# Programming

## Part 1: Build Visual Words Dictionary

### Q1.1 Extract Filter Responses

Figure 1: Chosen image from the data set

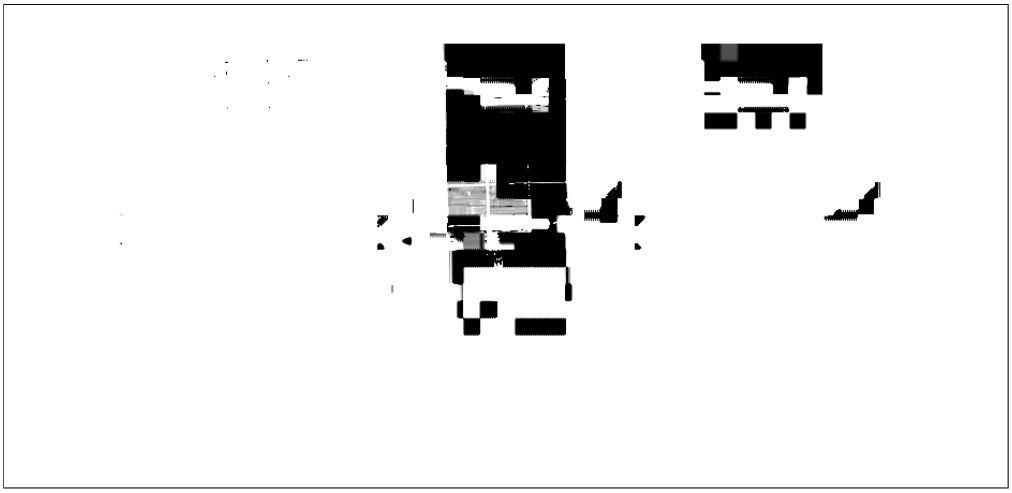

Figure 2: CIE Lab channels of the image (Left: L, Middle: a, Right: b)

