Learning to Localize Detected Objects

Qieyun Dai Derek Hoiem
Department of Computer Science
University of Illinois at Urbana-Champaign
{dai9, dhoiem}@illinois.edu

Abstract

In this paper, we propose an approach to accurately localize detected objects. The goal is to predict which features pertain to the object and define the object extent with segmentation or bounding box. Our initial detector is a slight modification of the DPM detector by Felzenszwalb et al., which often reduces confusion with background and other objects but does not cover the full object. We then describe and evaluate several color models and edge cues for local predictions, and we propose two approaches for localization: learned graph cut segmentation and structural bounding box prediction. Our experiments on the PASCAL VOC 2010 dataset show that our approach leads to accurate pixel assignment and large improvement in bounding box overlap, sometimes leading to large overall improvement in detection accuracy.

1. Introduction

Object localization is an important step in many computer vision tasks. In an analysis of detection error, we found that the largest single source of error was inaccurate bounding box localization. Detection performance could be improved considerably if we could accurately localize detected objects. More important, accurate localization provides valuable silhouette and contour cues for estimation of shape, pose, affordance, subordinate categories, and other object properties. Consider the boat in Figure 1: the detection on the left provides the rough location, but we need more precise localization to reason about the boat's shape or movement

Accurate object localization involves two major challenges: how to determine which pieces of evidence locally fit the object model and how to jointly infer object foreground/background regions. Local feature assignment is made difficult by large intra-class vairance, which we address by leveraging spatial cues and appearance estimates from an initial detection (Section 2). To achieve joint object region reasoning, we rely smoothness constraints (Section 4.1) or aggregated information (Section 4.2).

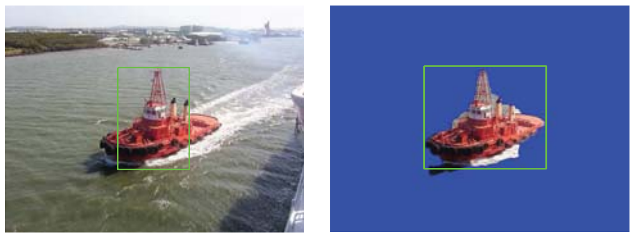
Figure 1. Given an initial detection (left), our goal is to assign features to the object and precisely localize it (right). We investigate cues for assigning color pixels and edge pixels to the object and for localizing the whole object with a segmentation or structural bounding box prediction (green box on right).

An overview of our approach is given in Figure 2. We first detect the object using a modified version of the deformable parts model (DPM) detector [9] (Section 2). Then, we predict which pixels are part of the object based on color and edge information (Section 3). To determine the full extent of the object, we propose two approaches (Section 4): segmentation using graph cut [3] on trained CRF potentials, and a structural learning approach to directly predict the bounding box. One interesting aspect of the segmentation approach is the use of occlusion boundary predictions to generate unary potentials, rather than using edges only for pairwise potentials, as is common. The direct bounding box prediction offers a useful way to determine the extent of objects not amenable to pixel segmentation, such as bicycles. Our experiments (Section 5) on PASCAL VOC [8] validate our color models and edge cues. We evaluate whole-object localization in terms of bounding box overlap, and demonstrate improvement in detection performance.

Background. Most work in object segmentation is based on user interaction, such as provided bounding boxes or strokes (e.g., [14, 19]). Their color models provide the basis for our own color models. Many other researchers work on category segmentation (e.g., [21]), where the goal is to label all pixels of a particular object category. Our problem differs in that we must identify pixels that belong to one particular instance, which might be surrounded by similar objects of the same category. A few, mostly recent, methods consider object localization from a detection window. Ramanan and colleagues [18, 25] incorporate position pri-

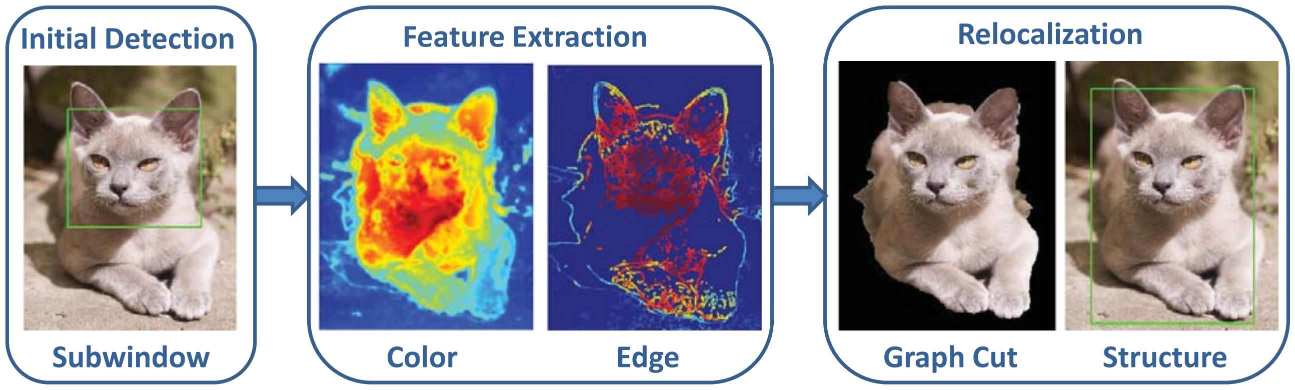

Figure 2. **Framework Overview:** (1) A subwindow detector is used to detect part of an object; (2) Color/edge features are computed from both category and instance level information; (3) We apply two approaches for accurately localizing the object: learned graph cut segmentation and structural bounding box prediction.

ors and simple color models; Parkhi et al. [16] use only color models; others use template weights to identify object pixels or edge features [4, 11]. Our approach provides a more thorough investigation into color models and edge features for the purpose of detector-initialized instance localization. Other approaches detect local object cues as a replacement for template detection (e.g., [6, 13, 20]). These approaches tend not to work as well as template detectors on the most challenging detection datasets, but their ideas, particularly about contour representation and matching, can (and should) be more fully incorporated into detector-based localization. Our approach of learning to predict bounding box localization builds on recent work in structural learning for sliding window detection [2]. Other works exist in structural learning of image segmentation [1], [22], and we follow [22] which learns parameters for different potentials and solves image segmentation using graph cuts.

