Computer Vision I

Project 4
Given: Nov 15, 2019, Due on Dec 6, 2019

Object Category Detection

In this project you will implement an algorithm to detect objects of a given class by using a Generalized Hough Transform (GHT) Approach.

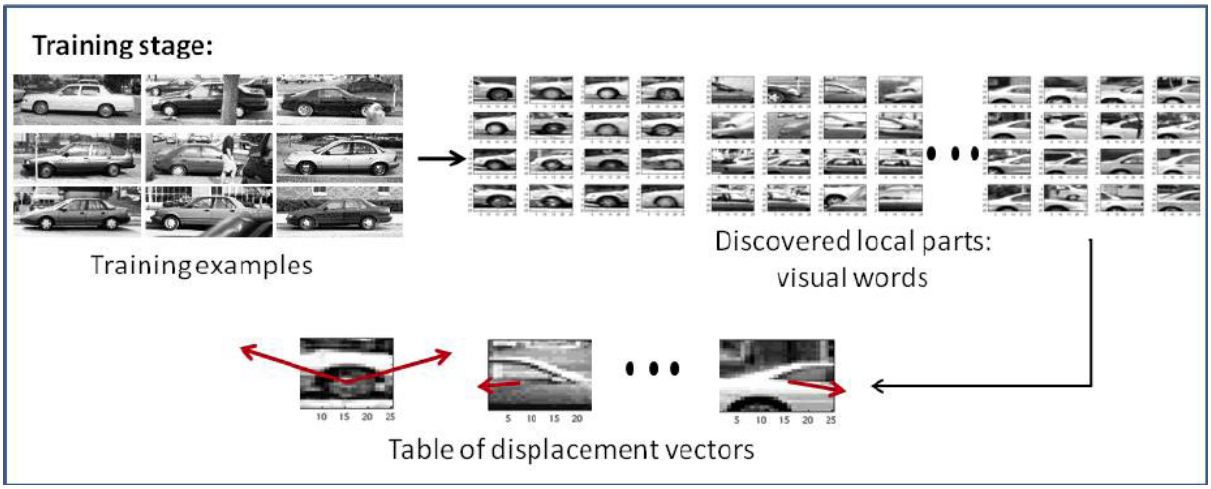
Given a set of cropped *training images* showing examples of objects in the class of interest, the method first generates a set of prototypical *"object parts"*. We will call these parts *visual words*. In this project, you will use small patches (25×25 pixels) around interesting points found using a Harris corner detector (you can use your code from the previous projects) to represent the parts. Once the parts have been identified, the method stores the possible displacements (translations) between each part and the object's center, as observed in the training examples.

Given a novel test image, the local patches are first mapped to object parts, by assigning the patch around each interesting point to the closest visual word. Then, each part uses the stored displacement vectors to vote for a position of the object. By looking for peaks in the voting space, one or more objects can be detected.

The object of interest may appear multiple times in the test image, at any location. However, we will assume that the scale and orientation of the object is the same as the ones used during training. That is, the voting space for the GHT is two-dimensional.

To summarize the approach consists of the following steps:

1. Training; Given a set of training cropped images prepare a vocabulary and GHT translation vectors:



(a) Use a Harris corner detector to collect interesting points in the training images.

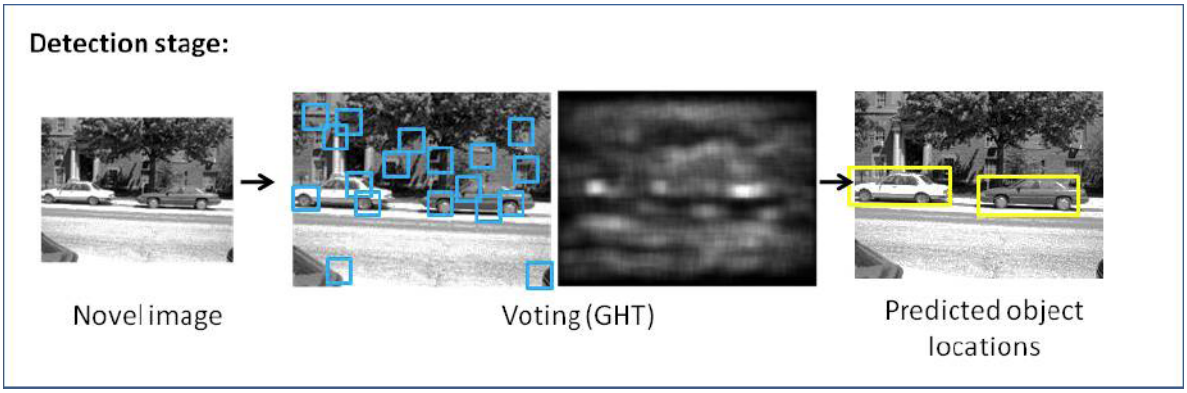
(b) At each interest point, extract a fixed size image patch (for example, 25×25 pixels) and use the vector of raw pixel intensities as the "descriptor".

(c) Cluster the patches from the training images into clusters using K-means (use `kmeans` command in Matlab). This step is meant to significantly reduce the number of possible visual words. The clusters that you find here constitute a *"visual vocabulary"*.

(d) Having found the vocabulary, go back to each training example and assign their local patches to visual words in the vocabulary. An image patch is assigned to the visual word which is closest using Euclidean distance (SSD).

(e) For each visual word occurrence in the training examples, record the possible displacement vectors between it and the object center. Assume that the object center is the center of the cropped training image. Note that a given visual word may have multiple displacement vectors, depending on its appearance (for example, the "wheel-like" word in car images.)

2. Testing; Given a novel test image, detect instances of the object:



(a) Run the corner detector to find interesting points.

(b) At each interesting point, use a fixed image patch (of the same size as the ones used during training) to create a raw pixel descriptor.

(c) Assign to each patch a visual word.

(d) Let each visual word occurrence vote for the position of the object using the stored displacement vectors.

(e) 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 that the object center is the bounding box center. Note that the object of interest may occur multiple times in the test image.

(f) Compute the accuracy of the 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.

Project Requirements:

1. Write a program to implement the above algorithm. The following data are provided in blackboard:

(a) `CarTrainImages`: 550 cropped training images of cars, each 40×100 pixels. If you encounter memory issues, it is fine to reduce the training set size as needed.

(b) `CarTestImages`: 100 test images. Each image has one or more cars among cluttered background.

(c) `GroundTruth`: Data file containing the ground truth bounding boxes of the test images. Load it using `load`. The variable `groundtruth` is a struct where `groundtruth(i)` contains the box for the i^{th} image. The box is specified in terms of the top left corner position, the width and height of the box. Each box gets a row in the list of locations: `groundtruth(i).topLeftLocs = [ x1, y1 ; x2, y2 ; ...; xn, yn ]` if there are n boxes in image i.

2. **Write a report.** The report should include:

(a) Abstract, description of algorithms, experiments, values of parameters used (How does the size of the vocabulary set affect your results?), observations and conclusions.

(b) A FLOWCHART.

(c) An appendix with your source code