Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2008-033 June 10, 2008

Fast concurrent object classification and localization

Tom Yeh, John J. Lee, and Trevor Darrell

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Fast concurrent object classification and localization

Tom Yeh
MIT EECS & CSAIL
Cambridge, MA, USA
tomyeh@mit.edu

John J. Lee
MIT EECS & CSAIL
Cambridge, MA, USA
jjl@mit.edu

Trevor Darrell
UC Berkeley EECS & ICSI
Berkeley, CA, USA
trevor@eecs.berkeley.edu

Abstract

Object localization and classification are important problems in computer vision. However, in many applications, exhaustive search over all class labels and image locations is computationally prohibitive. While several methods have been proposed to make either classification or localization more efficient, few have dealt with both tasks simultaneously. This paper proposes an efficient method for concurrent object localization and classification based on a data-dependent multi-class branch-and-bound formalism. Existing bag-of-features classification schemes, which can be expressed as weighted combinations of feature counts can be readily adapted to our method. We present experimental results that demonstrate the merit of our algorithm in terms of classification accuracy, localization accuracy, and speed, compared to baseline approaches including exhaustive search, the ISM method, and single-class branch and bound.

1 Introduction

In many object localization and classification applications, exhaustive search over all class labels and/or image windows can be too time-consuming to be practical. To improve localization efficiency, it is desirable to reduce the number of windows that need to be processed. Previous approaches have identified clusters of promising local features and searched over windows around these clusters [3], and/or used local features to vote for object locations and verify the voted locations though back-projection [8]. Recently, branch-and-bound approaches have been revisited in the computer vision literature, and have been shown to greatly speed up object detection [7] using contemporary object category recognition representations, e.g., visual word histograms. These methods all dramatically improve localization speed, but generally require a linear scan of model hypotheses, which can be costly when the number of categories or instances under consideration is large.

To improve classification efficiency with large numbers of categories or instances, tree-based data structures have been proposed to index class labels, such as the measurement-decision tree technique [10], the vocabulary tree technique [9], and the random-forests technique [1]. Given an image feature as input, these tree structures can quickly lookup relevant class labels. After every image feature has been considered, the most probable class label is simply the one that has been looked up the most number of times. Graph-based data structures have been proposed which organize 1-vs-1 SVM classifiers into a directed acyclic graph (DAG-SVM) so that only a small subset of SVMs need to be evaluated [11]. These methods improve classification speed, but assume the localization is irrelevant or already specified.

This paper deals with the problem of concurrent classification and localization, by proposing a multi-class formalism of branch-and-bound and introducing a data-dependent region hypothesis sampling scheme to more efficiently select promising candidate regions. Additionally, we incorporate a geometric verification stage into the method and show how the method can be extended to consider non-rectangular object regions, e.g., those defined by sets of stacked bounding boxes. Existing bag-of-features classification schemes, which can be expressed as weighted combinations of feature counts, can be readily adapted to our method. We present experimental results that demonstrate the relative merit of our algorithm in terms of classification accuracy, localization accuracy, and speed, in comparison with baseline approaches including exhaustive search, the ISM method of [8], and single-class branch and bound.

There has been a large body of work dealing with similar types of concurrent vision tasks. For instance, the problem of concurrent recognition and segmentation has been explored by [14], which uses graph partitioning to recognize patches containing body part, find the best configuration of these detected parts for a given object model, and return a segmentation mask based on this configuration, and by [13], which learns generative probabilistic models for recognition and segmentation capable of dealing with significant with-in class variations. These methods assume the object has been localized somewhere in the middle of an image. Also, the problem of concurrent object detection and segmentation has been studied by [6], which proposes a probabilistic method called OBJCUT that combines bottom-up and top-down cues to detect and segment a single instance of a given object category, such as cows and horses, by [12], which applies OBJCUT to detect and segment multiple objects (i.e., faces), and by [4], which searchers over affine invariant regions in images for matches consistent with a given object modal and uses an efficient algorithm for merging matched regions into segmentation masks. However, these methods either assume the class model is known or require repeated applications over all class models.