2. Subwindow Detector

Currently, most detectors are trained to provide high scores only for well-localized object windows. For example, the DPM (v4) detector [9] is trained to achieve at least 0.7 overlap ($\frac{\text{Area(gt}\cap\text{det)}}{\text{Area(gt}\cup\text{det)}}$) on all training examples. The problem is that, to fit a window to highly deformable objects, the appearance model must absorb the high variation in position of discriminative features, such as the face. For this reason, Parkhi et al. [16] recently observed that detecting cat and dog faces using DPM (with supervised annotations), followed by segmentation, provides a large boost in detection performance, compared to localizing full bodies.

Our experiments show that, even without head annotations, changing one training parameter can yield similar improvements. Instead of requiring the latent detector window to have 0.7 overlap with ground truth during training, we require $\frac{\text{Area(gt}\cap\text{det)}}{\text{Area(gt)}} > 0.3$ and $\frac{\text{Area(gt}\cap\text{det)}}{\text{Area(det)}} > 0.9$: the detection should be within the ground truth but does not need to occupy the full extent. We call this a "subwindow" detector. The subwindow detector often has fewer confusions with background and similar objects but does not localize the entire object body. For example, the top 250 cat detections by the original DPM detector include: 111 precisely localized (≥ 0.5 overlap) cats; 48 loosely localized (≥ 0.1 overlap) cats; 59 instances of other animals; and 32 other (background, dissimilar objects) mistakes. The top 250 detections of the subwindow detector includes 241 cats, with only 2 confusions with other animals and only 7 other mistakes, but 182 of the cats are not precisely localized ($0.1 \leq \text{overlap} < 0.5$).

This simple trick of changing the training overlap criteria helps with the most deformable objects, such as cats and dogs and, to a lesser extent, other animals. While increasing localization error, confusion with background and other objects is greatly reduced. For many vehicles, which are mostly rigid, however, our trick does not help and other methods to find good latent sub-object windows may be required. One additional advantage of the subwindow detector is that its detections are more likely to be contained within the object, making it easier to estimate appearance models. For simplicity of description and analysis, we use the subwindow detector for all further experiments.

3. Local Feature Assignment

One key challenge of object localization is to determine whether each feature, such as a color pixel, edge pixel, or interest point descriptor, belongs to the object. If object features can be localized accurately, simple grouping methods will succeed. With large errors in local estimates, even the best fitting or segmentation techniques will fail.

We explore, in Section 3.1, dense labeling based on color and position and, in Section 3.2, sparse labeling of edge pixels. Color often works well for the main body of the object but can fail at extremities, due to appearance difference (e.g., black cat with white feet) or confusion with background. Complementary to color cues, edge pixels are often more discriminative than interior color pixels because they correspond to object shape and can be labeled accurately at object boundary.

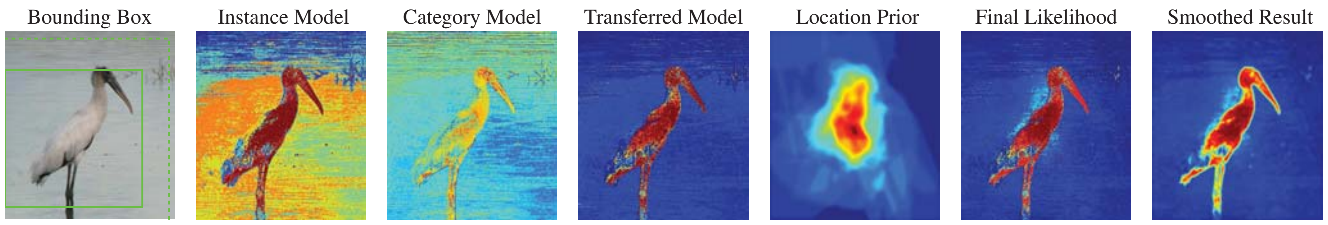

Figure 3. **Example of color pixel cues.** Left to right: detection window (the solid bounding box is the initial detection, the dashed bounding box is the predicted maximum window); probability predicted by instance model; probability predicted by category model; probability predicted by transfer model; position prior; final likelihood; smoothed likelihood.

Color Pixel Cues	
Instance color models: $\log P_{\text{ifg}}$, $\log P_{\text{ibg}}$, P_{i}	3
Category color models: $\log P_{\text{cfg}}$, $\log P_{\text{cbg}}$, P_{c}	3
Transfer color model: $\log P_{\text{transfer}}$, P_{t}	2
Position prior: P_L, $P_L > 0$	2
Edge Pixel Cues	
Position/orientation hists: $P_{\text{xy}\theta}$, $\log P_{\text{xy}\theta}$	8
Edge magnitude: $\|\|g_x, g_y\|\|$, $\|\|g_x, g_y\|\|^2$, Pb, $Pb > 0.1$	4
Color/texture: P, $\log P$, average on each side	24
Indicator for part of contour, straight segment	2
Contour likelihoods: P_{contour}, $\log P_{\text{contour}}$	8
Segment likelihoods: P_{segment}, $\log P_{\text{segment}}$	8

Table 1. Summary of cues for color and edge models. We train a logistic regressor on the trainseg set to predict the likelihood that each color/edge pixel belongs to the object based on these cues. Numbers on right indicate the number of features for each type.

