CS3242 assignment 3 report
Content-based image retrieval

Luong Minh Thang & Nguyen Quang Minh Tuan

1. INTRODUCTION

In content-based image retrieval (CBIR) system, color and edge are fundamental features used widely in many implementations. In this report, we investigate in various techniques involving around color and edge features. Besides, implementing those matching strategies that appears frequently in many CBIR system, we have also proposed some other new approaches, and demonstrated that our system has achieved significant improvement of 2.01% from the baseline system.

Our work consists of three main parts including color approach, edge/direction feature approach, and dynamic-confidence merging. In color approach, we compare histograms in YCrCb color space, adding color perceptual similarity, as well as color coherence. On the other approach with edge/direction feature, we have tried out several new techniques which are 4-step edge detection, and edge coherence together with entropy measurement. Lastly, we combined the results of color and edge features dynamically based on the query image with our new definition of confidence retrieval values.

2. COLOR FEATURE MATCHING

In this section, we describe our color feature used for image matching, in which we employ histogram matching technique. Each image histogram is first extracted out, and is represented by hist[1..N], in which N is the total number of colors, and hist[i] is the frequency value of each color i. The idea of histogram matching is that we compute the overall similarity between the query image histogram, *query[1..N]*, and each dataset image histogram, *data[1..N]*, by going through each bin in the color histograms to see how corresponding the values are. Our baseline similarity computation, which we called *exact similarity*, is formulated as below:

$$\text{overall\_sim} = \sum_{i=1}^{N} \text{query}[i] * sim\_exact(\text{query}[i], \text{data}[i]) \quad (1)$$

$$= \sum_{i=1}^{N} \text{query}[i] * (1 - \frac{|\text{query}[i] - \text{data}[i]|}{\max(\text{query}[i], \text{data}[i])}) \quad (2)$$

In the above formula, we multiply the $sim(\text{query}[i], \text{data}[i])$ to query[i] as a weight so that frequent colors n the query image will contribute significant to the overall similarity value. This formula, however, has a disadvantage that comparisons are only made

between two exactly the same colors, but not between colors that look very similar perceptually, e.g. between color i and color (i-1). Thus, an improvement for the exact similarity above is to consider perceptual similarity of each query color i with other similar colors in the dataset image. In order to accomplish that, we first compute the color similarity matrix by denoting each color i by its component values Y_i, Cr_i, and Cb_i. The similarity between color i and color j is based on $\text{diff}[i][j] = \frac{\sqrt{(Y_i-Y_j)^2+(Cr_i-Cr_j)^2+Cb_i-Cb_j)^2}}{\text{max\_diff}}$, in which:

$$\text{color\_sim}[i][j] = \begin{cases} 1 - \text{diff} & \text{diff} < \text{t\_color} \\ 0 & \text{diff} \geq \text{t\_color} \end{cases} \quad (3)$$

A color threshold *t_color* is used to exclude those colors which are very different from the reference color. We set *t_color* to be equal to the average value of all entries in the diff matrix, which is 0.5607. From the color similarity matrix, the perceptual similarity values for each color i in the query image is computed as below:

$$\text{sim\_per}_i = \sum_{j=1}^{N} \text{color\_sim}[i][j] * \text{sim\_exact}(\text{query}[i], \text{data}[j])$$

After having the perceptual similarity values, the abstract overall similarity formula:

$$\text{overall\_sim} = \sum_{i=1}^{N} f(\text{sim\_exact}, \text{sim\_per})$$

There are several different ways to specify f as a way to combine the exact and perceptually values. e.g. $f(\text{sim\_exact}, \text{sim\_per}) = \text{sim\_exact} * (1 + query[i] * \text{sim\_per})$ as mentioned in the lecture notes, or $f(\text{sim\_exact}, \text{sim\_per}) = \text{sim\_exact} * query[i](w + (1 - w)\text{sim\_per})$. However, after experimenting with many different combinations, we come to conclude that the performance is much better when sim_per is considered independently with sim_exact through $f(\text{sim\_exact}, \text{sim\_per}) = w*\text{sim\_exact}+(1-w)*\text{sim\_per}$. Detailed experiment results will be presented later in section 5.

