3D LayoutCRF for Multi-View Object Class Recognition and Segmentation

Derek Hoiem
Robotics Institute, Carnegie Mellon University
dhoiem@cs.cmu.edu

Carsten Rother, John Winn
Microsoft Research Cambridge, Cambridge, UK
{carrot,jwinn}@microsoft.com

Abstract

We introduce an approach to accurately detect and segment partially occluded objects in various viewpoints and scales. Our main contribution is a novel framework for combining object-level descriptions (such as position, shape, and color) with pixel-level appearance, boundary, and occlusion reasoning. In training, we exploit a rough 3D object model to learn physically localized part appearances. To find and segment objects in an image, we generate proposals based on the appearance and layout of local parts. The proposals are then refined after incorporating object-level information, and overlapping objects compete for pixels to produce a final description and segmentation of objects in the scene. A further contribution is a novel instance penalty, which is handled very efficiently during inference. We experimentally validate our approach on the challenging PASCAL'06 car database.

1. Introduction

In this paper, we address the problem of detecting and segmenting objects of a known class when seen from arbitrary viewpoints, even when the object is partially occluded. This task is extremely challenging, since inferring the position, orientation and visibility of an unknown number of objects involves reasoning within a high dimensional latent space.

Recently, Winn and Shotton [16] have introduced the Layout Conditional Random Field (LayoutCRF) algorithm to detect and segment objects while maintaining a consistent layout of parts (e.g., a nose above a mouth in a face) and reasoning about occlusions. They demonstrate success in detecting and segmenting side views of cars, but their method cannot handle multiple viewpoints or multiple scales.

Our main contribution is to relax this restriction, using a rough 3D model of the object class to register parts across instances during training, allowing detection of cars in a continuous range of viewpoints and scales. We also extend the object model to include a description of the color of the object. Further, we show how to include a per-instance cost in a CRF, while allowing efficient inference with graph cuts. Altogether, we are able, not only to detect objects across viewpoints and scales, but to label the pixels of the object into parts and describe the position, bounding box, viewpoint, and color of the object! In Figure 1, we show an example of our results on a test image.

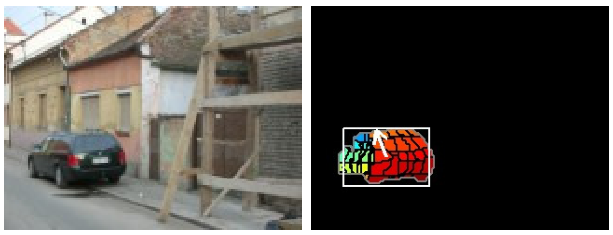

(a) Image (b) Parts/Object

Figure 1. We introduce the 3D LayoutCRF algorithm, which combines pixel-level and object-level reasoning to detect, segment, and describe the object.

The main challenge in detecting and segmenting objects across viewpoint and scale is that there is a huge space of possible solutions. How do we get from simple pixels to a complete description of the object? Our approach is to build up to our final model in several steps, with each step adding new information and providing a more precise hypothesis about the objects. First, for several wide viewpoint and scale ranges, we generate a set of proposals by labeling the pixels of the image into parts while maintaining a *local consistency* of neighboring parts. This gives us a rough idea of the viewpoint (e.g. within 45 degrees), scale (within a factor of $\sqrt{2}$), and position (within several pixels) of the object. To more precisely define each proposed object, we then enforce a *global consistency* of parts with respect to the object bounding box and search for the most likely part labeling and object description. After this refinement step, we compute a color model of each object (based on its current segmentation estimate) and of the background surrounding the object. This gives us several proposals at different viewpoints and scales, each with a precise object description and a pixel labeling into parts. Some of these proposals, however, will be incorrect, and others will be contradictory, claiming the very same pixels as parts. To decide which proposals are valid, we assign a per-instance cost to each object and find the best overall solution, considering both how well each object explains the image and how likely we are to see the object in a given position. In past CRF formulations, label smoothing terms have been unfairly burdened with the task of removing false positives. The incorporation of an instance cost provides a much more reasonable way of determining whether an object proposal has sufficient evidence.

A key idea in our approach is to use a coarse 3D model to roughly correspond physical parts across instances at different viewpoints. The benefits of 3D object models have been demonstrated by Ratan *et al.* [10], who find the object pose that provides the best appearance match with a template, Thomas *et al.* [14], who use an implicit 3D object model to improve detection, and Kushal and Ponce [6], who find objects in cluttered photographs based on 3D models obtained from stereo pairs. Our method allows us to take advantage of currently available large datasets (e.g. LabelMe [12]) of hand-segmented images, avoiding the need for multiple views [14] or stereo pairs [6] of the same object instance. Additionally, our 3D part correspondence enables feature sharing [15, 9] across viewpoints, rather than requiring separate appearance models for dozens of discrete viewpoints, as in [13].