3.1. Color Pixel Labeling

A color-based pixel likelihood model is foundational in many recent approaches to object instance localization. Such models typically include some combination of a shape or position prior, an instance-specific color model, and a category-wide color model. We mostly follow standard practice in creating our shape and color models (Table 1, Figure 3). Our main innovations are creating a "transfer" model and a simple method for combining category information, instance information and shape prior that seems to outperform previous strategies.

Shape prior. We incorporate a layout/position prior $P_L(o_i = 1|x_i, y_i)$, the likelihood that the i-th pixel belongs to the object ($o_i = 1$) given the pixel position (x_i, y_i) relative to the detection window. Similar to Yang et al. [25], we estimate a 100×100 pixel soft shape mask for each detector component, roughly accounting for object pose. In the trainseg set, each segmentation window is reshaped to 100×100, and the likelihood at each position is computed as the fraction of times that the corresponding pixel belongs to the object instance in the segmentation annotations. The mask is smoothed with a Gaussian filter ($\sigma = 1$).

Color models. Our color model aims to account for the typical colors of an object category (e.g. cats are rarely green) and the colors of the specific instance being localized. Each color model is a distribution of the pixel color c_i under some assumption for the foreground and background. The category model assumes that the instance follows the same distribution as general object pixels of the same category. The instance model assumes that the instance colors throughout the object are similar to those found within the detection window. The transfer model assumes that the instance colors have the same distribution as one similar example in the training set.

We learn the category model on the trainseg set, by computing a histogram in L*a*b* space with 32 × 16 × 16 bins using counting. The histogram density is smoothed with a uniform prior (weight=0.01) and an isotropic 3D Gaussian filter ($\sigma = 1.5$ for ranges of 100/50 for luminance/chrominance). The category foreground model $P_{\text{cfg}}(c_i|o_i = 1)$ is computed over all object pixels in trainseg; the category background model $P_{\text{cbg}}(c_i|o_i = 0)$ is computed over all background pixels in the segmentation windows of trainseg. Similarly, we compute an instance color model within the detection window $P_{\text{ifg}}(c_i|o_i = 1)$ and outside the maximum object window with a minimum border of 5 pixels $P_{\text{ibg}}(c_i|o_i = 0)$. Our subwindow detector is trained to be almost entirely within the ground truth window, so the window usually includes a subset of the object pixels and very few background pixels. To estimate the color distribution over the entire object, we transfer the color model $P_{\text{transfer}}(c_i|o_i = 1)$ from the trainseg set that best matches according to the P_{ifg} estimates. The best match is determined by histogram intersection, weighted by $\frac{P_{\text{cfg}}(c_i|o_i=1)}{P\text{cfg}(c_i|o_i=1)+P\text{cbg}(c_i|o_i=0)}$. Given these foreground and background models, we can compute posterior likelihoods $P_c = \frac{P_{\text{cfg}}}{P_{\text{cfg}}+P_{\text{cbg}}}$, $P_i = \frac{P_{\text{ifg}}}{P_{\text{ifg}}+P_{\text{ibg}}}$, and $P_t = \frac{P_{\text{transfer}}}{P_{\text{transfer}}+P_{\text{ibg}}}$.

Combining models. We combine different models to get a better estimate of the foreground/background color distribution. A typical fusion approach is taken by Parkhi et al. [16], which computes a weighted average of the instance and category posterior color likelihoods. Another possible approach is to weigh the log likelihood ratios, as in logistic regression. Our approach generalizes these two options by training a linear logistic regressor, with inputs of each color log likelihood ($\log P_{\text{cfg}}$, $\log P_{\text{cbg}}$, $\log P_{\text{ifg}}$, $\log P_{\text{ifg}}$ $\log P_{\text{transfer}}$), each posterior likelihood (P_{c}, P_{i}, P_{t}), and position likelihood (P_L, $P_L > 0$). Training features for category color and position models are computed on trainseg in a leave-one-image-out manner to avoid overfitting. Our experiments (Section 5,

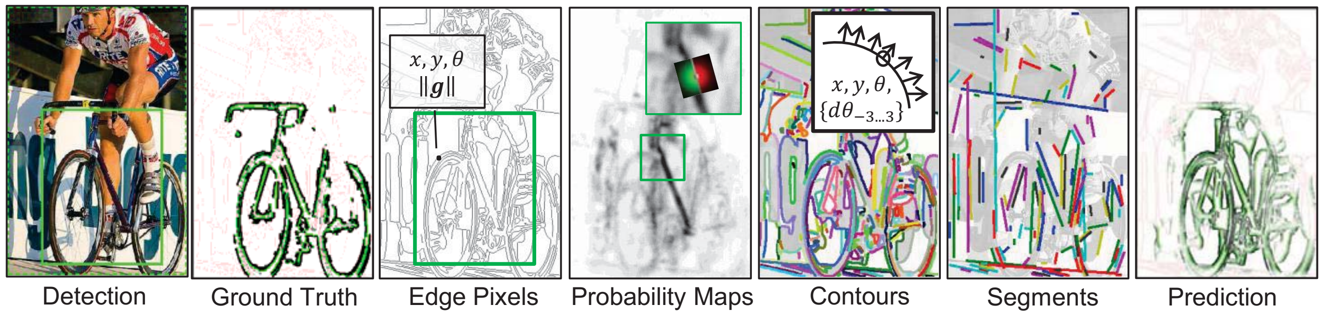

Figure 4. **Illustration of edge cues.** Detection window is shown as green box and maximum extent (whole image here) as dashed box. Object edge/foreground pixels are black; background edge pixels are red. Object side of exterior pixels is green. Foreground likelihood on each side is computed using one-sided Gaussian weighted averaging filter. Predictions are confidence-weighted (faded pixels are less confident). Ground truth is labeled based on PASCAL VOC segmentation maps, causing slight labeling noise.

