Multiple Object Detection with Optimized Spatial Weighting

Jay Ni, Kiat Chuan Tan, Takashi Yonebayashi

Abstract

This paper illustrates a method to enhance spatial weighting in object recognition as presented by Marszalek and Schmid [2]. The original algorithm uses spatial relations and boundaries to weight relevant areas of an image, and is robust for single object detection. We have derived a method that is both less expensive to run, and can detect multiple object clusters in an image in a single pass. The two methods described in this paper will be called **Nearest Neighbor Weighting (NNW)** *and* **Correlation Tree Elimination (CTE)**. *Our tests have shown CTE to be effective, but NNW to be somewhat ineffective.*

1 Introduction

The Bag-of-Features representation is a popular image classification technique that uses local descriptors to create a visual vocabulary to represent image data. We achieve this representation by first clustering a random set of local image descriptors, and then discretizing the descriptors of a particular image to the closest respective words in our visual vocabulary. While this technique performs very well under simple circumstances, it is still vulnerable to intra-class variation, background clutter, and pose changes. The resulting descriptors obtained from unwanted noise makes Bag-of-Features less effective by increasing false positives. Zhang et al. [4] suggested that using context information within an image could significantly improve the test accuracy under such conditions.

Marszalek and Schmid [2] extended this idea and proposed a method for object recognition that exploits the spatial relations of an object to decrease background clutter in training images. Using localized features, they were able to generate segmentation masks for training images that gave the target objects a stronger weight than the background (See Figure 1). The algorithm, known as spatial weighting, led to a 1.5% improvement over the original ERR (Error Equal Rates) of 94.5% demonstrated by Zhang et al on the PASCAL training data. [4]. This improvement was mostly due to a higher success rates on the more difficult PASCAL training sets containing high background clutter. There was no significant improvement over images which did lacked background noise, as there was little or no background to remove. Nevertheless, their results showed promising improvements over training images that few classifiers can utilize.

Figure 1: Test images of Graz02 data set (left), generated masks (middle), multiplication of two (on right). Source: Marszalek and Schmid [2]

Marszalek and Schmid [2] further proposed an extention their spatial weighting algorithm by using the segmentation masks to localize the target object of an image. We can achieve this by selecting the point in the segmentation mask with the highest probability, and applying thresholding to determine the boundaries of the image. However, the segmentation masks generated by the original spatial weighting algorithm are only effective for images where only one object is present. Furthermore, the original algorithm is extremely runtime intensive during testing, as it involves convolving the image several times for each descrip-

tor in our test image. We have developed two multi-class variations (NNW and CTE) of the segmentation mask that will allow a single mask to detect multiple objects in one image. These methods are not computationally intensive and have shown significant improvements when compared to no weighting during our initial tests.

We will first describe our implementation for feature extraction based on the work of Marszalek and Schmid [2] in section 2. In section 3, we present a summary of the original spatial weighting algorithm and our two algorithms, NNW and CTE. Section 4 describes our final results, and proposed extensions to our work.

2 Feature selection and Bag-of-Features representation

Our feature selection is primarily based on the work of Marszalek and Schmid [2] as an extension of Zhang et al. [4]. Our method first extracts several local image descriptors (32-dimensional HS-SIFT and LS-SIFT) from each class in our training set, which it then clusters to create the bag-of-features over several classes of objects. This differs from the original method, which clusters the centroids over all descriptors randomly sampled from the entire data set, regardless of class. For a given test image, classification is based on the confidence of which class a particular descriptor belongs to. Several descriptors with high confidence intervals are then chained together to create the boundary of the object.

2.1 HS-SIFT and LS-SIFT

For scale-invariant feature detection, we implemented both the Harris-Laplace (HS) and Laplacian (LS) detectors for extracting corner and edge regions of our image. Our detectors, based on Lowe's implementation [1], sampled descriptors over 3 octaves of the image, using difference-of-gaussians (DoG) to calculate relevant keypoints. For all training and test images, we limited our number of descriptors to 400 HS-SIFT and 600 LS-SIFT to speed up both training and classification.

We then used Lowe's SIFT descriptor [1] to compute the gradient orientations of the local regions extracted by the HS detector. Using the HS points as the centers of 8 x 8 pixel windows, we first weighted the magnitudes of the pixels in each window using a Gaussian window of $\sigma = 4.0$. For each pixel, we then calculated its discretized relative orientation and its weighted contribution to its 4 x 4 grid (See Figure 2). The result was a 32-dimensional feature vector, which we normalized to unity to negate the effect of illumination. The final vector was clipped to a threshold to filter out extremities, and then renormalized for training. Since our method differs from the conventional SIFT algorithm in keypoint extraction, we did not use an external library to compute the descriptors, as it was more practical to implement by hand.

