Using Hough Forests for Object Detection

Siamak Safarzadegan, Nam Vo, Unaiza Ahsan
Georgia Institute of Technology
North Avenue, Atlanta, Georgia 30332, USA
{ safarzadegan, namvo, uahsan3 }@gatech.edu

1. Problem

Object detection is a challenging problem in computer vision, despite recent progress. The main reason is the high intra-class variability between object class instances, and generalization to unknown classes [8]. The problems complexity increases with varying illumination conditions and background clutter. Hough forests [5] was recently introduced as a powerful method that produces state-of-the-art results on computer vision problems such as object detection and action recognition [10]. Hough forests are random forests [2] tailored to compute the generalized Hough transform efficiently [5]. They not only improve the performance of generalized Hough transform, but their application can be extended beyond object detection, to action recognition and object tracking.

2. Related Work

There have been many successful ideas over the last few years in the field of object detection in static images [3, 4, 9]. One of them is to utilize generalized Hough transform, which roughly refers to any process based on additive aggregation of evidences, for detecting generic parametric shapes [1] and then for detecting object class instances [6]. This methodology has been extended to Implicit Shape Models [7] using the idea of appearance codebooks. Codebook-based detectors store the relative position of clusters of image features with respect to the object center from training set and learn the mapping from image features into a Hough space. The problem with large generative codebooks is that computational complexity increases as the size of the codebook increases. Instead, Hough forest, proposed by Gall [5] for object detection and action recognition, maps the appearance of an image patch to a Hough vote, rather than using an explicit codebook of part appearances, which makes it an efficient method.

3. Approach

The Hough Forest (HF) is a random forest that clusters image patches, much like the way ISM builds the vocabulary of visual words. Based on its feature value, an image patch travels from the root node of the trees in the forest to one of the leaves. In this sense, each tree in the forest plays the role of a vocabulary, and each leaf is a visual word. The leaves record relative object center position density, which is learnt during training, then can be used to accumulate votes on the hough space during detection phase. Local maximas in this hough space correspond to detections of object center (Figure 1).

3.1. Feature extraction

While corner detection and image descriptor such as SIFT are used in ISM, HF employs densely sampling of fixed size image patches. Feature extraction is applied to compute several feature channels, forming appearance feature patches \((I_i)\) with size \((w, h, n)\) where \((w, h)\) is the image patch size and \(n\) is the number of feature channels. In our implementation, we use size of 16x16, and 32 feature channels including LAB color-space value, HoG, filtering response, e.g., 1st and 2nd order gradient, max and min-filtering, etc.

During training, collected feature patches are labeled \(c_i\) positive if within object bounding box and negative otherwise. Relative object center displacement \((d_i)\) to each positive patch is also recorded. These feature patches, represented as \(P_i = \{I_i, c_i, d_i\}\), are then used to train the forest.

3.2. Tree construction

The random forest consists of a number of binary decision trees; each tree is constructed as follows: Start at root node with full set of labeled patches. 2 subnodes are created and a good binary test (discussed in Section 3.3) is chosen to divide this set to 2 subsets, each is assigned to one of the subnodes. This process is applied iteratively at the subnodes, thus growing the tree. Growing stops at a node when the size of set of training patches at that node is smaller than a threshold, or the node reach the (predefined) maximum depth of the tree. Such a node is leaf of the tree; then foreground probability and object center displacement of each leafs training patches are recorded. In this manner, the entire forest is constructed.

3.3. Binary Test at each leaf node

Binary test, represented by \((f, x_1, y_1, x_2, y_2, k)\), is simple pixel tests that take input as appearance feature patch and compares feature value on a pair of locations on the patch.

\[
t_{f,x_1,y_1,x_2,y_2,k} = \begin{cases} 0 & I^f(x_1, y_1) < I^f(x_2, y_2) + k \\ 1 & \text{otherwise} \end{cases}
\]

(1)

where \(f\) is the channel number, \((x_1, y_1)\) and \((x_2, y_2)\) are pixel locations in the patch, and \(k\) is an offset value. A "good" test is the one that divide the set into 2 sets (corresponding to the
Figure 1: Detection pipeline: each sampled patch is classified to one of the leaf by the HF, the object center displacement at that leaf are accumulated to the hough space which represents the detection result.

test result is 0 or 1) such that they have small uncertainty in either class labels or object center displacements [5]. Given a set of appearance feature patches: $A = P_i = I_i, c_i, d_i$, the measurement for class uncertainty is defined as:

$$U(A) = -|A| \sum_{c} p(c|A) \ln(p(c|A)),$$

(2)

where $c$ is the label (e.g., either positive or negative), $|A|$ is the number of patches in the set $A$, and $p(c|A)$ is the proportion of patches with label $c$ in the set. The measurement for displacements uncertainty is defined as:

$$W(A) = \sum_{P_i,c_i=1} \|d_i - d'|^2$$

(3)

Where $d_i$ is the object center displacement of the positive patch $P_i$, $d'$ is mean value of $d_i$.

A pool of randomly generated binary test is evaluated according to one of these measurements, and the best test (the one with smallest measurement) is chosen for each node. By randomly choosing which measurement is used at each node, the set of all training patches is divided so that the set at each leaf node has small uncertainty in both class label and object center displacement.

3.4. Detection

Appearance patches are densely sampled from testing image and passed to the forests trees as inputs. At each node, the binary test is used to determine which child node the patch is assigned to. This continues till the patch reaches a leaf node, and the process is repeated for all patches. Next, the leaf nodes that the patches arrive at, are used to cast probabilistic votes for the object center on the Hough space (Figure 1). Finally, meanshift or local maxima on the smoothed Hough space can be applied to get object center detection (Figure 2).

4. Experiments

We experimented on the TUD pedestrian dataset which includes 400 training images and 250 testing images. Since the background in TUD training set is quite poor, we augmented the training set with negative examples from INRIA

Figure 2: Sample detection result from the challenging TUD pedestrian dataset at highest recall. Green box is groundtruth, Blue is true positive detection, red is false positive detection. The numbers are detection responses.
pedestrian dataset like [5] did. We built a 10 trees HF using 8000 appearance feature patches of size 16x16 and 32 channel features; a set of 2000 binary tests is generated at each nodes. Our parameter is much weaker than the original implementation (15 trees which are trained in 3 separate rounds using 50000 appearance feature patches; a set of 20000 binary tests is generated at each nodes and sampling rate at detection phase is 20 time higher than ours). The result is depicted in Figure 3. It took us about 1 day to train the HF and 1 day to test on our machine (with CPU of 2.5GHz and RAM of 6GB). Overall detection on image with size of 320x240 for pedestrian at 100 pixels height takes 1 minute. Detection was done at the same multiple scales as original work.

5. Discussion

The performance of our HF is much worse than the original HF and even ISM model. This is clearly due to the fact that we did not use enough training patches. More importantly, Gall et al boosted their HF by sampling difficult examples based on the last trained trees in order to train subsequent trees. He however did not show empirically how much this multiple round of training will help. In our experiments, training all the trees at the same time resulting inferior performance suggests that this boosted scheme plays an important part in learning phase. In term of speed, the cost of densely sampling examples makes HF slow (especially our Matlab implementation is 10 times slower than reported speed of the original C/C++ version). One way to mitigate this is decreasing the sampling rate, which results in worse performance in our experiment. Due to time constraint, we limited ourselves on the problem of pedestrian detection. However this same approach has been applied to other object detection or human action detection.

References