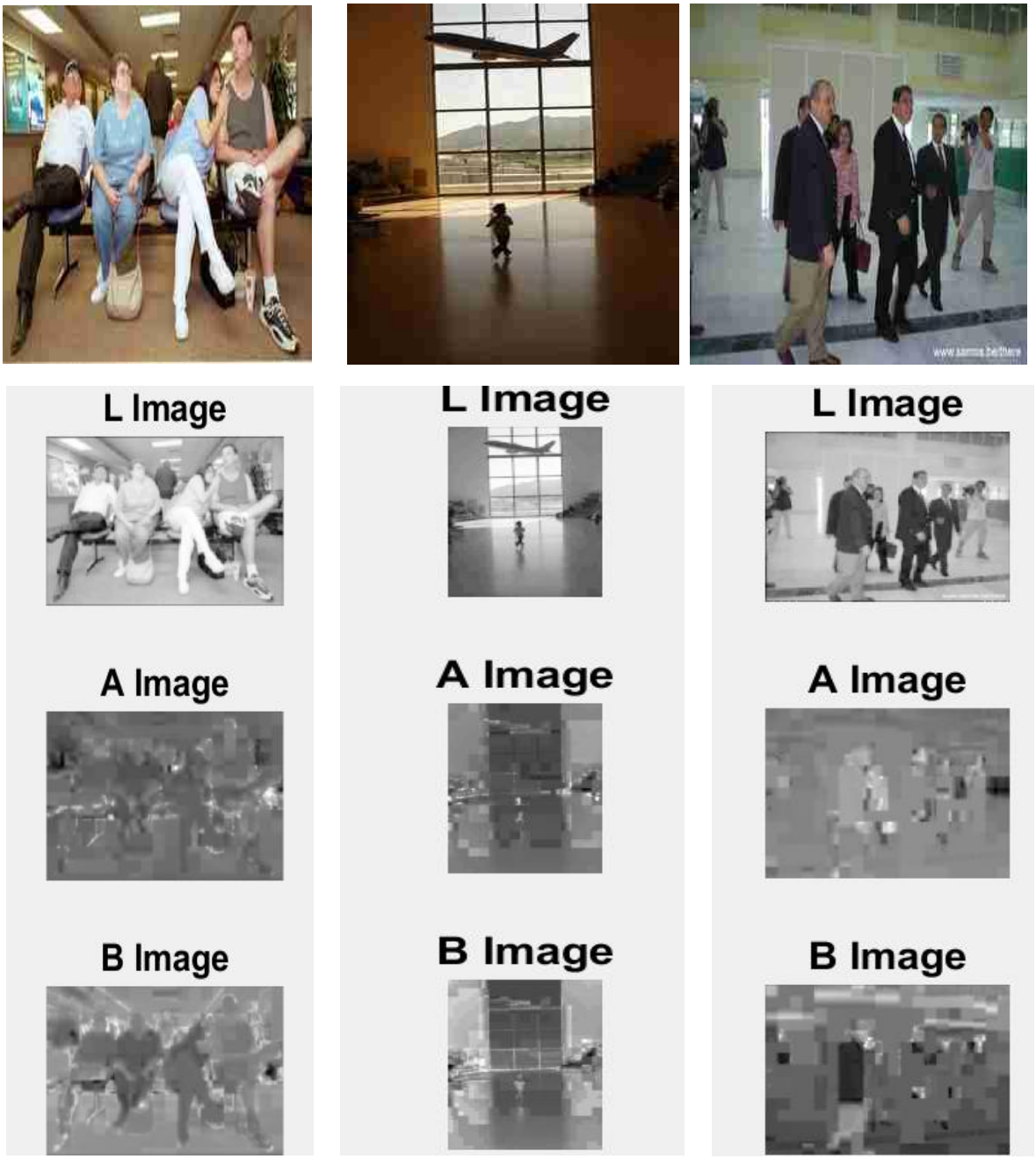
3.1.1) Extract Filter Responses

- The image has been converted from RGB to LAB using the RGB2Lab function.
- The n filters are applied on each of the channels of the input image.
- Have used the CIE Color Lab and it used the represent colour as L for lightness, a & b for green-red and blue-yellow components.
- Executed this in getHarrisRandomdict.m file for easier implementation. Kindly execute this file to get one of the sample outputs shown below.

