International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Scene Parsing with Object Instance Inference Using Regions and Per-exemplar Detectors

Joseph Tighe · Marc Niethammer · Svetlana Lazebnik

Received: date / Accepted: date

Abstract This paper describes a system for interpreting a scene by assigning a semantic label at every pixel and inferring the spatial extent of individual object instances together with their occlusion relationships. First we present a method for labeling each pixel aimed at achieving broad coverage across hundreds of object categories, many of them sparsely sampled. This method combines region-level features with per-exemplar sliding window detectors. Unlike traditional bounding box detectors, per-exemplar detectors perform well on classes with little training data and high intra-class variation, and they allow object masks to be transferred into the test image for pixel-level segmentation. Next, we use per-exemplar detections to generate a set of candidate object masks for a given test image. We then select a subset of objects that explain the image well and have valid overlap relationships and occlusion ordering. This is done by minimizing an integer quadratic program either using a greedy method or a standard solver. We alternate between using the object predictions to refine the pixel labels and using the pixel labels to improve the object predictions. The proposed system obtains promising results on two challenging subsets of the LabelMe dataset, the largest of which contains 45,676 images and 232 classes.

Joseph Tighe
University of North Carolina
Department of Computer Science
Chapel Hill, NC
Tel.: 415.359.3535
E-mail: jtighe@cs.unc.edu

Marc Niethammer
University of North Carolina
Department of Computer Science
Chapel Hill, NC
Tel.: 919.590.6149
E-mail: mn@cs.unc.edu

Svetlana Lazebnik
University of Illinois
Department of Computer Science
Urbana-Champaign, IL
Tel.: 217.300.2422
E-mail: slazebni@illinois.edu

1 Introduction

This paper addresses the problem of parsing scene images in terms of their constituent object classes. Our goal is achieving *broad coverage* – the ability to recognize hundreds of classes that commonly occur in everyday outdoor and indoor environments. A major challenge in meeting this goal is posed by the non-uniform statistics of object classes in realistic scene images. In a reasonably large and diverse scene dataset such as LabelMe [29], just a handful of classes constitute a large portion of all image pixels and object instances, while a much larger number of classes occupy a small percentage of image pixels and have relatively few instances each. The more frequent classes are predominantly "stuff" [1] such as sky, roads, trees, and buildings; while the less frequent ones tend to be "things" such as trucks, parking meters, dogs, vases, etc. "Stuff" categories have no consistent shape but fairly consistent texture, so they can be adequately handled by image parsing systems based on pixel- or region-level features [4, 7, 9, 23, 25, 30–32, 39]. On the other hand, "things" are better characterized by overall shape than local appearance, so to do better on such classes, it becomes necessary to incorporate detectors that model the object shape.

The first contribution of our paper is a novel broad-coverage image parsing approach that labels every pixel