Figure 7) show that the logistic regression on separate posteriors and log color likelihoods outperforms the weighted-vote-of-posteriors method used by [16].

3.2. Edge Pixel Labeling

Localization, pose estimation, and many other object interpretation problems require determining whether image edges correspond to background, object interior or occluding contours of an object. For occluding contours, we also want to know which side of the edge pertains to the object. Specifically, we want to classify each edge pixel into four labels: (1) background; (2) object internal; and (3,4) object external side 1 or side 2. Many researchers have studied the problem of identifying edges that correspond to a particular object category [17, 7]. Our problem is made easier by the rough detector localization but sometimes made more challenging by nearby objects from the same category.

We apply a variety of cues (Table 1, Figure 4), including edge position and orientation, boundary magnitude, color and texture on either side of the edge, as well as the position and shape of contours. We compute features for Canny edge pixels [5] and classify the edges using a trained logistic regression classifier. Our experiments (Section 5, Figure 8) indicate that each set of cues is helpful and that accuracy in edge prediction is high.

Local Edge Cues. Relative edge position (x_i, y_i) and orientation θ_i to the detection box are important cues for object shape. We compute histograms from the trainseg set for each label $e_i \in \{1, 2, 3, 4\}$, estimating $P_{\text{xy}\theta}(x_i, y_i, \theta_i | e_i)$. We use 16×16×16 histograms, computed and smoothed similarly to our color histograms (Section 3.1). All positions are defined relative to the maximun object window (computed from the detection window based on statistics gathered from the training set). We use $P_{\text{xy}\theta}$ and $\log P_{\text{xy}\theta}$ as logistic regression features. Edge magnitude is informative when differentiating between object interior and exterior edges, due to large contrast between object and non object regions, resulting in stronger edges. We represent boundary magnitude as the gradient magnitude $||(g_x, g_y)||$, gradient magnitude squared $||(g_x, g_y)||^2$, Global Pb [15], and thresholded Global Pb (Pb >0.1).

Color and Texture Cues. By estimating whether pixels on each side of an edge are likely to belong to the object, we can obtain valuable cues for differentiating between interior, exterior, and background edges. We first compute probability of color and texture maps using the instance and learned color models described in Section 3.1 and additional instance-based models learned for K-means clustered L*a*b pixels ($K = 128$) and MR4 texture filters [24] ($K = 256$). As features, we compute oriented, single-sided Gaussian-weighted averaging filter responses and log filter responses on either side of each edge pixel.

Contour and Segment Cues. Knowledge of connected contours and straight line segments can also be helpful as they are closely connected with object shape. For instance, airplanes are likely to be composed of straight lines, while dogs are more likely to have a curvy contour. Straight lines that extend well beyond the object are unlikely to be part of the object. To get contours from the edge map, we remove junction pixels and find connected components. For each edge pixel on a contour of minimum length (7 pixels), we store the position, orientation, and the relative orientations of two edge pixels on each side of the contour. In training, these contour features are clustered ($K = 250$), and histograms are estimated for each label to get P_{contour} features. For edges not on a contour, a default prior value is used for the features. We use, as an additional feature, an indicator whether an edge pixel is on a minimum-length contour. We compute straight line segments using an algorithm similar to Kosecka and Zhang [12] and then link co-linear, proximate edge segments. Features of position, segment length, and maximum distance to the bounding box, normalized by the detection window position and size, are computed, clustered, and histogrammed ($K = 100$), to yield P_{segment} features analogous to the contour features.

4. Object Localization

We propose two approaches to localize the entire object based on local cues. The first is a segmentation approach using graph cuts inference to assign each pixel to object or background. This approach makes it easy to incorporate data-driven smoothness constraints, but segmenting wiry objects like bicycles is difficult. We also propose a structural learning approach to directly predict the bounding box coordinates. This approach makes it easy to incorporate multiple sparse cues and its robustness comes from a compact model, rather than smoothing priors.

4.1. Segmentation Based Inference

We find the best pixel labeling by solving the following optimization problem:

$$\mathbf{L} = \underset{l_1,\ldots,l_n}{\operatorname{argmin}}\, \mathbf{w_1^T} \Sigma_i l_i \mathbf{\Phi}(I_i) + \mathbf{w_2^T} \Sigma_{i,j}(1-\delta(l_i,l_j))\mathbf{\Psi}(I_i,I_j)$$

where $\mathbf{L} = (l_1, l_2, \ldots, l_n)$ is the vector of labels assigned to each pixel in the image, I_i represents the i-th pixel in the image, $\mathbf{\Phi}(I_i)$ is the unary potential, $\mathbf{\Psi}(I_i, I_j)$ is the pairwise potential, $\delta(l_i, l_j)$ is the indicator function for whether l_i and l_j are the same, and $\mathbf{w_1}$, $\mathbf{w_2}$ are the weight vectors associated with different potentials.