Although the above mentioned approaches tackle multiple viewpoint detection and others have attempted to detect and segment objects from a *single* viewpoint (*e.g.* [1]), ours is the first, to our knowledge, to simultaneously detect and segment objects in a large range of viewpoints and scales. The key to our success is the ability to reason about local part appearance and occlusion relationships while maintaining a globally consistent description of the object.

2. The Model

We aim assign all pixels of an image $\mathbf{x}$ to an object instance or to the background. For each object instance, we also aim to capture the position, scale and viewpoint of that object. Hence, our model contains both pixel-level variables $\mathbf{h} = \{h_i\}$ and object-level variables $\mathbf{T} = \{T_m\}$. An overview of the entire model is shown in Figure 2.

At the pixel level, the part label h_i indicates the object instance that the pixel belongs to and the part of that instance the pixel lies on. Instances are numbered $\{0, 1, \ldots, M\}$ where the background is indicated by 0, and M foreground instances by $\{1, \ldots, M\}$. Each foreground instance is subdivided into H parts. Rather than defining parts according to a 2D rectangle, as in the original LayoutCRF, we define them over the surface of a 3D solid (Figure 3).

At the object level, the variables for the mth instance are denoted T_m and consist of the position and scale z_m, viewpoint V_m and color distribution C_m. The scale is anisotropic, so that instances of different aspect ratios can be detected.

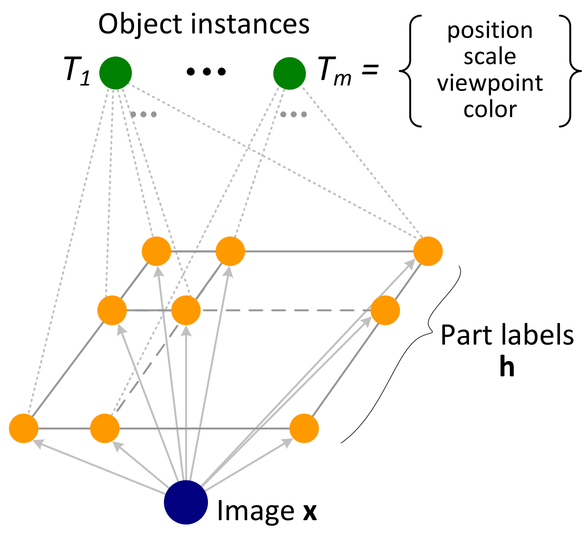


Figure 2. **The 3D LayoutCRF model.** The part labels $\mathbf{h}$ (orange) are conditioned on the image $\mathbf{x}$ (dark blue) and connected 4-wise with their neighbors. These pairwise potentials encourage neighboring parts to belong to the same object instance provided that they are consistent with the part layout of that instance. Each object instance has a set of variables T_m (green) relating to its position, viewpoint, color distribution and visibility. These instance variables affect the expected location/visiblity of the instance's parts via a rough 3D model. Each set of instance variables T_m is connected to all of the part labels.

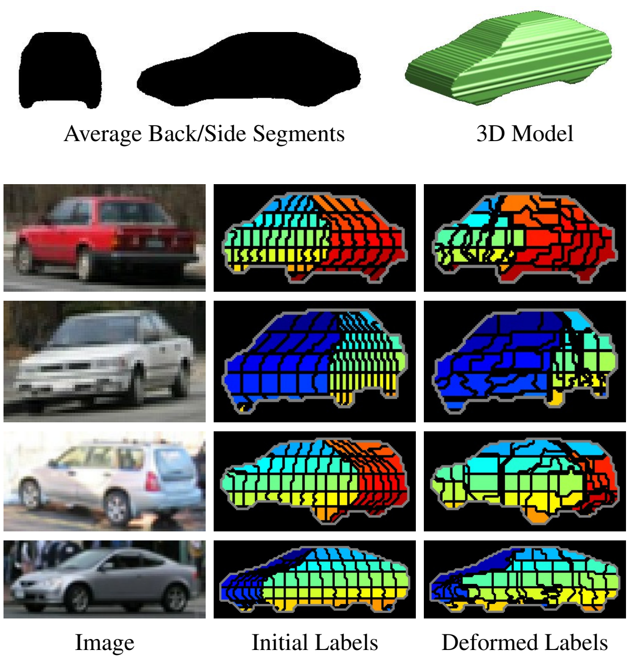


Figure 3. **3D LayoutCRF Part Assignment.** During initialization, we use a rough 3D model (top) to consistently assign the same physical part across different instances in different viewpoints. We then learn appearance models over those parts and use them to relabel the parts (bottom), allowing the part grid to deform slightly to better fit each training instance.

