MCA-Assignment1

Kshitiz Jain 2016051

28th February 2020

1 Question 1

Image Retrieval using color correlogram is done as following steps :

1.1 Color Quantization

The color space of a image is $256 \times 256 \times 256$, since this space will be too huge the correlogram of an image will be very sparse. Therefore the color space of all the images is reduced to $16 \times 16 \times 16$.

Again number of colors is still large and each color cannot be assigned different label, hence the color space is clustered into 16 colors (representatives of all color space). This is done by using KMeans clustering. When an image is read, firstly it is divided by 16 as the color space is reduced and then for each pixel KMeans assigns that pixel's color to the nearest color from the clusters.

Now for all images instead if 3 Channel grid, we have a single grid with 16 unique values.

1.2 Making of Color Correlogram

Correlogram is created at four distances $1, 3, 5, 7$, and is made using

$$P[Color_i|Color_i, Distanced]$$

. Formula used while calculating distance from a pixel is the following :

$$D_8(p,q) = \max(|\ p_x - q_x\ |, |\ p_y - q_y\ |)$$

1.3 Image Retrieval

The distance between 2 correlograms are calculated using the following formula :

$$|I - I'|_{\gamma} \equiv \frac{1}{m} \sum_{i \in [m], k \in [d]} \frac{|\gamma_{C_i}^{(k)}(I) - \gamma_{C_i}^{(k)}(I')|}{1 + \gamma_{C_i}^{(k)}(I) + \gamma_{C_i}^{(k)}(I')}$$

and images are retrieved from database accordingly.

1.4 Results

Precision, Recall and F1 score values are calculated over different number of retrievals from database.

Precision Values				
Images Retrieved	20	50	100	200
Average Precision	0.124	0.07878	0.0612	0.0474
Minimum Precision	0.05	0.02	0.01	0.005
Maximum Precision	0.35	0.3	0.23	0.2

Recall Values				
Images Retrieved	20	50	100	200
Average Precision	0.0687	0.0889	0.12061	0.1718
Minimum Precision	0.0150	0.0194	0.01941	0.05825
Maximum Precision	0.25	0.25	0.272	0.4545

F1-Score Values				
Images Retrieved	20	50	100	200
Average Precision	0.0671	0.05984	0.0594	0.0584
Minimum Precision	0.0247	0.02469	0.01526	0.009615
Maximum Precision	0.16	0.10811	0.1202	0.1459

Miscellaneous				
Images Retrieved	20	50	100	200
Average Good Images	1.0	1.33	1.666	2.333
Average Ok Images	1.0	1.33	2.666	4.333
Average Junk Images	1.0	1.0	2.33	3.333

Average Time Taken				
Images Retrieved	20	50	100	200
Time Taken (sec)	0.06096	0.0576	0.0619	0.06387

Percentage Retrieved				
Images Retrieved	20	50	100	200
Average(%) Good Images	3.395	4.012	4.629	7.0286
Average(%) OK Images	3.0162	4.405	7.659	11.51
Average(%) Junk Images	3.395	3.39	7.80	11.195

2 Question 2

Key/Blob detection using Laplacian of Gaussian is done in following sub parts.

2.1 Preprocessing of Images

- The image is converted to gray image for processing.
- The size of the image is very large and hence was not feasible to compute hence was down-sampled by 4 times, keeping the aspect ratio same.
- Since Laplacian function is sensitive to noise, Gaussian blurring is done on the image so as to suppress it.

2.2 Scale Space Extrema Detection

Now for different sigma values (scales), Laplacian of Gaussian filter is applied on the image and stored in a array, forming a grid of images stacked in order of the scale.

Sigma value is was started from 1.2 [1] and is increased to 2 times in next scale.

2.2.1 Keypoint Localization

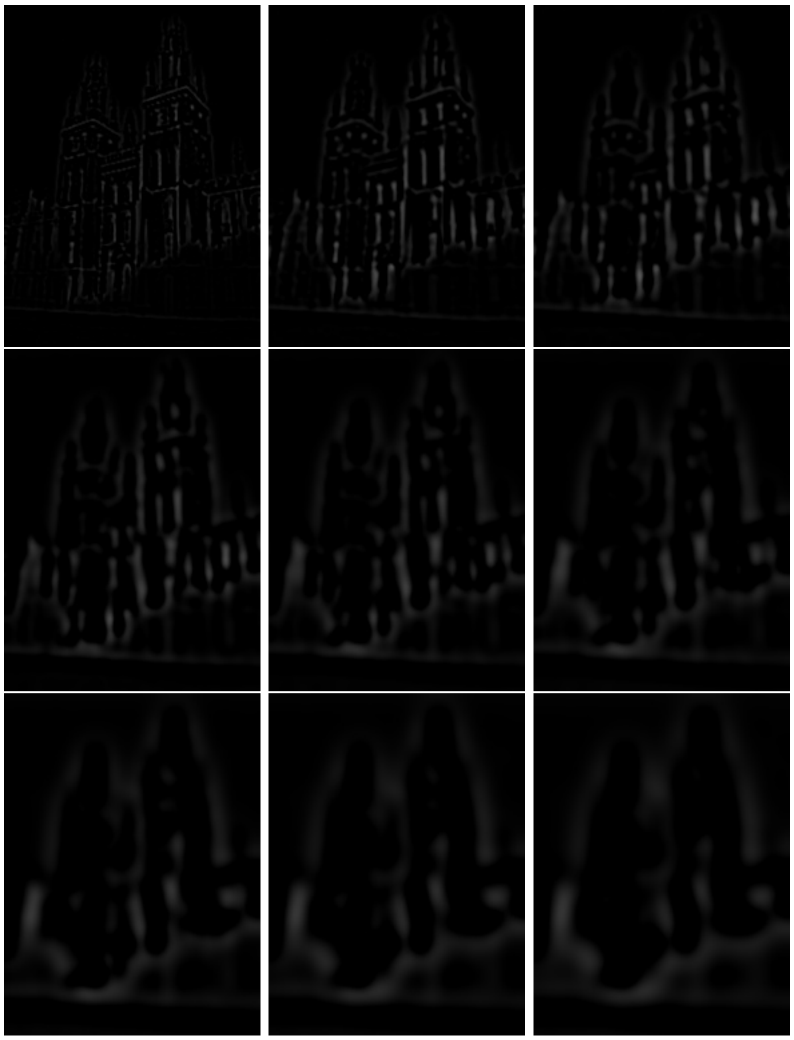
Now in each LOG image, for each pixel if it's value is higher than a threshold then a 3x3 grid is created by selecting the 8 neighbours in the same scale, and 9 neighbours from both 1 higher and 1 lower scale. Then if the value of pixel is strictly higher than all the values in the grid the keypoint is selecting as a blob.

