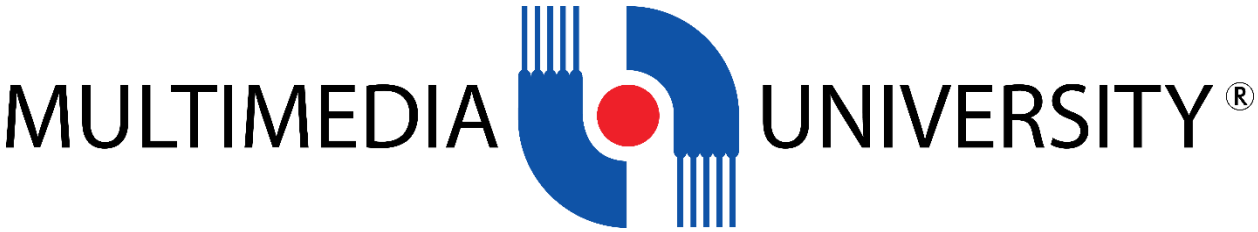

TDS 3651 Visual Information Processing

Assignment 2

Retrieval of Local Food through Similarity Measure

Prepared by:
Choe Choon Ho - 1132702963
Wong Tiong Kiat - 1132702943

Lecturer & Tutor:
Dr. John See

Contents

1. Abstract

Given 1000 images of local food along with their classes, we are tasked to extract the feature(s) of each image and compute the similarity measure between images. Types of local food contained in the database include 'Ais Kacang', 'Banana Leaf Rice', 'Cendol', 'Curry Laksa', 'Durian', 'Maggi Goreng', 'Nasi Lemak', 'Pisang Goreng', 'Roti Canai' and 'Satay'. There 100 images for each category in the database. The benchmark method was a RGB color histogram with 64 bins for each color. We propose a method based on SIFT. Each SIFT key points contains 128 features, which are obtained by taking the 16×16 neighbouring pixels around the key point, dividing them into 4×4 sub-region and constructing the histogram of orientations using 8 bins for each sub-region. We then cluster these features using k-means clustering to form a "codebook" or visual word "index". Then, we extract the SIFT features of the input image, perform vector quantization of these features with the codebook and construct a "visual word histogram" or "bag of words", and use cosine similarity measure as its distance.

2. Introduction

Given 1000 local food images as our database, we are tasked to extract certain feature(s) and compute the distance between these feature(s) of different images. The distance will be used for a retrieval task and will be judged on its precision and recall given by,

$$precision = TP \div (TP + FP)$$

$$recall = TP \div (TP + FN)$$

where *TP, FP and FN* are *true positives*, *false positives* and *false negatives* respectively. The benchmark algorithm given to us was a color histogram in RGB space with 64 bins for each color. The algorithm scored a precision of 20.96% and a recall of 15%. We have decided to use SIFT features and vector quantization to compare images as color histogram does not capture the spatial information of the images. Our other attempts are brief in the “Other Attempts” section. We used k-means clustering to cluster the SIFT features, k was determined empirically. After quantizing the SIFT features, we construct its “bag of words” histogram and use cosine distance to compare the images.

3. Methodology

3.1 Grayscale Conversion via PCA

We first converted the image into grayscale using PCA. PCA is used to transform the RGB image into its principal components where the first principal component captures the most variation in the image, the second captures the second most… We took the first component as our grayscale.

3.2 Reducing Region of Interest (ROI)

As the region of interest (food) is usually at the center of the image. We truncated the image by 5% on each side in order to reduce the background which may be captured by SIFT.

3.3 Contrast Limited Adaptive Histogram Equalization (CLAHE)

We then performed CLAHE on the grayscale image with a clip size of 4 and a tile size of 8×8 to improve the contrast of the grayscale image.

3.4 Extract SIFT Features

We then extracted the SIFT features of the grayscale image.

3.5 K-means Clustering

We performed k-means clustering on the SIFT features extracted from 200 random samples, 20 from each category to ensure a fair distribution for each category of food and set k to 68. The cluster centers are used as a “codebook”. Both 200 samples and 68 clusters or visual words was chosen arbitrary after some empirical results which can be seen in Table 1. We decided to stop after with 200 samples with k as 68 as empirically tuning the values is time consuming.