2 Concurrent object localization and classification

Given an input image with N local features and a set of M object class models learned previously, the task of concurrent object localization and classification can be cast as a joint optimization problem over a bounding region r and a class label i that maximizes the sum of the features bounded by r, weighted by the i-th class model. We can formally define this problem by expressing the N local features as a set of N triplets $T : \{(x_j, y_j, \vec{v_j})|1 \leq j \leq N\}$, where (x_j, y_j) marks the location of each feature and $\vec{v_j}$ is a m-dimensional vector whose i-th element $\vec{v}(i)$ stores the feature weight assigned by the i-th class model. Then, the objective function can be given as follows:

$$\arg\max_{r,i} \sum_{j \in T(r)} \vec{v_j}(i) \tag{1}$$

where $T(r)$ denotes the indices of the subset of triplets in T spatially bounded by the region r.

We initially consider bounding regions which are traditional rectangles, although below we introduce an extension of the method to consider composite box regions. Rather than uniformly sample image coordinates, we define the space of bounding region hypotheses in a data-dependent manner, using the actual locations of feature points to determine the four edges of a box. Our search space consists of five variables: c, x^-, x^+, y^-, y^+, where c is the class label, x^- and x^+ are the two extreme points of the box along the horizontal direction, and y^- and y^+ are the extreme points along the vertical direction. The time complexity of exhaustive search would be $O(mn^4)$. In the next section, we describe an efficient algorithm to solve this optimization problem based on branch-and-bound, which is also illustrated in Figure 1 and Algorithm 1.

2.1 Multi-class, data-dependent Branch-and-bound

We present a multi-class, data-dependent branch-and-bound method for concurrent object classification and localization. Branch-and-bound is a well-known search technique that is capable of finding the global optimal solution. In computer vision, this technique has bee applied in geometric matching [2], segmentation [5], and localization [7]. Two components critical to the branch-and-bound technique are the mechanism for branching—dividing the search space into different subspaces, and the criterions for bounding—estimating the upper and lower bounds of the optimal value that can be discovered within each subspace.

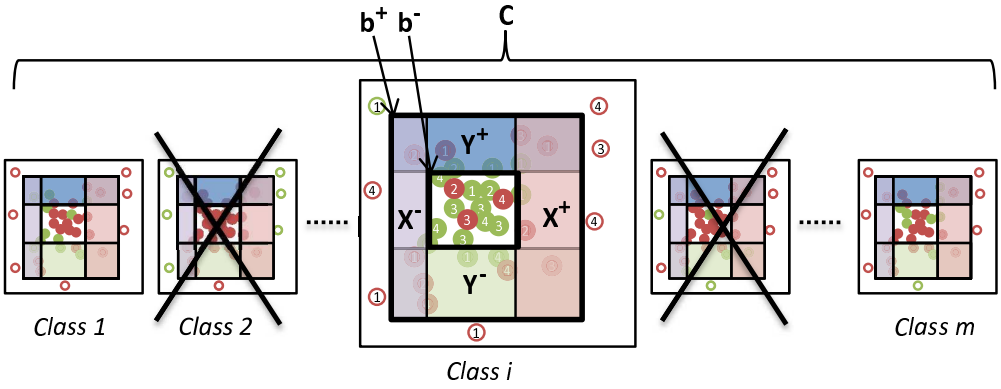

Figure 1: Searching the best bounding box and class model by branch-and-bound. Inputs are m weighted feature representations each weighted by one of m class models. Green and red markers denote positive and negative features w.r.t. a class model. The objective is to find a bound box and a class label that jointly maximize the sum of the weighted features inside the box (i.e., more green and fewer red). X^-, X^+, Y^-, Y^+, are the four candidate sets that specify a set of candidate bounding boxes (any box that can be drawn within the shaded regions). b^+ and b^- are the largest and smallest candidate bounding boxes definable by these four candidate sets. C is the fifth candidate set specifying the candidate class models. In this example, a subset of classes have been discarded from the current hypothesis (e.g., class 2) because their upper-bound estimates are low. To branch, we pick Y^- and split it into halves, because it has the most feature points (data-dependent sampling).