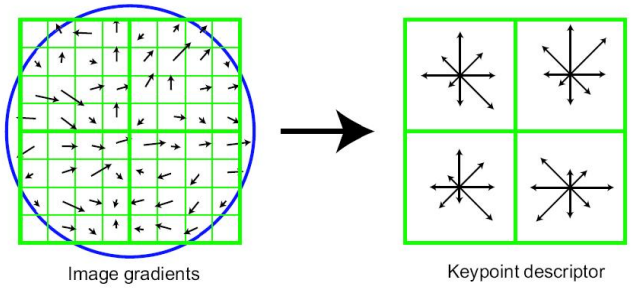

Figure 2: The calculation of a keypoint descriptor, weighted by a Gaussian window (in blue) and accumulated over 4x4 subregions. These generate a 4x8 = 32-dimensional feature space. Source: Lowe [1]

2.2 Class Based K-Means Clustering

During training, we sampled approximately 5000 descriptors from each of our five classes of images (mug, stapler, scissors, clock, and keyboard). Using K-Means clustering with k = 200, we clustered each set of 5000 descriptors, obtaining 5 centroid sets of 200 elements each (1000 centroids total). We can then assign each raw descriptor a confidence interval and a label by the closest (in 32-dimensional space) Euclidean distance centroid from all 1000 centroids.

During testing, after extracting our descriptors and running our filter (NNW/CTE), we determined each descriptor's closest centroid in our class based k-means centroids. The confidence level of the descriptor matching the centroid is given by the difference in the angles of their orientations in 32-dimensional space ($\frac{A \cdot B}{||A||_2^2 + ||B||_2^2}$.) We then filter the descriptors to only those that have a confidence level greater than

0.9, and for each descriptor, we search for other descriptors within δ pixel distance to the current descriptor (where δ increases with confidence), and repeat this process until we have a chain of descriptors. When the chain exceeds a certain threshold, the pixel location bounds of the chain determine the location of an object. For our purposes, we chose a threshold of 32 descriptors. Figure 3 illustrates the technique of chaining descriptors together.

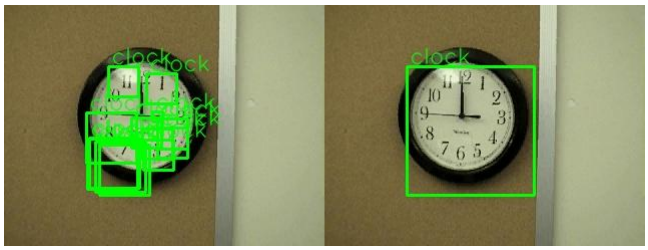

Figure 3: Chaining descriptors together to detect the location of an object.

```
01 TrainingSet.FilterFeatures();
02 for each TestImage in TestSet
03   Segmentation = 0;
04   for each TestFeature in TestImage
05     Hypothesis = 0;
06     for N closest TrainFeature in TrainingSet
07       M  = TrainFeature.GetImage().
            GetGroundTruthSegmentationMask();
08       T  = find_transformation(
            TrainFeature.GetPointOfView(),
            TestFeature.GetPointOfView());
09       M' = M.ApplyTransformation(T);
10       W  = gaussian(
            distance(TestFeature, TrainFeature),
            0, Sigma);
11       Hypothesis = Hypothesis + W * M';
12     Hypothesis.Normalize();
13     Segmentation = Segmentation + Hypothesis;
14   Histogram[TestImage] = 0;
15   for each TestFeature in TestImage
16     TestFeature.Weight(Segmentation);
17     Histogram[TestImage].Add(TestFeature);
```

Figure 4: Pseudocode for Marszalek and Schmid's algorithm. Source: Marszalek and Schmid [2]

3 Spatial Weighting

Spatial weighting is a technique that reduces the influence of background clutter and thus highlights descriptors that are unique to the image. For example, it is possible for a training image for a bike to have both a bike and a dog in the background, whereas we are only truly interested in the bike for training purposes. Using the visual vocabulary from HS-SIFT and LS-SIFT, we can hypothesize about the location of an object given its spatial relations to words in the visual vocabulary.

