. The L2 distance of these two images is $1.7119e+04$, which is even larger than the previous case.

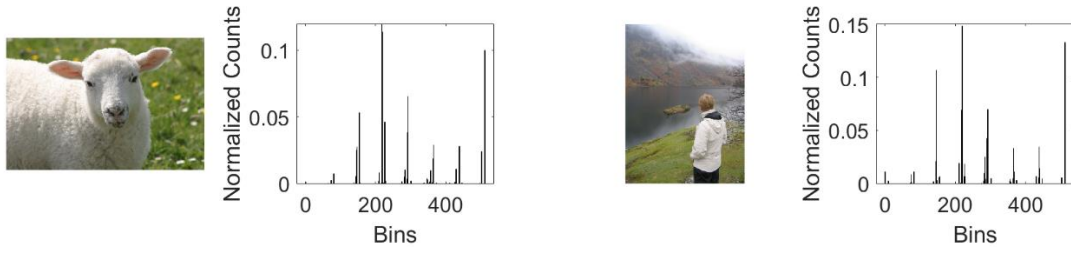


Figure 2-1 Images of Two Categories with Similar Global Color Histogram

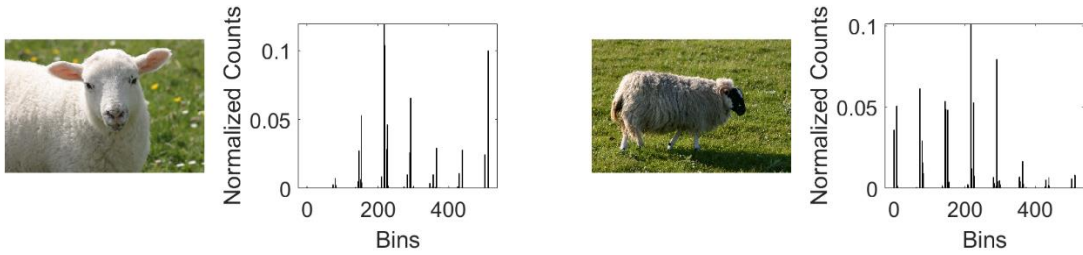


Figure 2-2 Images of the Same Category with Different Global Color Histogram

2.1.2 Grid Based Color Histogram

In addition to color information, the spatial information can also be used to measure similarities. The spatial information can be extracted by implementing a grid-based method. Specifically, we can divide the image into $m \times n$ grids. Then, the color histogram is calculated within every grid. The descriptor is obtained by concatenating all the descriptors in all grids. Figure 2-3 shows the result of the grid-based color histogram. The image is divided into 3×3 grids. The resulting L2 distance between the white sheep and man with white color is $1.2907e+04$, while the distance between two sheep is $9.2410e+03$. The result is more satisfying than the previous section.

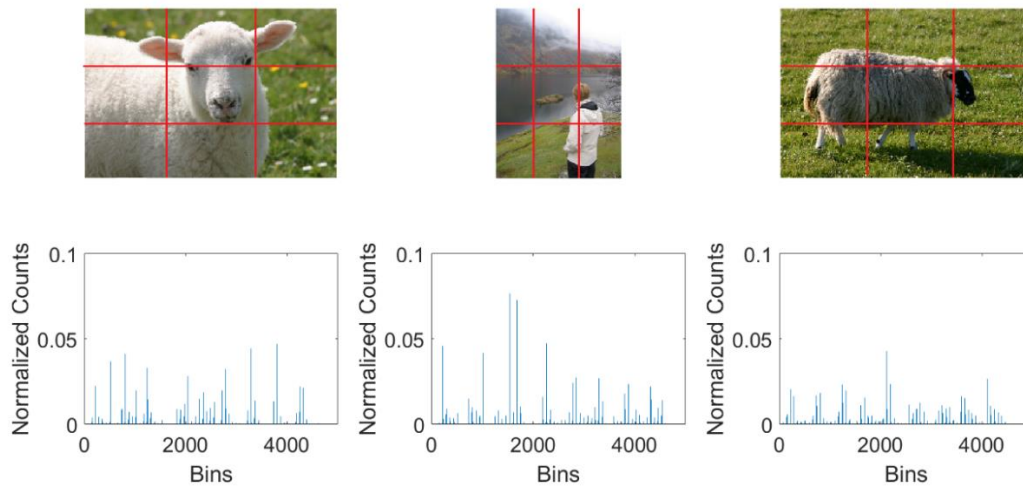


Figure 2-3 Grid Based Color Histogram

Similar to the quantization in color space, the grid size represents the spatial resolution of the method. The smaller the size of grids is, the better the spatial resolution is. Since the location and the scale of objects may vary, the grid cannot be too small.

It should also be noticed that the size of the descriptor is $m \times n \times q^3$ dimensional vector, which could be very large. It is suggested to use a dimension reduction method, such as PCA, which will be introduced in section 2.3.

2.2 Texture-Based Methods

2.2.1 Edge Detection

Edge feature is widely used in visual search problem. There are several choices of edge detectors. For example, we can calculate the first-order derivative of the image intensity by convoluting the image with difference filters. The choice of the filter can be Prewitt operator or Sobel operator. Then we use a threshold to select the strong responses as edges. Figure 2-4 (left) shows the detected edge by using Sobel operator. We should notice there are lots of scattered edges in the image and some edges are broken into pieces.

To improve the performance, we can use Canny detector instead. To suppress these scattered edges, Canny detector uses two thresholds. The detected edge whose gradient magnitude is above the upper threshold is called the strong edge. The edge gradient magnitude is below the lower threshold is considered as not an edge. The edge whose gradient magnitude is between two thresholds is called weak edge. For those weak edges, we can measure them according to the connectivity. If the weak edge is connected to a strong edge, it can be considered as a real edge. Otherwise, it will be suppressed. This principle is implemented as a tracking method called Hysteresis thresholding. Figure 2-4 (right) shows the detected edge by Canny detector, which show a good performance in continuity.

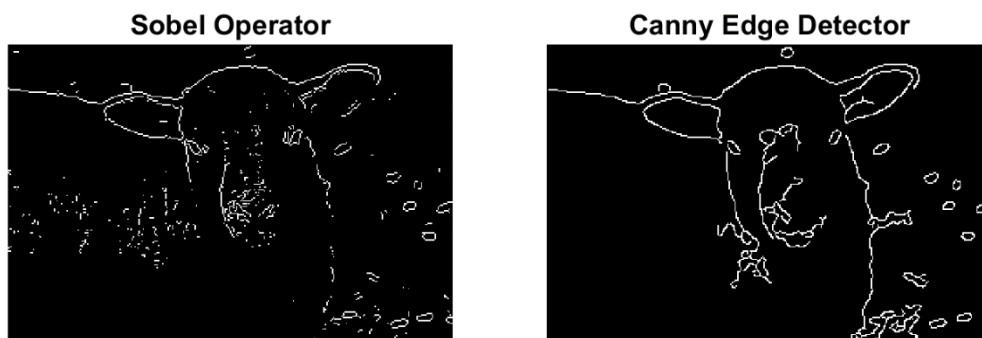


Figure 2-4 Edge Detection

2.2.2 Edge Oriented Histogram with Spatial Grids

To build an edge-oriented histogram, the image is firstly divided into $m \times n$ grids. Within each grid, we detect the edges and make an edge-orientated histogram with interval of $\pi/4$. The

histogram needs to be normalized, since there could be different amounts of edge response due to different scales of the same pattern. The resulting edge-oriented histogram contains information of both texture and space. However, edge orientation itself is always not discriminative enough for image retrieval problem [1]. It is always used together with color information. We can concatenate the edge histogram vector below the color vector for each bin. The simplest choice of color descriptor is to use mean color within each grid. Then, the size of the overall descriptor is $(11 * m * n) \times 1$, which is quite compact. Figure 2-5 shows the result of the histogram combined edge-orientation and normalized mean color. The size of grids is set as 8×8 . The L2 distance is 5.0447 between the sheep and human and 3.6939 between two sheep.

Another choice is to combine a color histogram with edge-oriented histogram. In this case, the size of the descriptor is $(q^3 * 9 * m * n) \times 1$, which could be very large. This inspires us to use some dimension reduction techniques.