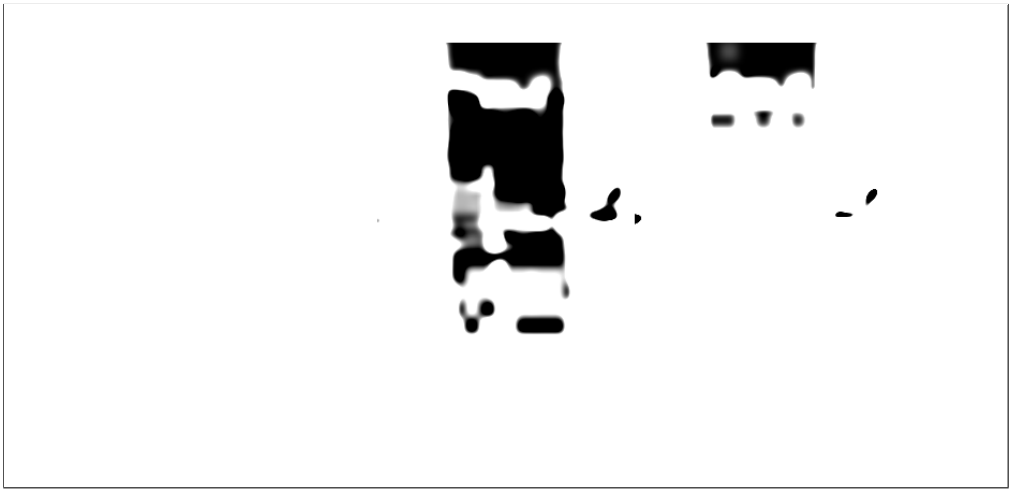

Figure 3: Gaussian 21-by-21 filter applied on the image

Figure 4: Laplacian of Gaussian 21-by-21 filter applied on the image

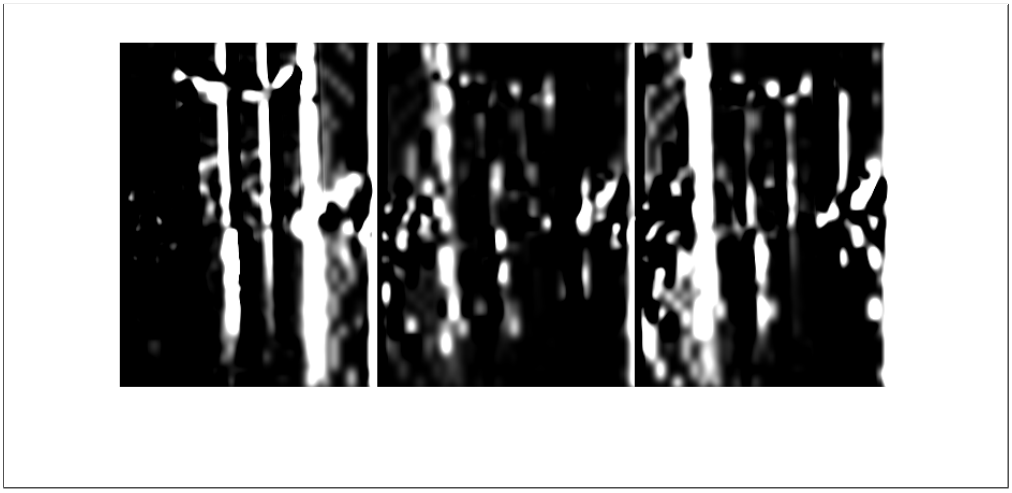

Figure 5: X-gradient of Gaussian 21-by-21 filter applied on the image

Noticed artifacts include blurring, brightening and sharpening of edges.

The CIE Lab color space expresses colors in terms of L for lightness from 0 for black to 100 for white, a for green (negative) to red (positive); b for blue (negative) to yellow (positive). This color model closely matches human perception, unlike the RGB model which models the output of physical devices. More accurate color balance corrections can thus be achieved by adjusting a and b, and L for adjusting lightness contrast.

### Q1.2 Collect sample of points from image

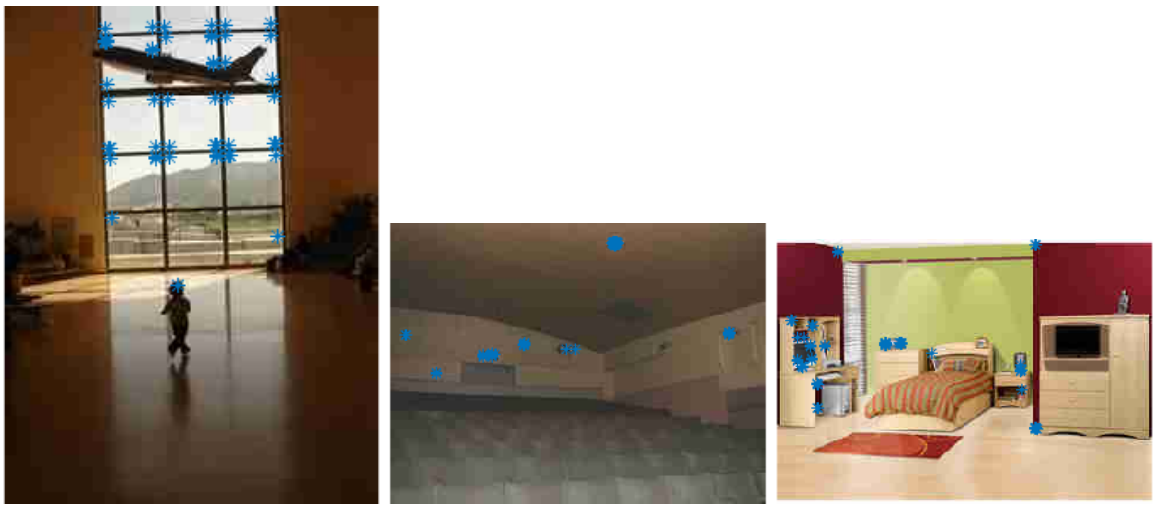

Figure 6: Harris corner detector on three images.