Unary potentials measure the affinity of pixels to the object class. Pixels with negative potentials are likely to be foreground. Our local cues provide us with: P_{color}, the likelihood that each pixel is assigned to foreground based on color; P_{ebg}, likelihood that edge is in background; P_{eint}, likelihood that edge is inside the object; and P_{eside1} and P_{eside2}, the likelihoods that the edge pixel is on the boundary with a particular object side. These likelihoods are encoded in our unary terms:

- $\Phi_{\text{color}}(I_i) = -\log \frac{P_{\text{color}}(I_i)}{1-P_{\text{color}}(I_i))}$
- $\Phi_{\text{edge}_{\text{interior}}}(I_i) = -\log \frac{1-P_{\text{ebg}}(I_i)}{P_{\text{ebg}}(I_i)}$
- $\Phi_{\text{edge}_{\text{exterior}}}(I_{i+\alpha}) = -\log \frac{P_{\text{eint}}(I_i)+P_{\text{eside1}}(I_i)}{P_{\text{ebg}}(I_i)+P_{\text{eside2}}(I_i)}$ and $\Phi_{\text{edge}_{\text{exterior}}}(I_{i-\alpha}) = -\log \frac{P_{\text{eint}}(I_i)+P_{\text{eside2}}(I_i)}{P_{\text{ebg}}(I_i)+P_{\text{eside1}}(I_i)}$, where α is an offset to one side of the oriented edge.
- $\Phi_{\text{min}}(I_i) = \min(\Phi_{\text{interior}}(I_i) + \Phi_{\text{exterior}}(I_i), \Phi_{\text{color}}(I_i))$
- $\Phi_{\text{const}(I_i)} = 1$ (bias term)

Binary potentials enforce smoothness and continuity of object regions. We use two measurements of object boundary: Global Pb and the edge probability $P_{\text{edge}}(I_i) = P_{\text{eside1}}(I_i) + P_{\text{eside2}}(I_i)$. For each (I_i, I_j) pair, we choose the pixel with stronger boundary response as the binary potential and calculate $\Psi_{\text{Pb}}(I_i, I_j) = \min_{t=i,j} \exp(-Pb(I_t))$, and $\Psi_{\text{edge}}(I_i, I_j) = \min_{t=i,j} \exp(-P_{\text{edge}}(I_t))$. We set Pb values below 0.1 to 0, and assign maximum binary potential value to pixels with zero Pb value. Similarly, for non canny edge pixels, we use the Pb value if it is non-zero, or the maximum binary potential otherwise.

Inference Procedure. Graph cuts [3] is used to find the minimum energy solution and the parameters associated with different potentials are learned on the trainseg set using the large margin structural framework proposed in [22]. In particular, we use the loss-augmented inference with margin rescaling, minimizing training loss directly in the objective function. To be consistent with our evaluation, we use bounding box overlap (4.2) as our loss function. The final prediction is obtained by fitting a bounding box to the largest connected component of the segmentation mask.

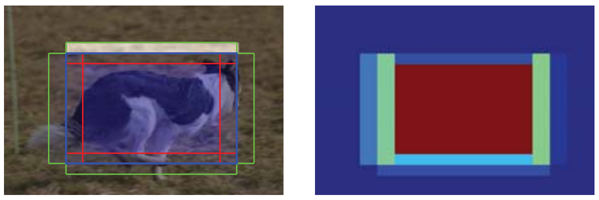
Figure 5. **Illustration of structural feature.** Left: the 10 regions used for computing structural features for a candidate detection window. The interior bounding box boundaries (4), the exterior bounding box boundaries (4), the interior of the bounding box (1), and the exterior of the bounding box (1). Right: The max value of each region for smoothed color map. The boundary width/height is set to 0.1 of the bounding box height/width.

4.2. Structural Prediction For Bounding Box

The structural prediction approach uses aggregated probability information to directly predict the location of the bounding box. We base our model upon the Structural SVM method of Tsochantaridis et al. [23] and use the four coordinates of the bounding box as the output structure.

Training Data Generation. Our training examples are detection window and ground truth pairs. For each ground truth object, we choose the highest scoring detection with sufficient overlap (at least 0.2 intersection/object, 0.8 intersection/gt) and ignore ground truth with no such corresponding detection. Our features are based on aggregated color and edge likelihoods at different positions. Given a candidate window, we divide the image into 10 regions, as shown in Figure 5. For each region we compute the max and average color/edge responses. For edge responses, we use the object probability and exterior object edge probability as defined in Section 3.2, and for color features we use the smoothed version of the learned color map and smoothed version of the instance color map such that the classifier is more robust to noise. To account for the location prior, we add bounding box overlap as an additional feature, defined as $(\frac{\text{Area(init}\cap\text{cand)}}{\text{Area(init)}}, \frac{\text{Area(init}\cap\text{cand)}}{\text{Area(cand)}}, \frac{\text{Area(init}\cap\text{cand)}}{\text{Area(init}\cup\text{cand)}})$ where Area(init) is the area of the initial detection and Area(cand) is the area of the candidate window. Together with the color/edge based features, we have a 83 dimension feature vector for each candidate bounding box. Our feature selection is based on the intuition that there should be large contrast in terms of edge/color response along the boundaries of a bounding box, with the inside having higher object probability.