The probability distribution for all latent variables conditioned on the image is given by

$$P(\mathbf{h}, \mathbf{T} \mid \mathbf{x}; \boldsymbol{\theta}) = \frac{\exp\left[-E(\mathbf{h}, \mathbf{T} \mid \mathbf{x}; \boldsymbol{\theta})\right]}{Z(\mathbf{x}, \boldsymbol{\theta})} \qquad (1)$$

where θ are the model parameters and E is the energy:

$$\begin{aligned} E(\mathbf{h}, \mathbf{T} \mid \mathbf{x}; \boldsymbol{\theta}) = & \sum_i \overbrace{\phi_i(h_i \mid \mathbf{x}, \{T_m\})}^{\text{part appearance}} \\ & + \sum_{i,j} \overbrace{\psi_{ij}(h_i, h_j \mid \mathbf{x}, \{V_m\})}^{\text{part layout}} \\ + \sum_m \Big[\sum_i & \overbrace{\mu_i(h_i, x_i; C_m)}^{\text{inst. appearance}} + \overbrace{\lambda_i(h_i, T_m)}^{\text{inst. layout}} + \overbrace{\beta_{\text{inst}}(V_m)}^{\text{inst. cost}} \Big]. \qquad (2) \end{aligned}$$

The *part appearance* potentials ϕ_i use local image information to detect which part is at pixel i. The *part layout* potentials ψ_{ij} encourage neighboring pixels to be layout consistent, i.e. to have part labels belonging to the same object instance and in the correct relative layout. The *instance appearance* and *instance layout* potentials $\{\mu_i, \lambda_i\}$ favor part labelings that are consistent with the appearance, position, scale and viewpoint of each object instance. Finally, the *instance cost* β_{inst} defines a prior on the existence of an object at a particular viewpoint in the image. We will now look at each of these potentials in more detail.

2.1. Part appearance

The part appearance potential captures the mapping from local appearance features of the image to object parts. In the original LayoutCRF, since parts were always seen from the same viewpoint, this term had to represent only the intra-class variability in the appearance of a part. In the 3D LayoutCRF, we have to consider how the appearance of a part varies with viewpoint. One possibility is to train the appearance model to recognize parts independent of the viewpoint. However, this leads to much greater variability in appearance and so reduces detection accuracy, while providing less bottom-up information about the viewpoint. Instead, we choose to provide multiple appearance models for each part, one for each $45°$ viewing range V.

As in [16], the appearance models we use are decision forests. While separate appearance models are learned for each viewpoint range, we share features between the models by ensuring that the decision forests for each $\phi_i(h_i \mid \mathbf{x}, T; \boldsymbol{\theta})$ have identical structures. This sharing of features helps to reduce over-fitting in the individual models, as demonstrated by [15]. For symmetrical objects, we also enforce parameter sharing between pairs of viewpoints that mirror each other (*e.g.* car facing left and car facing

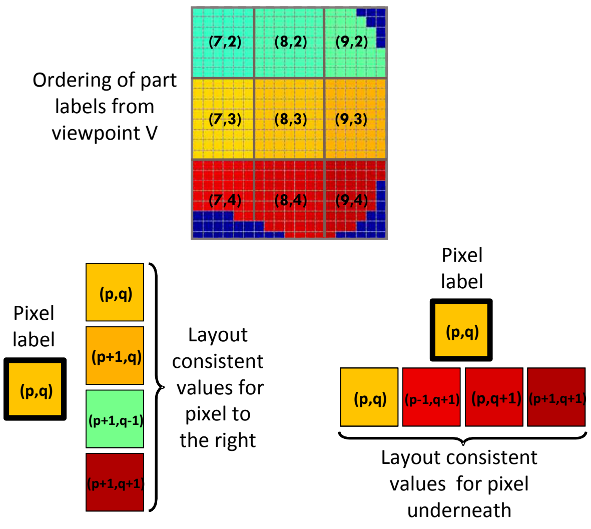


Figure 4. **Layout consistency.** From a given viewpoint V, object parts are expected to appear in a particular two-dimensional ordering (top). Neighboring part labels are *layout consistent* if they are consistent with this ordering (defined over the 3D surface of the object). To allow for object deformation and small rotations, the diagonally-related part labels are also considered layout consistent.

right), mirroring the image appropriately when evaluating features. Descriptions of the features used in the decision forests, along with details of the learning method are given in Sec. 3.1.

2.2. Part layout

The part layout potential favors part labels which are *layout consistent*, as defined in [16]. In essence, a part labeling is layout consistent if all pairs of neighboring parts are in the correct relative position, for example, in a face a nose part is above a mouth part (see Figure 4 for a more detailed explanation). For layout consistency to be applicable, it is necessary for the object parts to appear in the same relative position for any visible region of an object. When we fix the viewpoint, this is a good assumption for most rigid object classes. However, if we allow the viewpoint to change arbitrarily, the relative position of the parts can also change arbitrarily. For example, if a face can appear upside-down, then a nose part can appear below a mouth part. To avoid this, we fix the viewpoint to be within a $45°$ range for any proposed object instance. With this restriction on the viewpoint, layout consistency of any object region is a reasonable assumption, given that the LayoutCRF allows for small rotations/deformations. The assumption of a fixed viewpoint range also allows the appropriate appearance model to be selected for an object instance.