3. EDGE/DIRECTION FEATURE APPROACH

In this part, we will discuss about our methods to generate edge-direction histogram as well as ideas about edge coherence and entropy.

3.1. Four step edge/direction histogram generator(FSED)

To generate edge-direction histogram for a JPEG image, our methods contains of 4 steps:

Step 1 - Detecting edges: In a DCT block, AC values represent the variation of luminance and color in vertical and horizontal directions. The bigger the variation is, the more likely the block represents an edge. Thus, we compute the total variation

by summing all AC absolute values across Y, Cr, Cb channels. We determine that a block represents an edge if the total variation is greater than a predefined threshold. Experimentally, we set threshold = 1150 which gives the best result.

Step 2 - Filtering unimportant edges: After step 1, we found that many detected edges are not really important which results in inaccurate retrieval. For example, consider the image of a bird (the main object) in a complex background, the number of edges from the main bird object is often outweighed by that of the complex background. Another type of unimportant edges is those inside an object. For example, in the context of a bird image, normally, the important edges are at the "boundary" of the bird, not the edges "inside" the bird. In our implementation, we tried to omit the edges that are in the background or inside the object. We assume that the blocks at the boundary of an image represents its background, and go from the boundary of the images to its center and tried to detect whether an edge is at the boundary of an object or not by considering the properties of its neighboring blocks.

Step 3: Detecting direction In this step, we try to detect the direction of the important edges in step 1 and 2. People can argue that detecting the direction of all the blocks in the images is a better choice. However, we find that if the image background is very complex, a detection of all blocks' directions will affect adversely directions of the main object. Moreover, in most of the cases, the directions along the boundary of an object is sufficient for representing the direction of the whole object. For this step, we have tried two methods, one is from the paper (Eom & Choe, 2005), the other is our new approach:

- *Approach from paper* In this approach, the authors made use the properties of DCT cells $AC_{0,1}$ and $AC_{1,0}$ to approximate the direction of a whole block. According to the paper, the coefficient $AC_{0,1}$ denotes the edge strength in vertical direction and $AC_{1,0}$ denotes the edge strength in vertical direction. Hence, using the ratio $|AC_{0,1}|/|AC_{1,0}|$, we can approximates the direction of the whole block. For example, if the ratio is near 0, the vertical direction is dominating the block. From the ratio, we categorized the direction into 4 types: horizontal, vertical, 45- and 135-degree.

- *Our new approach* After trying the approach from the paper, we find that the result is not very good, so we decided to develop our own approach. In this approach, we consider those remaining edges from step 1 and 2, which are at the boundary of some objects. For each edge[i][j], we detect its direction by considering its 8 nearby blocks divided into 4 groups as below. For each direction group, we count the number of its blocks that contain an edge, and take the direction of group with maximum count to be the direction of block[i][j]. We break the tie by considering the first direction that satisfies.

- Vertical direction group: B[i-1][j] and B[i+1][j]
- Horizontal direction group: B[i][j-1] and B[i][j+1]
- 135-degree direction group: B[i-1][j-1] and B[i+1][j+1]
- 45-degree direction group: B[i-1][j+1] and B[i+1][j-1]

Step 4: Constructing edge-direction histogram We divide the images into 4*4=16 subimages. For each subimage, we calculate the number of edges in 4 directions. Hence, together, we have 4*4*4 = 64 bins for the edge histogram. The final step is to normalize the histogram so that its sum is equal to 1.

