Fast Concurrent Object Localization and Recognition

Tom Yeh
MIT EECS, CSAIL
Cambridge, M.A., USA
tomyeh@mit.edu

John J. Lee
Google Inc.
New York, N.Y., USA
jjl@jjl.nu

Trevor Darrell
U.C. Berkeley EECS, ICSI
Berkeley, C.A., USA
trevor@eecs.berkeley.edu

Abstract

Object localization and recognition are important problems in computer vision. However, in many applications, exhaustive search over all object models and image locations is computationally prohibitive. While several methods have been proposed to make either recognition or localization more efficient, few have dealt with both tasks simultaneously. This paper proposes an efficient method for concurrent object localization and recognition based on a data-dependent multi-class branch-and-bound formalism. Existing bag-of-features recognition techniques 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 recognition accuracy, localization accuracy, and speed, compared to baseline approaches including exhaustive search, implicit-shape model (ISM), and efficient subwindow search (ESS). Moreover, we develop two extensions to consider non-rectangular bounding regions—composite boxes and polygons—and demonstrate their ability to achieve higher recognition scores compared to traditional rectangular bounding boxes.

1. Introduction

In many object localization and recognition applications, exhaustive search over all object models 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 [4], and/or used local features to vote for object locations and verify the voted locations though back-projection [12]. Recently, branch-and-bound approaches have been revisited in the computer vision literature, and have been shown to greatly speed up object detection [10] using contemporary object category recognition representations, e.g., visual word histograms. These methods all significantly improve localization speed, but for problems involving many possible object models, may rely on a linear scan of all the models, which can be costly when the number of models is large.

To improve efficiency for large-scale recognition problems, tree-based data structures have been proposed to index a large number of object models, such as the measurement-decision tree technique [14], the vocabulary tree technique [13], and the random-forests technique [2]. These data structures typically assume an image representation based on histograms of local features. Given a single image feature, these data structures can quickly lookup a set of relevant object models. After all image features are considered, the most probable object model can often be determined as the one that has been looked up most frequently. 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 [15]. These methods improve recognition speed significantly, but assume localization is either already given or irrelevant.

There have been many efforts in the literature on concurrent recognition and segmentation; for instance, [18] uses graph partitioning to recognize patches containing body parts, find the best configuration of these detected parts for a given object model, and return a segmentation mask based on this configuration, and [17], 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. The topic concurrent detection and segmentation has be explored by [9], 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 [16], which applies OBJCUT to detect and segment multiple objects (i.e., faces), and by [6], which searchers over affine invariant regions in images for matches consistent with a given object model 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.

In this paper, we deal with the problem of concurrent object localization and recognition. In Section 2.1 we describe a multi-class formalism of branch-and-bound to simultaneously find optimal bounding regions and search object models. We introduce a data-dependent region hypothesis sampling scheme to more efficiently select promising candidate regions. Existing bag-of-features recognition schemes such as the Vocabulary Tree [13] and the Spatial Pyramid Match Kernel [11], which can be expressed as weighted combinations of feature counts, can be readily adapted to our method. Additionally, in Section 2.2 we describe methods to search for multiple objects and to incorporate geometric verification. In Section 2.3 we present experimental results to demonstrate the relative merit of our algorithm in terms of recognition accuracy, localization accuracy, and speed, in comparison with six baseline approaches such as exhaustive search, the implicit shape model (ISM) [12], and the efficient subwindow search (ESS) [10]. Finally, we extend our method to non-rectangular bounding regions in two ways—composite boxes (Section 3.1) and polygons (Section 3.2)—and present examples demonstrating their ability to find higher recognition scores compared to traditional bounding boxes for a variety of object types.

2. Concurrent localization and recognition

In this section we present an algorithm for fast concurrent localization and recognition (CLR). Given N local features extracted from an input image and a set of M possible object models, the objective of concurrent localization and recognition is to simultaneously find a bounding region r in the image and an object model i that jointly maximizes a prediction score f_i for the i-th object model (recognition) based on only the features inside r (localization). This prediction score can be either the output of an SVM classifier (e.g., Pyramid Match Kernel [8]) for classification problems or the output of a tree-based index (e.g., Vocabulary Tree [13]) for instance recognition problems. In both types of problems, the value of f can be derived by accumulating the partial scores contributed by individual input features. Thus, we can compute f incrementally as features are added to or subtracted from a candidate bounding region during the optimization process, which is the key to the efficiency of our algorithm.

