CSE 573 COMPUTER VISION AND IMAGE PROCESSING

Write up for hw1

- **Gautam Shende (UBITname: gautamav, Person #: 50245840)**

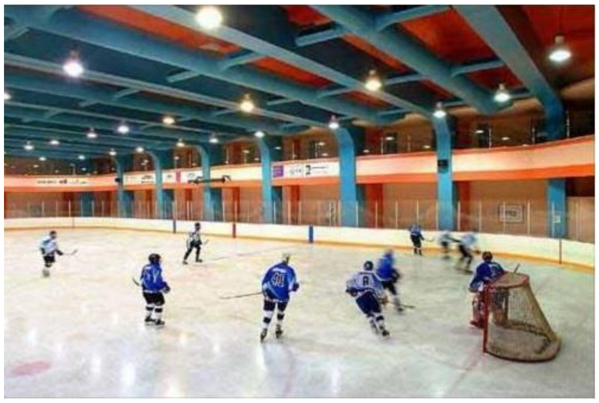
Index

1.0 Filters

The 4 filters used are:

1) Gaussian Filter:

Original image



Gausian Filter (default scale)

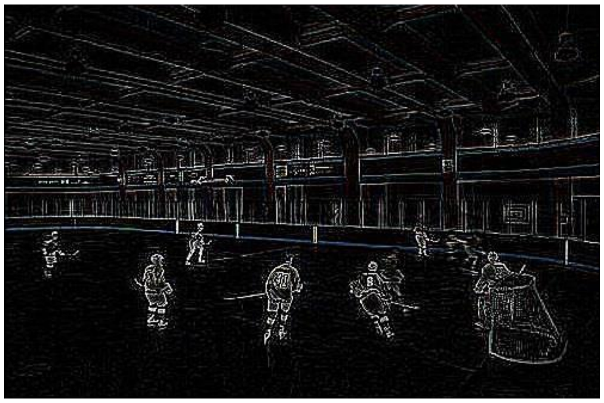
The default gaussian filter using the fspecial function has the following 3 x 3 representation in matlab:

0.0113	0.0838	0.0113
0.0838	0.6193	0.0838
0.0113	0.0838	0.0113

The filter is high at the centre and it falls off when we move towards the edges and further drops down as we move to the corners. This creates a smoothening effect and smoothens any sharp, thereby also trying to remove or reduce noise.

2) Laplace of Gaussian filter:

Original image

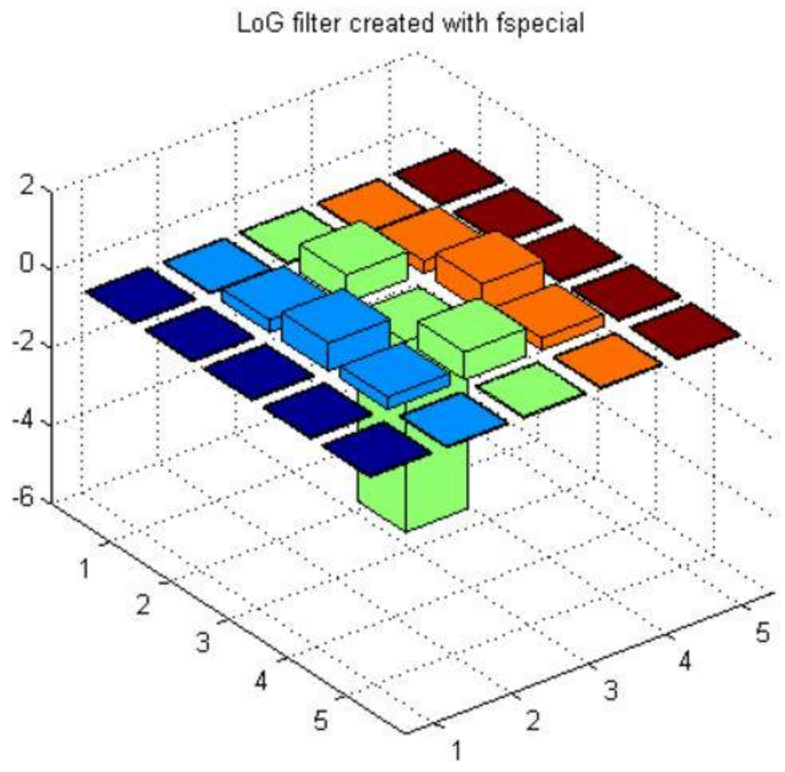


Laplace of Gausian Filter (default scale)