A subspace S can be defined by a set of five candidate sets X^-, X^+, Y^-, Y^+, C for the five variables x^-, x^+, y^-, y^+ , c respectively. A candidate set specifies a set of feature points or labels a variable can still take on. In contrast to [7], we initialize X's and Y's to be the set of all feature points, not the set of all image coordinates; C is the set of all class labels. At each iteration, the *branching* step is to pick a candidate and split it into two halves. Then, the *bounding* step is to estimate the bounds of the resulting new subspaces. These bounds are used to prioritize these subspaces in a queue. The subspace with the highest upper bound will be moved to the top of the queue and will be processed at the next iteration. As the search progresses, the size of each candidate set will gradually decrease. When every candidate set of the subspace at the top of the queue has only one member, a solution is found. The global optimality of this solution is guaranteed because all the subspaces remaining in the queue must have upper bounds below this solution.

To split a subspace S into two smaller subspaces S_1 and S_2, we choose the largest candidate set and split it into two equal-size subsets. For X's, we split the set horizontally, where points in one set are strictly to the left of the points in another set. For Y's, we split the set vertically, where points in one set are strictly above the points in another set. For C, we split the set according to the upper-bound estimates, where class labels in one set possess higher upper-bounds estimates than do those in the other set. After a candidate set is split into halves, V_1 and V_2, we first create two copies of S. Then, from each copy we remove the members in each half to obtain $S_i : \{X_i^-, X_i^+, Y_i^-, Y_i^+, C_i\}, i = 1, 2$. For instance, if Y^- is split, in order to obtain S_1, we remove points in V_1 not only from Y_1^- but also from X_1's as well. A benefit of our data-directed branching scheme when compared to a uniform sampling of the image coordinates is that fewer degenerate or redundant hypotheses will be considered. This efficiency advantage will be empirically demonstrated in Section 2.2.

Given a subspace S and its candidate sets X^-, X^+, Y^-, Y^+ , C, the largest box b^+ discoverable in this subspace must be the one defined by the left-most point in X^-, the right-most point in X^+, the bottom-most point in Y^-, and the top-most point in Y^+. Similarly, the smallest box b^- discoverable must be the right-most point in X^-, the left-most point in X^+, the top-most point in Y^-, and the bottom-most point in Y^+. For each class label $c \in C$, our goal is to find a box between b^+ and b^- that includes as many positive numbers and exclude as many negative points as possible at dimension c. Also, we know that this box cannot contain more positive points than does b^+ and cannot contain fewer negative points than does b^-. Thus, we can calculate the class-specific upper-bound f_c for the

Algorithm 1 Localize and classify the object in an input image I concurrently.

function LOCALIZEANDCLASSIFY(I)
  $X_0^-, X_0^+, Y_0^-, Y_0^+ \leftarrow$ all points in I
  Sort X's and Y's along x and y axis
  $C_0 \leftarrow$ all class labels
  $\mathcal{S}_0 \leftarrow \{X_0^-, X_0^+, Y_0^-, Y_0^+, C_0\}$
  $P \leftarrow$ empty priority queue
  Insert $\mathcal{S}_0$ into P
  while $\mathcal{S} \leftarrow$ POP(P) **do**
    if $\forall V \in S, |V| = 1$ **then**
      return $\mathcal{S}$
    end if
    $(\mathcal{S}_1, \mathcal{S}_2) \leftarrow$ SPLIT(S)
    Insert $\mathcal{S}_1$ to P with key $f_c(\mathcal{S}_1)$
    Insert $\mathcal{S}_2$ to P with key $f_c(\mathcal{S}_2)$
  end while
end function