Provided the Original images and L, a, b outputs for 3 sample images.



- Figure 1: -Sample image with a few filter responses.
- CIE Color Space defines the colours independently on how they are created or displayed. Hence, we would like to use it for all our images with different colours in the dataset.

1.2) Collect sample of points from image: -

Two functions are created namely getRandomPoints and getHarrisPoints and they are executed separately in harrisAndRandomPoints.m file for easier implementation. Kindly execute this file to get the output of the below Random and Harris points.

1.2.1) getRandomPoints: -

- Random pixel points are collected with a 50*2 matrix, and plotted as shown below: -

Figure 3,4,5: - Showing the sample random points for 3 sample images.

1.2.2) getHarrisPoints: -

- Harris corner detector is identified using the getrandHarrisPoints function.
- The function takes the input image and converted to a grayscale image.
- The covariance matrix M has been calculated for each pixel.
- Setting the threshold for the corners are plotted as shown below for 3 sample images: -
- Harris points are collected.

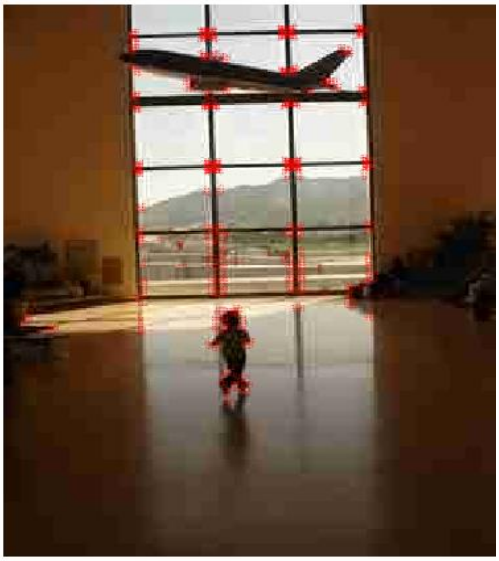


Figure 6,7,8: - Showing the sample Harris corner points for 3 sample images.

1.3) Compute Dictionary of Visual Words: -

Dictionary is computed based on the two methods.

- If the method is 'random' we call the getRandomPoints function.
- If the method is 'harris we call the getHarrisPoints function.
- Computing the alpha value to 50 and K value to 100, we call the getDictionary function.

- For every image sets that are provided filter response that has been created previously - extractFilterResponses is called and getRandomPoints based on the type of method as 'Random 'or 'Harris.
- Two dictionaries are created namely **dictionaryRandom.mat**, which used the random method to select points and another named **dictionaryHarris.mat** which used the Harris method.

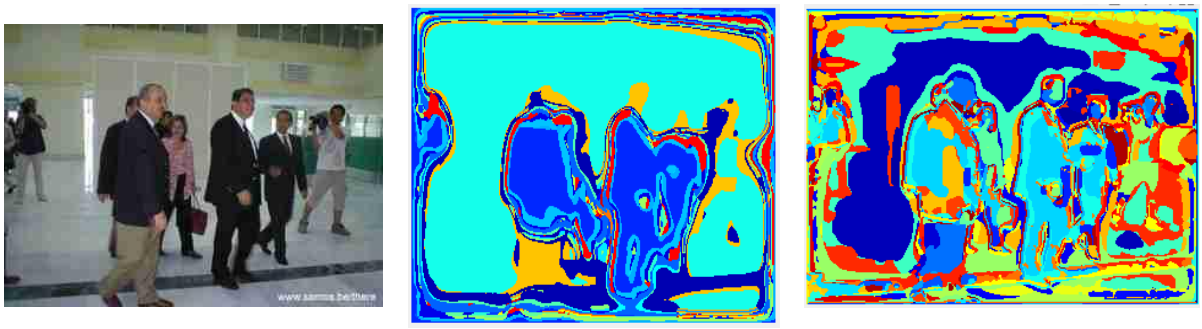
Part 2: Build Visual Scene Recognition System