These hypothesis allow us to weight relevant areas of the image, and the least weighted portions of the image correspond to background clutter and contribute significantly less to the overall image. Marszalek and Schmid's method first calculates the $N = 100$ closest descriptors to each descriptor in the test image, and generates a segmentation mask by repeatedly convolving the mask with Gaussians based on the descriptor locations. (See Figure 4).

We have developed two alternatives to spatial weighting that are less computationally expensive and support multiple-object detection. Both methods are effective at filtering out background descriptors, although both methods make the general assumption that background descriptors are generally weak and do not have a strong correlation with other descriptors in the image. Our tests indicate that CTE is more effective than NNW.

3.1 Nearest Neighbor Weighting

NNW is heavily based on the original method of spatial weighting. However, instead of obtaining information from the training information ground truth, it naïvely assumes that the general shape of an object can be approximated by a sum of Gaussians. We repeat the process of weighting the entire image over the descriptors which fall into the top $N = 25$ centroids.

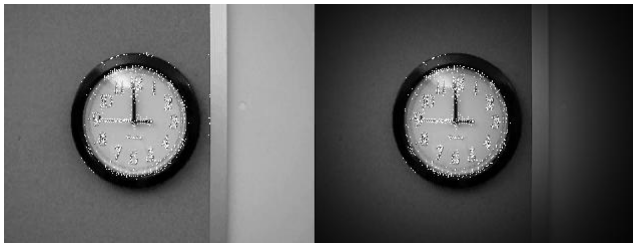

Figure 5: Result of using NNW on the first few frames of the CS221 test video.

Although NNW was effective at eliminating interest points beyond the weighted boundaries, the issue was that it could not eliminate enough points within the interest boundary that were not necessarily spe-

cific to the object. For example, the orientation of the hands, or the shape of the numbers on a clock are not indicative of whether the object is a clock itself. Furthermore, NNW was poor at eliminating interest points on more elongated objects (such as keyboards). We also noticed that although NNW eliminated background noise, the algorithm is not guaranteed to converge and can often alternate between different weighting configurations. In general, NNW did not improve performance overall and we will not discuss it further.

3.2 Correlation Tree Elimination

CTE is our original algorithm that uses the same idea of spatial weighting, but at a very localized level. The idea behind CTE is to construct several correlation trees out of the descriptors in our set, and classify only based on descriptors that are in a correlation tree that exceeds some size threshold T. To construct each correlation tree, we first compute the closest $N = 10$ centroids to each descriptor in our test image. For each descriptor, we then check every other descriptor to see if it matches the top N centroids, and add recursively add it to the tree, performing a depth-first search. We do not insert elements into the tree if they were previously encountered. The result is a partition of all descriptors into several trees of varying lengths. All descriptors that belong in a certain tree of length less than $T = 32$ are eliminated.

```
//single-tree construction
function CTE(descriptor D) {
  D.computeClosestCentroids();
  Foreach other_descriptor in TestSet {
    if (isCloseNeighbor(other_descriptor, D)
        && isNotExpanded(other_descriptor)) {
      CTE(other_descriptor);
    }
  }
}
//main code
Foreach descriptor in TestDescriptors {
  if(isNotExpanded(descriptor)) {
    CTE(descriptor);
  }
}
Foreach descriptor in TestDescriptors {
  Tree t = descriptor.getTree();
  if(t.numElements < THRESHOLD) {
    TestDescriptors.erase(descriptor);
  }
}
```

Figure 6: Pseudocode for CTE algorithm

CTE yielded very positive results for eliminating background noise. Figure 7 shows illustrates the result of running CTE on CS221 test video. It successfully eliminates almost poor keypoints caused by the granular texture of the wall and board.

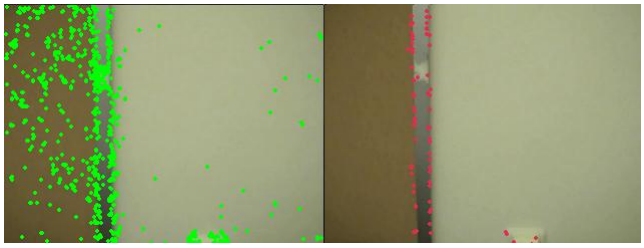

Figure 7: CTE example of keypoint elimination on the CS221 vision project. Before (left) and after (right).