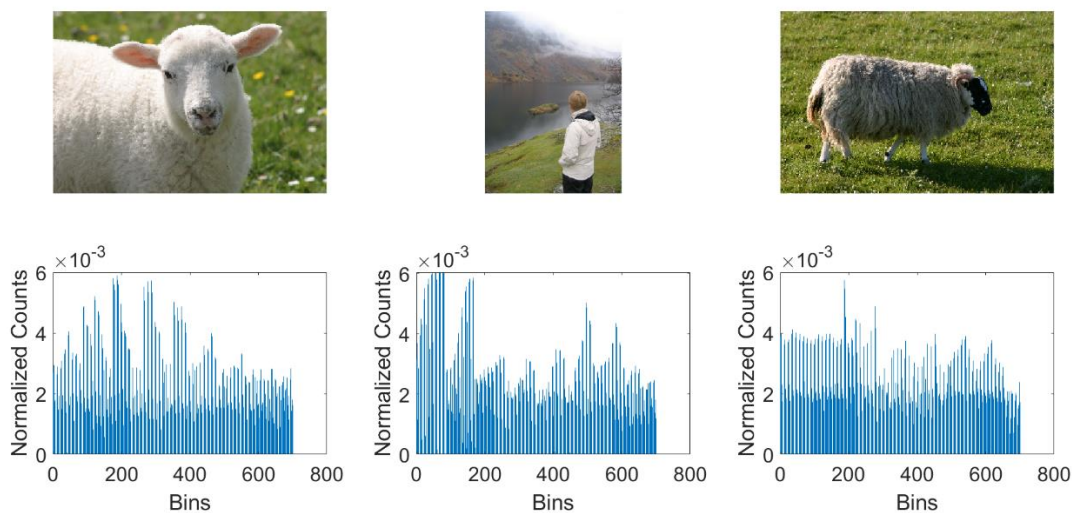


Figure 2-5 Edge Oriented Histogram + Mean Color

2.2.3 Gabor Wavelet Filter

Gabor filter is a band-pass filter which is widely used in extracting edge features. It can be used in both multi-frequency and multi-orientation analysis. It is proved to achieve the optimal performance [9] in the trade-off between time resolution and frequency resolution. We can further adapt it into the wavelet format to obtain the Gabor wavelet filter. However, Gabor wavelet filter is not guaranteed orthonormal to each other. It will lead to problems when we reconstruct the signal using wavelet coefficients. However, in feature detection case, we do not concern much about the orthonormality. The equation of Gabor wavelet is given by

$$\phi(x, y) = \frac{f^2}{\pi\gamma\eta} e^{-\left(\frac{f^2}{\gamma^2}x_r^2 + \frac{f^2}{\eta^2}y_r^2\right)} (e^{j2\pi f x_r} - K)$$

$$x_r = x \cos \theta + y \sin \theta, y_r = -x \sin \theta + y \cos \theta \quad (2.2)$$

where θ and f represents the orientation and center frequency respectively. And the other parameters are constants.

To extract edge information using Gabor wavelet filter. Firstly, we can construct a set of filters with different scales and orientations. Figure 2-6 shows the patterns of Gabor wavelet filter with 4 scales and 8 orientations. Then the image is convoluted with these filters to get the response. The mean and standard deviation of the Gabor wavelet coefficients are used as features. In the previous section, we obtain the local distribution of edges orientation as features. In addition, the Gabor wavelet filter can be used to provide some global edge information, in order to better describe the texture information.

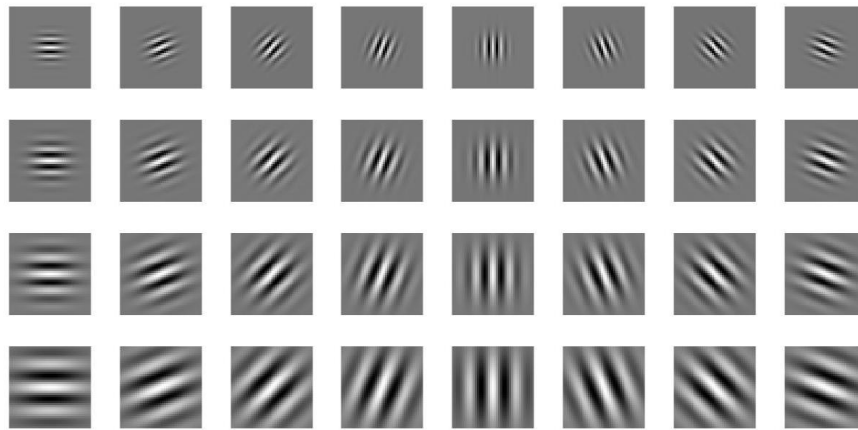


Figure 2-6 Gabor Wavelets with 4 scales and 8 orientations

2.3 PCA Dimension Reduction

The high dimensional data is always very sparse, but consumes a lot of memories and computational efforts. This phenomenon is called curse of dimensionality. In the above sections, some feature vectors could be very long. In fact, lots of features are redundant for classification in this dataset. For example, some features in the corner could be useless since the object are mostly in the middle. Therefore, it is reasonable to do some dimensional reduction to the feature vectors

Principal Components Analysis (PCA) is a popular dimension reduction technique. Firstly, we need to extract the principal components of the data. This is done by extracting the eigen model of the data. The eigenvectors represent the independent bases of the data. The eigenvalues measure the variance along each eigenvector. Next, the eigenvectors corresponding to the small eigenvalues are dropped. In practice, we will normalize the eigenvalues and sort them in descending order. Only the eigenvalues which add up to a certain percentage, for example 97%, are reserved. Finally, the reduced data is obtained by project the data into the new bases.

2.4 Similarity Metrics

After previous steps, every image in the database is represented as a feature vector. When a new query image comes, we extract its feature vector. Then, the similarity can be measured in feature space according to a specific similarity metric. In this section, I will introduce some common-used measurements as similarity metrics. The principle of metric selection is also discussed.

2.4.1 Different Kinds of Metrics

2.4.1.1 Euclidean Distance

The most commonly seen metric is Euclidean Distance. It represents geometric distance in the Euclidean space. The distance of two k-dimensional vectors \vec{x}_1 and \vec{x}_2 is in the form as

$$L2(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_{i=1}^k (x_2^i - x_1^i)^2} \quad (2.3)$$

Since we square the errors, the L2 norm is not robust to large errors. In other words, some large errors will dominate the result.

2.4.1.2 Manhattan Distance

Another classical distance metric is Manhattan distance. It is defined as the L1 norm of the deviation of two vectors.

$$L_1(\vec{x}_1, \vec{x}_2) = \sum_{i=1}^k |x_2^i - x_1^i| \quad (2.4)$$

Compared to Euclidean Distance, it is more robust to large deviations.

2.4.1.3 Mahalanobis Distance

In practice, the data in different dimensions are always on different scales. The simplest way is to normalize the data by subtracting the mean and dividing by the variance along each dimension. However, different dimensions are not always independent. If a small-scaled data is linearly combined with large-scaled data, the normalization will hide the small-scaled data.

The Mahalanobis distance can remove the effects of data-correlation on normalization. We need firstly compute the eigenvectors and project the data into these independent bases. Then, we can do the normalization. The derived equation of Mahalanobis distance is given by

$$M(\vec{x}_1, \vec{x}_2) = \sqrt{(\vec{x}_2 - \vec{x}_1)^T \Sigma^{-1} (\vec{x}_2 - \vec{x}_1)} \quad (2.5)$$

where Σ is the covariance matrix². It should be noticed that the covariance matrix is very likely to be ill-conditioned if the dimension is very high. Therefore, the Mahalanobis distance is usually used combined with the PCA.

2.4.1.4 Pearson Correlation Coefficient

If there is a linear transformation between two vectors, the distance-based metric may no longer be a good choice. For example, if the same object appears in different locations in two images, we should use the correlation-based metric to measure the similarity. The most commonly used correlation-based method is Pearson correlation coefficient.

$$Pearson(\vec{x}_1, \vec{x}_2) = \frac{\sum_{i=1}^k (x_2^i - \bar{x}_2)(x_1^i - \bar{x}_1)}{\sqrt{\sum_{i=1}^k (x_2^i - \bar{x}_2)^2 \sum_{i=1}^k (x_1^i - \bar{x}_1)^2}} \quad (2.6)$$

2.4.1.5 Cosine Correlation Coefficient

Cosine correlation measures the angular between two vectors. The coefficient is only related to the angle between two vectors rather than their lengths.

$$\text{Cosine}(\vec{x}_1, \vec{x}_2) = \frac{|\sum_{i=1}^k (x_2^i x_1^i)|}{\sqrt{\sum_{i=1}^k (x_1^i)^2 \sum_{i=1}^k (x_2^i)^2}} \quad (2.7)$$

2.4.2 Principles of Selection

In order to reduce the search space of the system, I will introduce some principles of metric selection. The choice is close-related to the resolution. In terms of both resolutions in color space and spatial space, if the resolution is coarse, L1 or L2 norms will be sufficient. Because, different bins in the histogram can be considered uncorrelated. On the other hand, if the resolution is very fine, we should give some tolerations to the variations, i.e. shifts in the histogram. Thus, Pearson coefficient or Cosine coefficient will be a better choice.

² The covariance matrix could be in ill-conditioned. In this case, we can use pseudo-inverse instead.

The use of Mahalanobis depends on the scale of different dimensions. In the case of the color-based features, I will not suggest Mahalanobis method. Because Mahalanobis distance may strengthen the weights of small-scale noisy color, leading to poor performance.

3. Bag of Visual Words

The idea of bag of visual words (BoVW) comes from speech area, where the frequencies of the words are accumulated to form a dictionary. In computer vision, the visual words denote the distinctive image patches. A codebook can be generated by accumulating these patches. Different combinations of these visual words can provide some semantic information of the image.

There are three steps for building a BoVW system. Firstly, the features are extracted from every image in the database. Secondly, the features are clustered into different categories and store in the codebook. Finally, when a query image comes, we represent it as a sentence using the codebook. Then, we can compare different sentences to find the similar images.

3.1 Feature Extraction

The features are some distinctive points in an image. A feature detector should consist of two part: a key-point detector and a descriptor. Corner is widely used as the key-point. However, corner itself contains little semantic information. Thus, the descriptor is manually designed to store the surrounding information of key-points. We can measure the similarities of two images according to their descriptors. There are various types of detectors and descriptors. We can arbitrarily combine them to get an appropriate representation of the feature.

