Cognitive Computing HW2 Report

Author name: 劉家維 school ID: b05902052

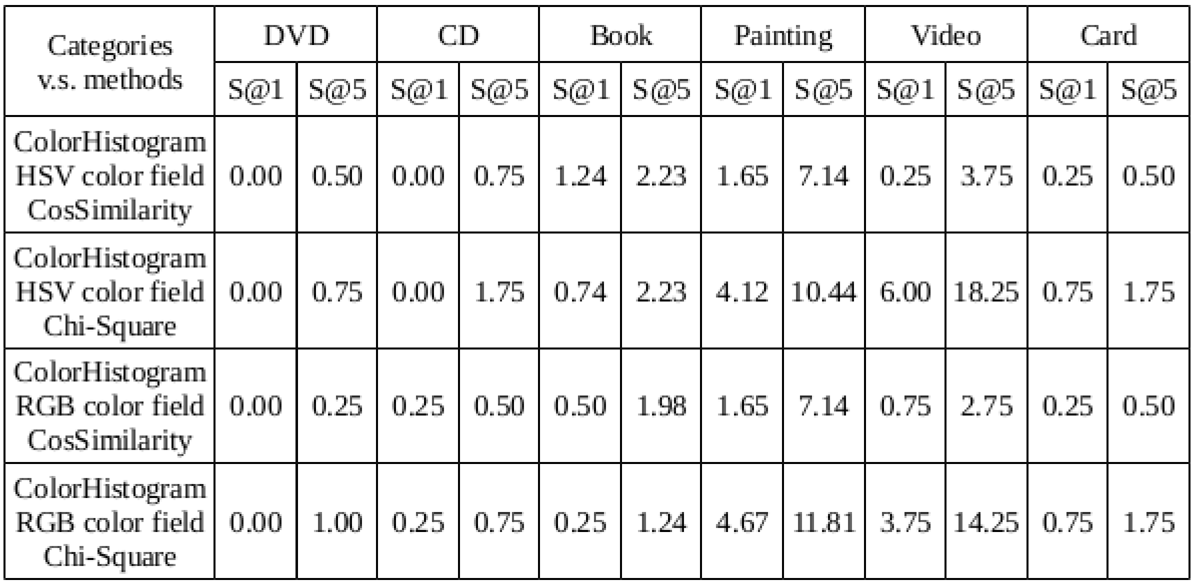
The units of all the scores shown in the tables below are percentage(%). That is, 40.00 in a slot of S@5 means that "Every 100 queries, there are 40 queries whose top5 of retrieval results contain the correct reference."

Color Feature

Procedures

- Preprocess the images.
- Calculate Color Histogram of images.
  - The algorithm is implemented in `opencv`
  - Introduction webpage: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html
- Use any histogram comparison methods (cosine similarity, l1, l2, chi-square, intersection, etc.) to find the most similar reference of every query image.

Result



Categories v.s. methods	DVD		CD		Book		Painting		Video		Card	
	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5
ColorHistogram HSV color field CosSimilarity	0.00	0.50	0.00	0.75	1.24	2.23	1.65	7.14	0.25	3.75	0.25	0.50
ColorHistogram HSV color field Chi-Square	0.00	0.75	0.00	1.75	0.74	2.23	4.12	10.44	6.00	18.25	0.75	1.75
ColorHistogram RGB color field CosSimilarity	0.00	0.25	0.25	0.50	0.50	1.98	1.65	7.14	0.75	2.75	0.25	0.50
ColorHistogram RGB color field Chi-Square	0.00	1.00	0.25	0.75	0.25	1.24	4.67	11.81	3.75	14.25	0.75	1.75

Discussion

1. The performances of global color histogram in DVDs, CDs, books, cards are very bad. Since the query images are usually with different light, shadow, and a large part of background, which will interrupt the

computation of histogram value of important part.
2. The performances of global color histogram in paintings and videos are better, since the query images in paintings include just white background, and the query images in video only include a very small part of background.
3. The difference of the results between HSV and RGB color field is not that obvious. Though it makes sense that HSV is usually a better color field than RGB, it is actually not shown in the experiment.
4. The difference between cosine similarity and chi-square is large, especially in videos. Maybe cosine similarity is not an appropriate metrics.
5. Though I haven't tried yet, the most efficient way to boost the performance I think is to crop the query images into smaller images, without those irrevelant backgrounds. However, it is difficult to determine the crop area of every image. Thus, we can use several sliding windows to mask the images and calculate the histogram by summed-area tables (just like the computation of Haar features) to avoid heavy computation caused by several sliding windows. After we calculate the histograms of all the sliding windows, we can query all the sliding windows and choose the most similar (or most closest) reference image among the results of all the sliding windows.