3.6 Vector Quantization

Our feature extraction consists of step 3.1 to 3.4. Next we performed vector quantization on the features with the “codebook” from step 3.5 to construct a histogram or “bag of visual words”.

3.7 Cosine Distance

We used the cosine similarity to compare the “bag of visual words” obtained from 3.6.

Samples	**k**	**Precision** (%)	**Recall** (%)
100	128	25.92	15
200	48	28.41	16
200	**64**	**28.67**	18
200	**68**	27.72	**21**
200	72	27.41	18
200	81	27.32	17
300	64	27.92	6

Table 1: Empirical Results to Determine k

*** Note: Results may vary even with the same parameters as k-means clustering may not return the same cluster centers.**

4. Results & Analysis

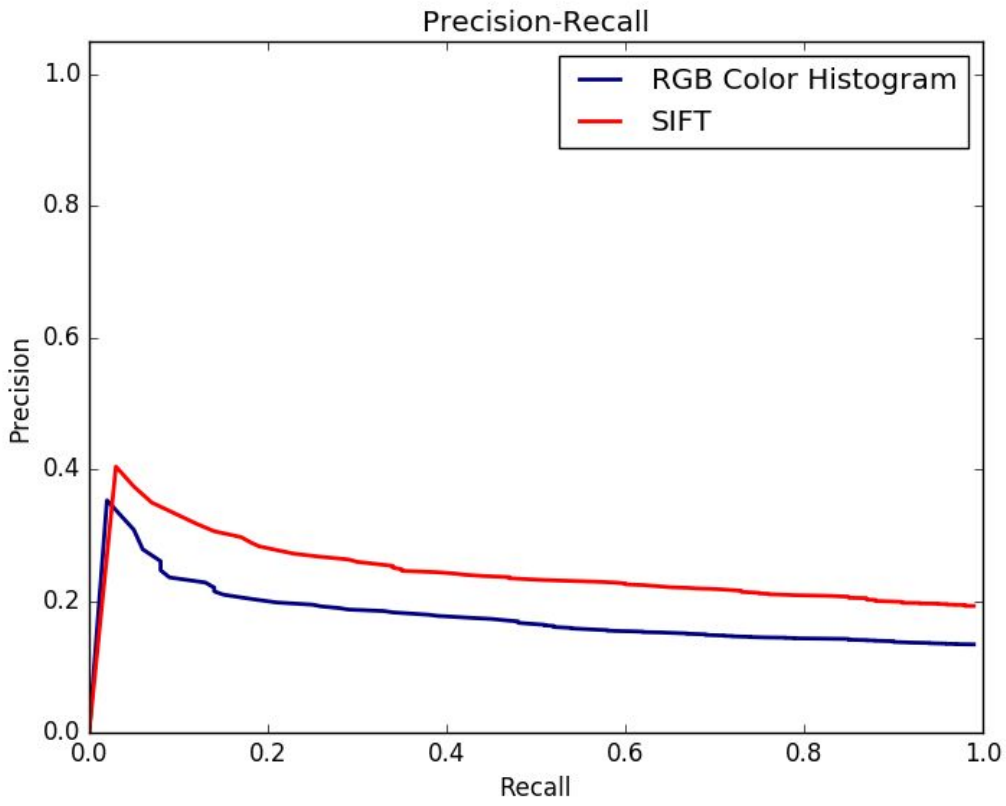

Image 1: Precision-Recall Curve

The RGB histogram differences for every pair of images in the database are computed and stored in a distance matrix. The top (number of retrievals) that have the least distance to the query image are selected out, then number of relevant images (having the same class as the query image) are counted.

Next, the distance between all pairs of images in the database are calculated using RGB histogram and stored in a 2D array. Then, we compared each image with its actual class whereby the number of correctly retrieved images are used to calculate the average precision using the formula given below,

$$Average\ Precision,\ AP_q = (1 \div K) \times \sum_{k=1}^{K} (r_k \div k)$$

The average precision array is then divided by the number of retrievals using the formula given below,

$$Mean\ Average\ Precision,\ MAP = (1 \div Q) \times \sum_{q=1}^{Q} AP_q$$

The recall is then calculated by taking the average precision dividing it by the number of images per class, where the formula is given below as,

$$Recall\ rate = (1 \div Q) \times \sum_{q=1}^{Q} r_k \div R$$

The similar method is then applied to the proposed method. However, we used cosine similarity instead of Euclidean distance.