3.1.1 Feature Detector

3.1.1.1 Harris Corner Detector

Corner points are very suitable for using as key-points. Harris corner detector is simple and fast, thus is widely-used in many real-time applications, such as KLT tracker. In the corners, if we move an observing window, the intensity will change in both x and y directions. The change can be evaluated through the cost function of

$$\sum_w g * [I(x + u, y + v) - I(x, y)]^2 \quad (3.1)$$

where w represents the observing window, g is the Gaussian kernel, $I(x, y)$ is the intensity of the pixel, (u, v) represent a shift distance. Then we take a Taylor expansion of $I(x + u, y + v)$ and rearrange the equation into a matrix

$$\begin{bmatrix} u & v \end{bmatrix} \cdot \left(\sum_w \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \cdot \begin{bmatrix} u \\ v \end{bmatrix} \quad (3.2)$$

The covariance matrix in the middle is called Harris matrix. The eigenvalues of the matrix indicate the intensity changes if we move the window. According to Figure 3-1, if both eigenvalues are large, the point will be considered as a corner. Furthermore, a measurement called R-score can be used as an indicator.

$$R = \det(M) - k \cdot \text{tr}(M)^2 \quad (3.3)$$

where M denotes the covariance matrix and k is a constant. A large R-score indicates a corner.

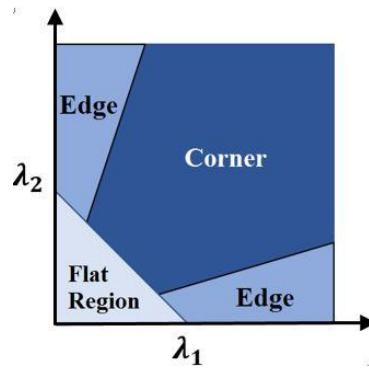


Figure 3-1 Eigenvalues of Harris Matrix

Figure 3-2 shows the response of Harris detector. We can see that several corner-like points are returned. Harris detector is fast and rotation-independent, but it is not robust to scale. In the right figure, a gaussian pyramid is implemented for down-sampling. We can find that many detected points in the original image are no longer corners in the scaled image. In practice, depending on the distance to the image plane, the object always shows different scales in the image. This requires us to explore some scale-invariant features

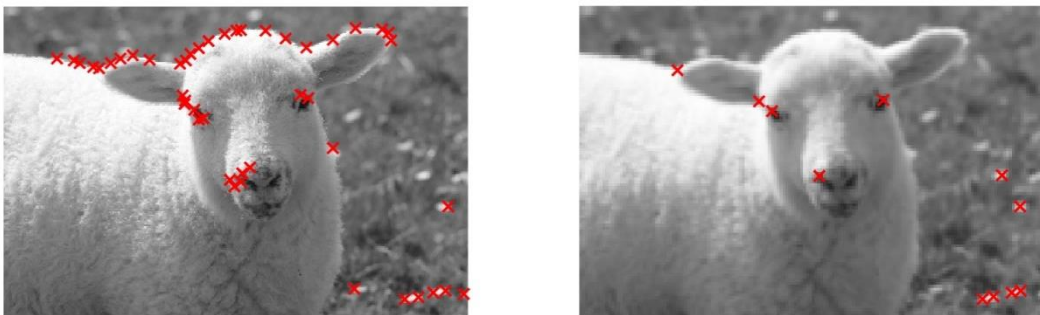


Figure 3-2 Detected Harris Corners in Different Scales

3.1.1.2 SIFT Detector

Scale Invariant Feature Transform (SIFT) is a very stable local feature which is both scale-invariant and rotation-invariant. It consists of both the detector and the descriptor. To solve the scale problem, SIFT builds a discrete scale space by using a Gaussian pyramid and find the key-points in different scales. In each octave layer, blob-like features are detected using a multi-scale detector. Compared to Harris corner, blob features are stable across scales. Laplacian of Gaussian (LoG) can be used to detect blob. The idea is to take second order derivatives of the intensity to find blob like features. Because the higher order derivatives are always noisy, we can use a Gaussian filter to smooth the image. Combining these two filters yields the equation as

$$LoG(x, y, \sigma_I) = \left| \frac{\partial^2 g(x, y, \sigma_I)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma_I)}{\partial y^2} \right| * f(x, y) \quad (3.4)$$

where σ is the Gaussian variance to control the scale of the filter. In order to reduce computation effort, we can use Difference of Gaussian (DoG) as an approximation of LoG, by taking difference of two Gaussians at different scales.

$$DoG(x, y, \sigma_I, k) = G(x, y, k\sigma_I) - G(x, y, \sigma_I) \quad (3.5)$$

Figure 3-3 shows an example of 2-D DoG.

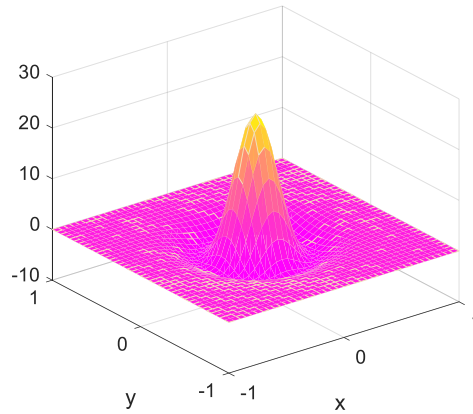


Figure 3-3 Blob Detector

In different octave layers, we convolute the image with multiple DoG detectors. The blob features can be described by the local extreme between its eight spatial neighbors. By carefully designing the Gaussian pyramids and the sizes of DoG filter, the SIFT key-points can be considered as scale-invariant.

Since DoG filter is homogeneous in all direction, it has no rotation information. To solve this, SIFT builds the edge-oriented histogram. Specifically, the orientation is divided into 360 bins. The peak in the histogram corresponds to the orientation of the key-point. As a result, we get

the rotation-invariant key-points.

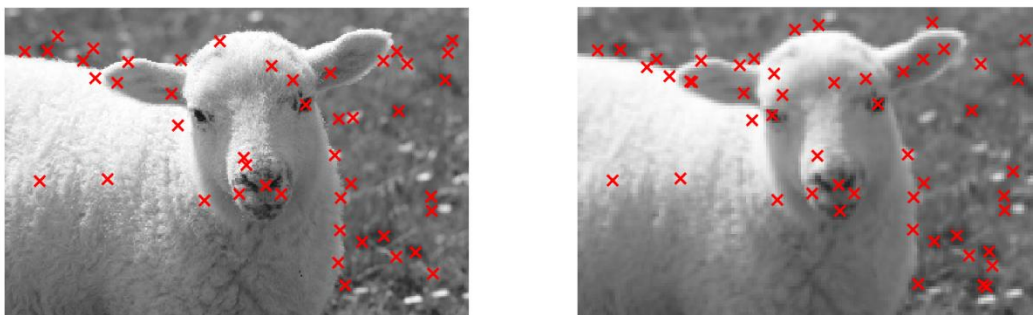


Figure 3-4 Detected SIFT Features in Different Scales

Figure 3-4 shows the SIFT key-points. Compared to the Harris corners, the SIFT key-points are very stable across different scales. However, SIFT is very computationally intensive. In addition to Harris and SIFT, there are lots of other selections. For example, oriented FAST detector, which is used in ORB feature, is very popular in real-time applications like SLAM. SURF is another widely used detector, which is very similar to SIFT. But its speed is much more tolerable. Due to the time limitation, the details will not be introduced. Further information can be found in [6].

3.1.2 Dense Feature Detector

There are some problems of using local feature detector for BoVW. Firstly, the number of detected key-points varies among different images. It means we cannot fully describe the images with few corners. This could lead to poor performance if we take these images as query images. Moreover, we have to dynamically allocate memory to the key-point matrix, which is time-consuming. Secondly, the key-points are imbalanced distributed in the image. They tend to accumulate in the area with strong textures. We will lose lots of global spatial information due to this imbalance.

To solve these problems, we can use dense features. As Figure 3-5 shows, the image is divided into $m \times n$ grids. We extract key-points at intersections of different grid lines. By this way, the dimension of each descriptor is fixed and some global information is added by using the grids.

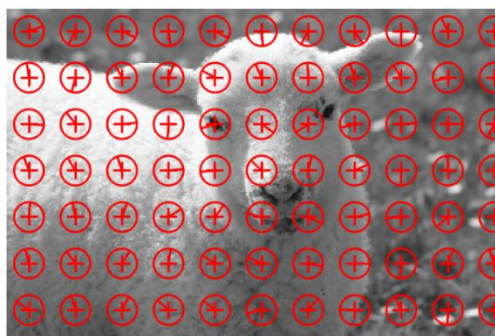


Figure 3-5 Dense Features

3.1.3 Feature Descriptor

The key-point detector gives us the location of the features as well as its intensity and orientation information. However, a single point is not enough for comparison. We need to design a descriptor to give more information about the key-point. SIFT descriptor is proposed with SIFT detector. It uses the information of the 16x16 neighbours around the key-point. Then, these pixels are further divided into 16 4x4 blocks. In each block, an edge-orientation histogram with 8 bins is built to describe the image patch. The results are stretched out to form a 128-dimensional vector.

3.2 Codebook

In the previous section, we extract the feature vector for each image. The next step is to identify the visual words and assemble them into a codebook. The visual words are considered as some representative features. To find them, we can use the K-means clustering.