The part layout potential for a given viewpoint V, takes the form

$$\psi_{ij}(h_i, h_j \mid \mathbf{x}, V; \boldsymbol{\theta}) = \begin{cases} 0 & \text{Layout Consistent} \\ \beta_{\text{oe}}.e_{ij} & \text{Object Edge} \\ \beta_{\text{oo}}.e_{ij} & \text{Object Occlusion} \\ \beta_{\text{inc}} & \text{Inconsistent} \end{cases}$$

where e_{ij} is an edge cost that encourages object boundaries to align with image contrast edges (see [16]) and the four cases are:

Layout Consistent: Both h_i and h_j are layout consistent foreground labels as seen from viewpoint V, or both are background labels.

Object Edge: One label is the background, the other is an edge part (i.e., a part that lies on the edge of the object when seen from viewpoint V).

Object Occlusion: One label is an interior (non-edge) part label, the other is the background label or an edge part label. This represents the case where an object is occluded by another object or a 'background' object.

Inconsistent: Both labels are interior part labels which are not layout consistent.

For the experiments in this paper, we set the cost parameters to $\{\beta_{\text{oe}} = 3, \beta_{\text{oo}} = 6, \beta_{\text{inc}} = \infty\}$ when generating proposals and $\{\beta_{\text{oe}} = 1, \beta_{\text{oo}} = 2, \beta_{\text{inc}} = \infty\}$ for the later stages (when instance layout can also be considered).

2.3. Object Instance Model

The instance model ensures that the part labeling is consistent with the color, position, scale and viewpoint of the instance. It also provides a prior on the existence of an object at a particular viewpoint, through the use of a per-instance cost.

Instance appearance: Though colors may vary widely within an object class, any particular instance typically consists of a small set of colors. For instance, red and blue car doors are common, but a single car rarely has both. Thus, we require the color of the parts to be consistent with the overall color distribution of an instance. We represent the color distribution for instance m as a mixture of Gaussians model with parameters C_m, as used in [11]. We also learn a localized color distribution C_0 for the background. The instance appearance potential is defined to be

$$\mu_i(h_i, x_i; C_m) = -\beta_{\text{c}}\delta(h_i \in m) \log \frac{P_{\text{MoG}}(x_i|C_m)}{P_{\text{MoG}}(x_i|C_0)} \quad (3)$$

where P_{MoG} is the mixture of Gaussians model and $\delta(h_i \in m)$ is 1 if h_i is a part of instance m and 0 otherwise. Since this potential depends on the test image, it is learned during inference (see Section 3.2). In our experiments, its weight β_{c} is set to 0.25.

Instance layout: While layout consistency ensures that parts have a reasonable *local* layout, instance consistency ensures that parts are *globally* consistent with the position, scale and viewpoint of the instance. Unlike the single-viewpoint LayoutCRF, the global position of parts can no longer be specified using a 2D rectangular coordinate frame. We now specify the likelihood of each part given its position and the object's position, scale, and viewpoint. Thus, our object representation is expanded into viewpoint and scale space. The quantization for determining instance layout is finer than for part appearance sharing. In our experiments, we subdivide each viewpoint range of 45 degrees and height range of $\sqrt{2}$ into three subviewpoints, three height ranges, and two aspect ratio (bounding box width:height) ranges.

The instance layout potential is a look-up table for each quantized viewpoint V_m,

$$\lambda_i(h_i, T_m) = -\delta(h_i \in m) \log \frac{P(h_i|\text{loc}(z_m, i), V_m)}{P(h_i)} \quad (4)$$

where $\delta(h_i \in m)$ is as specified above, $\text{loc}(z_m, i)$ returns the position of pixel i in the object-coordinate system given that the object is at position/scale z_m, and $P(h_i)$ is the part prior. During the proposal stage, when object-level information is unavailable, we instead apply a constant penalty β_{bg} for assigning a pixel to background, in order to offset the low prior probability of any individual part.

Instance cost: We introduce a per-instance cost $\beta_{\text{inst}}(V_m)$ to the MRF formulation, which acts as a prior favoring image interpretations with fewer object instances. Effectively, it determines whether the *total evidence* for an object outweighs our prior bias against its existence. The instance cost is commonly used in object classification (e.g. [13]), and can be justified as the odds ratio in a log-likelihood ratio test or as the object description length in the MDL principle.

The use of an instance cost is a more natural way to remove false detections than relying on smoothing terms, resulting in better segmentations. It also provides the ability to determine that disconnected regions are part of the same object (for example, when a lamp post divides a car in two). We define a cost that depends only on viewpoint; dependence on scale and position could allow methods such as [3] to be employed. In Sec. 4.1, we demonstrate improvement due to our use of the instance penalty.

3. Training and Inference

3.1. Training

The full training process is summarized in Figure 5.