Loss Function. We define our loss based on the overlap with ground truth as:

$$\Delta(y_{\text{i}}, \hat{y}_i) = \begin{cases} 1 & \text{if overlap}(y_{\text{i}}, \hat{y}_i) \le 0.25 \\ 0 & \text{if overlap}(y_{\text{i}}, \hat{y}_i) \ge 0.75 \\ 1 - \text{overlap}(y_{\text{i}}, \hat{y}_i) & \text{otherwise} \end{cases}$$

where $\text{overlap}(y_{\text{i}}, \hat{y}_i) = \frac{\text{Area}(y_i \cap \hat{y}_i)}{\text{Area}(y_i \cup \hat{y}_i)}$.

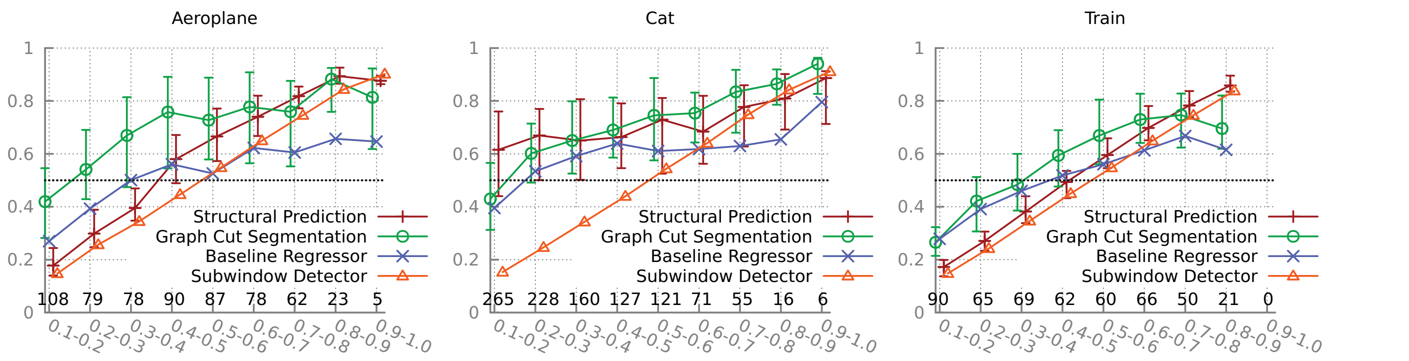

Figure 6. **The improvement in localization measured as overlap with ground truth.** We show error bars (25 percentile, median, 75 percentile) for the two integration methods, and lines (median) for the subwindow detector and the baseline method. The x axis is the overlap for subwindow detector, the y axis is the final prediction. The numbers above the x axis are the number of detections with the initial overlap specified on the x axis.

Objective Function and Optimization Algorithm. The Structural SVM (margin-rescaling) requires larger margins for examples with higher loss by solving the following optimization problem:

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_i C\xi_i$$

$$\text{s.t. } \forall i : \mathbf{w}^T\Phi(y_i) \geq \max_{\hat{y}_i}(\mathbf{w}^T\Phi(\hat{y}_i) + \Delta(y_i, \hat{y}_i)) - \xi_i$$

where $\Phi(y_i) = \phi(x_i, \hat{y}_i)$ is the feature vector associated with prediction $\hat{y}_i$, $\Delta(y_i, \hat{y}_i)$ is the loss function, and ξ_i is the slack variable. We modify this by removing any constraints based on zero-loss solutions, so that any zero-loss solution is acceptable.

Inference Procedure. To solve the structural learning problem, we need to find the highest scoring zero loss bounding box and the bounding box with the highest loss augmented score. Though searching over all possible bounding boxes in a sliding window fashion seems straightforward and easy to implement, examining $O(n^2)$ possible bounding boxes (n being the number of pixels in the given image) is infeasible in terms of memory and time. Therefore, we approximate the best solution using a "coordinate ascent" approach that searches over a subset of all possible bounding boxes, iteratively maximizing the score over each bounding box coordinate. When searching for the highest scoring ground truth bounding box, we use the real ground truth as start point, and perform coordinate ascent with 4 different coordinate ordering, choosing the highest scoring bounding box among the four.

5. Experiments

Our experiments measure the accuracy of our local cues in pixel label assignment and the improvement in object localization achieved by our integration method. We show that our proposed set of features can accurately determine which pixels are part of the object and our integration method provides great improvement in localization for most categories, which also leads to improved average precision when measuring detector performance. We conduct our training on the trainseg and train set of PASCAL VOC 2010 [8] and test on the valseg set for feature accuracy and the val set for localization and detector performance. Qualitative results are shown in Figure 9.

	sw	sw+lr	sw+seg	sw+struct
aeroplane	0.446	+0.036	+0.184	+0.087
bicycle	0.546	+0.009	-0.016	+0.049
boat	0.364	+0.028	+0.040	+0.041
bus	0.542	+0.002	+0.051	+0.052
car	0.558	-0.003	+0.041	+0.043
motorbike	0.547	+0.000	+0.042	+0.037
train	0.439	+0.040	+0.100	+0.045
vehicle$_{\text{avg}}$	0.492	+0.016	+0.063	+0.051

	sw	sw+lr	sw+seg	sw+struct
bird	0.400	+0.044	+0.145	+0.106
cat	0.363	+0.176	+0.257	+0.279
cow	0.465	+0.031	+ 0.108	+0.073
dog	0.351	+0.135	+0.212	+0.212
horse	0.483	+0.029	+0.054	+0.057
sheep	0.501	-0.022	+0.042	+0.028
animal$_{\text{avg}}$	0.427	+0.066	+0.136	+0.126

Table 2. **Improvement in overlap with ground truth bounding box compared with the initial detection.** The first column shows the initial overlap and the rest of the table shows improvement over subwindow results. sw: subwindow detector, lr+sw: baseline regressor, sw+seg: graph cut segmentation, sw+struct: structural prediction.