3.2.1 K-means Clustering

K-means clustering is a popular unsupervised classification method. The classical K-means method follows 5 steps:

- 1 Pick K centroids uniformly at random
- 2 For each point, calculate the L2 distance to each of the centroids
- 3 Allocate each point to its closest centroid
- 4 Update the centroid as the mean of the data in each cluster.
- 5 Repeat steps 2-4 until convergence or reaching max iterations

The initialization of centroids strongly affects the performance of the problem. Matlab uses K-means++ algorithm for a better initialization. The method iteratively selects centroids with the probability of

$$P(x_i \text{ as new centroid}) = \frac{\|x_i - \mu_j^i\|^2}{\sum_{i,j} \|x_i - \mu_j\|^2} \quad (3.6)$$

where μ_j^i is the allocated centroid to x_i in the previous iteration. By using K-means++, it is more likely to select the points which are far from the existed centroids as the new centroid. Thus, the centroids can be evenly distributed in the dimensional space.

3.2.2 Approximate K-Means

In Matlab, there is a built-in k-means function which is easy-to-use. However, when I use it in my program, I find K-means is very slow when applied to a high-dimensional case. To speed up the program, I adopt the approximate K-means method suggested in paper[7]. In classical K-means, the calculation of Euclidean distance for every point (step 2) consumes a large amount of time. Alternatively, a forest of 8 randomized k-d trees [7] can be used to find the nearest neighbors of each centroid.

The k-d tree is a binary search tree to partition data in k-dimensional space. In classical k-d tree, in every iteration, we choose the dimension with the largest variance as the partition dimension. However, if a point lies near the boundary, it is very likely to be assigned to the wrong side of the node. In practice, multiple dimensions may have similar variance. Perhaps one point is ambiguous for classification in the largest dimension may be more obvious in the second-largest dimension. Randomized k-d trees improve this by randomly generating multiple independent trees with overlapped partitions. The process of building a randomized k-d tree is summarized as below

- 1 Starting from the root node, which contains all the data points. For every point in that node, extract max and min in each dimension.
- 2 Search for the dimensions with top-N largest variance and randomly select one of them as the partition dimension. Set the median in that dimension as the partition point.
- 3 All the lower points in that dimension consist of the left node. All the higher points consist of the right node.
- 4 Repeat until the node is undividable. The undividable node is called a leaf node.

Rather than calculating L2 distance to every point, we can simply find the neighbors of the centroid by comparing it with the boundaries of the hyperrectangle. If all the points in the boundary are close to the centroid, this node altogether with its child nodes will be assigned to this centroid. Otherwise, we will further detect the child node. To make use of all these random trees, the priority search algorithm[8] is implemented. By using this method, the algorithm complexity is reduced from $O(NK)$ to $O(N \log(K))$, where N is the data number and K is the clustering number.

3.2.3 Codebook Generation

The clustering algorithm returns K centroid vector, i.e. the visual word. All of these K visual words are assembled into a set of basis as a codebook. Then, for every image, we correlate the visual words with the image and use a histogram to count the number of responses for each word. Since every image has a different response, it is important to normalize the histogram. For every enquiry image, we again build the histogram according to the codebook and calculate the distance to other images.

4. Experimental Evaluation

4.1 Dataset

The visual search algorithm is evaluated on the MSRC-v2 dataset. The dataset consists of 591 images in total. The images are divided into 23 object classes with 30 images per class³. Table 4.1 shows some example images. In fact, this dataset is not designed for image retrieval test. Therefore, not all of the images are suitable to evaluate the performance. Some poor performance cases will lower the overall accuracy. But we can still use the mean average score to compare the relative performance of different algorithms.







Class 2	Class 4	Class 7
		
Class 8	Class 9	Class 13
		

Table 4.1 Some Example Classes

³ Exceptions: Class 10(32 images), Class 12(34 images), Class 15(24 images), Class 20(21 images)

4.2 Evaluation Methodology

One way to visualize the performance of the system is to use the Precision-Recall curve(PR-curve). Suppose we retrieve the first N similar images. Among them there are T images with correct labels and totally S images with this label in the dataset. Then the precision and the recall are defined as

$$Precision(N) = \frac{T}{N} \quad (4.1)$$

$$Recall(N) = \frac{T}{S} \quad (4.2)$$

As N varies, each (Precision, Recall) pair corresponds to a point in the PR curve (Figure 4-1). The ideal curve will approach the upper-right corner.

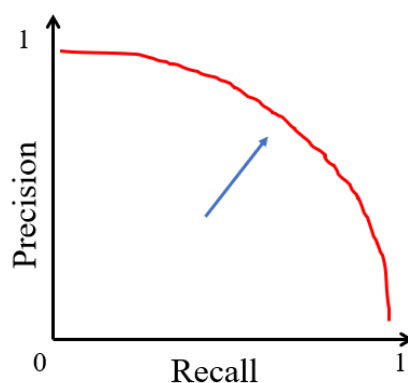


Figure 4-1 Precision-Recall Curve

PR curve is suitable for visualization for the results. To evaluate the performance quantitatively, we can calculate the Average Precision (AP) at N [10] for each query image

$$AP \text{ at } N = \frac{1}{N} \sum_{i=1}^N P(i) \cdot rel(i) \quad (4.3)$$

where $P(i)$ denotes the precision at i . $rel(.)$ is a binary function, which will return 1 for correct image and 0 for incorrect image. N is the number of the retrieved image we evaluate on. If we set N as the total number of relevant images, the equation is called AP. AP is suitable for a dataset with imbalanced classes. In this case, since the number of samples in each class is constant, there is no difference for us to use AP or AP at N . For simplicity, we will omit the “at N ” in the expression of AP at N .

AP can be used to evaluate the performance of each enquiry. To evaluate the overall performance, we can calculate the mean of AP through all the images in the database.

$$MAP \text{ at } N = \frac{1}{K} \sum_{j=1}^K AP_j \text{ at } N \quad (4.4)$$

where K is the total number of images in the dataset.

4.3 Performance Analysis

4.3.1 Performance of Color-Based Methods

4.3.1.1 Performance of Global Color Histogram

Figure 4-2 shows the average precision of Global Color Histogram. From the heatmap, we can find the best performance is achieved when we use 16 quantization bins in each color channel without PCA. However, the size of the descriptor is 4096 for this case, leading to an intensive computation⁴. If we use PCA, the dimension can be dramatically reduced into 159*1, which is quite compact. The performance drop is acceptable in most of the cases. Therefore, in later experiments, PCA will be implemented as a mandatory process rather than an option.

Another variable is the number of quantization bins. If we increase it from 8 to 16, the performance increase since we have a better color resolution. The impact of L2, Cosine and Pearson distance metrics seems to be homogeneous in this test. However, Mahalanobis distance is quite slow and leads to inferior performance. The reason is already discussed in section 2.4.2. In short, the Mahalanobis distance will magnify the importance of the small-variance dimensions, which makes no sense in this problem.

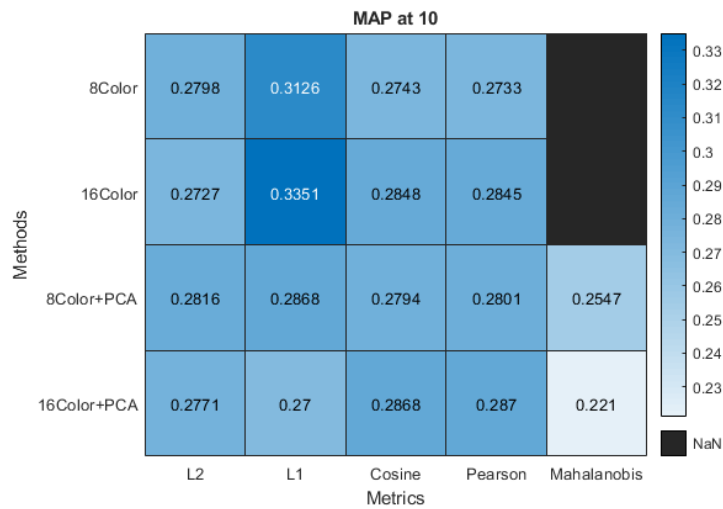


Figure 4-2 Average Precision of Global Color Histogram

⁴ 2.06 seconds and 21.73 seconds for q=16 with/without PCA

Especially, in the non-PCA cases, the performance will improve a lot if we use L1 distance. To explain this, we can further investigate the precision of different classes in Figure 4-3. It is shown that L1 norm improves the accuracy in class 2(trees), 4(cows), 9(sheep), 13(books) and 17(streets). All these three classes have very distinctive color distribution. For example, the sheep are all white and the grasses are all green. We can simply distinguish these images directly by the major color. Thus, it is reasonable to use L1 to filter out some outlier features, such as soils and rivers.