## Part 2: Build Visual Scene Recognition System

### Q2.1 Convert image to word map

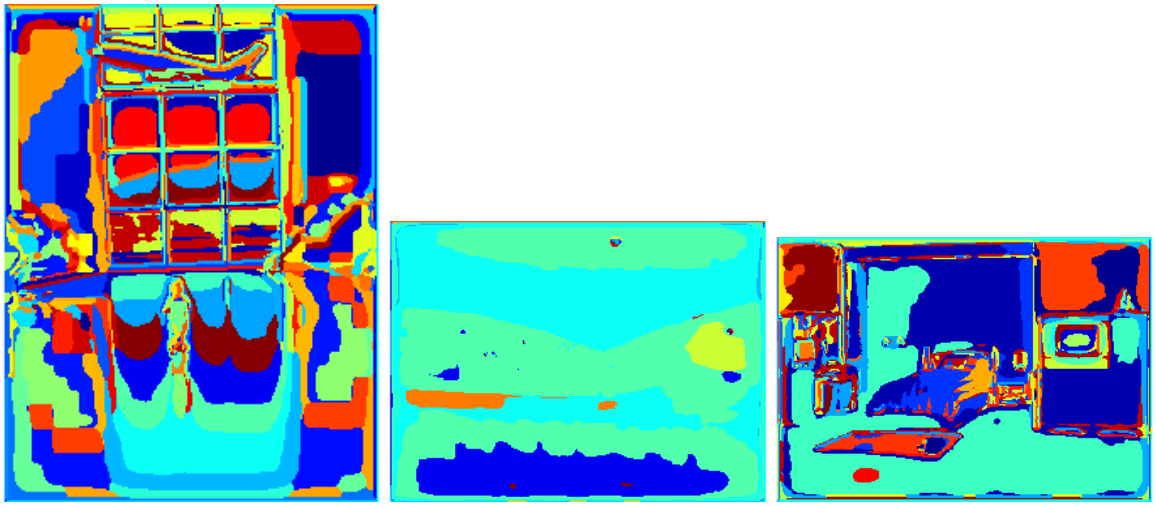

Figure 7: Word Map using Random Points Dictionary.

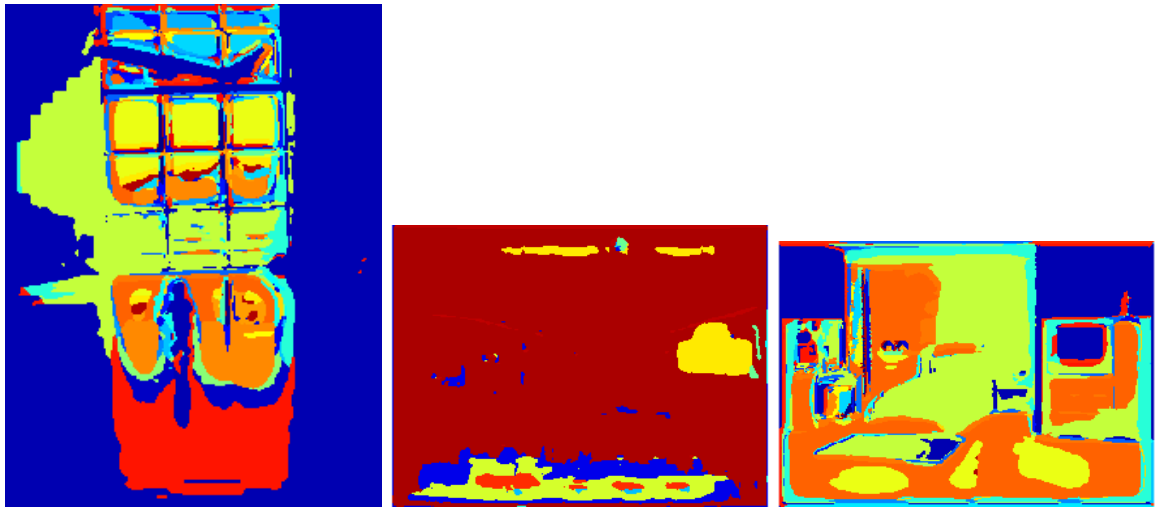

Figure 8: Word Map using Harris Points Dictionary.

The word maps do kind of capture semantic meanings. However, comparing the performances between using random points and Harris points, the former ones have nosier word maps and therefore we believe it does not perform as well as the Harris' ones.