Localization Improvement. To measure gains in localization (i.e. overlap with ground truth bounding box compared to the subwindow detector), we compare with a simple linear regressor baseline. The *linear regressor baseline* is trained on the training set to repredict the coordinates of the new bounding box, based on the relative location of the initial detection within the image and the width/height of the initial detection window. We take subwindow detections with above 0.1 ground truth overlap, and compute overlap for the relocalized bounding boxes. The average overlap improvement is shown in Table 2. The results show that both methods achieve great improvement in localization for almost all classes, except for the graph cut method on bicycle. We also show improvement in overlap as error bar plots

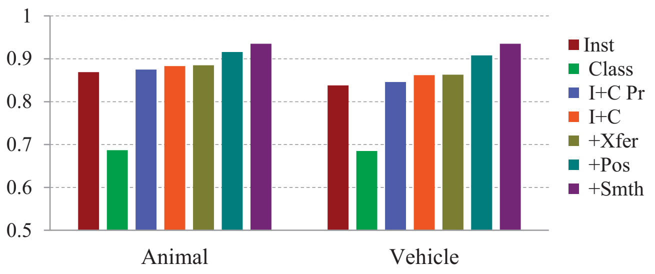

Figure 7. **Color-based Pixel Classification Accuracy.** We measure the accuracy of assigning a pixel to a particular object instance. The numbers are AUC, averaged over instances for each category, then averaged over categories for animals and vehicles. Inst: instance model; Class: class model; I+C Pr: combination of instance and class probabilities; I+C: combination of instance/class probabilities and instance/class log likelihood; +Xfer: addition of transfer model to I+C; +Pos: addition of position feature; +Smth: with Gaussian smoothing. Linear logistic regression models are learned on trainseg and evaluated on valseg.

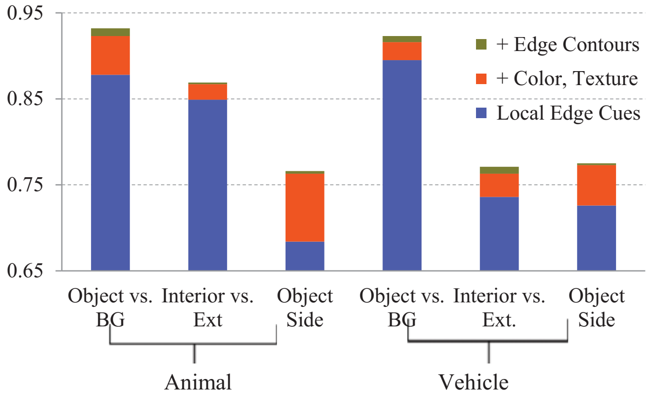

Figure 8. **Edge Classification Accuracy.** The figure displays accuracy of classifying edge pixels into: (1) object vs. background; (2) object interior vs. exterior boundary; (3) for exterior boundaries, which side is object. The numbers are AUC, first averaged over instances for each category, then averaged over categories, for animals and vechicles. Feature combinations are learned via linear logistic regression on the trainseg set, with evaluation on valseg.

in Fig 6.

Feature Improvement. We evaluate our proposed local features on the valseg set using pixel accuracy. We show in Figure 7 the accuracy of different color model and combination mechanism measured as area under the ROC curve(AUC). For both animal and vehicle categories, we see improvements brought by model combination, position information and local consistency as a result of smoothing. Figure 7 evaluates the effectiveness of different features in predicting different types of edge labels. We see gains in classification accuracy introduced by reasoning about color and material of image patches on both sides of the edge pixel and object shape information gathered from edge contours.

Detection Performance. We show the improvement in detection performance (Table 3) resulting from better object localization. Results of the DPM detector [10] are included

	DPM	sw	sw+lr	sw+seg	sw+struct	sw ub
aeroplane	0.442	0.346	0.366	0.430	0.411	0.490
bicycle	0.496	0.456	0.481	0.376	0.482	0.534
boat	0.066	0.035	0.051	0.042	0.045	0.108
bus	0.535	0.509	0.520	0.522	0.556	0.601
car	0.379	0.333	0.351	0.340	0.368	0.419
motorbike	0.388	0.359	0.367	0.346	0.376	0.411
train	0.342	0.289	0.319	0.369	0.313	0.453
vehicle$_{avg}$	0.378	0.332	0.351	0.346	0.364	0.431
	DPM	sw	sw+lr	sw+seg	sw+struct	sw ub
bird	0.054	0.039	0.049	0.071	0.053	0.079
cat	0.237	0.079	0.265	0.438	0.438	0.352
cow	0.079	0.061	0.084	0.059	0.084	0.118
dog	0.078	0.040	0.093	0.141	0.128	0.152
horse	0.355	0.308	0.367	0.300	0.338	0.395
sheep	0.260	0.214	0.176	0.217	0.203	0.243
animal$_{avg}$	0.177	0.124	0.172	0.204	0.207	0.223

Table 3. **Detection performance as average precision for different methods.** Last column: upper bound on sw detectors when all initial detection with >0.2 overlap is localized correctly.

as reference. We compute average precision on the 5000 most confident detections produced by each method and perform non-maximum suppression with a threshold of 0.5 before computing the average precision values. For our integration method, we use the confidence of the initial detection. Note that even without reranking or pruning subwindow results, our relocalized detectors achieved comparable results with the DPM detector and outperformed it for several categories. To compare with [16], we implemented a color based graph cut baseline using the the same unary and binary potentials as their work. (Direct comparison is not possible due to lack of annotation.) Our implementation differs from theirs in that we use our subwindow detector as initial detection instead of trained cat/dog face detectors and rather than choosing a specific portion of the detection window as object foreground, we determine instance level foreground and background the same way as Section 3.1. The *graph cut baseline* achieved comparable results (with an AP of 0.401) for cat and dog (with an AP of 0.147), but performed poorly for the rest of the categories (with an average AP of 0.135 for animals and 0.182 for vehicles). A few failed examples suggest that for more rigid objects, edge information and location prior are important cues for object location while color based models sometimes suffer from background pixels included in the initial detection window.