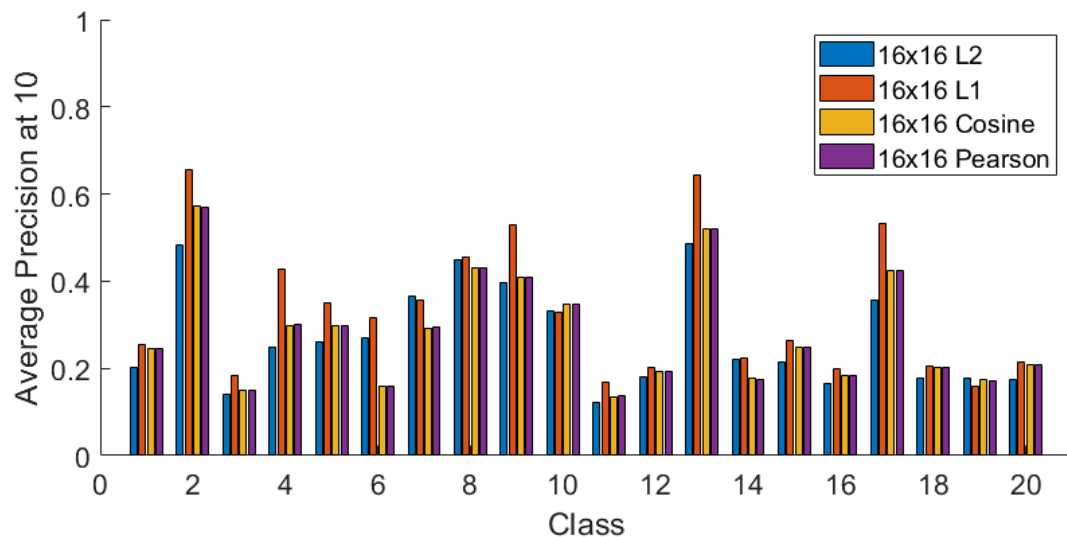


Figure 4-3 Performance on Different Classes of Global Color Histogram

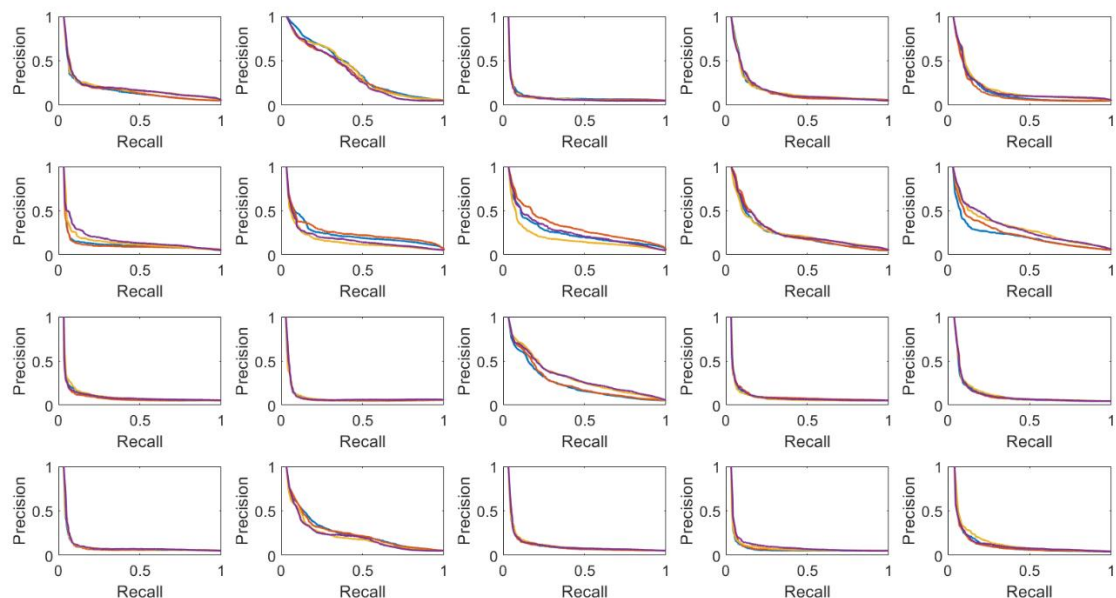


Figure 4-4 PR Curve of Different Methods

4.3.1.2 Performance of Grid-based Color Histogram

In the grid-based method, since we need to store the covariance matrix for PCA, the descriptor cannot be too large. Considering this memory limitation, the maximum number of bins is set to 8. From Figure 4-5, we can find the most appropriate choice is 3x3 grids + 8 bins histogram + Pearson metric. In most of the images, the objects are roughly of the same size and the same location. A coarse grid, e.g. 3x3, is enough for spatial resolution.

Comparing the first two columns in Figure 4-5, we can find the distance-based metrics have poorer performance as the grids become finer. Because if we use a finer grid, we need some tolerance of spatial shifts of the objects. The L1 and L2 norm compare each dimension independently. If the same object appears in two different image patches, L1 will treat them as outliers, while L2 will give a large penalty to them, both leading to low accuracy. In contrast, the correlation-based metrics give tolerance to object shifts. In other words, it allows the comparison of adjacent grids. Therefore, the correlation-based metric will be a good choice for methods with fine grids.

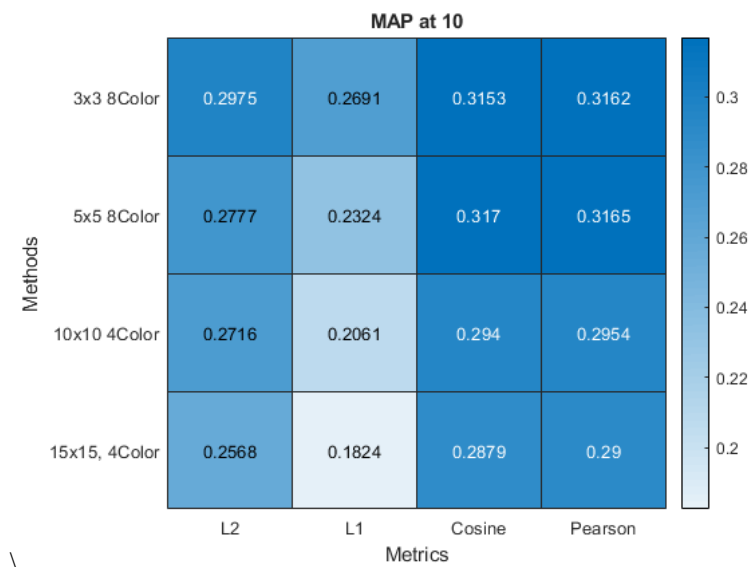


Figure 4-5 Average precision of Gridded Color Histogram

Figure 4-6 shows the average precision of different classes. We can find the gridded methods behave pretty good on class 2(tress). The spatial information helps to improve the accuracy in this case.

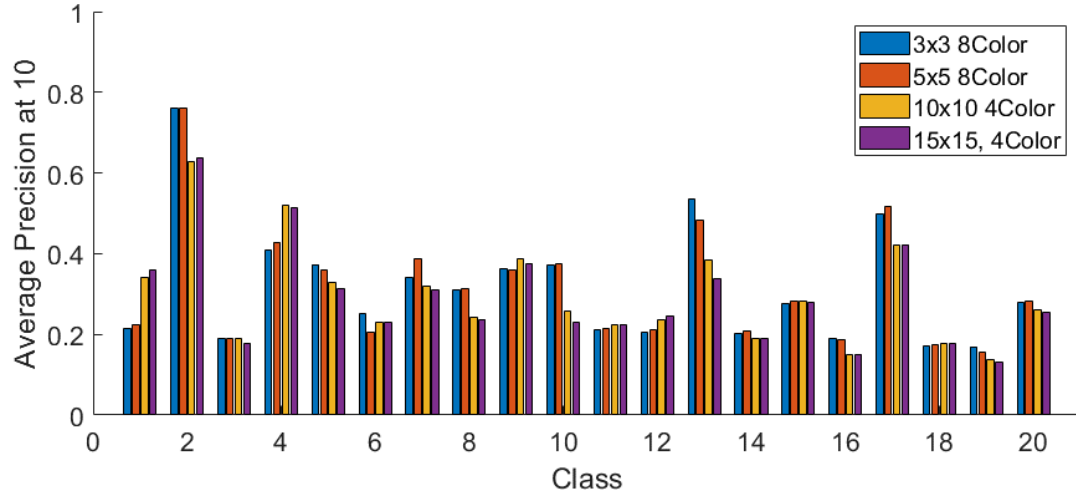


Figure 4-6 Performance on Different Classes of Gridded Color Histogram

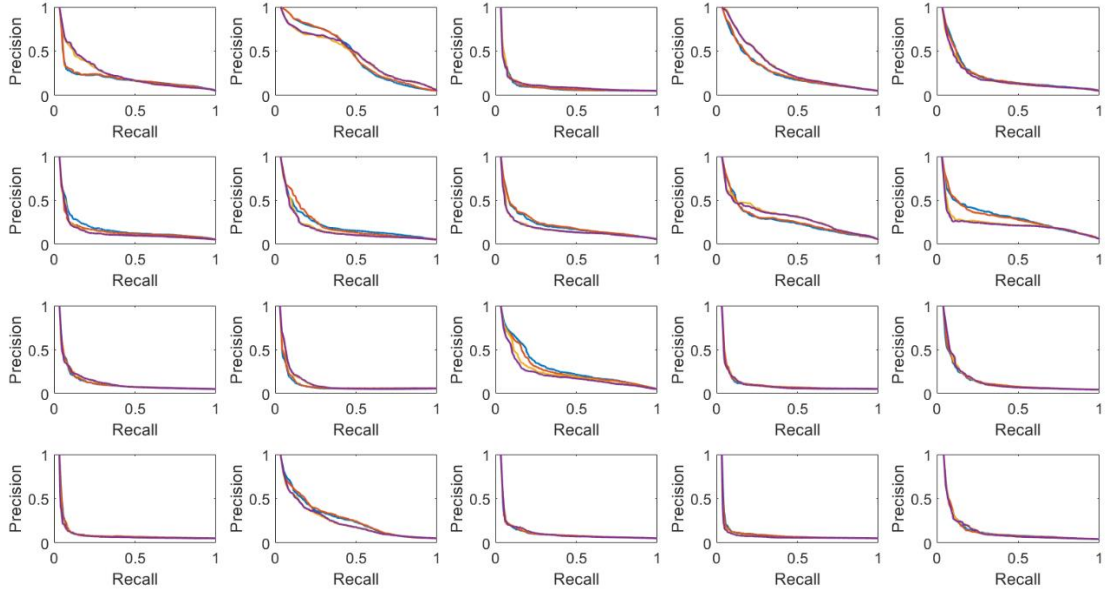


Figure 4-7 PR Curve of Different Methods

4.3.2 Performance of Edge-Based Methods

4.3.2.1 Performance of Edge Oriented Histogram

In EOH, we use the edge-oriented histogram and mean edge color to assembly the descriptor vector. It can provide us with both the texture information and the spatial color distribution.

