# MCA HW-1

Priysha - 2016256

28 February 2020

## 1 Question 1

Color Auto-Correlogram :

### 1.1 Feature Descriptor

Technique used to represent spacial correlation between pixels of certain color at a certain distance. It can be considered as a 2-D array with number of colors (quantized) on one axis and different values of distance d considered on other axis. So order of this method is :

$$O(m \times d)$$

where m is the number of colors and d is the list of distances considered. Each element of this array represents the following probability:

$$\gamma^k_{C_i}(I) = Pr_{p1 \in I_{C_i}, p2 \in I}[|p1 - p2| = k, p2 \in I_{C_i}]$$

where $C_i \in$ list of m colors and k $\in$ d = [1,3,5,7] and p2 $\in$ list of pixels at D8 distance = k.
In particular, the $256 \times 256 \times 256$ RGB color space is quantized into 32 colors by using KMeans clustering technique and stride of 10 for each color channel. So, m = 32.
Each element of the correlogram is calculated as :

$$Pr_{colorC_i, dist.d} = \frac{Number - of - neighbours - with - same - color - at - distance - d}{Number - of - existing - neighbours - at - distance - d}$$

Hence a $32 \times 4$ descriptor is obtained for each image in the database.

### 1.2 Image Retrieval

Descriptor of a query image is compared to descriptors of all images in the database and similarity matching is done based on the distance metric :

$$|I - I'| = \frac{1}{m} \sum_{i \in m, k \in d} \frac{|\gamma^k_{C_i}(I) - \gamma^k_{C_i}(I')|}{1 + \gamma^k_{C_i}(I) + \gamma^k_{C_i}(I')}$$

The image retrieval results are sorted as increasing distances and top 'k' are taken to calculate the results.

### 1.3 Results

- Precision: Finds how precise the model is, i.e. how many of those images retrieved for a query actually belong to the ground truth(good+ok+junk) of that query.

$$Precion = \frac{No.of images retrieved \in ground-truth}{Total-images-retrieved(k)}$$

- Recall: Finds what portion of the actual ground truth is predicted by the model.

$$Recall = \frac{No.of images retrieved \in ground-truth}{Total-images \in ground-truth(good+ok+junk)}$$

- F1 score: Harmonic mean of precision and recall. This measures the accuracy of the model while maintaining a balance between precision and recall.

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall}$$

- Respective values for all queries :

| k = 10 | | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| Precision | 0.1666 | 0.1000 | 0.6000 |
| Recall | 0.0528 | 0.0057 | 0.2307 |
| F1 score | 0.0655 | 0.0111 | 0.2608 |

| k = 30 | | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| Precision | 0.0929 | 0.0333 | 0.3333 |
| Recall | 0.0705 | 0.0090 | 0.2307 |
| F1 score | 0.0596 | 0.0142 | 0.1395 |

| k = 100 | | | |
|---|---|---|---|
| | Average | Minimum | Maximum |
| Precision | 0.0629 | 0.0111 | 0.2666 |
| Recall | 0.1087 | 0.0388 | 0.2727 |
| F1 score | 0.0581 | 0.0185 | 0.1192 |

Minimum precision comes for : all_souls_000026 image, for k = 30.
Maximum precision comes for : oxford_002904 image, for k = 30.

- Average retrievals in each category:

| Average value per query | | | |
|---|---|---|---|
| | k=10 | k=30 | k=100 |
| Good | 1.3030 | 1.848 | 3.2121 |
| Ok | 0.3636 | 0.6363 | 1.4242 |
| Junk | 0.0 | 0.3030 | 1.0303 |

| Average percentage for whole dataset | | | |
|---|---|---|---|
| | k=10 | k=30 | k=100 |
| Good | 5.269% | 7.475% | 12.990% |
| Ok | 1.351% | 2.364% | 5.292% |
| Junk | 0.0% | 1.115% | 3.790% |

| Average time taken for a retrieval per query (in sec) | | |
|---|---|---|
| k=10 | k=30 | k=100 |
| 0.1107 | 0.1078 | 0.1143 |