function SPLIT($\mathcal{S} : \{X^-, X^+, Y^-, Y^+, C\}$)
  Let V be the largest candidate set
  if $V = C$ **then**
    Let $\mathcal{S}_c$ denote $\{X^-, X^+, Y^-, Y^+, c\}$
    Sort all $c \in V$ by $f_c(\mathcal{S}_c)$
    $V_1 \leftarrow$ top half of all C_is
    $V_2 \leftarrow$ bottom half of all C_is
  else
    $V_1 \leftarrow$ first half of V
    $V_2 \leftarrow$ second half of V
  end if
  $\mathcal{S}_1 \leftarrow$ copy of $\mathcal{S}$ with V_1 removed
  $\mathcal{S}_2 \leftarrow$ copy of $\mathcal{S}$ with V_2 removed
  return $(\mathcal{S}_1, \mathcal{S}_2)$
end function

subspace S as follows:

$$f_c(S) = \sum_{j \in T(b^+)} h^+(\vec{v_j}(c)) + \sum_{j \in T(b^-)} h^-(\vec{v_j}(c)) \tag{2}$$

where $h^+(x)$ and $h^-(x)$ are functions that evaluate to x if x is positive or negative respectively, and 0 if otherwise. Similarly, the lower-bound is attained by including as many negative numbers and excluding as many positive numbers as possible. The class-specific lower-bound g_c can be obtained by

$$g_c(S) = \sum_{j \in T(b^-)} h^+(\vec{v_j}(c)) + \sum_{j \in T(b^+)} h^-(\vec{v_j}(c)). \tag{3}$$

Finally, we compute the overall upper-bound $f(S)$ and lower-bound $g(S)$ for the subspace S by taking the maximum and minimum of the individual f_c's and g_c's for the classes in C respectively. The lower bound measurements are useful for pruning the search space, discarding those subspaces whose upper-bounds are lower than the lower-bounds of known subspaces.

To deal with multi-instance localization and classification, we optionally let our search algorithm continue to run even after it has discovered the optimal solution. To avoid redundant detections we remove features used in previous detections. Finally, we also add an optional post verification step to check the spatial consistency between features enclosed by the bounding region, simply by applying the ISM method using these features. If these features are consistent, the hypothesis made by ISM will also be consistent with the bounding box; we would expect the localization predicted by ISM would be within the bounding box. If not, we can reject the current bounding box hypothesis and try the second best bounding box, until it passes the ISM verification step.

Our joint optimization problem for concurrent object localization and classification operates on a set of triplets $(x_j, y_j, \vec{v_j})$. Many common object category classification schemes, including voting and SVMs with kernels defined on visual word histograms, can be straightforwardly expressed in this weighed-feature representation, as we show in the following paragraphs.

Voting-based classification schemes simply allow each feature to cast votes on a set of class labels, ideally with an efficient indexing scheme to retrieve relevant labels for a given feature. E.g., with a vocabulary tree each leaf node in the tree corresponds to a visual word and stores the id's of the training images that contains such word and the counts of this word in each training image. To obtain the desired representation, we add a triplet $(x_j, y_j, \vec{v_j})$ for each feature f_j. (x_j, y_j) are the coordinates of the feature location. To obtain $\vec{v_j}$, let $W(j)$ be the set of id's of the training examples indexed by the word looked up by f_j. Let k be the id of a training image, l_k its class label, and c_k

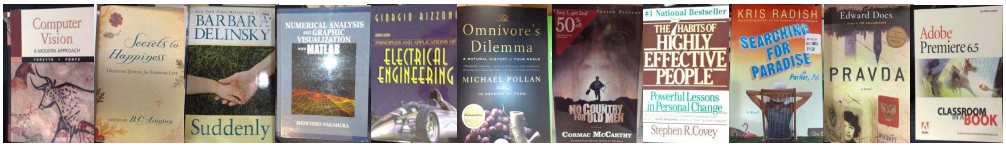

Figure 2: Examples of training images.