The radius of the blob is calculated by multiplying the sigma with 1.414 (square root of 2), and is saved along with the coordinates for that pixel. Overlapping blobs are pruned using the scipy.feature library.

2.3 Example

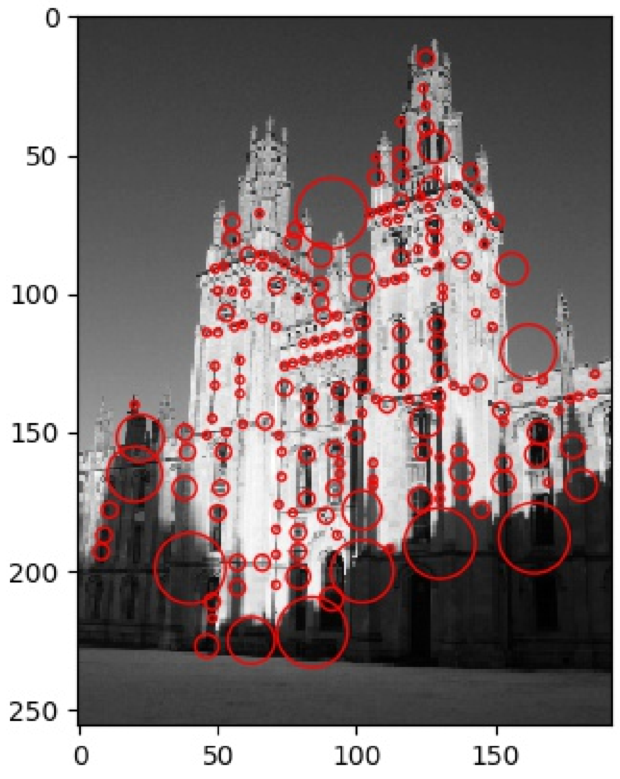
Average time to retrieve blobs from image : 0.8532 (depends upon the threshold, lower the threshold higher the blobs detected, higher the time)

2.3.1 LOG images at different Scales



With increase in sigma/scale, the bigger blobs appear in the LOG.

2.3.2 Blob Detected

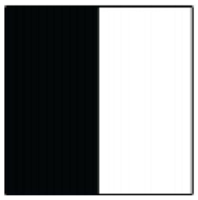
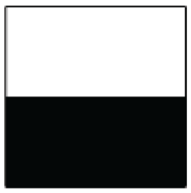


2.4 Feature Descriptor

Features for LOG blobs are calculated in following way

1. For each blob calculated previously, a 16×16 grid is taken about the center of that blob, and following feature vector is created :

$$v_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right]$$

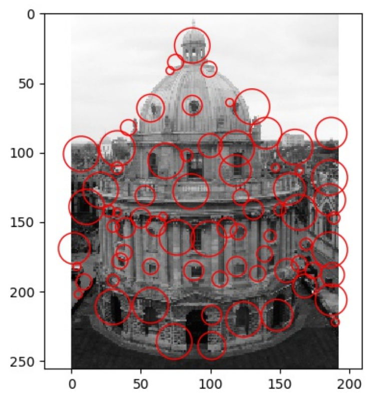
dx:  dy: 

This feature can be calculate using the Discrete Wavelet Transform (HAAR), where LH and HL band are used.

3 Question 3

For the SURF feature keypoints, everything is same as done in blob detection in LOG. The difference between them is that in LOG we used Laplacian of Gaussian at different scales/sigma, whereas in SURF we use Determinant of hessian of a image at different levels. For calculating hessian $skimage.feature.hessian\_matrix\_det$ library is used.

Example Image with SURF blobs :



Average time to retrieve blobs from image : 1.053 (depends upon the threshold, lower the threshold higher the blobs detected, higher the time)

References

[1] Bay, Herbert and Tuytelaars, Tinne and Van Gool, Luc. SURF: Speeded up robust features, Computer Vision-ECCV 2006.