# Part 3: Evaluate Visual Scene Recognition System

### Q3.2 Evaluate Recognition System - NN and kNN

In NN:

Let $C_{\text{VISION,METRIC}}$ be the confusion matrix using vision data VISION $\in \{h, r\}$ for `visionHarris.mat` and `randomHarris.mat`, and distance metric METRIC $\in \{e, c\}$ for `euclidean` metric and `chi2` metric respectively; similarly let $\text{acc}_{\text{VISION,METRIC}}$ be the corresponding prediction accuracy.

- $C_{h,e} = \begin{bmatrix} 8 & 3 & 4 & 0 & 0 & 0 & 1 & 4 \\ 4 & 11 & 3 & 1 & 0 & 0 & 1 & 0 \\ 3 & 4 & 6 & 1 & 4 & 1 & 0 & 1 \\ 2 & 3 & 1 & 4 & 0 & 3 & 6 & 1 \\ 0 & 1 & 5 & 3 & 5 & 1 & 4 & 1 \\ 4 & 2 & 2 & 3 & 1 & 4 & 1 & 3 \\ 4 & 2 & 1 & 3 & 1 & 1 & 7 & 1 \\ 3 & 1 & 0 & 2 & 0 & 0 & 0 & 14 \end{bmatrix}$ with $\text{acc}_{h,e} = 0.3688$.
- $C_{h,c} = \begin{bmatrix} 12 & 1 & 2 & 0 & 0 & 0 & 1 & 4 \\ 5 & 9 & 2 & 1 & 2 & 1 & 0 & 0 \\ 3 & 4 & 9 & 1 & 2 & 1 & 0 & 0 \\ 3 & 3 & 1 & 6 & 1 & 2 & 4 & 0 \\ 0 & 2 & 3 & 0 & 10 & 0 & 4 & 1 \\ 1 & 4 & 4 & 2 & 1 & 4 & 3 & 1 \\ 2 & 1 & 1 & 6 & 1 & 2 & 6 & 1 \\ 1 & 1 & 0 & 3 & 0 & 0 & 1 & 14 \end{bmatrix}$ with $\text{acc}_{h,c} = 0.4375$.
- $C_{r,e} = \begin{bmatrix} 11 & 2 & 3 & 0 & 0 & 0 & 0 & 4 \\ 4 & 11 & 1 & 0 & 1 & 0 & 1 & 2 \\ 3 & 3 & 9 & 0 & 2 & 2 & 0 & 1 \\ 1 & 1 & 2 & 6 & 0 & 1 & 4 & 5 \\ 2 & 2 & 1 & 1 & 7 & 1 & 3 & 3 \\ 2 & 2 & 3 & 1 & 0 & 9 & 2 & 1 \\ 3 & 2 & 1 & 6 & 0 & 2 & 5 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 16 \end{bmatrix}$ with $\text{acc}_{r,e} = 0.4625$.

- $C_{r,c} = \begin{bmatrix} 12 & 2 & 1 & 1 & 0 & 0 & 0 & 4 \\ 6 & 9 & 1 & 1 & 2 & 1 & 0 & 0 \\ 2 & 3 & 11 & 0 & 2 & 0 & 0 & 2 \\ 4 & 2 & 2 & 6 & 1 & 1 & 2 & 2 \\ 2 & 2 & 3 & 3 & 5 & 2 & 3 & 0 \\ 1 & 3 & 3 & 1 & 1 & 9 & 1 & 1 \\ 0 & 1 & 2 & 7 & 1 & 2 & 6 & 1 \\ 3 & 0 & 1 & 1 & 0 & 0 & 0 & 15 \end{bmatrix}$ with $\text{acc}_{r,c} = 0.4652$.

Both dictionaries have similar performances, with the random points one being slightly better. It is a bit surprising because Harris points carry semantic meaning but apparently that is not quite useful.

Using `chi2` metric performs better. We believe that is because `chi2` metric makes more sense in this situation. The vectors we are comparing with one another are essentially pdfs, which `chi2` is designed for. Euclidean norm, on the other hand, carries no meaning.