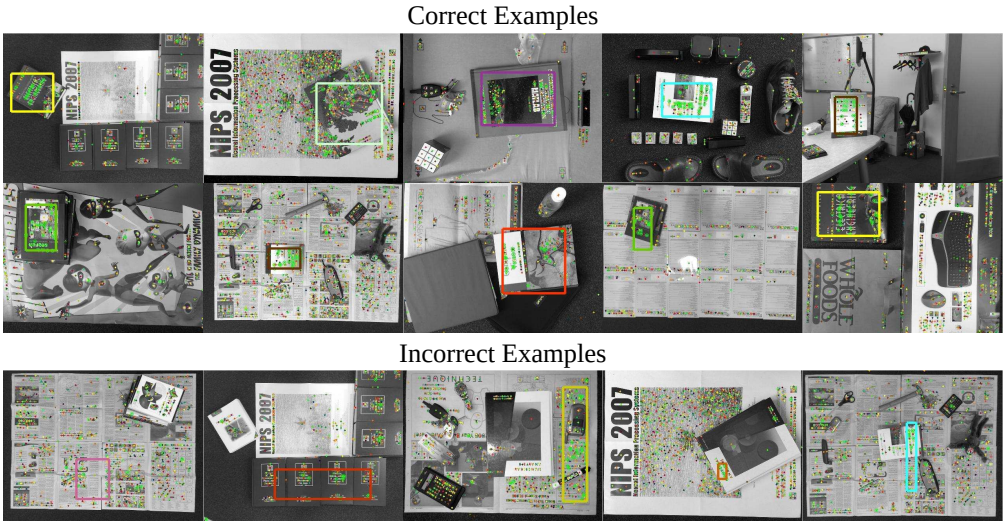

Figure 3: Examples of localization and classification results of our method. The top rows show correct results. The first three incorrect examples are cases when classification failed; our algorithm mistaken background regions as some books. The last two incorrect examples are cases when localization failed (the object center is outside the bounding box) despite correct classification. These mistakes are mainly due to the failure to extract salient features from some regions of a book, but can be remedied by geometric post-verification.

the count of its votes. Then, the i-th element of $\vec{v_j}$ can be calculated as

$$\vec{v_j}(i) = \sum_{k \in W(j)} h(l_k = i) c_k \tag{4}$$

where $h(x)$ evaluates to 1 if x is true and 0 if otherwise.

Typical SVM-based classification schemes use bag-of-visual-word representations to compute pairwise similarity scores between training images as kernels to learn support vectors, and between support vectors and the test image to calculate margins. Since the bag-of-feature similarity score is simply the aggregate of the partial similarity scores contributed by individual image features, the margin for the i-th class is the weighted sum of similarity scores to a set of training examples plus an offset β_i; this can be computed by aggregating the weighted partial similarity scores of individual image features. For the k-th training image, let $\vec{\alpha_k}$ be a vector containing its support vector weights where i-th element denotes its weight in the i-th SVM. Then, the partial score contributed by a local image feature f_j to the overall score of the i-th SVM can be calculated as

$$\vec{v_j}(i) = \sum_{k \in W(j)} h(l_k = i) c_k \vec{\alpha_k}(i) \tag{5}$$

Also, class-specific bound estimates must be adjusted by the offset of each SVM as follows:

$$f'_c = f_c + \beta_c \quad , \quad g'_c = g_c + \beta_c \quad . \tag{6}$$

2.2 Experiments

The objective of our first experiment is to evaluate the effectiveness of our algorithm in localization and classification tasks in relation to baseline methods. We choose a typical large-category recogni-

Method	BOF	ISM	ISM$^+$	EES	EES$^+$	CCL$^-$	**CCL**
Classification accuracy	59.2%	86.1%	83.8%	85.3%	85.3%	85.3%	85.3%
Localization accuracy	N/A	39.2%	82.3%	80.0%	80.0%	80.0%	80.0%
Time per image (sec)	0.12	61.32	524.28	606.65	182.53	2.27	1.14

Table 1: Empirical comparisons of our method to six alternatives (Section 2.2).

tion task, recognizing book covers in a set of input images. Our training set was a database of book covers from an online retailer (e.g., Figure 2). For the test set, we collected images with multiple books placed in varying orientations in a cluttered environment. Figure 3 shows some examples of test images with bounding boxes and labels (indicated by different colors) found by our algorithm.