Formally, we express the input (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 an M-dimensional vector whose i-th element $\vec{v}(i)$ stores the partial score this feature would contribute to the total prediction score f_i of the i-th object model. Then, the optimization objective 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.

To compute $\vec{v_j}$ for object instance retrieval problems, let us consider the typical process of evaluating a tree-based index such as a Vocabulary Tree [13]. Given an input feature f_j, we can quickly look up a visual word and retrieve a set of object images indexed by this word. The object model associated with each object image receives some partial score based on the corresponding word counts. After all the features are considered, we tally all the partial scores received by each object model and determine the most probable object model. Thus, to facilitate this process in the formalism above, let $W(j)$ be the set of *id*'s of the object images indexed by the word looked up by f_j. Let k be the *id* of an image, l_k its object model label, and c_k the word count. Then, the i-th element of $\vec{v_j}$ can be calculated as

$$\vec{v_j}(i) = \sum_{k \in W(j)} h(l_k = i) c_k \tag{2}$$

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

Similarly, to compute $\vec{v_j}$ for object classification problems, let us consider a typical classification scheme based pm linear SVMs such as the Pyramid Match Kernel [8]. Suppose an SVM classifier has been previously trained for the object class i. To evaluate the margin of this SVM given an input image, we need to first compare this image to each training image that is the support vector of this SVM and obtain a set of similarity scores. Then we compute the sum of these scores weighted by support vector weights and add the corresponding SVM offset β_i. Since an image similarity score is essentially the aggregate of the partial similarity scores contributed by individual image features, we can directly express the margin as a weighted sum of these partial similarity scores. Thus, to apply the desired formalism, let $\vec{\alpha_k}$ be a vector containing the support vector weights of the k-th training image, where the 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 (margin) of the i-th SVM can be calculated as

$$\vec{v_j}(i) = \sum_{k \in W(j)} h(l_k = i) c_k \vec{\alpha_k}(i). \tag{3}$$

2.1. Multi-class data-dependent branch-and-bound

To solve the problem stated above, we develop a multi-class, data-dependent branch-and-bound method, which is also illustrated in Figure 1 and Algorithm 1. Branch-and-bound is a well-known search technique that is capable of

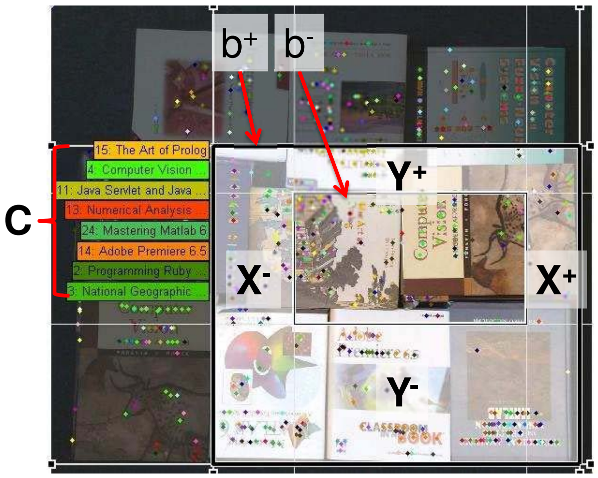

Figure 1. **Searching for the best bounding box and object label in a concurrent localization and recognition task** (Section 2.1). X^-, X^+, Y^-, and Y^+, are the four candidate sets that specify a set of candidate bounding boxes (any rectangle that can be drawn within the highlighted regions). b^+ and b^- are the largest and smallest candidate bounding boxes respectively definable by these four candidate sets. C is the candidate set specifying the candidate object models. In this example, only a subset of models have remained (colored book titles). To branch, we split Y^- into halves because it is the largest set (data-dependent sampling).

finding the global optimal solution. In computer vision, this technique has been applied in geometric matching [3], segmentation [7], localization [10], structural SVM learning [1], and in this paper is applied to concurrent localization and recognition. Two components critical to the branch-and-bound technique are the mechanism for branching (dividing the search space into disjoint subspaces), and the criterions for bounding (estimating the upper and lower bounds of the optimal value that can be discovered within each subspace).