Learning the 3D Model: We create the 3D model (shown in Figure 3) by space carving [7] from thresholded segmentations of the rear and side views of cars, assuming an orthographic projection. To do this, we first center and rescale (according to height) the segmentations of each viewpoint. We then take the mean of the segmentations and threshold

TRAINING

1. Gather training examples with segmented and viewpoint-labeled objects
2. Construct rough 3D object model
3. Assign part labels to training examples
4. Learn part appearance (randomized decision trees)
5. Learn instance layout (simple counting over part locations)
6. Refine labeling (run inference steps 1 and 2); go to step 4 (one iteration)

Figure 5. **Training procedure for 3D Layout CRF.**

(at 0.5) to get an "average" segmentation from each view. Assuming an orthographic projection, we then carve voxels out of a solid cube. Finally, we assign parts to the surface of the 3D object model, projecting a grid onto each side.

To assign parts to a new training instance, we require a segmentation and orientation (obtained by clicking on the corner of the car). We rotate and scale (separately in each axis) our 3D prototype to match the orientation and segmentation of the training instance as closely as possible and back-project the part labels onto the object segment, again assuming an orthographic projection.

Choice of image features: Our local parts appearances are based on RGB intensity, filter responses of random patches selected from the training images (similarly to [15]) and a distance transform of a Canny edge image. The latter can effectively model unique edge appearances, such as the circle of a wheel, and long-range properties, such as that a pixel in large uniform regions will have a high distance to the nearest edge, and, thus, is unlikely to be part of a car or person.

Training Decision Trees: We apply the randomized trees method [8] to estimate the likelihood of a pixel label given its image features. Randomized trees allows efficient large-label learning (we have 120 parts in our experiments) and is robust to over-fitting. We learn a single set of 25 randomized trees, each with 250 leaf nodes, on a subset of our data. We then re-estimate the parameters of the trees, without changing the structure, using the millions of pixels in our training set. By sharing the tree structure across viewpoint models, we reduce over-fitting and allow more efficient training and testing.

3.2. Inference

To optimize the full objective function in one step is intractable and prone to poor locally optimal solutions. Therefore, the inference is split into three different optimization steps. In the first two steps, each potential instance in each viewpoint/scale range is optimized individually. The final step operates on the full objective function that considers total evidence across the entire image. The inference procedure is outlined in Figure 6 and illustrated in Figure 7.

INFERENCE

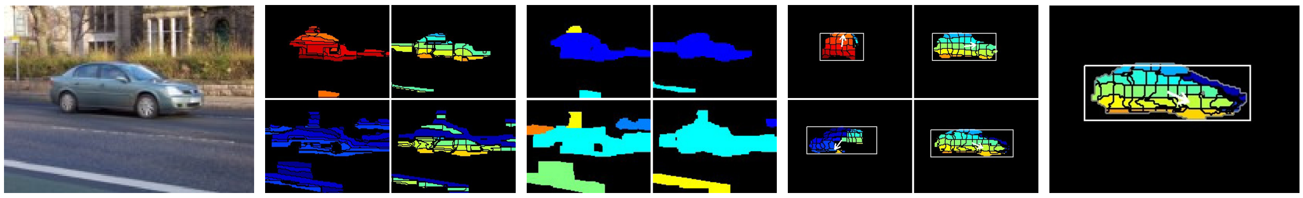
1. **Generate Proposals** (scale/viewpoint separately)
   (a) Compute $\mathrm{P}(\mathbf{h} \,|\, \mathbf{x})$
   (b) Label pixels into parts using only part-level appearance and consistency (TRW-S)
   (c) Connected components (by layout consistency) become proposals
2. **Refine Proposals** (instance/scale/viewpoint separately)
   For each proposal, iterate until convergence:
   (a) Find all likely object configurations $\{\hat{\mathbf{T}}\}$ given parts $\hat{\mathbf{h}}$ (s.t. $\mathrm{P}(\hat{\mathbf{T}} \,|\, \hat{\mathbf{h}}) > 0.01$)
   (b) Find most likely parts $\hat{\mathbf{h}}$ given $\{\hat{\mathbf{T}}\}$ (TRW-S)
   Compute object and background color distributions, object appearance terms
3. **Create Final Labeling** (instance/scale/viewpoint jointly)
   Input: part labeling for each proposal with unary potentials
   Assign object labels m to pixels using complete model (alpha-expansion graph cuts)

Figure 6. **Inference algorithm for 3D Layout CRF.**

1) Generate proposals: We use sequential tree-reweighted message passing (TRW-S) [4] to obtain the MAP solution and create layout-consistent connected components of parts to get our initial proposals.
2) Refine proposals: To refine each proposal, we iteratively (1) estimate the distribution of likely instance descriptions ($\mathbf{T}$) and marginalize to get the instance consistency terms for that proposal; and (2) find the most likely parts given the current instance estimate (also using TRW-S). By maintaining a distribution of instance descriptions during this iterative process, we robustly converge to a good solution.
3) Combine proposals: To determine which proposals are valid and the final segmentations of the objects, we apply alpha-expansion [5], with each expansion being a potential switch to a different object label. It can be shown that each expansion is submodular, making graph cuts well suited for this phase of the inference. The proposal and refinement stages take most of the computational time (typically 1-5 minutes per scale in a 120x160 image). The final alpha-expansion stage requires only a few seconds.