On a task with 120 books in the index and 130 test images, we compared our concurrent classification and localization method (CCL) to six alternatives: (1) simple bag-of-feature voting (BoF), which performed no localization and used all the image features to vote on class labels, (2) the implicit shape model (ISM) [8], which provided localization by letting image features to vote on scales and locations using class-specific codebooks, (3) an improved, robust-to-rotation version of ISM (ISM+), which augmented the training set with rotated versions of each training image in 45 degree increments, (4) efficient sub-window search (EES) [7], which also uses branch-and-bound to find optimal bounding boxes but does not split thee search space in a data-dependent manner, (5) EES with data-dependent sampling (EES$^+$), and (5) a reduced version of our method without data dependent hypothesis sampling (CCL$^-$), which aimed to demonstrate the benefit of this sampling technique. Note that the first five alternatives all require repeated applications over all the class models in order to find the best class label [1].

Table 1 reports the result of this comparative experiment along three criteria: classification accuracy, localization accuracy, and running time. All the methods involved were implemented in C++ and executed on a machine with a 3.2GHz CPU. In terms of speed, BoF achieved the fastest performance at 0.12 sec. per image; however, it had the lowest classification performance at 59.2% and provided no localization. The original ISM method achieved the best classification accuracy, but it only managed to correctly localize the book in 39.2% of the test images, mainly when the book in a test image happens to be in an orientation close to that of the training image of the same book. By using 8 rotated images of books as training images, ISM+ improved the localization accuracy to 82.3%; yet, it took close to 10 minutes to process each test image because of the need to check every codebook. The running time of M applications of EES also took about 10 minutes. Using data-dependent sampling (EES$^+$), this time was reduced to about 3 minutes. In comparison, our method was able to provide localization and classification with an accuracy comparable to that of the two variants of ISM, while recording a computation time of a little more than a second per test image, at least two-orders of magnitude faster than ISM+ and EES. Moreover, the benefit of data-dependent sampling has been verified; when this feature was turned off (i.e., CCL$^-$), the running time more than doubled (1.14 versus 2.27 sec.).

Figure 4 demonstrates the scalability of our method. It shows the classification accuracy and per-image running time of our method using the same set of 130 test images but against a database of up to 1000 books. Our method still achieved a classification accuracy above 70% at 1000 books with running time growing linearly to the number of classes. Figure 5 shows some examples of multi-instance localization and classification results demonstrating that our method can handle multiple objects in the same image. Figure 6 shows examples of adding ISM verification after a bounding box and class label is discovered by our algorithm.

3 Search with composite box regions

While bounding boxes have been widely used for localization because of computational simplicity, there are situations when these boxes are too coarse for localization. For example, when the target is a long, thin object positioned at a 45 degree angle, any rectangular bounding box is deemed to

[1] In all experiments, SIFT were the features descriptors. In ISM-related experiments, the Harris detector was used on images resized to 640x480 and the threshold for agglomerative clustering was -40000. In all other experiments, the MSER detector was used on images resized to 1024x768.

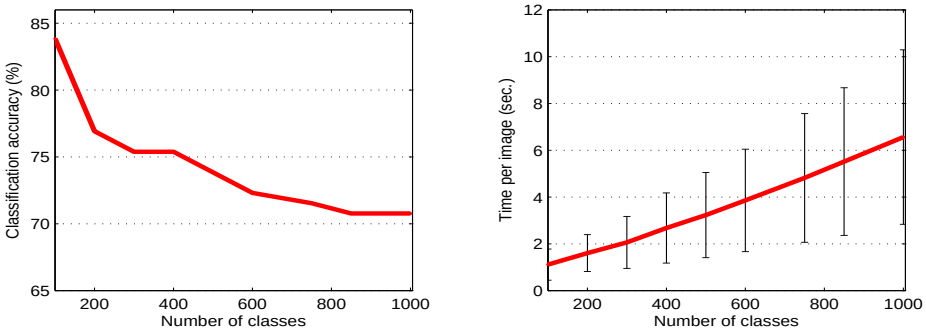

Figure 4: Scalability of our multi-class branch-and-bound algorithm. As the number of classes increases (i.e., up to 1000 books), the classification accuracy slowly declines but still stays above 70%. The running time grows linearly instead of quadratically to the number of classes.