We initially consider bounding regions which are traditional rectangles similar to [10], although later in Section 3 we introduce two extensions that consider more flexible, non-rectangular regions. Also, rather than uniformly sampled 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 bounding box. Thus, our search space consists of five variables: c, x^-, x^+, y^-, y^+, where c is the object model 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 $O(mn^4)$.

Since the search space consists of five variables: x^-, x^+, y^-, y^+ , and c, we define a subspace S by a set of five candidate sets X^-, X^+, Y^-, Y^+, and C corresponding to each variable. A candidate set specifies a set of feature points

Algorithm 1 Localize and recognize an object in an input image I concurrently (Section 2.1).

```
function LocalizeAndRecognize($I$)
    $X_0^-, X_0^+, Y_0^-, Y_0^+ \leftarrow$ all points in $I$
    Sort $X$'s and $Y$'s along $x$ and $y$ axis
    $C_0 \leftarrow$ all object model 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 $\{X^-, X^+, Y^-, Y^+, c\}$ w
        wSort all $c \in V$ by $p_c(\mathcal{S}_c)$
        $V_1 \leftarrow$ top half of all $C_i$s
        $V_2 \leftarrow$ bottom half of all $C_i$s
    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
```

or labels a variable can take on. We initialize X's and Y's to be the set of all feature points (not the set of all image coordinates as in [10]); C is the set of all possible labels. At each iteration, the *branching* operation picks a candidate set and splits it into halves, resulting in two new subspaces. Then, the *bounding* operation estimates the bounds of the new subspaces. These bounds are used to prioritize the subspaces stored 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. When it is no longer possible to split the top subspace (every candidate set 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 that of this solution.

Here we give the details of the branching operation. 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 equalsize 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 it according to the upper-bound estimates, where object models 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-dependent 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 2.3.

Next we provide the details of the bounding operation. First we need to determine the largest and smallest bounding region definable by a given subspace S and its the candidate sets X^-, X^+, Y^-, Y^+ , and C. The largest rectangular box b^+ discoverable 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 one defined by 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 object model label $c \in C$, our goal is to find a box between b^+ and b^- that includes as many points with positive values ($\vec{v_j}(c) > 0$) and exclude as many points with negative values ($\vec{v_j}(c) < 0$) as possible at dimension c. Also, we know that this box cannot contain more positive values than does b^+ and cannot contain fewer negative values than does b^-. Thus, we can calculate the upper-bound p_c of the subspace S for the object model c as follows:

$$p_c(S) = \sum_{j \in T(b^+)} h^+(\vec{v_j}(c)) + \sum_{j \in T(b^-)} h^-(\vec{v_j}(c)) \qquad (4)$$

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 values and excluding as many positive values as possible, Thus, the object-specific lower-bound q_c can be obtained by

$$q_c(S) = \sum_{j \in T(b^-)} h^+(\vec{v_j}(c)) + \sum_{j \in T(b^+)} h^-(\vec{v_j}(c)). \qquad (5)$$

Finally, we compute the overall upper-bound $p(S)$ and lower-bound $q(S)$ for the subspace S by taking the maximum and minimum of the individual p_c's and q_c's for the object candidates 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. Note that for SVM classification problems, we need to adjust the bound estimates by the offset values of respective SVMs as follows:

