MCA HW-1 Report

-Priyanshi Jain, 2017358

Color Correlogram:

References:

- https://github.com/enthalpy-yan/Color-Texture-Shape
- https://raw.githubusercontent.com/dermotte/LIRE/master/src/main/java/net/semanticmetadata/lire/imageanalysis/features/global/correlogram/NaiveAutoCorrelogramExtraction.java

All the images were resized to (360,360) pixels.
The first thing to be done was divide the colours of the image into some set number of bins. For this, a universal thresholding was used. 256x256x256 colours were divided into 216 bins.
A color array with all the 216 colors was made.
For the distances, 1,3,5 and 7 are used
To optimize the code, instead of checking the neighbours for each and every pixel, pixels at a jump of W/6 and H/6 were considered, where W and H denotes the width and height of the image respectively.
Now, instead of checking all the neighbours of a pixel at distance d, a gap of distance d/2 was considered to optimize the process.
Now, if the value of the neighbour's pixel is the same as that of the pixel in consideration, we increase the count of the pixel in the correlogram array by 1.
At the end, correlogram[d][c] represents the probability of having color c at a distance d.

These correlograms for all the given images are saved.

Similarity Matching:

Firstly, the saved correlogram features of all the images are imported.
Then, for a given query, the absolute distance between it's correlogram and all the other correlogram are saved.
After getting all these 5063 distances, they are sorted and top 100 are retrieved for the results.

The results are as follows:

Recall = Relevant images retrieved/ total number of relevant images
Precision = Relevant images retrieved/ total number of images retrieved

Measure	Maximum	Minimum	Average
Precision	0.26	0.01	0.047

Recall	0.31	0.009	0.095
F1 score	0.116	0.0094	0.045

Average time for retrieval: 3.73 sec

% of images retrieved:
Good: 16%
Ok: 7%
Junk: 3%

Analysis:
Max recall occurs for image "`cornmarket_000047.jpg`"
Max precision and f1 occurs for image "`radcliffe_camera_000523.jpg`"
The reason could be the unique color composition of these images. The second image has pixel values close to each other. So when we divide the pixels into bins, not a lot of information gets lost.

Min recall, precision, f1 occur for image "`all_souls_000013.jpg`"
The reason behind it could be that the colour composition of this image is similar to a lot of other images which are not even matches for this image. So when we use color correlogram, those images get matched with this image.

It can be seen that the results obtained using matching by color correlogram are not very accurate. The results could be:

- Resizing the image: The given images got reduced by half or more. This may have resulted in a loss of a lot of information.
- Color bins: The colors of the images were reduced to 216 bins.
- All the pixels weren't considered for checking and all the neighbours were not checked for each pixel.

Such optimization techniques would have resulted in loss of information.

Scale Invariant Blob Detection - LoG:
References:

- https://projectsflix.com/opencv/laplacian-blob-detector-using-python/
- http://slazebni.cs.illinois.edu/spring18/assignment2_py.html

For each image, LoG was calculated by convolving the image with the Log filter at different scales. For scales, sigma was initialized with 1 and was progressively increased by a factor of (1.5)^i, where i ranged from 1 to 10.

After getting the results of the image at different scales, non-maximum suppression was performed to take the local maximum at neighbouring scales and for a 3x3 region.
After taking the maximum, thresholding was performed with a threshold value of 0.03 and the top 1000 features were saved with their x,y and pixel values.
Link to extracted features: Blob_points_final