The default laplace of gaussian filter using the fspecial functoin has the following 5 x 5 representation in matlab:

0.0448	0.0468	0.0564	0.0468	0.0448
0.0468	0.3167	0.7146	0.3167	0.0468
0.0564	0.7146	-4.9048	0.7146	0.0564
0.0468	0.3167	0.7146	0.3167	0.0468
0.0448	0.0468	0.0564	0.0468	0.0448

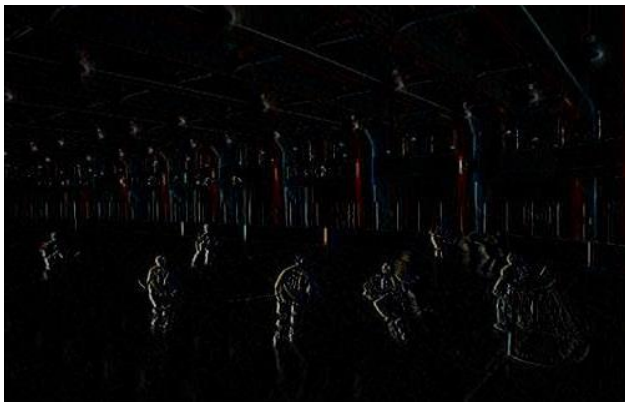
The filter is highly negative only at the center and then has decreasing positive values towards the edges that further decrease down the corners. The filter thus has kind of a reverse effect when compared to the gaussian filter. On convolution, the borders have negative values and the pixels next to the borders have higher values. This helps in edge and line detection in the image.



Source: https://stackoverflow.com/questions/10127264/laplacian-of-gaussian-log-edge-detector-in-matlab

3) dxscale filter

Original Image


dxscale filter

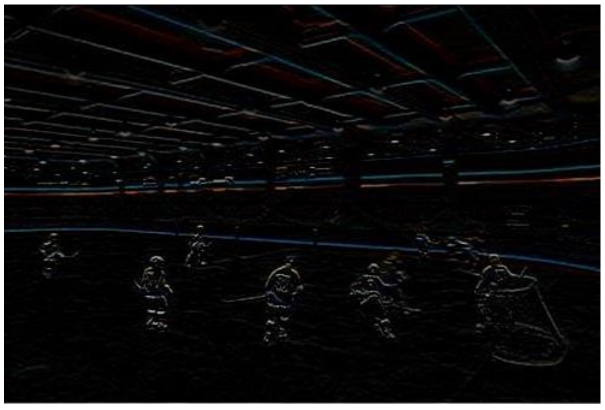
The default dxscale filter is obtained by just filtering the original gaussian filter with the matrix [-1 0 1]. It has the following 3 x 3 representation in matlab:

0.0838	0	-0.0838
0.6193	0	-0.6193
0.0838	0	-0.0838

The dxscale filter tries to highlight the edges and lines along the x axis i.e along the vertical direction in image processing.