$$p'_c = p_c + \beta_c \quad , \quad q'_c = q_c + \beta_c \quad . \qquad (6)$$

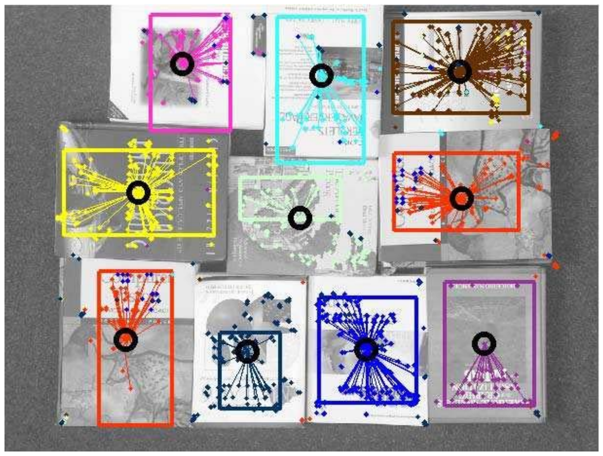
Figure 2. **Concurrent localization and recognition of multiple objects with geometric post-verification** (Section 2.2). Our method quickly find the best bounding box and label. Then, only the geometric model corresponding to the label is evaluated to predict the object center (black circle). Here, all centers are inside the bounding boxes, thus they are spatially consistent.

2.2. Multi-instance CLR and geometric verification

To deal with multi-instance localization and recognition, we optionally let our search algorithm continue to run even after it has discovered an optimal solution. To avoid redundant detections, we remove features used in previous detections. We also add an optional post-verification step to check the spatial consistency between features enclosed by an optimal bounding region, by applying the implicit shape model (ISM) [12] of the identified object using these features. If these features are consistent, the hypothesis made by our algorithm and ISM should be consistent; we would expect the localization predicted by ISM would be within the bounding box found by our method. If not, we can reject the current hypothesis and try the second best bounding box, until it passes the ISM verification step. Figure 2 shows an example of concurrent multi-instance localization and recognition with ISM post-verification.

2.3. Experiments

To properly evaluate the effectiveness and efficiency of our method in solving concurrent localization and recognition problems, we need a suitable dataset not only with a large number of object models (so exhaustive search over all object models would be too costly) but also with significant cluttered background (so localization matters). Thus, we designed a typical large-scale concurrent localization

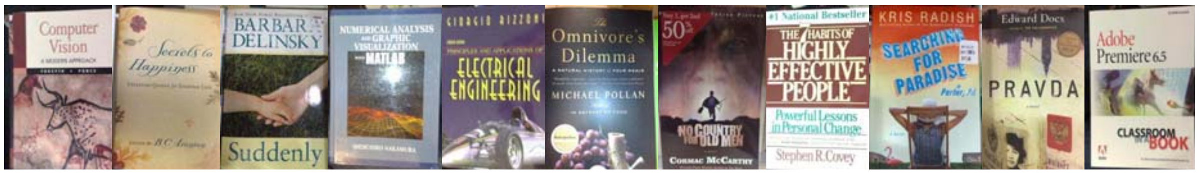

Figure 3. **Examples of the training images used in our experiment** (Section 2.3).