Applying the instance cost: We are the first, to our knowledge, to show that an instance cost can be handled appropriately with the alpha expansion procedure (i.e., that the objective function is expansion submodular).

The instance cost is defined as a cost per instance that is present in the image. If at least one pixel is labeled as part of an object, then that object must be visible, incurring a fixed cost; otherwise, the object may be invisible (non-existent) at no cost. Formally we may write this as hard constraint



(a) Input Image (b) Initial Parts (c) Connected Comps (d) Refined Proposals (e) Final Result

Figure 7. **Inference illustration.** From left to right, we show the input image, the initial part labelings (Figure 6, step 1b) that enforce local consistency (4 of 8 total viewpoints per scale), the corresponding layout consistent connected components (step 1c), four (of 24 total) refined proposals (step 2), and the final labeling after performing inference over the full model.

which is added to our objective function in Equation 2

$$\sum_m \sum_i \infty \ ((1 - s_m)\, IP_m(h_i))\,, \tag{5}$$

where the instance part function IP_m has a binary output indicating whether h_i is a part of instance m or not. The instance variable s_m is 1 if instance m exist in the image, otherwise 0. Note that this function prohibits the configuration that $s_m = 0$ and a part of instance m is present in the image. This term is indeed submodular [5], i.e. $E(0,0)+E(1,1) \leq E(0,1) + E(1,0)$. It is $E(0,0) = E(1,1) = E(0,1) = 0$ and $E(1,0) = \infty$, where $E(h_i, s_m)$ is the pairwise term between a pixel node and an instance node.

Learning the instance appearance: Since the color appearance of an object is instance specific the respective parameters in Equation 3 are learned during inference, in the end of the refinement stage (step 2 in Figure 6). We generate two proposed segmentations: a conservative estimate (increase β_{bg} by 0.05) in which all pixels are highly likely to be object and a loose estimate (decrease β_{bg} by 1) such that all pixels outside of it are highly unlikely to be object. We define a local background region of pixels within an enlarged bounding box (by ten pixels on each side), excluding pixels in the loosely estimated segment. The object color distribution and background color distribution are each estimated over their respective regions using a mixture of three diagonal-covariance Gaussians in L*a*b* space. The class-conditional log likelihood ratio of the color likelihoods is factored into our appearance term.

4. Experiments

In this paper, we have shown how to reason about object-level properties, such as viewpoint, size, and color, while also reasoning about pixel-level appearance and part consistency. We have also introduced a 3D model to allow part assignments on different instances to correspond roughly to physical parts on the object. Finally, we have shown how to incorporate a per-instance cost into the CRF, allowing object proposals to be rejected or accepted based on the entire evidence, instead of relying on local pairwise smoothing costs to remove false positives. Our goal in these experiments is to demonstrate the value of each of these contributions.

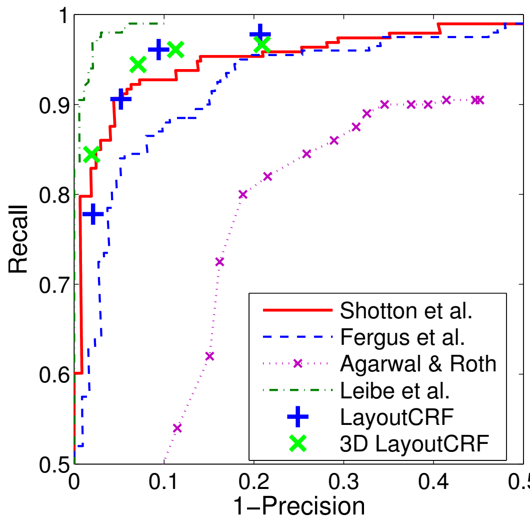


Figure 8. **Precision-recall on UIUC car test set.**

4.1. Comparison to the Original LayoutCRF

The key contribution of our algorithm is the ability to detect and segment objects in the presence of viewpoint and scale variation. To do this, we have introduced many modifications to the original LayoutCRF inference algorithm and improved the basic model by including an instance cost. To compare, we perform the experiment described by Winn and Shotton [16] on the UIUC car dataset. Note that, since this dataset is single-scale with only side views of cars, we do not incorporate our 3D model for this experiment.

In Figure 8, we show that our algorithm achieves higher recall (by about 8%) than Winn and Shotton at the high-precision regions of the precision-recall curve, with similar recall elsewhere. The benefit of incorporating an instance cost into our model can be seen in the segmentation and qualitative results (see Figure 9). The instance cost allows the smoothness and contrast costs to be reduced, since they are no longer responsible for removing false positives. Thus, our algorithm has a segmentation accuracy (average intersection-union ratio of ground truth and inferred object regions) of 0.77, compared to 0.66 for the original