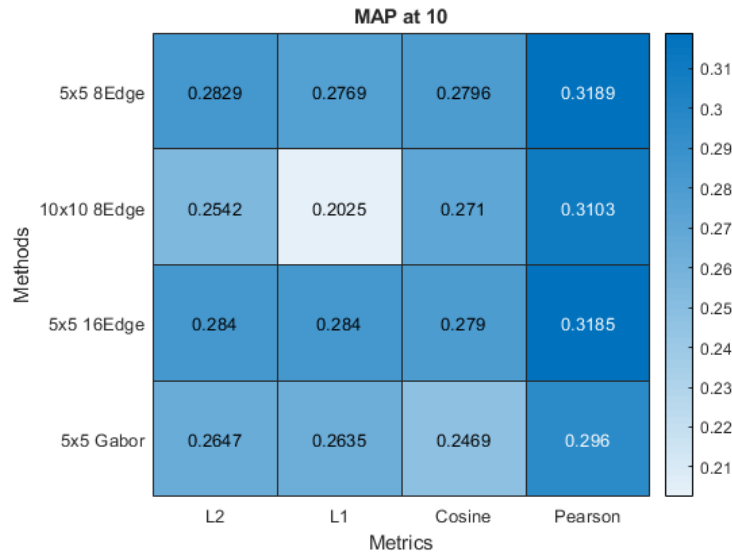


Figure 4-8 Average Precision of Edge Oriented Histogram

In figure , comparing the first and the third row, if we increase the edge resolution, the overall performance is almost the same. The reason for that is the edges are continuous theoretically. For example, if there is a circle, the increase of edge orientation resolution will take no effects for identifying it. Thus, we can think 8 bins are enough for the edge orientation. Again, the finer grids do little help with the accuracy. Comparing the performance of Edge Oriented Histogram and Gabor Wavelet, we can find the Edge orientation has better performance.

Comparing Figure 4-6 and Figure 4-9, the most significant improvement occurs in class 4(jet planes). The texture information helps to identify it from trees and houses, which all have similar backgrounds. There are also some drops in class 2(trees) and 17(streets). If we check the image of streets, we can find it includes many objects such as cars trees and houses, which can lead to the confusion of the texture-based method.

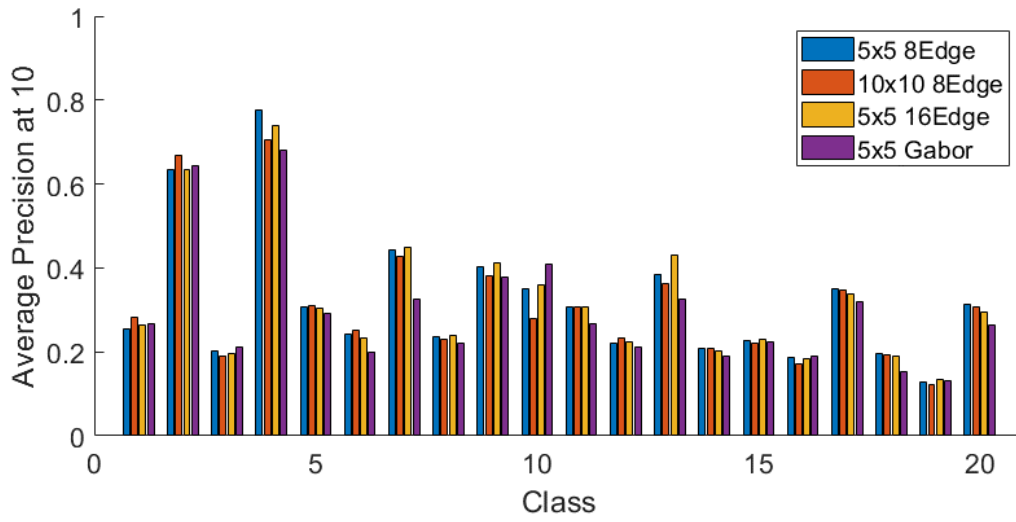


Figure 4-9 Performance on Different Classes of Edge Oriented Histogram

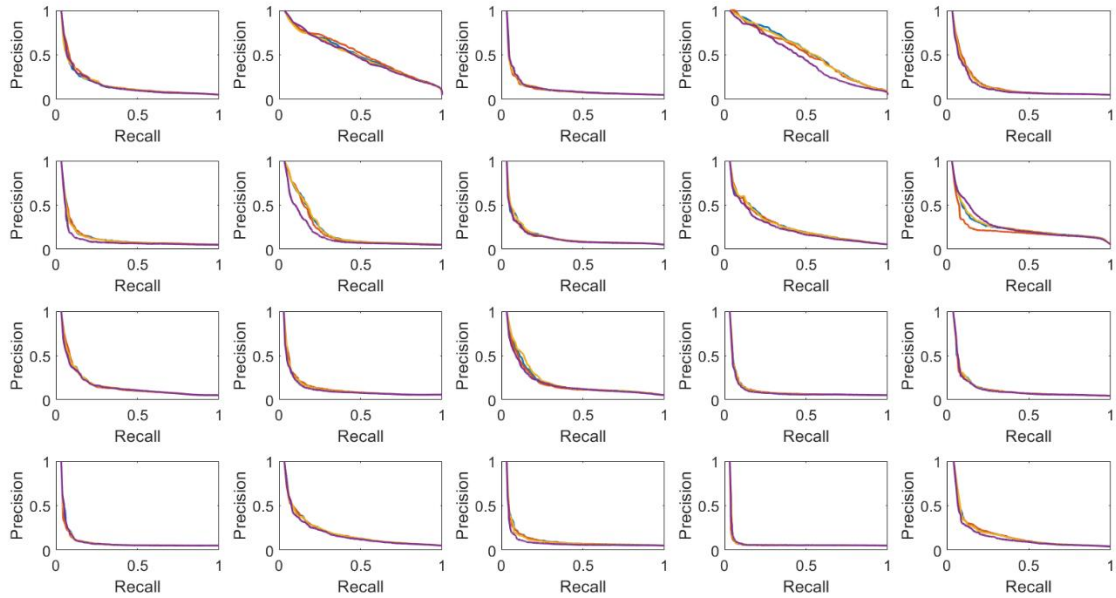


Figure 4-10 PR Curve of Different Methods

4.3.2.2 Performance of Color & Edge Oriented Histogram

In the previous section, we find the gridded color histogram can achieve better performance. Thus, we can combine the color histogram and the edge-oriented histogram together. From Figure 4-11, we can find the average precision do increase while the size of the descriptor is on the same level after PCA⁵. Different combinations of edge bins and color bins are tested. The best performance is achieved when there are 12 color bins and 8 edge bins, which is 0.2 higher

⁵ The details are recorded in APPENDIX.

than the edge + mean color method.

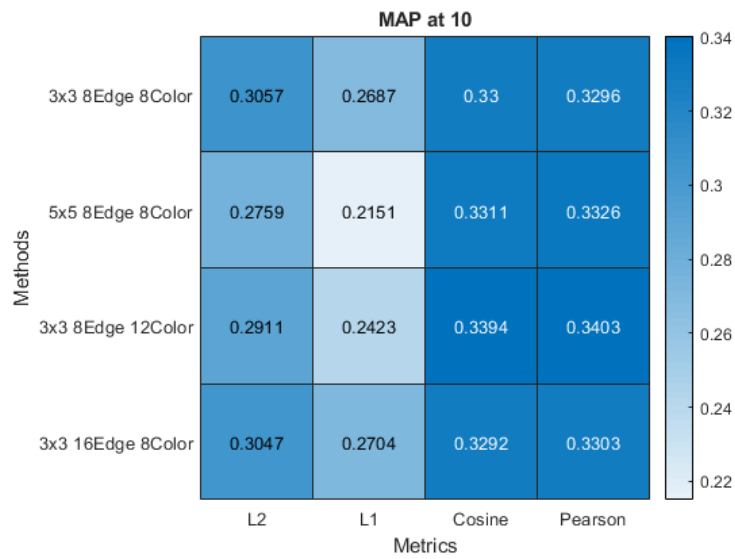


Figure 4-11 Average Precision of Color & Edge Oriented Histogram

In Figure 4-12, the AP for different classes is provided. Compared to the PA of color grid in Figure 4-6, we can find there are some improvements in class 6(faces), 7(cars), 8(bikes) and 10 (flowers). In these classes, the objects have different colors but similar textures. Thus, edge oriented provides useful information for similarity measurement.