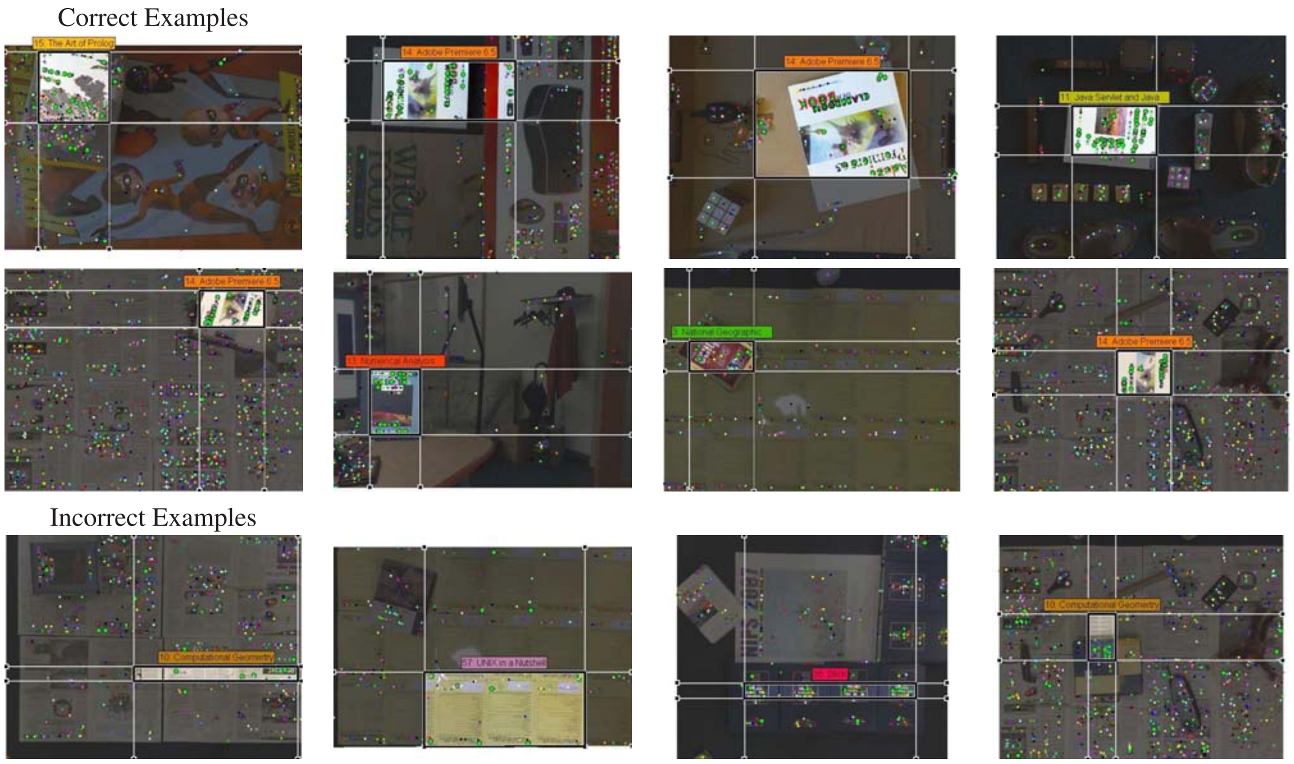

Figure 4. **Examples of concurrent localization and recognition results in our experiment** (Section 2.3). The top two rows are correct examples where our algorithm not only finds an optimal bounding box covering most of the target object but also identifies the correct label (name of the book). The bottom row are incorrect examples. The first three are cases when recognition fail; our algorithm mistakenly labels background regions as some books. The last example is a typical case when localization fails (the object center is outside the bounding box) despite correct recognition. This mistake is due to the inability to extract enough salient features that cover the entire object, but can be overcome by geometric post-verification to infer the correct object center.

and recognition task—recognizing books situated in cluttered environments, and constructed a dataset accordingly. Our training set was images of 120 books downloaded from an online retailer. Figure 3 shows a small sample of the training images. For the test set, we placed physical copies of these books in varying orientations at varying scales in cluttered environments and took pictures of them using a camera phone. Figure 4 shows some examples of the test images with optimal bounding boxes and labels (indicated by different colors) discovered by our algorithm.

Based on this dataset, we conducted an experiment to evaluate the effectiveness and efficiency of our concurrent localization and recognition (CLR) method in relation to six baseline methods on a task with 120 books in the index and 130 test images. These six baseline methods were: (1) simple bag-of-feature voting (BoF), which performed no localization and used all the image features to vote on object models, (2) the implicit shape model (ISM) [12], 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 subwindow search (ESS) [10], which also uses branch-and-bound to find optimal bounding boxes but does not split the search space in a data-dependent manner, (5) ESS with data-dependent sampling (ESS$^+$), and (6) a reduced version of our method without data-dependent sampling (CLR$^-$), which aimed to demonstrate the benefit of this sampling technique. Note that the first five alternatives all require repeated applications over all the object models in order to find the best object label.

Table 1 reports the result of this comparative experiment along three criteria: recognition accuracy, localization accuracy, and running time. All the methods involved were implemented in C++, optimized to the best of our ability, and executed on a machine with a 3.2GHz CPU. BoF achieved the fastest performance at 0.12 sec. per image; however, it had the lowest recognition accuracy at 59.2% and provided no localization. The original ISM method achieved the best recognition 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 happened 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 one of the 120 codebooks. The running time of 120 applications of ESS also took about 10 minutes. Using data-dependent sampling (ESS$^+$), this time was reduced to about 3 minutes. In comparison, our method was able to provide localization and recognition 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 ESS. Moreover, the benefit of data-dependent sampling has been verified; when this feature was turned off (i.e., CLR$^-$), the running time almost doubled (1.14 versus 2.27 sec.).