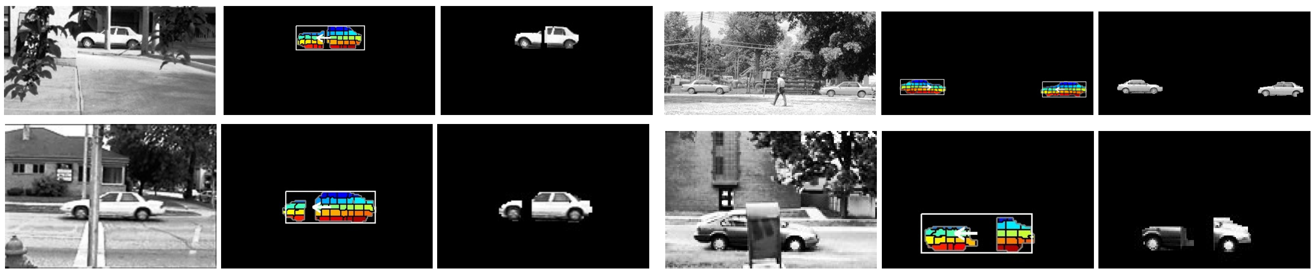


Figure 9. **Test results on the UIUC dataset.** Note the accurate segmentations and the ability to determine that disconnected car regions can be explained by a single instance. In (b), parts are labeled with separate colors, bounding boxes indicate the estimated extent of an object, and arrows indicate the estimated orientation. The instance cost allows disconnected object regions to be explained by a single instance (left, and right bottom row). The orientation is incorrectly estimated in (right bottom row), leading to a poorer segmentation.

LayoutCRF algorithm. More importantly, the instance cost allows our algorithm to correctly assign disconnected car parts to a single instance, when they are separated by an occlusion (see Figure 9 for examples). The instance cost allows the algorithm to follow Occam's razor – that if the same pixels can be explained as well by the presence of one car as by two, then the single-car hypothesis is preferred.

4.2. Multi-view Car Detection

To demonstrate our ability to recognize and segment cars in a variety of viewpoints, we experiment on images from the PASCAL 2006 challenge [2], a very difficult dataset. In Figure 10, we show some example detections and segmentations at different viewpoints and scales on test images.

Training: We trained using roughly 700 pre-segmented cars from the LabelMe database [12] and 300 cars from the PASCAL training set, which we manually segmented. We train appearance models for four viewpoint ranges (45 degrees each) with a scale range of 26 to 38 pixels tall. When estimating our instance consistency terms, we rescale the cars to 30-34 pixels tall, divide them into groups with viewpoint ranges of 15 degrees, and subdivide those into two groups according to aspect ratio (width to height of bounding box). Thus, during the refinement stage, we are able to accurately recover the viewpoint and bounding box of the cars. We set $\beta_{bg} = 4.75$ and an instance cost for each viewpoint at $\alpha * \{1000, 750, 1100, 1100\}$, where α determines the precision-recall trade-off. We set the weight of the object color term to 0.25. We have not found the algorithm to be highly sensitive to these parameters, except β_{bg}, which must be set sufficiently high to allow high recall, but low enough so that the entire image is not initially assigned to object parts (i.e., no pixels assigned to background).

Testing: To create our test set, we downsample the PASCAL car test set to 160x120 pixels and test on the first 150 images that contain cars at least 26 pixels tall. In a multi-scale search, downsampling the image in steps of $\sqrt{2}$, we process only those scales for which at least one car is between 26 and 38 pixels tall. This constraint is due to the high computational cost of our current algorithm (it still takes 1-10 minutes per image, depending on the number of initial proposals) and not due to any fundamental limitations of our approach. When searching within a 26-38 pixel scale range, we produce separate proposals for each 45-degree range of viewpoints and repeat with the mirrored the image to cover the full 360 degree range (taking advantage of car symmetry).

Results: Considering the large interclass variability, heavily occluded objects, and viewpoint and scale variation in the dataset, our quantitative and qualitative results are quite good. We achieve equal precision-recall at 61%. For reference, the highest reported results [2] in the 2006 PASCAL challenge had an equal precision-recall rate of about 45% (but note that this rate is for the full-scale test set, which is a much more difficult test). In Figure 10, we demonstrate the ability to accurately detect, segment, and determine the viewpoint of cars in a wide variety of cases. We also show several examples of failure. Often the mistakes, such as getting viewpoint wrong by 180 degrees or thinking that a double-decker bus is actually two cars are reasonable in the absence of high-level contextual information.

We also measure the value of our 3D model and of modeling the color distribution of the object. For the former, we assign a 2D grid of parts for each viewpoint range and relabel based on appearance, as is described in [16]. After learning appearance models under this part-labeling method, we then run our inference keeping other aspects of the algorithm equal (e.g., we include instance costs and the object color term). Our 3D model outperforms the 2D grid method of initial part assignment in accuracy (by about 5% recall at the equal precision-recall point of 60%), produces better segmentations, and a more precise viewpoint estimation. Similarly, including the color model improves recall by about 5% at 60% precision and improves segmentation.