We increased the number of retrievals by 10 every iteration until 999, and the average precision, mean average precision and recall obtained each iteration are stored in an array and used to plot the precision recall curve.

We tested our SIFT-based approach using the command: $python\ queryEval.py\ -d\ 100$

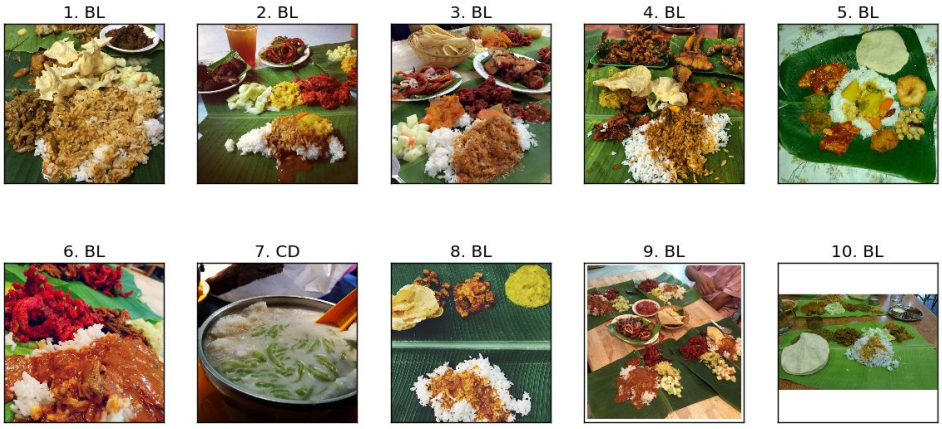

Image 2: Example of Query

9 out of 10 returned results are banana leaf where the average precision for this query is 0.8264, and recall rate is 0.56.

5. Suggestions for Improvement

In our proposed method, we only trained the codebook with 200 images and set k to 68. We could further improve this by feeding it the full dataset and further tuning the k empirically or personally eyeballing the data set and select the best representative(s) for each class to build the codebook. However, this will require a lot of processing power and time. Alternatively, we could perform hierarchical clustering instead of k-means.

As the shape and color of the food varies even in the same category, a lot of distinct points will be captured by SIFT in which they will produce a different visual word. Therefore, SIFT features alone is not suitable for this task. Color histogram and textons may be better, however, pyramids might be needed to extract the texture information as the region of interest (food) varies in scale. A weighted sum can then be used to measure the distance where more emphasis should be given on the texture information.

6. Other Attempts

6.1 Convolutional Neural Network (CNN)

Our first attempt was using CNN to aid in classification of the image. We first truncated the input image by 5% on each sides to reduce background information and further resize the truncated image to a size of 64 x 64 to reduce the time needed for CNN. We then train the model with the preprocessed images. For our “computeFeatures.py”, we first obtained the class label of the image. Then created a tuple for label with the color histogram as features. For our “computeDistance.py”, we compared the classes of the images and set the distance to 0 if the images have the same class. If they do not match, we applied color histogram comparison using Euclidean distance to obtain the similarity measure. However, we decided not to proceed with CNN as CNN was too much of a “black box” and we could not write much in our report. Furthermore, we are lacking in time to study CNN due to the time constraints.

6.2 Textons using Gabor kernels

Our second attempt was extracting the histogram of textons together with the color histogram in which we will apply a weighted distance to place more emphasis on the textons histogram. We used scipy Gabor Kernels to generate 31 filters of different orientation and frequency and convoluted them with the image. Next, we encoded each pixel with the index of the filter which gave the highest response. We then construct a histogram of the encoded image. We used Euclidean distance to compute the distance between the histograms. However, precision and recall yield was slightly higher than the benchmark as we only used filters that detect edges. We can improve this method by adding different type of kernels like Gaussian, Laplacian (Blob), cylinder-like kernels to detect rice shape… since Gabor kernels detects edges only. However, this method requires a lot of computation as it performs x convolutions on each image where x is the number of kernels. Empirically running different types of kernels will be too expensive and therefore decided to try a different approach.