6. Conclusion and Discussion

We proposed an approach to improve the localization of a given detection based on color and edge features. Our contributions are: 1) We show that for flexible objects, if we do not force the detector to localize well, false positives caused by confusion with background or other objects can be greatly reduced; 2) We evaluate various cues in pixel assignment and provide models that labels color and edge pixels with high accuracy based on initial estimates of object bounding box; 3) We show that we can greatly improve the localization of the initial detection and generate accurate segmentation by integrating edge and color cues.

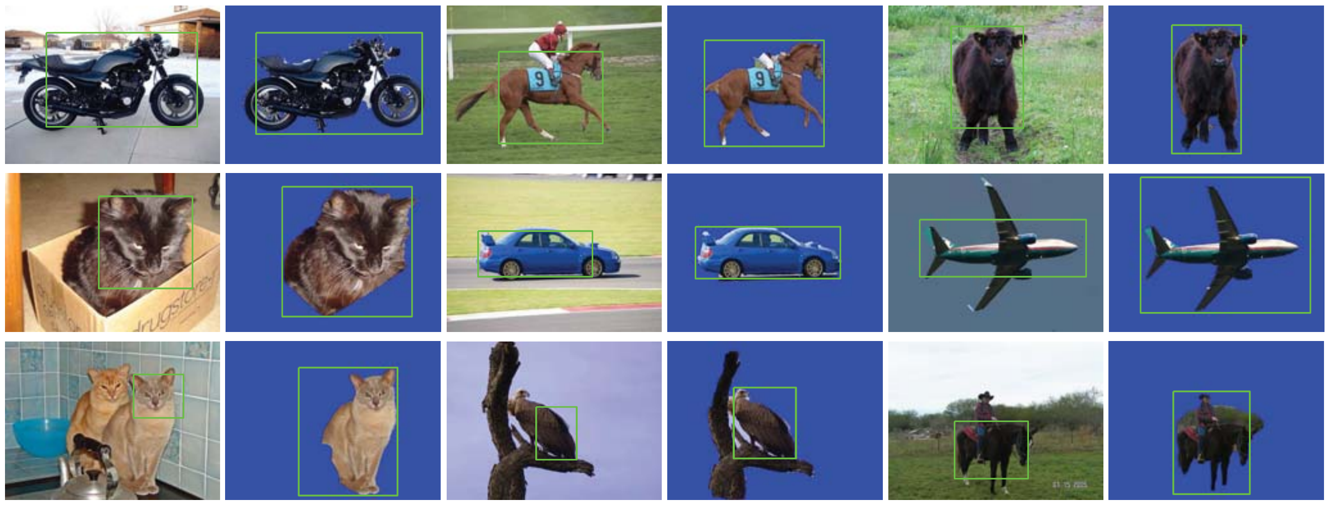

Figure 9. **Qualitative Results.** For each image pair, the left shows initial detection and the right shows segmentation and structural prediction result. First two rows: success. Last row: failure. Left to right: structural failure, segmentation failure and failure for both.

Our method only takes into account cues computed from a single detection. Therefore, it can be further improved if we leverage information from other nearby detections. For instance, if we could successfully identify different detections as belonging to the same object or different objects, we could reduce the number of false positives caused by duplicated detections and deal with the difficulties caused by nearby objects from the same category.

Acknowledgement This work was supported by NSF CAREER Award 10-53768 and NSF Award 09-04209.

References

[1] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized structural svm learning for supervised object segmentation. In *CVPR*, 2011.

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

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

[4] T. Brox, L. Bourdev, S. Maji, and J. Malik. Object segmentation by alignment of poselet activations to image contours. In *CVPR*, 2011.

[5] J. Canny. A computational approach to edge detection. *PAMI*, 1986.

[6] T. Deselaers, B. Alexe, and V. Ferrari. Learning object classes with generic knowledge. *IJCV*, 2011.

[7] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 2009.

[10] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[11] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[12] J. Kosecka and W. Zhang. Video compass. In *ECCV*, 2002.

[13] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR*, 2005.

[14] V. Lempitsky, P. Kohli, and C. Rother. Image segmentation with a bounding box prior. In *ICCV*, 2009.

[15] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *CVPR*, 2008.

[16] O. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. The truth about cats and dogs. In *ICCV*, 2011.

[17] M. Prasad, A. Zisserman, A. Fitzgibbon, M. Kumar, and P. Torr. Learning class-specific edges for object detection and segmentation. In *ICCV*, 2006.

[18] D. Ramanan. Using segmentation to verify object hypotheses. In *CVPR*, 2007.

[19] C. Rother, V. Kolmogorov, and A. Blake. Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH'04)*, 2004.

[20] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *PAMI*, 2008.

[21] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[22] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *ECCV*, 2008.

[23] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[24] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *IJCV*, 2005.

[25] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *CVPR*, 2010.