4) dyscale filter

Original Image

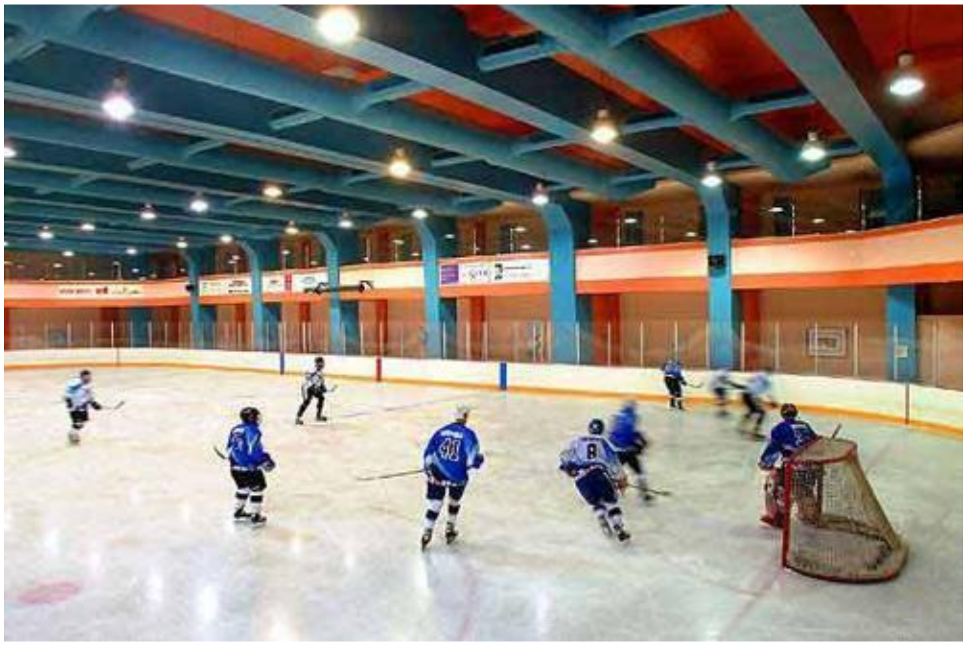

dyscale filter

The default dyscale filter is obtained by just filtering the original gaussian filter with the transpose matrix [-1 0 1]. It has the following 3 x 3 representation in matlab:

0.0838	0.6193	0.0838
0	0	0
-0.0838	-0.6193	-0.0838

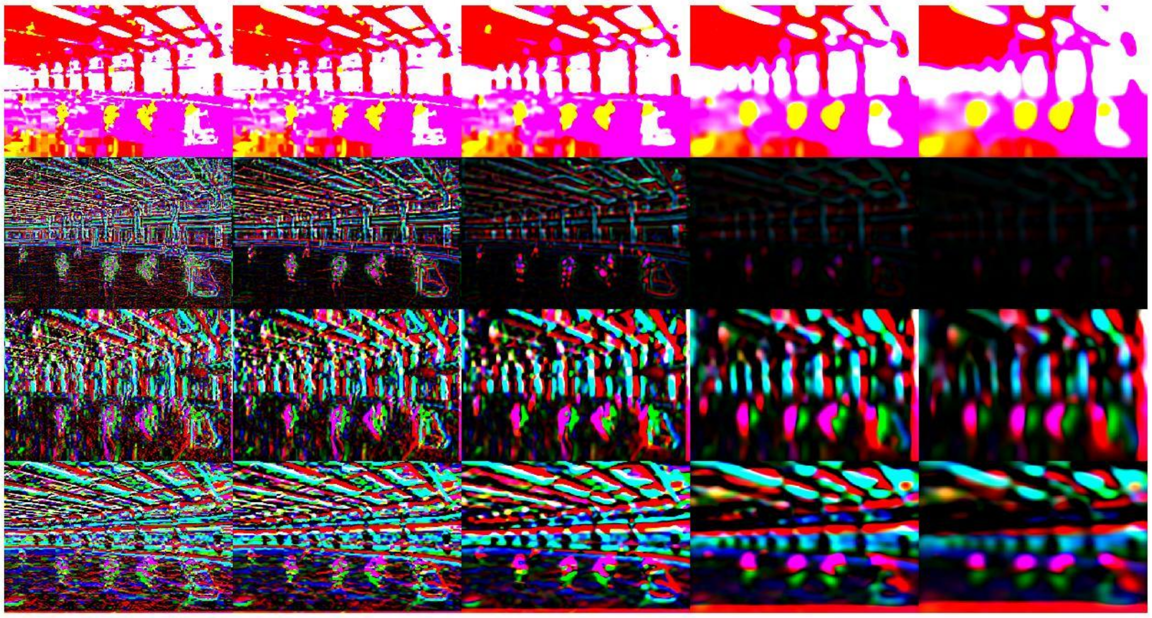
The dyscale filter tries to highlight the edges and lines along the y axis i.e along the horizontal direction in image processing.