We also performed an experiment with the PASCAL Visual Object Classes (VOC) dataset [5] with 20 categories, using the same setup as [10] to compute bag-of-word representations over densely sampled features and train linear SVMs. We measured the computation time of our algorithm versus that of ESS in finding the optimal bounding box and the best class label in a test image. With the same empirical setup, our algorithm was to reproduce the result as [10] within margin of errors, but much faster Averaging over 1000 randomly sampled test images, our algorithm took 2.58 seconds (std = 1.07), whereas 20 runs of ESS over all classes took a total of 52.84 seconds (std = 11.51). The speedup is attributed only to the multiclass branch-and-bound scheme but not to the data-dependent sampling scheme since the features were densely sampled.

3. Beyond rectangles

While rectangular bounding boxes have been widely used for localization because of computational simplicity, sometimes they may be too coarse. For example, when an object is long and positioned at an odd angle or has a concave shape, rectangular bounding boxes may inadvertently include excessive background features due to its rigidity and result in low recognition scores. In this section, we present two alternatives to rectangular bonding boxes—composite boxes and polygons—that can be employed with the branch-and-bound optimization technique above.

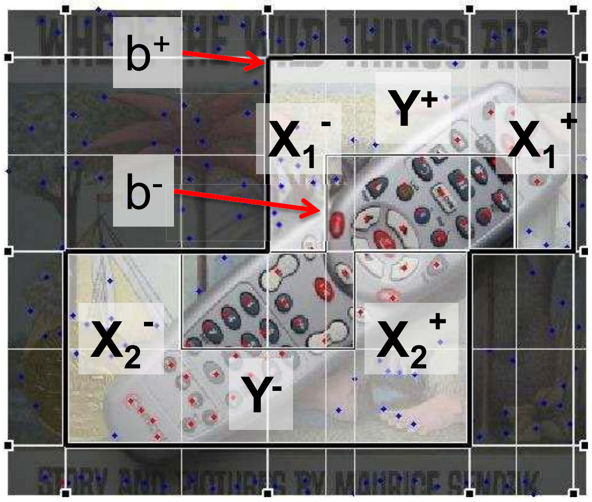

Figure 5. **Finding the best composite box** (Section 3.1).

3.1. Composite bounding boxes

The first alternative to a bounding box is a composite bounding box. A composite bounding box is comprised of a series of k bounding boxes in a vertical stack and can offer more flexibility than a rectangular bounding box. Figure 5 illustrates a typical state of branch-and-bound search with $k = 2$. 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 score improves no more.

We created six shape classes and randomly generated foreground, background, and occlusion features. The recognition score f is the sum of the features within the bounding area. Figure 6 shows that composite bounding boxes achieve higher scores, since their geometric flexibility allows them to include many foreground features without inadvertently including many background features. In contrast, single bounding boxes often can not avoid the inclusion of background features due to their rigidity.

3.2. Bounding polygons

The second alternative to a rectangular bounding box is a bounding polygon. As illustrated in Figure 7, we can define a polygon by a set of straight lines each of which intersect the image boundaries at two locations u and v. These lines may intersect between themselves somewhere

Method	BOF	ISM	ISM$^+$	ESS	ESS$^+$	CLR$^-$	**CLR**
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 2.3).