2.1) Convert image to word map: -

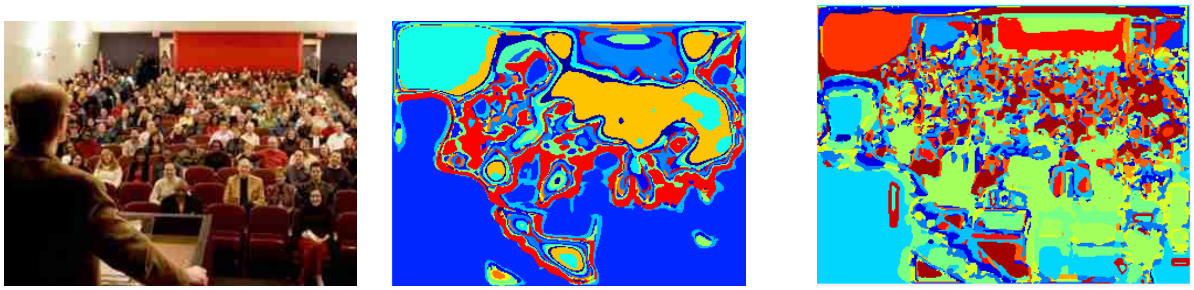
- Word maps are created with H*W matrix.
- Executed this in getHarrisRandomdict.m file for easier implementation. Kindly execute this file to get one of the sample outputs shown below.
- Pdist2 is used to find the word maps and results are shown using the label2rgb functions as shown below: -

Figure 9,10,11,12: - Sample visual word plot.

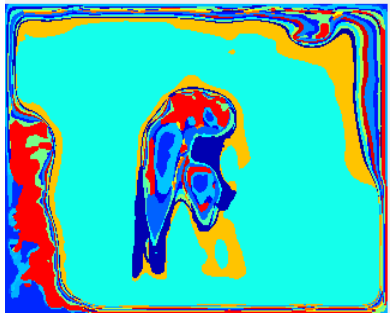
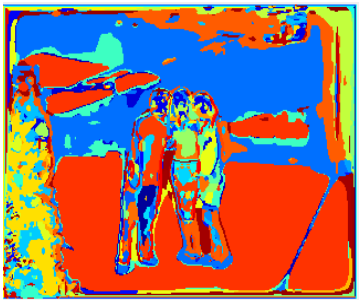
Sample Random and Harris for Image 1



Sample Random and Harris for Image 2



Sample Random and Harris for Image 3

2.2) Get Image Features

- Extracted the visual words in the image.
- L1 normalized the histogram.

2.3) Build Recognition System - Nearest Neighbors

- train_imagenames and train_labels are loaded from traintest.mat.
- Loaded the dictionary from dictionaryRandom.mat and dictionaryHarris.mat that has been saved in 1.3 and written in buildRecognitionSystem.m

PART 3: Evaluate Visual Scene Recognition System

3.1 Image Feature Distance

- Histograms for two images are created and distance has been calculated using hist1 and hist2.
- The Euclidean distance is calculated using the pdist2 function else the chi squared distance is calculated. If the two images are very similar, then the distance is small. If the images are having mismatches, then the distance is large. By calculating the distance, this has been found out.

3.2 Evaluate Recognition System - NN and kNN

evaluateRecognitionSystem_NN()

- The closest neighbour recognition system is calculated using evaluateRecognitionSystem_NN.m
- **Harris-** Word maps are created using getVisualWords() function for all the image sets provided. Image distances are calculated separately for chi squared and Euclidean using the getImageDistance() function.
- The accuracy & 8*8 matrix of harris and chi squared is shown as below: -

```
The accuracy of harris and chi-squared is 0.39
The confusion matrix for harris and chi-squared is:
     9     4     2     1     0     0     2     2
     4     9     2     1     1     0     2     1
     2     5     7     0     3     2     1     0
     3     1     0     5     4     0     6     1
     2     2     4     2     5     3     2     0
     3     0     3     3     2     7     2     0
     1     0     1     3     3     2     7     3
     3     2     1     0     0     0     1    13
```

- The accuracy & 8*8 confusion matrix of harris and Euclidean is shown as below: -

```
 The accuracy of harris and euclidean is 0.25
The confusion matrix for harris and euclidean is:
    10     5     3     0     1     0     3     3
     4    10     2     2     2     0     3     2
     3     5     7     1     4     3     2     0
     3     0     1     5     5     0     7     2
     3     2     0     3     5     3     3     0
     4     1     4     4     2     7     3     1
     2     1     2     4     4     3     4     4
     4     3     2     1     0     0     2    14
```

- **Random**- For all the image sets provided image, and using the wordMap that has been created and used in getImageFeatures to calculate h1 distance and h2 using trainFeatures() function. Using the h1 and h2 calculated, the Image distances are calculated separately for chi squared and Euclidean using the getImageDistance() function.