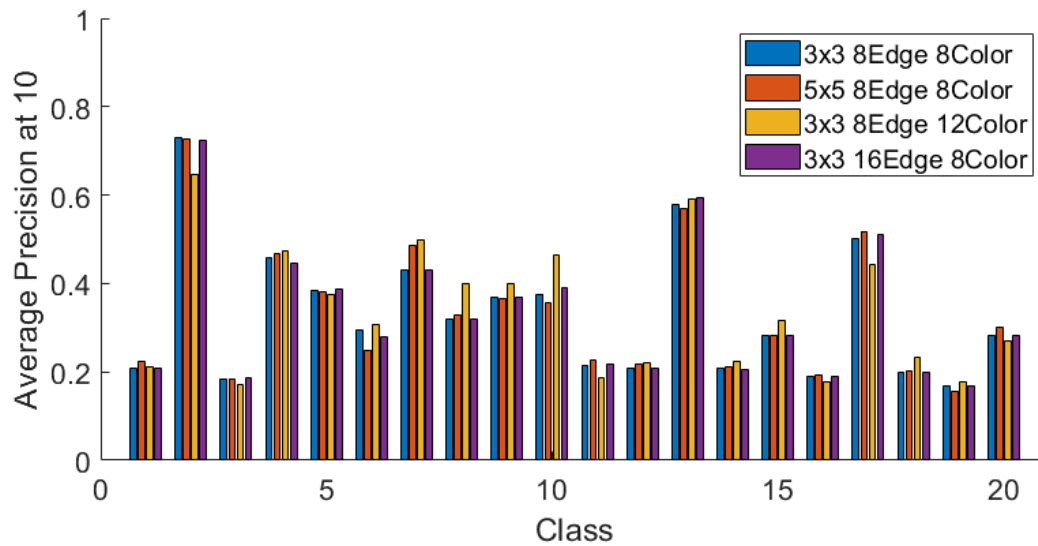


Figure 4-12 Performance on Different Classes of Color & Edge Oriented Histogram

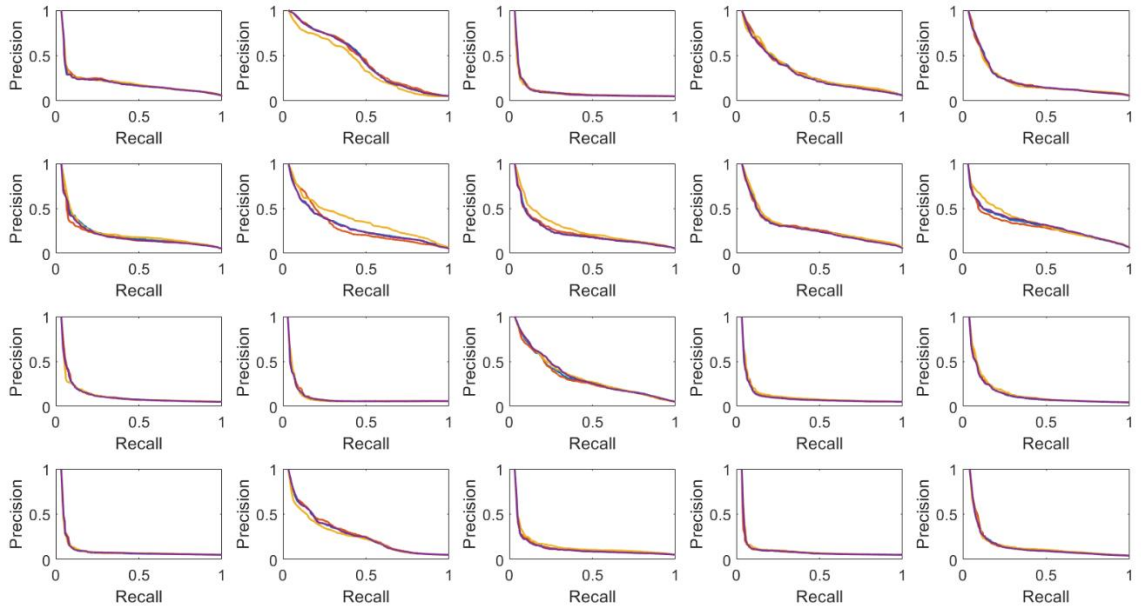


Figure 4-13 PR Curve of Different Methods

4.3.3 BoVW Methods

In this section, the performance of BoVW methods is tested. Both the advantages and limitations are discussed.

Firstly, the computation effort of the built-in k-means function and the approximate k-means are compared. The test case is to cluster 115081 64-dimensional feature vectors into 1000 words. The elapsed time is 185.34 seconds and 18.92 seconds for k-means and approximate k-means respectively. The approximate k-means is 100 times faster in Matlab. In figure , we can find the performance of approximate k-means method (denoted by AK) is acceptable. Therefore, it is reasonable to use approximate k-means to replace the k-means for this visual search problem.

In Figure 4-14, the first four rows represent different combinations of features and number of words. We can find the accuracy can improve a little bit by increasing the number of words. The performance of Harris detector + SIFT descriptor is comparable to complete SIFT features. In fact, the number of extract features is much larger than the number of words. Both of the detectors will provide redundant information, leading to a similar performance. However, the different choices of detector may affect some specific cases. This is beyond the scope of this report. In these four rows, the selection of L2 norm yields to high performance. It is consistent with the metric selection when building the word histogram.

The two methods ended with BI, means they use built-in Matlab bog functions. We can find the performance of L1 norm is extremely low in these two cases. The reason is Matlab does not do normalization to its built-in BoVW histogram. Therefore, the L1 norm can return a big error while the correlation-based metrics show a good performance.

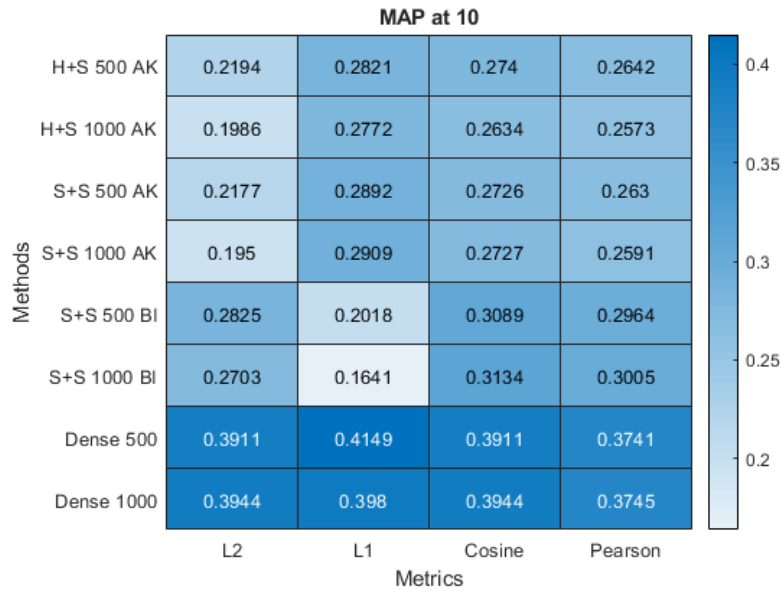


Figure 4-14⁶ Average Precision of BoVW Histogram

In the last two rows of Figure 4-14, the dense feature method shows the best performance among all the other tested methods. From Figure 4-15, we can find the dense feature method outperform the others in class 4(jet planes), 6(faces), 7(cars), 10(flowers), 12(birds) and 20(boats). All of these images have distinctive objects of similar size, for example, the human faces and flowers. Therefore, the dense feature based BoVW, which both provides spatial and semantic information, performs very well in these classes. By contrast, in class 11(road signs), the performance of the dense method is much worse than the local method. Because there are no textures in most of the grid point in those road sign images.

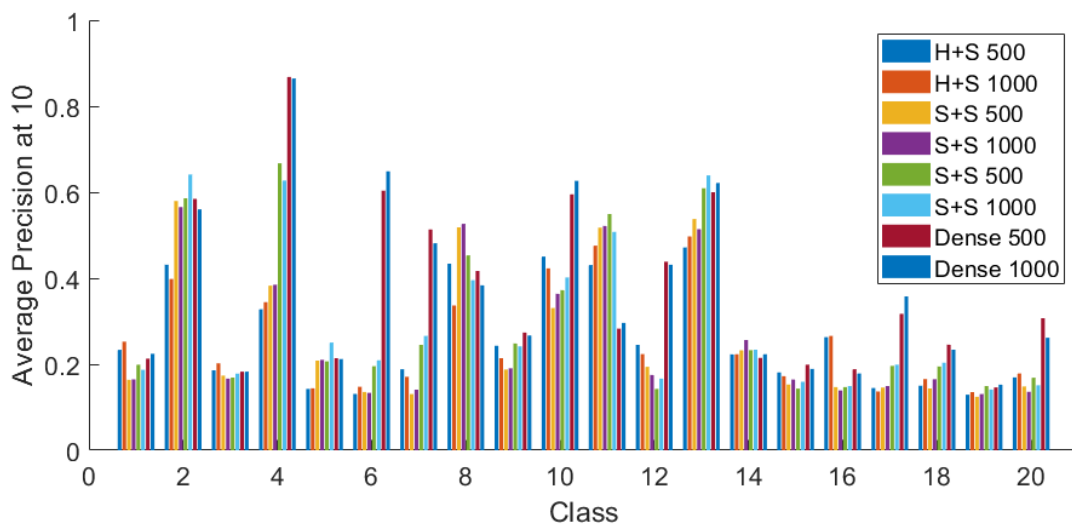


Figure 4-15 Performance on Different Classes of BoVW Histogram

⁶ H denotes Harris, S denotes SIFT, AK denotes Approximate K-means; BI denotes Built-in and S denotes SURF