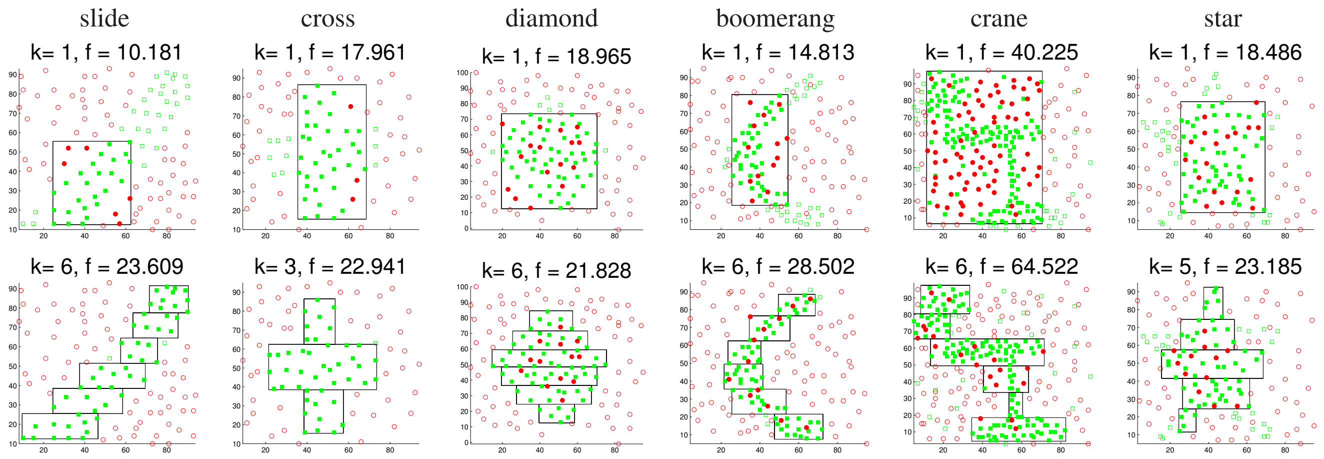

Figure 6. **Single (top) versus composite (bottom) bounding boxes** (Section 3.1). Green and red markers are positive and negative features, which are solid if bounded. Single bounding boxes achieve lower recognition scores f because they can not include all the positive features (solid green markers) without inadvertently including many negative features (solid red markers).

inside the image and these intersections form the vertices of a bounding polygon. We can parameterize u or v by t between 0 and 1 in terms of its distance from the lower-left image corner along the image boundaries clockwise. We set t to be 0, 0.25, 0.5, 0.75 for the lower-left, upper-left, upper-right, and lower-left corners respectively, and interpolate t for other locations along the boundaries between these corners. Let $U_i = [u_i^-, u_i^+]$ and $V_i = [v_i^-, v_i^+]$ denote the intervals in which we will search for the best u_i and v_i that defines the i-th line. Also, we order all the lines clockwise. This way, the smallest bounding region b^- will always be the polygon defined by the lines pairs u^-'s and v^+'s, whereas the largest bounding region b^+ will be defined by the lines based on pairs of u^+'s and v^-'s. This formulation not only allows us to define both convex and concave polygons but also lends itself to branch-and-bound search. To apply branch-and-bound, we initialize U's and V's to $[0, 1]$. For a k-sided polygon, there are $2k$ variables to optimize. At each branching step, we split the largest interval into two halves with equal number of points. The bounds can be computed in the same manner as before based on the smallest and largest bonding polygon regions b^+ and b^-. In theory it is possible to search for polygons with any number of edges using our method, but in practice it is advisable to keep k small (i.e., $k \leq 5$) as the computational cost can be high if k is too large.

Figure 8 shows examples of objects with certain geometries where bounding polygons found higher localization and recognition scores than did rectangular bounding boxes.

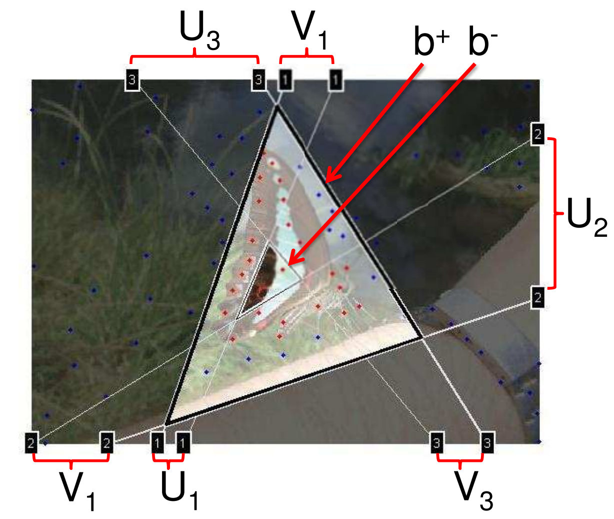

Figure 7. **Finding the best bounding polygon** (Section 3.2).