1.1 Extracting filter responses



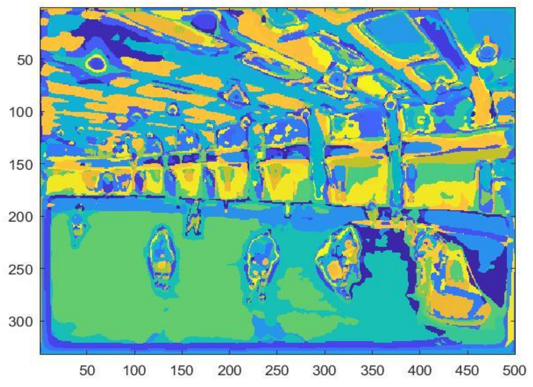
Input image of indoor ice skating

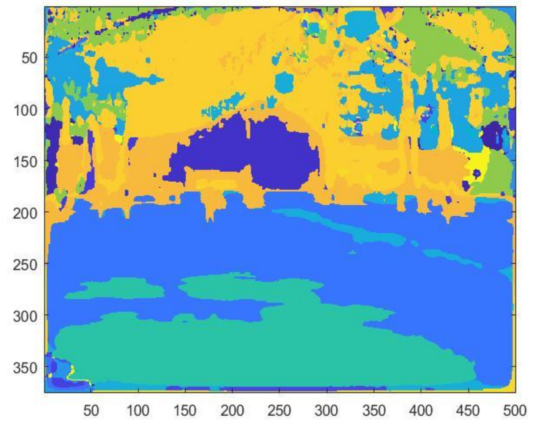
path = ../data/ice_skating/sun_aamerccwdeptadrz.jpg



Filter Responses of the L*a*b version of the image using the montage function.

1.3 WordMap Visualization





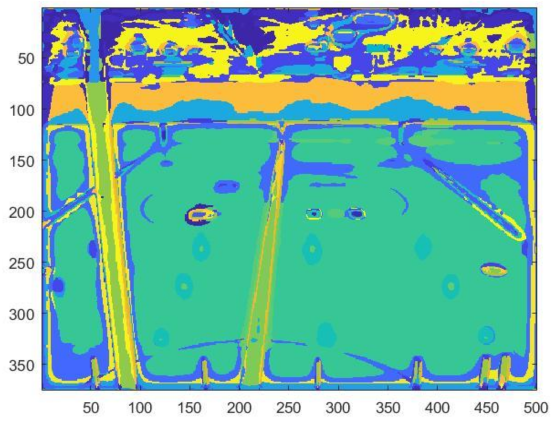


Image Paths = (‘../data/ice_skating/sun_aamerccwdeptadrz.jpg’, ‘../data/ice_skating/sun_absxwroozitnpiwm.jpg’, ‘../data/ice_skating/ sun_abxvvrflfkhpimek.jpg)

Comments:

What is a Wordmap?

It’s a representation of the image in 2D where the pixel values have only range from 1 to k, where k is the maximum number of clusters or the dictionary size.

Practical significance:

When we visualize the wordmap, we can see that it is divided into different parts based on the different textures. Each pixel is replaced by the index of its closest cluster. Simplistically, the clusters can be viewed as representing different elements based on the properties such as color,intensity,textures,etc.

For instance, a type of ice having a specific texture and color could be one cluster say k1 and another type of ice could be another cluster say k2. We replace all ice pixels of type 1 with k1 and the other ice pixels with k2. This is seen in the images above where ice is replaced by value of k that gives a greenish color. Similarly we have k such different color codes that represent k different elements.

