# Problem 1: Representing the World with Visual Worlds

### 1.1: Extracting Filter Responses

1. What properties do each of the filter functions pick up? (See Fig 3) Try to group the filters into broad categories (e.g. all the Gaussians). Why do we need multiple scales of filter responses?

   The properties of the filters are as follows:

   (a) **Gaussian Filters** are used to blur images. The extent of blurring is determined by the standard deviation of the filter; the higher the sigma, more the blur. Traditional Gaussian filters usually act as low pass filters as blurring results in loss of high frequency features such as edges.

   (b) **Laplace of Gaussian** filters are primarily used for edge detection. They are used to extract high frequency features such as edges and localizing the location of such features using ingenious approaches such as zero crossing.

   (c) **derivative of Gaussian in x** filter is also known as a vertical edge detector as there is a patent change in gradient of intensity in x direction for vertical edges.

   (d) **derivative of Gaussian in y** filter is also known as a horizontal edge detector as there is a patent change in the gradient of intensity in the y direction for horizontal edges.

   We need multiple scales of filter responses because filtering can sometimes lead to loss of critical information. In particular, for our case we would be dealing with a multitude of images that might differ in sizes. Thus, having filters of multiple scales would ensure that we can extract salient features of the given images.

### 1.1.2: Extracting Filter Responses:

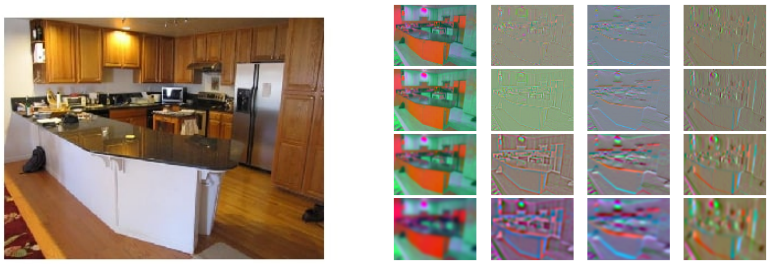

Figure 1: Collage of kitchen image with various filter scales: 1,2,5,10 (top to bottom)

## 1.2: Creating Visual Words

1. The following parameters were used to create the dictionary for the given training set:
    - Number of clusters(K): 10
    - Number of pixels randomly sampled from each image for creating a wordmap ($\alpha$): 25

## 1.3: Computing Visual Words

The images below show the effect of using wordmap on various images from different datasets. The wordmap was constructed for each image by combing four different filter responses at four different scales. The scales used were 1, 2, 5, and 10.

Figure 2: Visual wordmap for the kitchen image with varying number of clusters

Overall, it can be concluded that as the number of clusters is increased, the color map becomes more vivid and the image has more color differentiation with a larger color palette. However, it was noted that increasing number of clusters also increased the computational load. Similar trends can be observed in the images below. The word **"boundaries"** does make sense to me based on the huge gradient on the boundaries of these images where there is a patent variation in color for each instance.

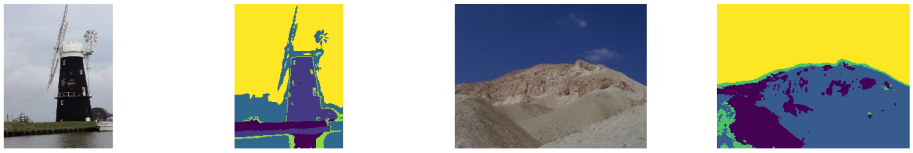

Figure 3: Original image and visual wordmap for the windmill image with 77 clusters

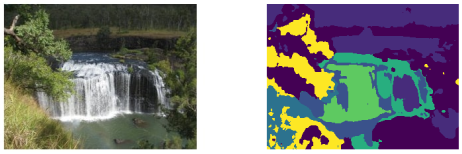

Figure 4: Original image and visual wordmap for the windmill image with 25 clusters