As can be seen, our method can find the optimal triangles, quadrilaterals, and pentagons, both convex and concave.

4. Conclusion and Future Work

This paper described a fast algorithm for concurrent object localization and recognition based on a data-dependent multi-class branch-and-bound formalism. We empirically demonstrated the superior performance of our method in terms of accuracy and speed compared to six baselines. In

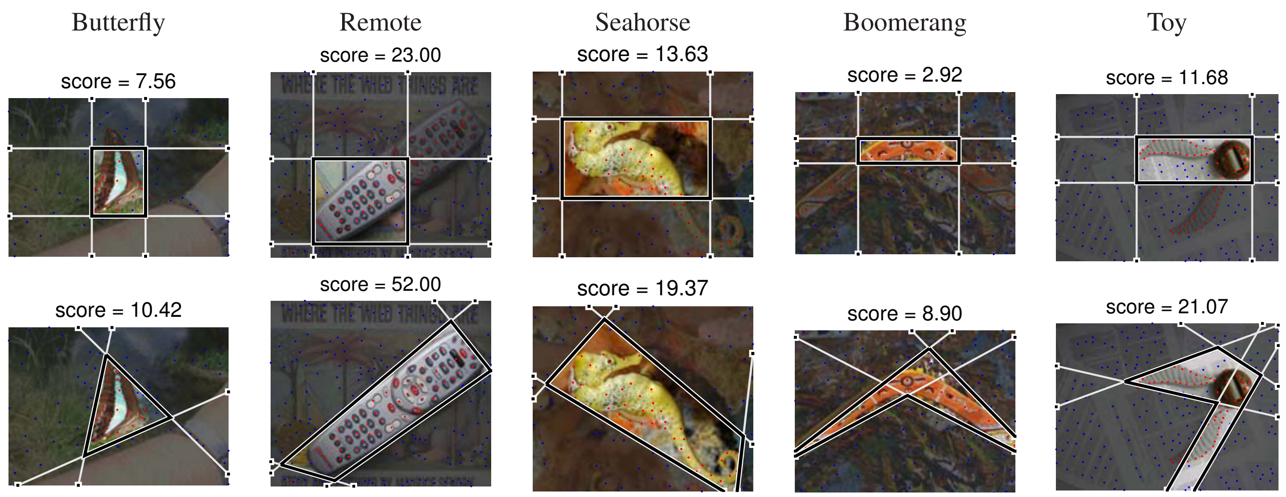

Figure 8. **Bounding boxes (top) versus bounding polygons (bottom)** (Section 3.2).

addition, we introduced flexible bounding geometries based on composite bounding boxes or polygons as alternatives to rectangular bounding boxes. As future work, we will perform quantitative analysis of these alternatives on larger datasets, develop methods to distinguish unique instances of the same object in close proximity, and explore the possibility to learn object-specific bounding geometries.

Acknowledgement

Funding for this research was provided in part by DARPA and by NSF IIS-0704479.

References

[1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *Proc. of ECCV*, pages 2–15, 2008.

[2] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proc. of ICCV*, pages 1–8, 2007.

[3] T. M. Breuel. A comparison of search strategies for geometric branch and bound algorithms. In *Proc. of ECCV*, pages 837–850, 2002.

[4] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proc. of CVPR*, 2007.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[6] 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.

[7] Y. Gat. A branch-and-bound technique for nanostructure image segmentation. In *Proc. of CVPR Workshop*, 2003.

[8] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[9] P. M. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proc. of CVPR*, pages 18–25, 2005.

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

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. of CVPR*, volume 2, pages 2169–2178, 2006.

[12] 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.

[13] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. of CVPR*, volume 2, pages 2161–2168, 2006.

[14] S. Obdrzálek and J. Matas. Sub-linear indexing for large scale object recognition. In *Proc. of BMVC*, 2005.

[15] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Proc. of NIPS*, volume 12, 2000.

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

[17] J. Winn and N. Jojic. Locus: learning object classes with unsupervised segmentation. In *Proc. of ICCV*, volume 1, pages 756–763, 2005.

[18] S. Yu, R. Gross, and J. Shi. Concurrent object recognition and segmentation with graph partitioning. In *Proc. of NIPS*, 2002.