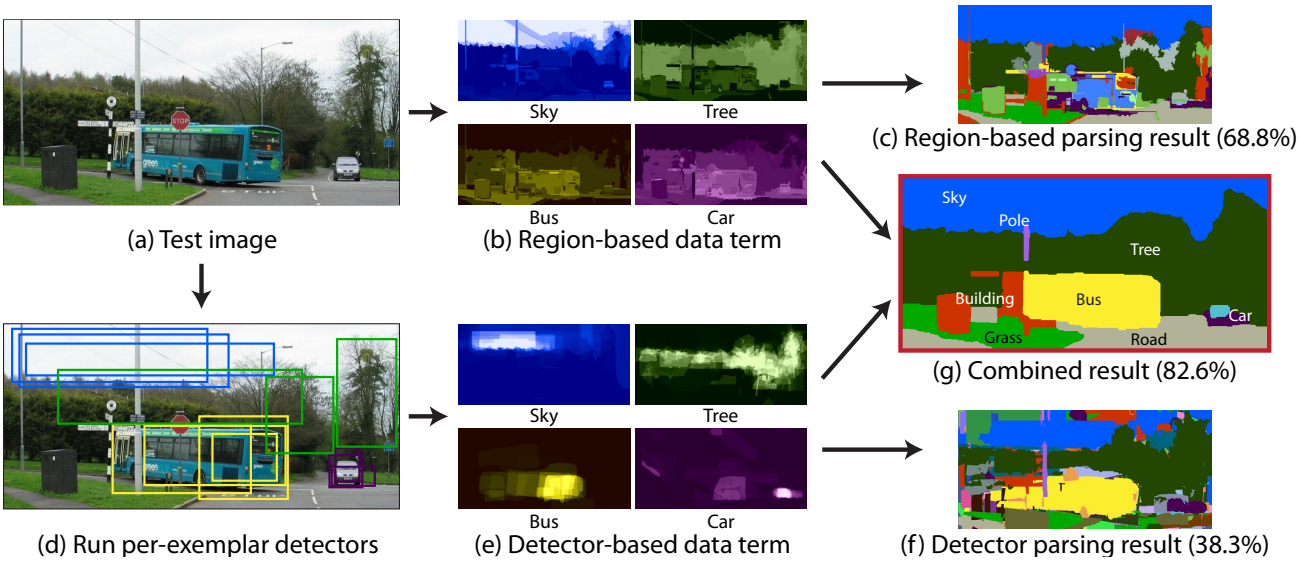

Fig. 1 Overview of our region- and detector-based pixel labeling approach (see Section 3 for details). The test image (a) contains a bus – a relatively rare "thing" class. Our region-based parsing system [32] computes class likelihoods (b) based on superpixel features, and it correctly identifies "stuff" regions like sky, road, and trees, but is not able to get the bus (c). To find "things" like bus and car, we run per-exemplar detectors [24] on the test image (d) and transfer masks corresponding to detected training exemplars (e). Since the detectors are not well suited for "stuff," the result of detector-based parsing (f) is poor. However, combining region-based and detection-based data terms (g) gives the highest accuracy of all and correctly labels most of the bus and part of the car.

with its class by combining region- and detector-based cues (see Figure 1 for an overview). We obtain the region-based cues from the superpixel-based parsing system we have developed earlier [32]. For the detector-based cues, we rely on the relatively new framework of *per-exemplar detectors* or *exemplar-SVMs* proposed by Malisiewicz et al. [24]. Specifically, we train a per-exemplar detector for each ground truth object instance in our training set. When a per-exemplar detector fires in a test image, we take the segmentation mask from the corresponding training exemplar and transfer it into the test image to form a segmentation hypothesis. We accumulate the per-exemplar detector masks for each class individually to generate a detector-based data term and combine this with the region-based data term from our earlier work [33] to produce a dense pixel labeling. While Malisiewicz et al. [24] first suggest the idea of using per-exemplar detectors to help segment objects, our work is the first to provide large-scale experimental evidence that this can be used to improve the accuracy of dense multi-class parsing.

The use of per-exemplar detectors gives us several advantages over other existing image parsing systems that rely on sliding window detectors [11,13,15,18,22,38]. First, standard detectors such as Deformable Part Models (DPMs) [10] produce only bounding box hypotheses, and inferring a pixel-level segmentation from a bounding box is an open research problem in its own right. By contrast, with per-exemplar detectors we can directly transfer ground truth masks into detected locations in the test images to obtain candidate instance segmentations. Second, traditional bounding box detectors typically require lots of training examples per category in order to achieve good performance, while per-exemplar detectors are more appropriate for sparsely sampled classes with high intra-class variation.

Unfortunately, merely labeling each pixel in the image with its class produces a scene interpretation that is impoverished and ambiguous. In particular, a pixel labeling does not directly allow us to infer object instances. For example, in Figure 2(a) it is impossible to know whether the big blob of "car" labels represents a number of overlapping cars or a single large car. As our second contribution, we extend our system to infer object instances defined by segmentation masks, as well as overlap relationships between different instances. Briefly, we start with our pixel labeling and per-exemplar detections to obtain a set of candidate object instances. Then we select a subset of instances that agree with the current pixel labeling and respect overlap and occlusion ordering constraints learned from the training data. For example, we may learn that headlights occur in front of cars with 100% overlap,[1] while

[1] Technically, headlights are attached to cars, but we do not make a distinction between attachment and occlusion in this work.

