# EN1610 Image Understanding

# Lab # 5: Image Classification, Object Detection

The goal of this fifth lab is to

- Understand a SIFT descriptor
- Learn about the bag of words model, object detection systems

All the images you will need for the lab can be downloaded off the class website, `http://vision.lems.brown.edu/engn161/fall2011/Labs`.

## Installing SIFT

For this lab we will be using the implementation of SIFT from `http://www.vlfeat.org/`. Please download, and install the code. Go through the tutorial `http://www.vlfeat.org/overview/sift.html`

## Region Descriptors - SIFT

**Problem 1. Implement a SIFT descriptor for the region** We will continue with the Harris corners you developed in the previous lab. Now we will attach a SIFT descriptor to them.

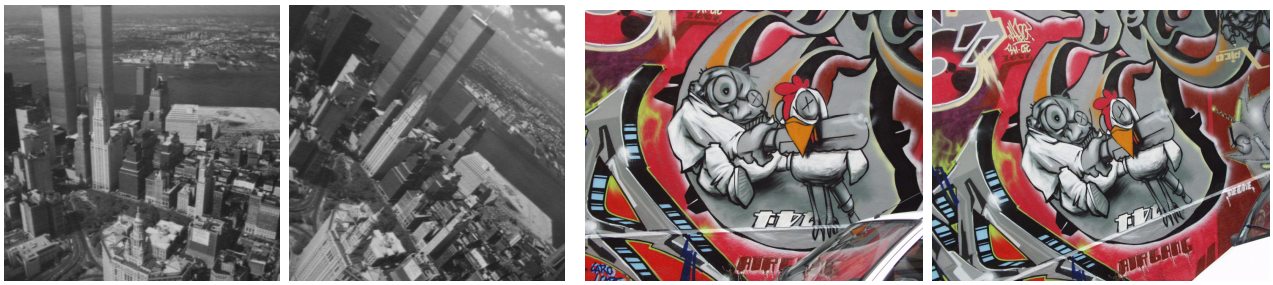
SIFT features are formed by computing the gradient at each pixel in a 16 x 16 window around the detected feature points. In each 4 x 4 quadrant of the window, a gradient orientation histogram is formed by adding weighted gradient value to one of the eight orientation $(0, \frac{1}{4}\pi, ..., \frac{7}{4}\pi)$ histogram bins, which can be done by projecting the gradient onto the 8 orientations, but only considering the positive projected values. The resulting 128 (8 orientations x 16 cells) non-negative values from a raw version of the SIFT descriptor vector. The reduce the effects of contrast or gain, the 128 dimensional descriptor should be normalized.

Repeat the same procedue as in Problem 3 in Lab4 on the images in Figure 1. Compare against the best forming region descriptor (pixel or histogram) in the previous lab.

## Image Classification

**Problem 3. Bag-of-Words Classifier** In this problem, you will implement a simplified *Bag-of-Words* model to solve a simple classification problem. The goal is to perform *four-class image classification*, with the four classes being *airplanes, motorbikes, faces , and cars.* See Figure 2.

1. Training State - Form a **Codewords Dictionary**.



(a) Plane Rotations (b) ViewPoint Changes

Figure 1: Images to Test Descriptors

Figure 2: Image Classification Data Set

- Compute SIFT descriptors on a random subset of images across all categories. The number of images from each category, constitutes the size of your training set. Try a training set size of 1,5,10,15 where a training size of 1 means, one image from each of the four categories
- Cluster the SIFT descriptors using the $k$-means algorithm. Let the $k$ centers be the number of **codewords**. You will have to play around with $k$, maybe on the order of 500 to 1000
- Now go back over all training examples, and map the SIFT features into its **bag-of-words histogram**. The histogram for image $I_j$ is a k-dimensional vector: $F(I_j) = [f_{1,j}, f_{2,j}, ..., f_{k,j}]$ where each entry $f_{i,j}$ counts the number of occurrences of the $i$-th visual word in the image, and $k$ is the number of total words in the dictionary. The $i$-th visual word, SIFT descriptor, is assigned to the cluster center that has the minimum Euclidean distance. In other words, a single image's list of $n$ SIFT descriptors yields a $k$-dimensional bag of words histogram.
- At the end of this stage, you should have a histogram for each training image, If you used 5 training examples, for each of the four categories, you would have 20 **bag-of-words** histograms

2. Testing State - Given a new image, classify it into one of four categories
   - Randomly, select a group of images , not from the training set, this will serve as your

test set.