# 2: Building a Recognition System

## 2.5: Quantitative Evaluation

**Confusion Matrix:**

$$\begin{bmatrix} 32 & 1 & 1 & 3 & 1 & 1 & 7 & 4 \\ 0 & 19 & 6 & 13 & 7 & 0 & 2 & 3 \\ 0 & 7 & 22 & 0 & 4 & 2 & 6 & 9 \\ 2 & 4 & 2 & 23 & 13 & 2 & 4 & 0 \\ 1 & 3 & 3 & 15 & 18 & 2 & 7 & 1 \\ 1 & 0 & 3 & 1 & 4 & 29 & 7 & 5 \\ 15 & 0 & 3 & 1 & 3 & 4 & 20 & 4 \\ 1 & 5 & 5 & 2 & 5 & 5 & 10 & 17 \end{bmatrix}$$

**Accuracy: 45%**

## 2.6: Failure Modes

As the classifier only works accurately ∼65% of the test images, there are some images that the classifier was unable to classify correctly. The following issues could explain the reasons for the inaccuracies in the image classifier:

- Difference in scaling and zoom level could prevent the classifier from determining the distinctive features in the testing image.
- Images being compared from similar scenes can have a lot of unwarranted and different objects that could throw-off the classifier into believing that the images being juxtaposed are dissimilar.
- The images could have been taken at starkly different lighting conditions, which could have impacted the pixel intensities, resulting in seemingly different images for the classifier. The angles at which the images were captured also matter a lot.
- The differences between the images being compared might be too subtle and there might not be a lot of features to extract and compare with. **Some examples of failures are shown below:**

Figure 5: (Left) Example of a training image. (Right) A test image that was not classified accurately.

The comparison above shows that although the two images look similar, they were classified incorrectly. This could have to do with the plethora of small fish in the image from the training dataset. Therefore, in this case, even though some of the features from the large fish would be similar, the images might be too varied overall to be classified as similar.

- The failure modes above can be addressed by implementing one or more of the following solutions:

  1. Increasing the number of clusters (K)
  2. Increasing the number of random pixels being sampled from each image ($\alpha$)
  3. Increasing the number of layers in the Spatial Pyramid Matching (SPM) process
  4. Increasing the training data to have more diverse settings for each scenario

# 3. Improving Performance

## 3.1 Hyperparameter Tuning

| Filter Scales | # Visual Words(K) | # Random Samples ($\alpha$) | # SPM Layers(L) | Accuracy(%) | Run-time(minutes) |
|---|---|---|---|---|---|
| [1, 2, 5, 10] | 10 | 25 | 3 | 45 | 14.22 |
| [1, 2, 5, 7, 10, 25, 50] | 10 | 25 | 3 | 44.5 | 37.5 |
| **[1, 2, 5, 7, 10]** | **150** | **100** | **3** | **61.75** | **26** |

Table 1: Iteration results from hyperparameter tuning. (Optimal results in bold)

**Conclusions:**

The tuning of the hyperparameters in 2 highlighted the following pertinent trends:

- Increasing the number of filter scales does not help in bolstering the accuracy of the system. However, it does impact the run-time in a negative manner, decreasing the overall efficiency of the code.
- Increasing number of visual words/clusters had a positive impact on the performance as it bumped the accuracy of the model upwards by $\sim$10%. Moreover, increasing the clustering resolution also increases the execution time noticeably.
- Increasing the sampling portion from each image for creating more accurate wordmaps did help with improving the performance of the model by $\sim$4% with a marginal $\sim$6% trade-off in computation time.
- Varying the number of layers employed for spatial pyramid matching yielded interesting results. Creating and matching higher resolution histograms did not improve accuracy from the previous test case. The run-time remained commensurate to the previous iteration.

### 3.2: Further Improvement

For further improving the performance of the recognition system the weight of the layer with the highest resolution was increased. This allowed the similarities or discrepancies in layer with most sections to be more pertinent in the final calculation(s). In my case, changing the relative weights for layers 0,1,and 2 decreased the accuracy by 3%. The reason for the slight dip in accuracy could be attributed to the uneven distribution of weights where the lower layers, which are weighed are weight too low resulting in loss of critical matching information, and thus reducing the accuracy of the system.

| Filter Scales | # Visual Words(K) | # Random Samples ($\alpha$) | # SPM Layers(L) | Accuracy(%) | Run-time(minutes) |
|---|---|---|---|---|---|
| **[1, 2, 5, 7, 10]** | **150** | **100** | **3** | **58.25** | **32** |

Table 2: Iteration results from further hyperparameter tuning