3.2. Edge histogram similarity

After applying FSED for two images A and B, we have 2 edge histogram H_a and H_b. This section is about how to obtain a similarity value from H_a and H_b. We have tried 3 different ways to measure this similarity value:

Uniform approach: $sim = max(1 - \sum_{i=0}^{63} |H_a[i] - H_b[i]|, 0)$

Central approach: give more weight to the center of the images

$$sim = max(1 - (\sum_{i:centralpart} \alpha * |H_a[i] - H_b[i]| + \sum_{i:boundarypart} \beta * |H_a[i] - H_b[i]|), 0)$$

In the formula, a>b to represent the importance of the central part.

- Uniform approach: $sim = max(1 - \sum_{i=0}^{63} |H_a[i] - H_b[i]|, 0)$
- Central approach: give more weight to the center of the images. In the formula, we set $\alpha > \beta$ to represent the importance of the central part.

$$sim = max(1 - (\sum_{i:centralpart} \alpha * |H_a[i] - H_b[i]| + \sum_{i:boundarypart} \beta * |H_a[i] - H_b[i]|), 0)$$

- Clustering approach: try to match sub-images with high edge density together and sub-images with low edge density together. The idea is because high edge density subimages have higher chance to represent the important object:

$$sim = max(1 - \sum_{i=1}^{16} \sum_{j:direction} |B_a[i][j] - B_b[i][j]|, 0)$$

where B is a [16][4] array sorted by the density of edge in block.

After trying 3 approaches, we found that uniform approach works slightly better than the other two. The reason is that in the given dataset, many objects do not stay close

to the center of the image, and many objects expand the whole image, so the clustering approach and the central approach won't work well. Therefore, we will use the uniform approach in our final evaluation. To support the uniform approach, we also calculate some other parameters such as edge or direction percentage in the image.

3.3. Edge coherence and entropy

Besides direction histogram, we have tried edge coherence. This techniques is applied after the first 3 steps of FSED. The idea of edge coherence is that the longer an edge is, the more important it is. For example, in building, there are many edges in vertical direction. These edges form a long vertical boundary of the building. In mountain, there are a many edges in the diagonal direction forming a long diagonally line. Therefore, if an image has some long diagonal lines, it may matches up with another having the same property. In our implementation, we counted the total edge length in each direction, giving more weight to long boundary, creating a 4-bin edge-length histogram for each image and match them with each other.

In addition, we also made use of edge entropy in the image. The edge entropy is the measurement of how edges 'scatter' in the image. The more scattering the edges are, the larger is the entropy. The entropy is used together with edge coherence in our system.

4. DYNAMIC-CONFIDENCE MERGING

To combine the strengths of both color and edge features, we introduce *confidence values* when combining the results of the two features. The intuition comes from several of our observations during experiments; for example, images whose histograms centralize on a few major colors will be better matched than images where histograms span across many unimportant colors. On edge feature side, we notice that images with moderate percentage of edges will be retrieved better. As such, we compute the *color confidence value* based on the number of important colors, i.e. colors whose frequency values are greater than *freq_thres*. On the other hand, *edge confidence value* is given high value when the edge percentage is within [*low_thres*, *high_thres*]. Below is the combined similarity:

$$\text{combined\_sim} = \frac{\text{color\_conf} * \text{color\_sim} + \text{edge\_conf} * \text{edge\_sim}}{\text{color\_conf} + \text{edge\_conf}}$$

5. EVALUATION & DISCUSSION

We carries our experiments on the dataset of 100 images consisting of 10 categories such as bird, road, or desert. Each category contains 10 images, so we evaluates based on the *11 standard recall levels* (Baeza-Yates & Ribeiro-Neto, 1999). We have three main comparisons: among color-feature techniques, among texture-feature techniques, and among different combined color-texture techniques.

As described in subsection 2.1, we concluded that the retrieval performance is much better when sim_per is considered independently with sim_exact. Table 1 shows the statistics to support that conclusion at the last column sim_per is considered independently with sim_exact $f(\text{sim\_exact}, \text{sim\_per}) = w * \text{sim\_exact} + (1 - w) * \text{sim\_per}$. We have the best average precision **0.4099** compared to others.