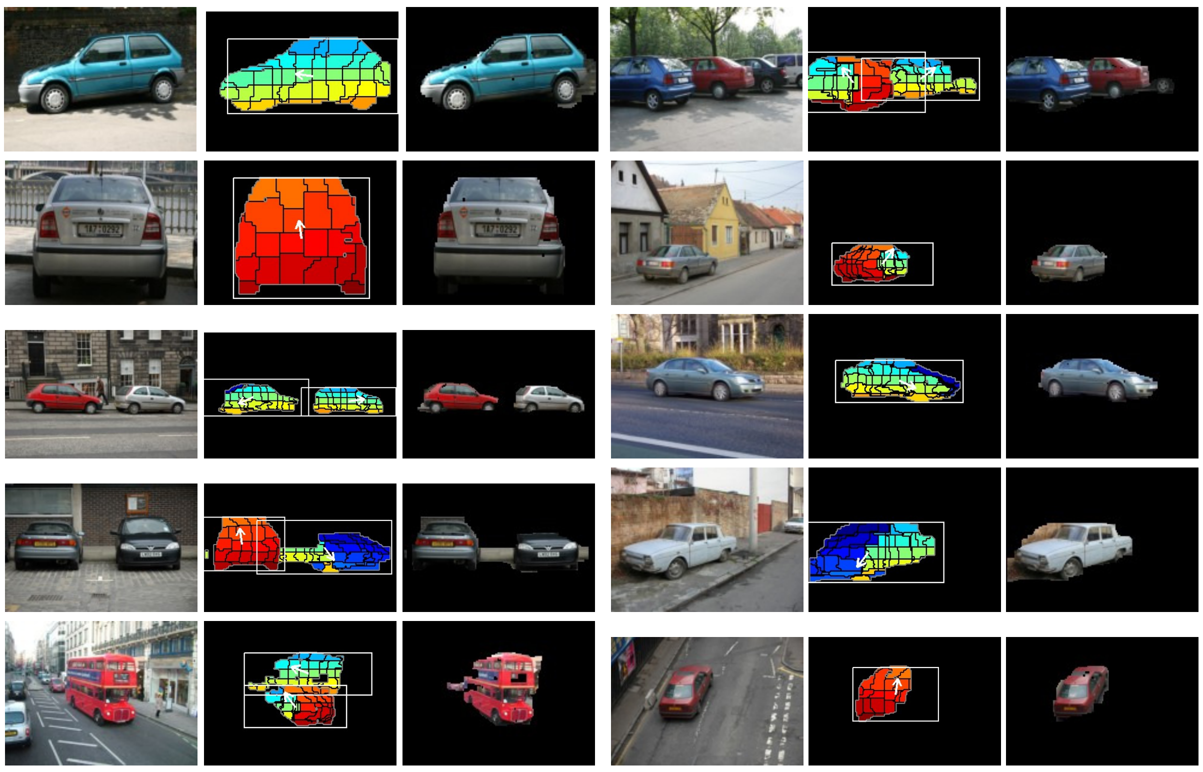

(a) Image (b) Parts/Object (c) Segmentation (d) Image (e) Parts/Object (f) Segmentation

Figure 10. Results of multi-view car detection and segmentation on test images of the challenging Pascal Dataset.

5. Discussion and Future Work

We introduce a method which we believe is the first to combine multi-viewpoint class recognition with segmentation. Our 3D LayoutCRF model makes it possible to reason about object-level properties, such as viewpoint, size and color, while also reasoning about pixel-level appearance, part consistency and occlusion. Another important contribution is an instance cost, which improves segmentation accuracy and allows non-contiguous regions to be assigned to the same object.

Some conceptually simple (but perhaps technically difficult) extensions include reducing the currently-prohibitive computational time, modeling a larger number of objects, and modeling scale dependencies (e.g., as in [3]). Major challenges include extension to non-rigid or articulated objects and integration with methods that are more appropriate for objects without a well-defined shape, such as buildings or grass.

Acknowledgements: We would like to thank Vladimir Kolmogorov for insights on the submodularity of the instance cost and for improving the TRW-S inference speed.

References

[1] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *CVPR*, 2004.

[2] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The Pascal VOC2006 results. Technical report, 2006.

[3] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006.

[4] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. PAMI*, 28(10):1568–1583, 2006.

[5] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. PAMI*, 26(2):147–159, 2004.

[6] A. Kushal and J. Ponce. Modeling 3d objects from stereo views and recognizing them in photographs. In *ECCV*, 2006.

[7] K. N. Kutulakos and S. Seitz. A theory of shape by space carving. In *TR692, Computer Science Dept., U. Rochester*. May 1998.

[8] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, June 2005.

[9] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, 2006.

[10] A. L. Ratan, W. E. L. Grimson, and I. William M. Wells. Object detection and localization by dynamic template warping. *Int. J. Computer Vision*, 36(2):131–147, 2000.

[11] C. Rother, V. Kolmogorov, and A. Blake. GrabCut -interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004*.

[12] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *MIT AI Lab Memo AIM-2005-025*, 2005.

[13] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *CVPR*, 2000.

[14] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool. Towards multi-view object class detection. In *CVPR*, 2006.

[15] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. *CVPR*, 2004.

[16] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *CVPR*, 2006.