Texture Feature

Procedures

- Preprocess the images.
- Apply FFT on three channels of images and transform them into magnitude spectrum.
  - The algorithm is implmented in `opencv`
  - Introduction webpage: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_transform.html
- Use any distance or similarity metrics (cosine similarity, l1, l2, etc.) to calculate the distance of two spectrums.

Result

Categories v.s. methods	DVD		CD		Book		Painting		Video		Card	
	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5
FFT reshape-200,200 L2-distance	0.25	0.75	0.25	0.75	0.00	0.74	4.40	11.26	10.50	22.25	0.75	1.75
FFT centercrop reshape-200,200 L2-distance	0.75	1.25	0.50	0.75	0.99	3.22	6.04	12.64	12.75	26.25	1.50	2.75

Discussion

1. The texture-based feature are sensitive to different rotation and vision angle. As the same result as color-based method, my texture-based method can only get higher scores in paintings and videos, as same as those methods in color-based.
2. FFT generates the magnitude spectrum of an image, and the feature is as same as image size. Since I can not come up with other proper methods to handle the different sizes of different images, I reshape

all the images into 200x200 and then compare two spectrum images by L2-distance directly.
3. Since the aspect ratio will be wrong when reshaping all images into 200x200, I tried a different method - Center-crop out the biggest square within the images and then reshape it. By this way, the performance is slightly improved in every kind of images.

Local Feature

Procedures

- Apply SIFT to all images (both queries and references) and get 128-dimension visual word representation of the keypoints.
  - The algorithm is implemented in `opencv`
  - Introduction webpage: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
- Collect the keypoints of reference images and apply K-Means to cluster them into `n_clusters` clusters.
  - The algorithm is implented in `scikit-learn`
  - Introduction webpage: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans
- Collect the keypoints of each image and find the the closest cluster of every 128-dimension keypoint feature. If the distance to the closest cluster $d_1$ and the distance to the second closest cluster $d_2$ satisfies $d_1 <= d_2 \times threshold$, assign the index of the closest cluster to the keypoint feature. Calculate the appearance of the assigned cluster of every keypoint, and view it as a histogram.
- Use any histogram comparison methods (cosine similarity, l1, l2, chi-square, intersection, etc.) to find the most similar reference of every query image.

Result

Categories v.s. methods	DVD		CD		Book		Painting		Video		Card	
	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5	S@1	S@5
DoG+SIFT n_cluster=512 threshold=0.75	1.00	2.75	2.50	6.75	3.71	10.64	5.22	9.89	8.25	19.75	17.25	24.75
DoG+SIFT n_cluster=512 threshold=1.00	6.25	12.50	4.25	11.75	12.38	15.35	4.95	10.16	23.00	34.75	15.50	24.50
DoG+SIFT n_cluster=4096 threshold=1.00	8.50	17.00	6.25	15.00	19.80	30.69	9.07	13.74	21.50	40.00	26.75	31.75

Discussion

1. The procedures take lots of time. However, there are some faster algorithms I haven't tried, such as SURF (to approximate SIFT), PCA-SIFT (to reduce the size of codebooks).
2. The number of clusters actually plays an important role. With more clusters, the performance is more better, and the improvement is unneglectable. (But it also takes much more time to compute.)

3. Some articles on the websites set the threshold when finding the closest cluster of the keypoints. That is, if a keypoint is at the middle point of two cluster, the cluster we should assign to the keypoint is ambiguous. However, when I set threshold (threshold=0.75), I got worse performance than not setting the threshold in most of the images, except the paintings and cards.
4. Another interesting thing is that the performance of paintings is even the worst among all kinds of images. It is different from color-based or texture-based method. I think it is because the backgrounds of all paintings are almost the same, so some keypoints in the background are useless and can even lead to mismatching to other painting images.

Conclusion

1. Among all methods I have tried, SIFT is the most robust method to detect object. Since color-based method will fail due to color-difference and some light and shadow caused by different cameras, and texture-based method will fail due to the different rotation or vision angles.
2. Though I think SIFT has the best performance, maybe it is unfair to compare the all methods directly. One of the most important parts of the experiments in local features part is to cluster all the ~1500k keypoints. It is difficult and time/memory-consuming to cluster such many sample points. Some of the power of SIFT actually comes from the clustering, which can reduce the computation complexity when constructing the codeword histogram. However, other methods - such as color histogram or gradient histogram method, can compute the histogram of an image directly and rapidly.