Recall level	**Sim_exact**	**Depend. sim_per**	**Depend. sim_per**	**Indep. sim_per**
10%	1.0000	1.0000	1.0000	1.0000
20%	0.6124	0.6091	0.6139	0.6315
30%	0.4589	0.4533	0.4604	0.4744
40%	0.3910	0.3819	0.3859	0.3882
50%	0.3298	0.3314	0.3314	0.3341
60%	0.3056	0.3026	0.3029	0.3123
70%	0.2827	0.2818	0.2820	0.2845
80%	0.2523	0.2530	0.2531	0.2522
90%	0.2306	0.2302	0.2312	0.2304
100%	0.1902	0.1898	0.1911	0.1916
Average	0.4054	0.4033	0.4052	**0.4099**

Table 1. Color feature comparison among: exact similarity, dependent perceptually with w=0.9, dependent perceptually with t_color = 0.2, and independent perceptually similarity w = 0.9.

For the second evaluation, we perform comparison over edge/direction techniques, which are FSED, and edge coherence, and across normal queries as well as object queries. Object queries consisting of bird, building, penguin and sculpture category and are retrieved much better than the others (Table 2) due to clearer boundary edge. This suggests a way to further improve the system by asking users about the presence of object, and give a higher weight for FSED.

Another observation is that edge coherence and entropy's performance is not as good as FSED. We believe that it is because when implementing edge coherence, we only consider global feature. If we can divide the image into sub-images, and consider edge coherence value for each sub-images, the result for edge-coherence may significantly be improved.

Lastly, we compare the performance with our final retrieval systems in which we have combined color and edge features with/without the participation of confidence values. Table 3 has shown that our combined system with confidence values outperforms all other systems. We perform better than the baseline system at almost all recall levels, and obtain an improvement of **2.01%** from the baseline system, which we believe significant after trying out many experiments where improvements are normally less than 0.1%.

Recall level	FSED - normal	FSED - object	Edge coher.	Edge coher. - object
10%	1.0000	1.0000	1.0000	1.0000
20%	0.5681	**0.6797**	0.3738	0.4229
30%	0.4125	0.5540	0.2125	0.2352
40%	0.2929	0.3721	0.1747	0.1793
50%	0.2295	0.2886	0.1398	0.1463
60%	0.2009	0.2572	0.1309	0.1373
70%	0.1723	0.2227	0.1230	0.1277
80%	0.1560	0.1939	0.1201	0.1232
90%	0.1475	0.1817	0.1161	0.1220
100%	0.1369	0.1597	0.1140	0.1161
Average	0.3312	0.3909	0.2500	0.2610

Table 2. Texture feature comparison among: FSED - normal queries, FSED - object queries, edge coherence, and edge coherence - object queries.

Recall level	Baseline	Color	Edge	Non-confidence	Confidence
10%	1.0000	1.0000	1.0000	1.0000	1.0000
20%	0.6124	0.6315	0.5681	0.6302	0.6721
30%	0.4589	0.4744	0.4125	0.4860	0.4970
40%	0.3910	0.3882	0.2929	0.4143	0.4316
50%	0.3298	0.3341	0.2295	0.3579	0.3589
60%	0.3056	0.3123	0.2009	0.3316	0.3238
70%	0.2827	0.2845	0.1723	0.2989	0.2940
80%	0.2523	0.2522	0.1560	0.2667	0.2651
90%	0.2306	0.2304	0.1475	0.2330	0.2252
100%	0.1902	0.1916	0.1369	0.2016	0.1870
Average	0.4054	0.4099	0.3312	0.4220	**0.4255**

Table 3. Final retrieval comparison among: baseline, color feature, edge feature, combined system, and combined system with confidence values

6. REFERENCES

[1] Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison Wesley.

[2] Eom, M., & Choe, Y. (2005). Fast extraction of edge histogram in dct domain based on mpeg7. *Proceedings of world academy of science, engineering and technology, 9*, 2005, 209–212.