4 Results and Extensions

We ran our test algorithm on both the easy and moderate CS221 testing videos. The easy video has almost no background clutter, and the moderate video has a fair amount of background noise. We used the scoring algorithm as specificed in CS221, which computed the area overlap between pairs of correct objects penalized by the number of true negatives and false positives. The results are shown by the following graphs:

Improved results using the CTE algorithm were mostly from filtering out false positives, as expected. In both cases, the CTE yields significantly higher scores than running on our entire set of HS-SIFT and LS-SIFT descriptors. It should be noted, however, that our algorithm for detection was not particularly accurate for all objects, and our current results are not representative of an improvement over the original method my Marszalek and Schmid. However, CTE was able to serve our purposes by eliminating background keypoints quickly while still having the ability to identify multiple objects in one frame. The reason for this is because CTE keeps all relevant keypoints that belong to some contour of some object. To show that CTE does not in fact decrease the rate of general object detection, we ran both test videos with all labels untagged. The CTE scored an average of 0.8696 on the moderate video, compared with 0.8697 without. On the easy video, CTE scored 0.8809 and 0.8916 without. The insignificant difference between score shows that our improved results from CTE is not due to a lack objects being detected.

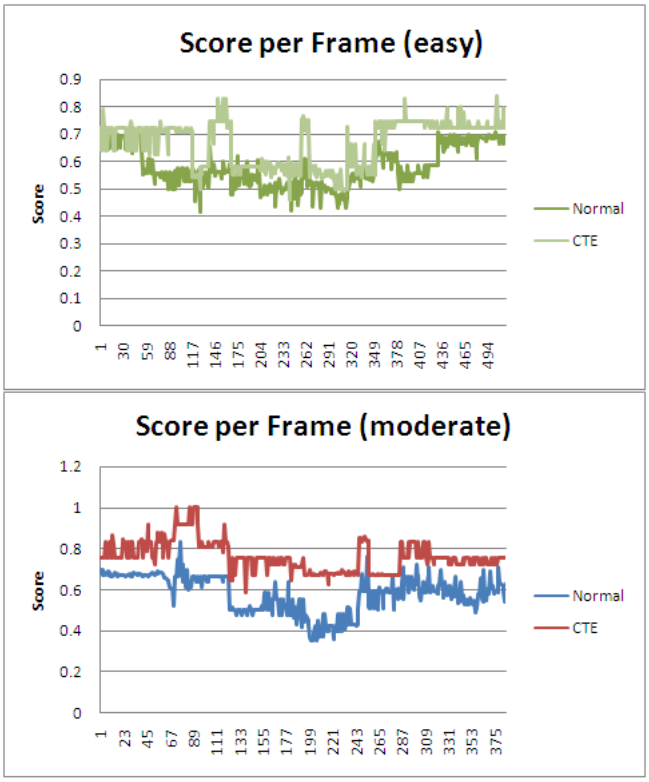

Figure 8: Score per frame for the easy and moderate CS221 test videos. Results shown are run with and without CTE.

4.1 Future extensions and Summary

There are several potential extensions to our CTE algorithm. One potential extension is the addition of CTE to the training process. When training images contain a significant amount of background clutter, it is possible to approximate centroids using descriptors of previously trained images, and use those centroids to eliminate the clutter in future images. We believe that some of the inaccuracy from testing is due to inaccurate descriptors during training, such as logos on mugs, numbers on clocks, and other various noise that may appear in the background. Another possible extension is to use the partitions generated by CTE as features for bag-of-features. Partitions using CTE generate edge curvature like features that may be effective as an addition to our HS and LS-SIFT descriptors.

In this paper, we have shown a new and efficient algorithm for filtering out unwanted background keypoints of an image. We demonstrated that CTE is effective and significantly improved our initial test results in the CS221 test videos. CTE is a viable alternative to spatial weighting, especially in cases where speed is needed, such as classifying a streaming video.

References

[1] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, (submitted), 2003

[2] M. Marszalek, C. Schmid, "Spatial Weighting for Bag-of-Features," *IEEE Computer Society Conference on & Computer Vision and Pattern Recognition*, V2, pp. 2118-2125, 2006

[3] S. Winder, M. Brown, "Learning Local Image Descriptors," *IEEE Computer Society Conference on & Computer Vision and Pattern Recognition*, pp. 1-8, 2007

[4] J. Zhang, M. Marszalek, S. Lazebnik, C. Schmid, "Local features and kernels for classification of texture and object categories: An in-depth study." Technical Report RR-5737, INRIA Rhone-Alpes, 2005