- We repeat the procedure we did for training images, compute SIFT descriptors on each test image, and compute a bag-of-words histogram
- For each test image, determine a category using a $k$nn classifier. Try k=1,3,5

3. Evaluation - How good is your classifier?

   - Compute the accuracy by looking at the percent classified correctly. For example, if you tried to classify 100 images, and you got 80 correct report 80 percent
   - Plot the percent classified correctly as a function of your training size. For example, if I tried classifying 100 images, with one training example per category what is my classification rate? a training set of 5? etc. (You will notice that as your training size increases your classification rate will go up)

# Object Detection - Template Matching

**Problem 2. Edge Based Template Matching** We will try to detect objects using template matching. We will define a template, $T$, and try to find that template in an image $I$.

1. Construct a template around the object of interest, in this case, a binary image
2. Compute the edges of the image, and get a binary map
3. Compute the distance transform of the binary edge map, use `bwdist`
4. Slide your template around the image, exactly as you would apply a filter, and at each pixel in the image compute a distance
5. For this problem you will try two different metrics, again in both cases here the object will be found where the distance is minimized

   - The *Chamfer* distance between $T$ and $I$,Equation 1 is the average distance to the nearest feature

   $$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t) \tag{1}$$

   where $d_I(t)$ represents the distance to the nearest feature.

   - The $Hausdorff$ Measure, Equation 2, is the distance of the worst matching template pixel to its closest image pixel.

   $$D_{hausdorff}(T, I) = \max_{t \in T} \min_{i \in I} \|t - i\| \tag{2}$$

6. Threshold your resulting image, in this case, it will be either $chamfer$ distance map, or $Hausdorff$ distance map, and declare an object detection

# Object Detection -

**Problem 4.** The goal of this assignment is to perform object detection using a voting-based Generalized Hough Transform approach. Given a set of cropped training images showing the class of interest, the method first generates a set of prototypical object parts. Well use visual words formed from local image patches extracted at Harris corner interest points to represent the parts. That is, object part = visual word. Having identified the object parts, the method stores the possible displacement vectors (translations) between each part and the objects center, as observed in all the training examples. Given a novel test image, the local patches are first mapped to object parts, by assigning the patch around each interest point to the nearest prototype (visual word). Then each part uses the stored displacement vectors to cast votes for the position of the object. By looking for peaks in that vote space, one or more detections are made. Note that the object of interest may appear multiple times in a test image, at any location. However, we will assume that the scale and orientation of the object is fixed; that is, the scale and orientation at which it appears in the training images are both constant, and will also be similar in the test images. Thus the vote space is only 2-dimensional.

To summarize the main steps required:

- Training stage - Prepare the visual vocabulary and GHT displacement vectors:
  - Run the Harris corner detector on each training example to collect its interest points.
  - At each interest point, extract a fixed size image patch (e.g., 25 x 25 pixels), and use the vector of raw pixel intensities as the descriptor.
  - Cluster the patches from the training images with k-means to form a visual vocabulary, where each mean represents an object part discovered in the training data.
  - Having found the vocabulary, go back over the training examples and assign their local patches to visual words (i.e., the prototype object parts). An image patch is assigned to the visual word to which it is closest using Euclidean distance.
  - For each visual word occurrence in the training examples, record the possible displacement vectors between it and the object center. Let the object center be the center of the cropped training image. Note that a given visual word may have multiple displacement vectors, depending on its appearance. (For example, see the wheel-like word in the figure above.)
- Testing stage - Given a novel test image, detect instances of the object:
  - Run the Harris corner detector on the test image to collect its interest points.
  - At each interest point, extract a fixed size image patch, using the same scale selected above. Use the vector of raw pixel intensities as the descriptor.
  - Assign each patch to a visual word.
  - Let each visual word occurrence vote for the position of the object using the stored displacement vectors.

- After all votes are cast, analyze the votes in the accumulator array, threshold, and predict where the object occurs. (To predict the fixed-size bounding box placement, assume object center = bounding box center.) Note that the object of interest may occur multiple times in the test image.
- Compute the accuracy of the detectors predictions, based on the overlap between the predicted and true bounding boxes. Specifically, a predicted detection is counted as correct if the area of the intersection of the boxes, normalized by the area of their union, exceeds 0.5.