Computational significance:

Thus, from possible 256 values for a single pixel in each of the 60 filter responses for an image, we have gone down to possible k values for a single pixel. The value of k used in my model is 100. Thus from a possibility of (256)^60 values for a single pixel in feature extraction, we have reduced it to just 100 possible values in our wordmap. This helps largely in reducing the computation time without loss of too many important features.

2.5 Evaluation

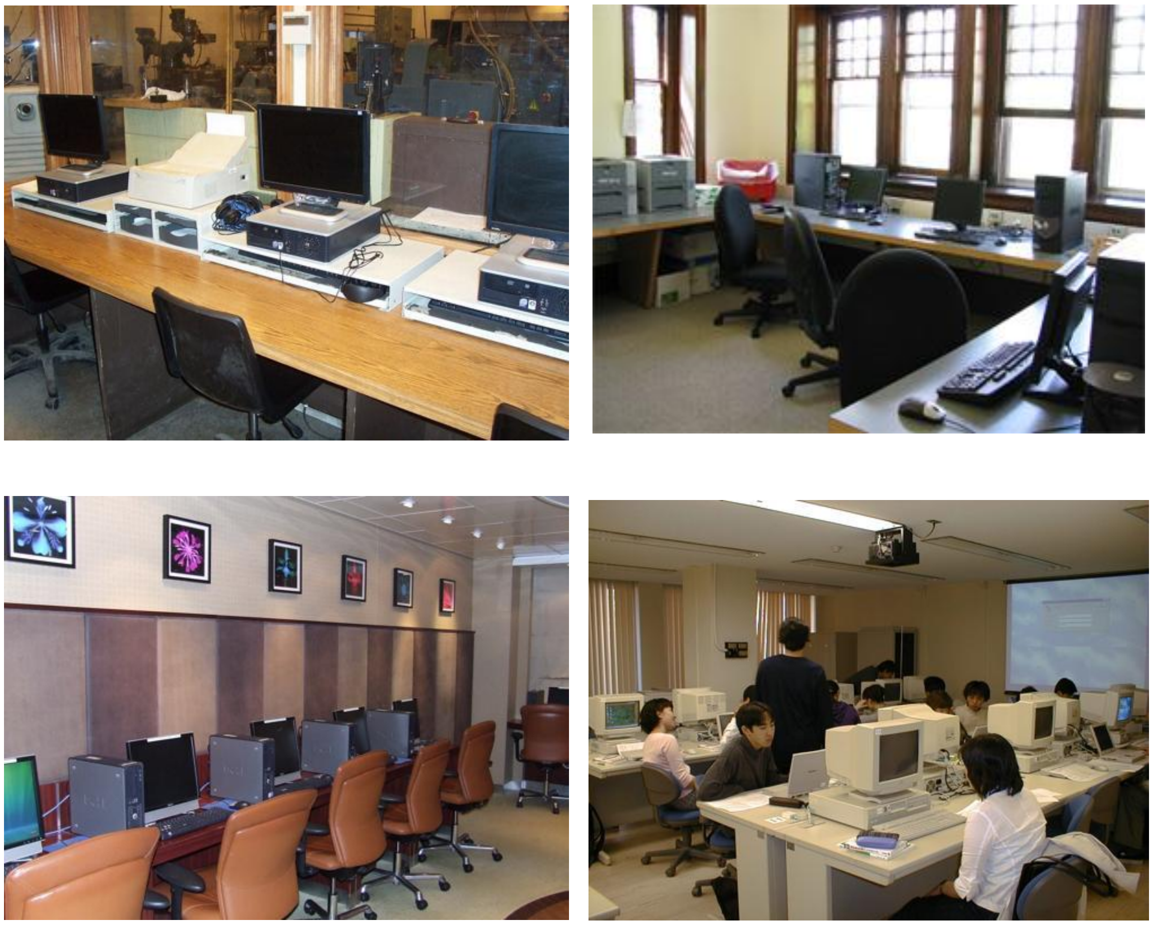
1) Confusion matrix (8 x 8) :