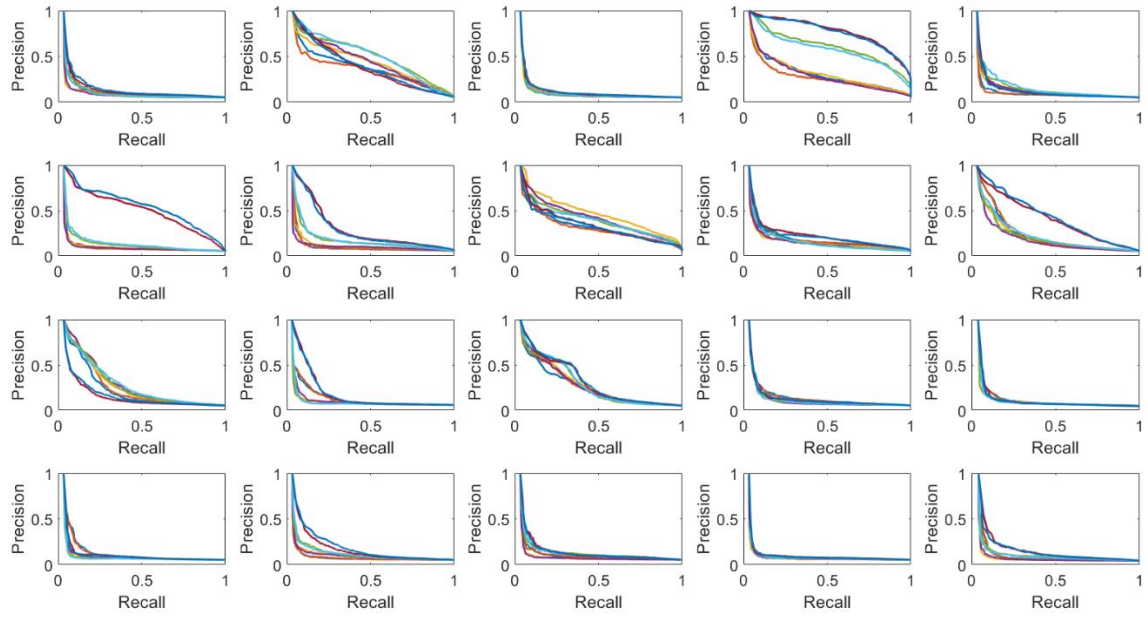


Figure 4-16 PR Curve of Different Methods

5. Conclusion

In this report, several visual search algorithms are introduced and compared. The best performance is achieved in the dense-feature-based BoVW method. Moreover, the effect of grid size and bin size for color and texture channels are discussed. The key point is to select a suitable resolution which is suitable for the dataset.

From the experiment results, we can find that the proposed algorithms only have a good performance in some specific classes. In general, low-level features are not enough for describing all kinds of class in visual search problem. To further improve the performance, we need to use more semantic information.

References

- [1] C. Tsai, "Bag-of-Words Representation in Image Annotation: A Review", *ISRN Artificial Intelligence*, vol. 2012, pp. 1-19, 2012. Available: 10.5402/2012/376804.
- [2] J. Collomosse, "Computer Vision and Pattern Recognition Lecture 7-9", University of Surrey, 2019.
- [3] R. Datta, D. Joshi, J. Li and J. Wang, "Image retrieval", *ACM Computing Surveys*, vol. 40, no. 2, pp. 1-60, 2008. Available: 10.1145/1348246.1348248.
- [4] J. Mitro, "Content-based image retrieval tutorial", *arXiv preprint*, *arXiv:1608.03811*, 2016.
- [5] Yu-Gang Jiang, Jun Yang, Chong-Wah Ngo and A. Hauptmann, "Representations of Key point-Based Semantic Concept Detection: A Comprehensive Study", *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 42-53, 2010. Available: 10.1109/tmm.2009.2036235.
- [6] Uchida, Yusuke, "Local feature detectors, descriptors, and image representations: A survey.", *arXiv preprint*, *arXiv:1607.08368*, 2019.
- [7] Philbin, James, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. "Object retrieval with large vocabularies and fast spatial matching." In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8. IEEE, 2007.
- [8] Pelleg, Dan, and Andrew Moore. "Accelerating exact k-means algorithms with geometric reasoning." Carnegie Mellon University, 2000.
- [9] T. Lee, "Image representation using 2D Gabor wavelets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 959-971, 1996. Available: 10.1109/34.541406.
- [10] M. Ozsu and L. Liu, *Encyclopedia of database systems*. New York: Springer, 2009.

Appendix

I. Data for Global Color Histogram

Parameters	Distance	Precision At 5	Precision At 10	Precision At 15
8 bins 512*1	L2	0.4648	0.3355	0.2758
	L1	0.5028	0.3702	0.3068
	Cosine	0.4568	0.3313	0.2756
	Pearson	0.4548	0.3299	0.2579
8 bins + PCA 59*1	L2	0.4621	0.3353	0.2768
	L1	0.4745	0.3432	0.2835
	Cosine	0.4628	0.3347	0.2800
	Pearson	0.4606	0.3374	0.2827
	Mahalanobis	0.4352	0.3033	0.2444
16 bins 4096*1	L2	0.4556	0.3264	0.2620
	L1	0.5341	0.3875	0.3199
	Cosine	0.4658	0.3402	0.2812
	Pearson	0.4651	0.3402	0.2810
16bins + PCA 157*1	L2	0.4616	0.3290	0.2687
	L1	0.4517	0.3193	0.2557
	Cosine	0.4706	0.3420	0.2867
	Pearson	0.4659	0.3425	0.2883
	Mahalanobis	0.4099	0.2590	0.1984

II. Data for Grid Based Color Histogram

Parameters	Distance	Precision At 5	Precision At 10	Precision At 15
5x5 PCA 8 bins 363*1 (12800*1)	L2	0.4131	0.2709	0.2125
	L1	0.4557	0.3209	0.2641
	Cosine	0.5045	0.3731	0.3115
	Pearson	0.5039	0.3728	0.3126
	Mahalanobis	0.2906	0.1810	0.1407
3x3 PCA 8 bins 248*1 (4608 *1)	L2	0.4562	0.3176	0.2563
	L1	0.4857	0.3500	0.2869
	Cosine	0.5028	0.3722	0.3122
	Pearson	0.5036	0.3733	0.3132
	Mahalanobis	0.3642	0.2265	0.1756
10x10 PCA	L2	0.4438	0.3259	0.2727

4 bins 331*1 (6400*1)	L1	0.3732	0.2452	0.2015
	Cosine	0.4775	0.3538	0.2997
	Pearson	0.4804	0.3544	0.3028
	Mahalanobis	0.2645	0.1664	0.1364
15x15 PCA 4 bins 395*1 (14400*1)	L2	0.4270	0.3112	0.2602
	L1	0.3391	0.2205	0.1781
	Cosine	0.4728	0.3479	0.2938
	Pearson	0.4739	0.3498	0.2970
	Mahalanobis	0.2581	0.1582	0.1287

III. Data for Edge-based Methods

Parameters	Distance	Precision At 5	Precision At 10	Precision At 15
5x5 8 bins 68*1 (original275*1)	L2	0.4625	0.3324	0.2738
	L1	0.4686	0.3392	0.2759
	Cosine	0.4538	0.3306	0.2705
	Pearson	0.4980	0.3785	0.3164
10x10 8 bins 206*1 (1100*1)	L2	0.4297	0.3105	0.1969
	L1	0.3765	0.2432	0.2572
	Cosine	0.4472	0.3200	0.2591
	Pearson	0.4909	0.3684	0.3109
5x5 16 bins 66*1 (475*1)	L2	0.4627	0.3423	0.2818
	L1	0.4712	0.3399	0.2773
	Cosine	0.4525	0.3308	0.2695
	Pearson	0.4988	0.3766	0.3151
5x5 Gabor 139*1 (1275*1)	L2	0.4143	0.2957	0.2433
	L1	0.4274	0.3062	0.2594
	Cosine	0.4259	0.3046	0.2497
	Pearson	0.4372	0.3200	0.2705

IV. Data for Edge + Color Histogram

Parameters	Distance	Precision At 5	Precision At 10	Precision At 15
5x5 8Color 8Edge 379*1 (13200*1)	L2	0.4630	0.3224	0.2638
	L1	0.3923	0.2546	0.2012
	Cosine	0.5210	0.3879	0.3276
	Pearson	0.5207	0.3894	0.3296

3x3 8Color 8Edge 265*1 (4752*1)	L2	0.4910	0.3550	0.2920
	L1	0.4622	0.3154	0.2525
	Cosine	0.5196	0.3878	0.3270
	Pearson	0.5185	0.3894	0.3282
3x3 12Color 8Edge 347*1 (15696*1)	L2	0.4260	0.2821	0.2239
	L1	0.4770	0.3343	0.2734
	Cosine	0.5262	0.3961	0.3369
	Pearson	0.5255	0.3969	0.3397
3x3 8Color 16Edge 363*1 (4896*1)	L2	0.4877	0.3516	0.2874
	L1	0.4593	0.3154	0.2514
	Cosine	0.5125	0.3833	0.3227
	Pearson	0.5122	0.3839	0.3239