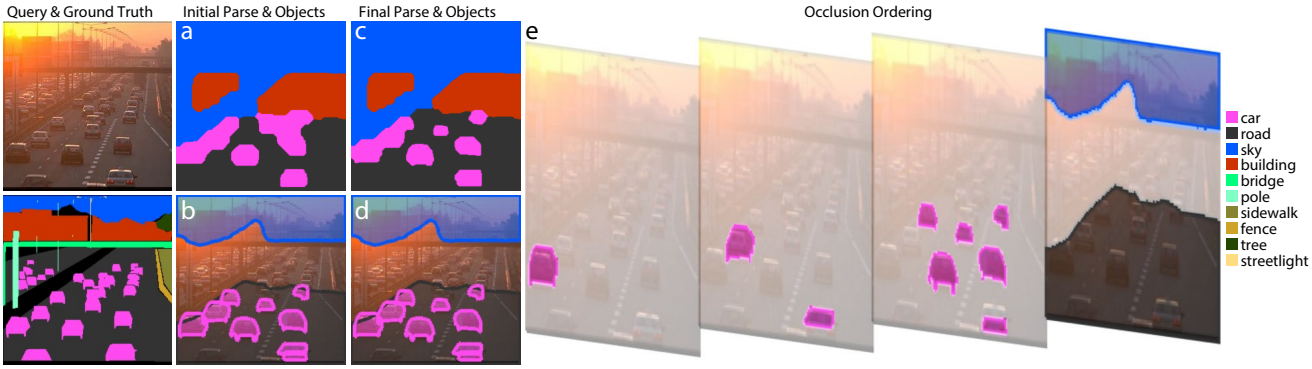

Fig. 2 Overview of our instance inference approach (see Section 4 for details). We use our region- and detector-based image parser (Figure 1) to generate semantic labels for each pixel (a) and a set of candidate object masks (not shown). Next, we select a subset of these masks to cover the image (b). We alternate between refining the pixel labels and the object predictions until we obtain the final pixel labeling (c) and object predictions (d). On this image, our initial pixel labeling contains several "car" blobs, some of them representing multiple cars, but the object predictions separate these blobs into individual car instances. We also infer an occlusion ordering (e), which places the road behind the cars, and puts the three nearly overlapping cars on the left side in the correct depth order. Note that our instance-level inference formulation does not require the image to be completely covered. Thus, while our pixel labeling erroneously infers two large "building" areas in the mid-section of the image, these labels do not have enough confidence, so no corresponding "building" object instances get selected.

cars usually overlap other cars by at most 50%. We formulate instance inference as an integer quadratic program and minimize it either using a greedy method or a standard solver. We also show how to refine the inferred pixel labels and instances by alternating between the two prediction tasks. The result is a layered scene representation as illustrated in Figure 2.

Our presentation is organized as follows. Section 2 surveys related work on image parsing and object detection. Section 3 introduces our method for combined region- and detector-based pixel-level inference, which was first published in [32]. Section 4 describes our method for instance-level inference, which has appeared in [34]. Section 5 presents an experimental evaluation of both methods. Our system produces state-of-the-art results on two challenging datasets derived from LabelMe: LMO [23] and LM+SUN [33]. In particular, LM+SUN, with 45,676 images and 232 labels, has the broadest coverage of any image parsing benchmark to date. Complete code and results for our work can be found at `http://www.cs.unc.edu/SuperParsing`.

2 Related Work

A growing number of scene interpretation methods combine pixel labeling and detection [11,13,15,18,22,35,38]. However, as mentioned in the Introduction, the detectors used by most of these methods produce only bounding box hypotheses. Ladický et al. [22], for example, augment a conditional random field (CRF) pixel labeling approach with detector-based potentials defined over object segmentation hypotheses. To obtain these hypotheses automatically from detected bounding boxes, Ladický et al. [22] use GrabCut [27], which relies only on local color cues and does not take object shape into account. Yao et al. [38] propose a more expressive CRF formulation that jointly infers image-level class labels, region labels, and bounding boxes. This approach makes use of shape prior masks corresponding to individual DPM components, but it does not explicitly associate a segmentation mask with each inferred object instance, and does not model overlap between instances. Kim et al. [18] introduce another augmented CRF for relating segment labels and bounding box hypotheses, one that is capable of producing instance-level segmentations. But while their detected bounding boxes may overlap, the segmentations cannot as their system only predicts a single label per pixel and does not model occlusion. Furthermore, the approaches of [18,22,38] rely on rather complex inference and are evaluated only on small datasets such as Stanford [12], CamVid [4] and MSRC [30], which contain hundreds of images and tens of classes, whereas we want to scale parsing to datasets consisting of tens of thousands of images and hundreds of classes.