.

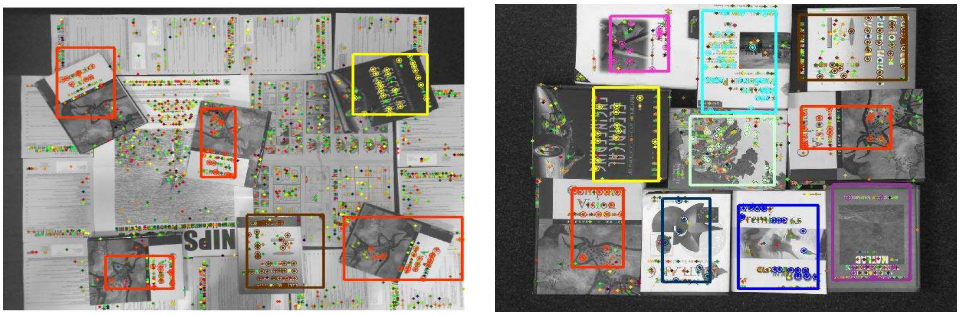

Figure 5: Examples of concurrently localizing and classifying multiple copies of different books.

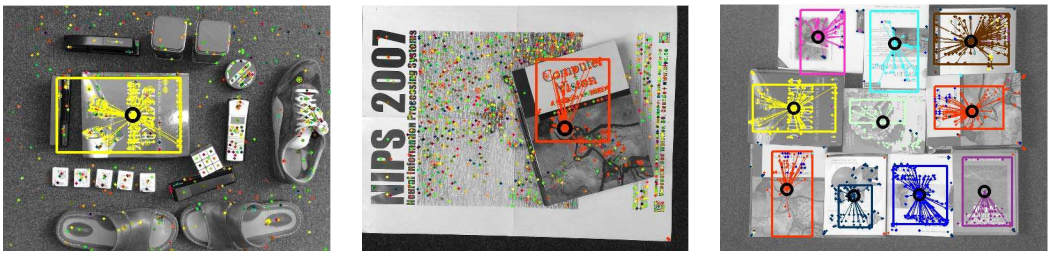

Figure 6: Examples of using implicit shape model (ISM) to verify the spatial consistency of the feature points in a bounding box. Our method quickly find the best bounding box and the class label. Then, only the ISM corresponding to the label is evaluated to predict the object center (black circle). Here, all centers are inside the bounding boxes, thus spatially consistent.