- The accuracy and 8*8 confusion matrix of random and chi squared is calculated and shown as below: -

```
 The accuracy of random and chi-squared is 0.49
The confusion matrix for random and chi-squared is:
    12     3     2     0     0     0     1     2
     3    12     3     0     1     1     0     0
     2     4    12     2     0     0     0     0
     2     2     1     8     1     2     4     0
     1     2     2     0    12     0     3     0
     1     4     4     2     0     6     2     1
     3     1     0     8     2     0     5     1
     6     0     0     1     0     0     1    12
```

- The accuracy and 8*8 matrix of random and Euclidean is calculated and shown as below: -

```
 The accuracy of random and euclidean is 0.37
The confusion matrix for random and euclidean is:
    13     3     3     1     1     1     2     3
     4    13     2     0     2     0     1     1
     3     5    13     3     0     1     0     0
     3     2     2     5     0     3     5     0
     2     3     3     1    11     0     4     0
     2     5     3     3     0     7     2     0
     2     2     0     8     1     0     6     0
     7     0     0     2     0     1     1    13
```

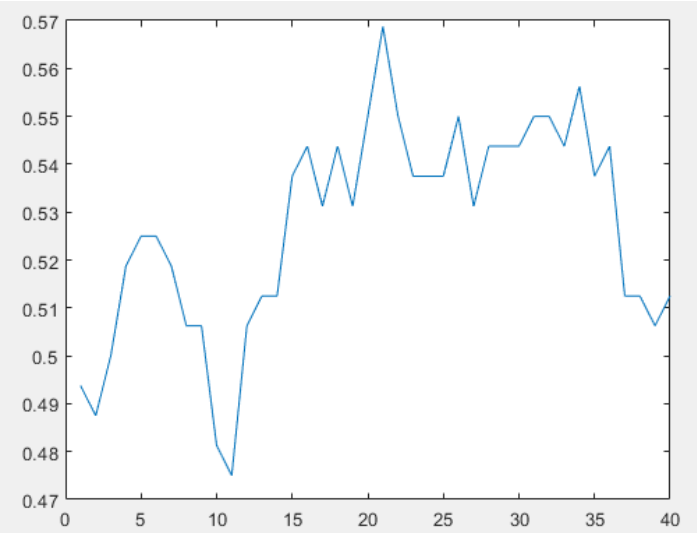
- The performance of random seems a bit better than the Harris as we could see from the above.
- The performance of random distance metric is closer than the expected distance metric for harris and this would be different because of different images in the data sets that are provided.

evaluateRecognitionSystem_kNN()

- Images are created for all the image sets provided.
- h1 is calculated using getImageFeatures() with the provided wordMap and K.
- h2 is calculated using trainFeatures() function.
- Using h1 and h2, the distance is calculated and finding the k nearest neighbours from 1 to 40.
- Votes are calculated for each neighbour and labels are created with the maximum vote.
- Best of K is created and 8*8 confusion matrix is created which is shown as below: -

```
The confusion matrix for the best k is:
    14     1     3     1     0     0     0     1
     3    13     3     0     0     0     1     0
     3     5    12     0     0     0     0     0
     3     1     2    11     0     0     3     0
     1     3     4     1    10     0     1     0
     3     4     3     1     1     6     0     2
     7     0     1     2     3     0     7     0
     2     0     0     0     0     0     0    18
```

- The accuracy is achieved at 0.57 for k=21 and k is not always better as we could see in the below plot that there are ups and downs at each k iterations. The spike is inconsistent because of the training and the testing datasets.



EXTRA CREDITS

QX.1 Evaluate Recognition System - Support Vector Machine

- I couldn't test the desired using SVM but I learned the SVM and their kernels and tried to created two kernels namely linear kernel and polynomial kernel and provided the code logic in get_kernel.m under ../ec/get_kernel.m.