9	1	0	4	4	0	0	2
1	8	1	3	5	1	0	1
0	0	19	0	0	1	0	0
0	1	0	18	0	0	0	1
3	4	0	3	9	1	0	0
1	0	3	0	2	8	5	1
0	1	0	0	3	1	14	1
1	2	1	3	2	1	2	8

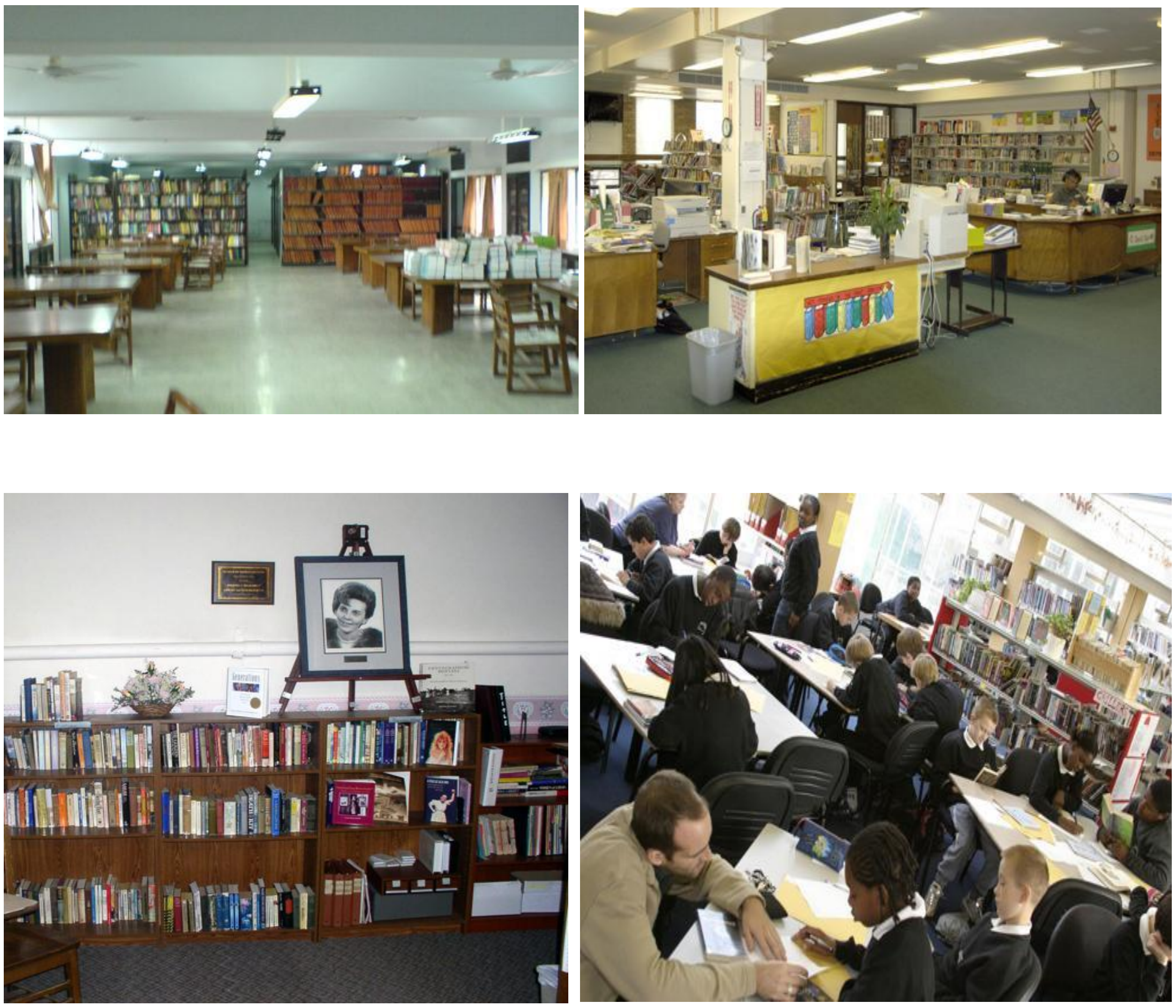
2) Accuracy = trace (conf) / sum (conf(:))= 93/160 = **0.58125**
Thus the accuracy achieved traditionally with minor adjustments in k and alpha was 58.125 %