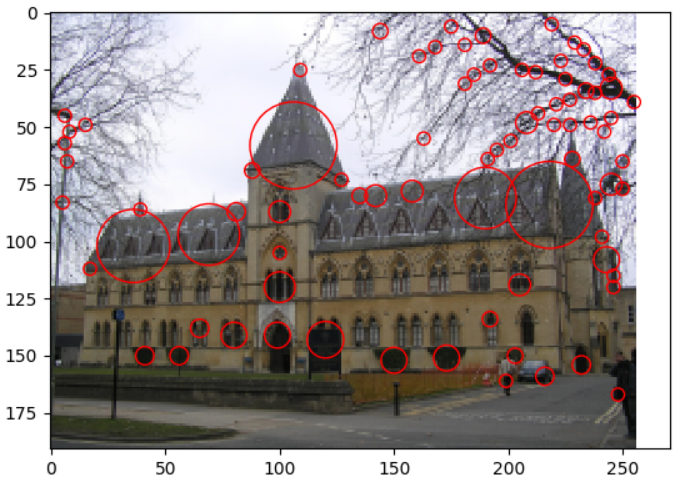
# 2 Question 2

Scale-Invariant Blob Detection – LoG

## 2.1 Feature Detection

- Images in the database are converted to gray, downsampled to 25% their original size, while maintaining their aspect ratio and blurred to remove noise.
- The obtained images are then passed to gauss_laplace() function to obtain laplacian outputs at different sigma levels. To form the scale space, we do not need to reduce the size of image after each iteration since sigma ($\sigma$) is getting updated in each iteration.
- Sigma ($\sigma$) starts from 2 and is increased by factor $k = 2^{0.25}$ in each iteration[2].
- Scale space obtained is scale normalized by multiplying by $\sigma^2$.
- Non-maximum suppression is performed in a $3 \times 3 \times 3$ neighbourhood to retain only the locally significant keypoints. Thresholding and pruning is done to retrieve the most significant blobs.

### 2.2 Example




**Average time taken to detect SURF keypoints per image : 3.2284 sec**

### 2.3 Feature Descriptor

A 64 length desciptor can be made for each keypoint (blob). Consider a $16 \times 16$ neighbourhood and divide it into 16 grids of $4 \times 4$ each. The following descriptor is concatenated for each of the 16 grids :

$$(\sum dx, \sum dy, \sum |dx|, \sum |dy|)$$

where dx and dy are haar wavelet outputs of the image for x and y direction respectively[3].
So descriptor of each image is of the shape : Number of blobs x 64.

### 2.4 Image retrieval

Brute force matching can be done for keypoint desciptors of each query with all those of all images in the database. The images with most keypoints matched can be retrieved.

## 3 Question 3

SURF: Speeded-Up Robust Feature

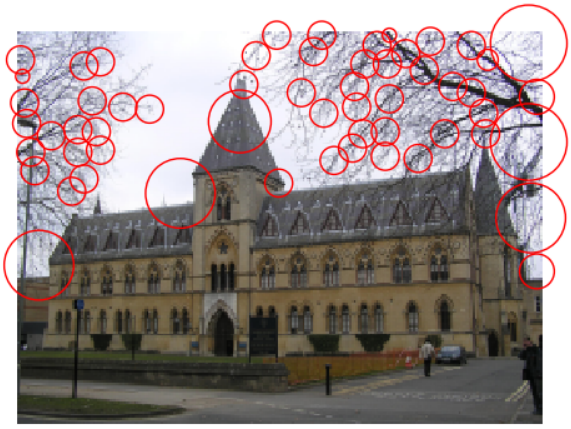
### 3.1 Feature Detection

- Images in the database are converted to gray, downsampled to 25% their original size, while maintaining their aspect ratio and blurred to remove noise.
- Obtain integral images for the above output and pass to hessian_matrix() function to obtain hessian matrix at different sigma levels.
- Sigma ($\sigma$) starts from 1.2 and is increased by factor $k \in [1, 2, .., 12]$ in each iteration[3].
- Determinant of the hessian matrix at each pixel is calculated as [3] :

  $$Det = H_{xx}H_{yy} - (0.9H_{xy})^2$$

  The same can be directly obtained using hessian_matrix_det() function.
- Non-maximum suppression is performed in a $3 \times 3 \times 3$ neighbourhood to retain only the locally significant keypoints. Thresholding and pruning is done to retrieve the most significant blobs.

### 3.2 Example



**Average time taken to detect SURF keypoints per image : 0.7582 sec**

## References


[1] Precision-Recall-F1

[2] DIENS Computer Science department

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