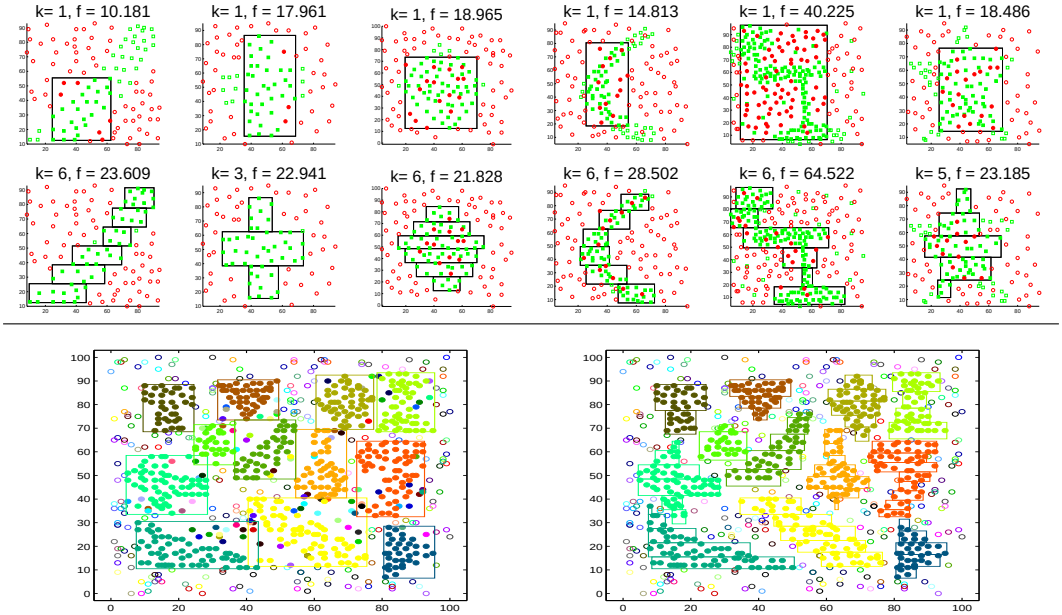

Figure 7: Composite versus single bounding boxes (Section 3). In single-class examples (top), green and red markers are positive and negative features, which are filled if bounded. Single boxes (1st row) cannot maximize the sum of the weighted features in them, whereas composite boxes can capture more positive features and discover higher scores (2nd row). In multi-class examples (bottom), each class has a unique color and bounded features are filled. Similarly, composite boxes (right) can find higher scores by avoiding more negative features than single boxes (left).

include many unwanted background features. In the other extreme are methods that do not assume any bounding shape, such as [8], that are immune to this problem; however, such methods can be computationally costly.

An attractive compromise between the two extremes that can be efficiently implemented with our method is to use composite bounding regions comprised of a series of k bounding boxes in a vertical stack. Instead of using two extreme points to mark the left and right sides of a single bounding box, we use k pairs of left-right feature points to define the two sides of each of the k bounding boxes. Namely, we split X^- and X^+ into $\{X_1^- \dots X_k^-\}$ and $\{X_1^+ \dots X_k^+\}$ respectively. Thus, together with Y^-, Y^+, C, the total number of candidate sets in the new joint optimization problem is $2k+3$. Moreover, k can be determined automatically by incrementally increasing it by one until the optimal score improves no more. Figure 7 show examples of complex distribution of features that can be bounded by composite boxes to find optimal scores but not by single boxes.

4 Conclusion

This paper described an efficient method for concurrent object localization and classification based on a data-dependent multi-class branch-and-bound formalism. In our experiments, we compared our approach to existing methods, and demonstrated the superior performance of our method in terms of overall classification and localization accuracy, and speed.

References

[1] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proc. of International Conference on Computer Vision*, pages 1–8, 2007.

[2] T. M. Breuel. A comparison of search strategies for geometric branch and bound algorithms. In *Proc. of European Conference on Comptuer Vision*, pages 837–850, 2002.

[3] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[4] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67, 2006.

[5] Y. Gat. A branch-and-bound technique for nano-structure image segmentation. In *Proc. of Conference on Computer Vision and Pattern Recognition Workshop*, 2003.

[6] P. M. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pages 18–25, 2005.

[7] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: object localization by efficient subwindow search. In *Proc. of CVPR*, 2008.

[8] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289, May 2008.

[9] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.

[10] S. Obdrzálek and J. Matas. Sub-linear indexing for large scale object recognition. In *Proc. of British Machine Vision Conference*, 2005.

[11] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Proc. of Advances in Neural Information Processing Systems*, volume 12, 2000.

[12] J. Rihan, P. Kohli, and P. Torr. Objcut for face detection. In *Computer Vision, Graphics and Image Processing*, pages 576–584, 2006.

[13] J. Winn and N. Jojic. Locus: learning object classes with unsupervised segmentation. In *Proc. of International Conference on Computer Vision*, volume 1, pages 756–763, 2005.

[14] S. Yu, R. Gross, and J. Shi. Concurrent object recognition and segmentation with graph partitioning. In *Proc. of Advances in Neural Information Processing Systems*, 2002.