2.6 Failed Test Cases

In the observed confusion matrix, there are some general misclassifications in the testing set however some of them are more prominent than others. One such example is the library vs computer room pair.



Some Computer Rooms wrongly classified as library



Some libraries wrongly classified as computer room

Since the libraries and computer rooms have a lot of things in common like desks, chairs, indoor lighting conditions and many other features that have common textures and colors, the model fails to efficiently distinguish between the two classes.

In the feature extraction step, a lot of similar features are extracted for these two classes. Thus, there will be many instances where the pixels in the wordmap of these images will have been mapped to the same cluster.

As a result, our algorithm makes similar or comparable histograms in the train_features matrix for both library and computer room images and its hard to distinguish between the two.

There are other such instances as well where tennis courts are classified as gardens and this is due to similarity of features in images like lawn/grass whose pixels are mapped to the same cluster index thereby leading to similar/comparable histograms.

2.7 Custom Improvements

After playing with values of k and alpha in the original models, I was achieving an accuracy of 58.125%. So to improve the model further, I tried to use a different method for comparing histograms and it improved the accuracy to 63.75% yielding an improvement of 5.625%.

Histogram matching using K nearest neighbors

I tried applying the k nearest neighbor for histogram comparison and image guessing. Essentially, while classifying the incoming images, instead of finding just the closest instance in the training data set, I found out k closest instances to the input test image and then used the k nearest neighbors' algorithm to predict the class. I have used unweighted voting where in each of the neighbors votes is given equal weightage and thus the label with maximum number of votes is selected as the output label. This can be simply achieved by using the mode function.

For k = 1, we have the initial confusion matrix given by traditional solution with no adjustments as seen in section *2.5*

Here is how the confusion matrix and accuracy vary with different values of k in knn :

For k = 25 in knn,

10	0	0	4	6	0	0	0
2	12	0	2	4	0	0	0
0	0	18	0	1	1	0	0
1	1	1	16	1	0	0	0
1	7	0	1	11	0	0	0
0	0	2	0	1	11	6	0
1	1	0	0	2	2	14	0
2	1	4	2	4	0	2	5

Accuracy = 60.625 %

For k = 30 in knn,

Confusion matrix :

11	0	0	4	5	0	0	0
2	13	0	2	3	0	0	0
0	0	19	0	0	1	0	0
1	0	1	17	1	0	0	0
1	7	0	1	11	0	0	0
0	1	3	0	1	11	4	0
1	1	0	0	2	1	15	0
2	1	5	1	4	1	1	5

Accuracy = 63.75 %

For k = 40 in knn,

10	1	0	4	5	0	0	0
2	12	0	2	4	0	0	0
0	1	19	0	0	0	0	0
1	0	1	17	1	0	0	0
2	7	0	1	10	0	0	0
0	1	3	0	1	10	5	0
1	1	0	0	2	1	15	0
2	3	7	1	2	1	1	3

Accuracy = 60 %

To summarize:

Accuracy for different values of k in knn for histogram matching :

k in knn	Accuracy
1	58.125 %
25	60.625 %
30	**63.75 %**
40	60 %

Thus, I realized that 30 was a sweet spot for out data set and above that the accuracy started to drop signifying that there was just too much noise being taken into consideration.

The *guessImageknn.mat* file found in custom should be used to guess images instead of the default *guesImage.mat* with knn variable set to 30 for 63.75% accuracy.