In kNN using `visionRandom.mat`:

Confusion matrix $C = \begin{bmatrix} 15 & 0 & 1 & 0 & 0 & 0 & 0 & 4 \\ 2 & 14 & 2 & 1 & 0 & 0 & 1 & 0 \\ 4 & 2 & 12 & 0 & 1 & 0 & 0 & 1 \\ 5 & 2 & 1 & 8 & 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 1 & 11 & 0 & 3 & 1 \\ 2 & 2 & 4 & 3 & 1 & 7 & 0 & 1 \\ 2 & 1 & 3 & 2 & 3 & 2 & 6 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 18 \end{bmatrix}$.

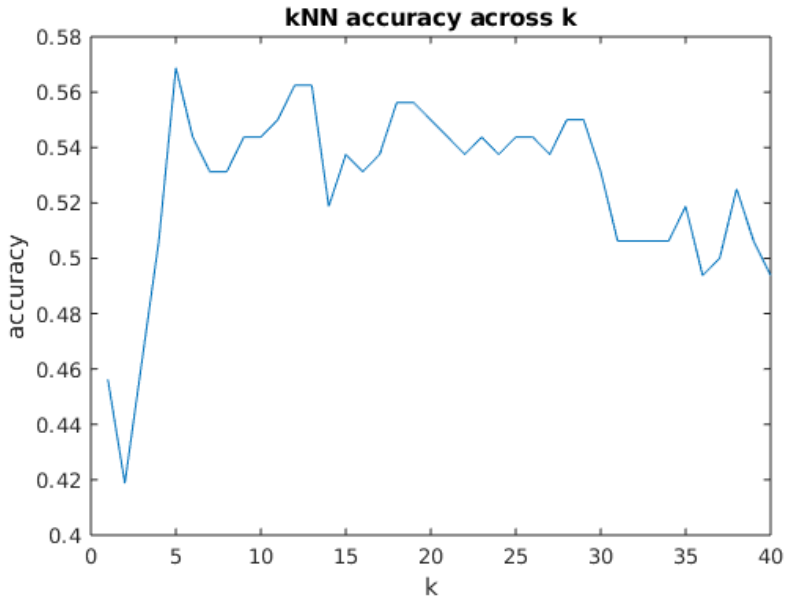


Figure 9: Plot of kNN accuracies across $k$

A larger $k$ is not always better, because when too many neighbors are considered in deciding the predicted labels, the class labels of data points that are further away from the target but are larger in number, can overwhelm the correct class label, making the prediction more noisy; in the extreme case, if $k$ equals to the total number of data points, then the predicted class label is effectively the mode of class labels of all data points - irrelevant of the prediction.

In the case of having two different class labels that have the same maximum number of neighbors, ties are broken by just choosing the first occurrence of the class labels according to the order of test images in the folder.

## Extra Credits

### QX.1 Evaluate Recognition System - Support Vector Machine

The SVM library used is LIBSVM. The SVM model used is `nu-SVM`. The two kernels used are linear are rbf (gaussian) kernels, and their accuracies are respectively 0.6250 and 0.63125. Model is trained with `visionRandom.mat` as the `visionSVM.mat`.

The model performs better than NN, it might be because SVM is a more robust model. The non-linear rbf kernel works slightly better than the linear kernel, probably because it can also handle non-linear class separating boundaries.

### QX.2 Inverse Document Frequency

Using IDF on NN (`visionRandom.mat` and `chi2` metric) gives 0.4562 accuracy, which is lower than that without IDF. Using IDF on kNN (`visionRandom.mat` and `chi2` metric) gives 0.5437 with best $k$ being 37, which is also lower than that without IDF. This makes sense because frequent words maybe important features in determining the classification results, and appearing more frequently does not necessarily make the visual word less important as assumed by IDF. On the other hand, if such important but frequent words are weighted close to zero, the classifier cannot use much of these features and this adversely affect the accuracy.

### QX.3 Better pixel features

We experimented with HOG, in which each histogram is properly normalized to remove variance across image sizes. Using this new way of representing features, SVM is then used to perform classification. It performed reasonably well given the relatively simple features, with 0.53125 accuracy after some tuning.