An important aspect of our work is its ability to produce a layered scene representation. In this, it is related to the work of Guo and Hoiem [13], who infer background classes (e.g., building, road) at every pixel, even in regions where they are not visible. They learn the relationships between occluders and background classes (e.g., cars are found on the road and in front of buildings) to boost the accuracy of background prediction. We go further, inferring not only the occluded back-

ground, but all the classes and their relationships. More similarly to us, Yang et al. [37] infer object instances and their depth ordering. However, their system is aimed at foreground segmentation on the PASCAL dataset [8], which is a different task from ours and has its own system requirements. Namely, PASCAL segmentation is concerned about two dozen "thing" classes and does not require all image pixels to be labeled, whereas we are interested in explaining every pixel – both the foreground "things" and the background "stuff" – in datasets consisting of hundreds of categories. Accordingly, our system uses both segmentation regions and detectors to generate object hypotheses, while [37] uses only detectors. Since detectors do not work well on background classes, Yang et al. [37] would find it challenging to get the background pixels correct. On the other hand, they use DPMs, which are more accurate for PASCAL categories than per-exemplar detectors, and have a more sophisticated foreground segmentation model for the smaller number of classes they consider, so we could not compete with them on their task.

The recent approach of Isola and Liu [17] is perhaps the most closely related to our work both in terms of its task and its output representation. This approach parses the scene by finding a configuration or "collage" of ground truth objects from the training set that match the visual appearance of the query image. The transferred objects can be translated and scaled to match the scene and have an inferred depth order. The system of [17] can run on datasets on the same scale as the ones used in this work. However, as the experiments of Section 5 demonstrate, our system considerably outperforms theirs in terms of pixel-level accuracy on the LMO dataset.

3 Pixel Label Inference

This section presents our system for parsing images by combining region- and detector-based cues. An overview of this system is given in Figure 1. Given a test image, it applies the following sequence of steps to produce a dense pixel-level labeling:

1. Obtain a *region-based data term* from our earlier SuperParsing system [33] (Section 3.1).
2. Obtain a *detector-based data term* from transferred segmentation masks associated with per-exemplar detections (Section 3.2).
3. Combine the two data terms into a single unary potential with the help of a learned support vector machine (Section 3.3).
4. Compute the final pixel labeling by optimizing a CRF objective function based on the combined unary potential, pairwise smoothing terms, and a spatial prior (Section 3.4).

3.1 Region-Based Data Term

For region-based parsing, we use the scalable nonparametric system we have developed previously [33]. Given a query image, this system first uses global image descriptors to identify a *retrieval set* of training images similar to the query. Then the query is segmented into regions or superpixels; each region is matched to regions in the retrieval set based on 20 features representing position, shape, color, and texture; and these matches are used to produce a log-likelihood ratio score $L(s, c)$ for label c at region S:

$$L(S, c) = \log \frac{P(S|c)}{P(S|\bar{c})} = \sum_k \log \frac{P(\mathbf{f}^k|c)}{P(\mathbf{f}^k|\bar{c})}\,, \tag{1}$$

where $P(\mathbf{f}^k|c)$ (resp. $P(\mathbf{f}^k|\bar{c})$) is the likelihood of feature type k for region S given class c (resp. all classes but c). The likelihoods are given by nonparametric nearest-neighbor estimates (see [33] for details). We use this score to define our *region-based data term* E_{R} for each pixel location $\mathbf{p}$ and class c:

$$E_{\mathrm{R}}(\mathbf{p}, c) = L(S_{\mathbf{p}}, c)\,, \tag{2}$$

where $S_{\mathbf{p}}$ is the region containing $\mathbf{p}$. Refer back to Figure 1(b) for examples of some region-based data terms for the test image in Figure 1(a).

3.2 Detector-Based Data Term

Following the exemplar-SVM framework of [24], we train a per-exemplar detector for each labeled object instance in our dataset. The feature representation we use is the histogram of oriented gradients (HOG) [5] over the instance bounding box. While it may seem intuitive to only train detectors for "thing" categories, we train them for *all* categories, including ones seemingly inappropriate for a sliding window approach, such as "sky." As our experiments will demonstrate, this actually yields the best results for the combined region- and detector-based system. We follow the detector training procedure of [24], with negative mining done on all training images that do not contain an object of the same class. For our largest LM+SUN dataset we only do negative mining on 1,000 training images most similar to the positive exemplar's image (we have found that using more does not increase the detection accuracy).

At test time, given an image that needs to be parsed, we first obtain a retrieval set of globally similar training

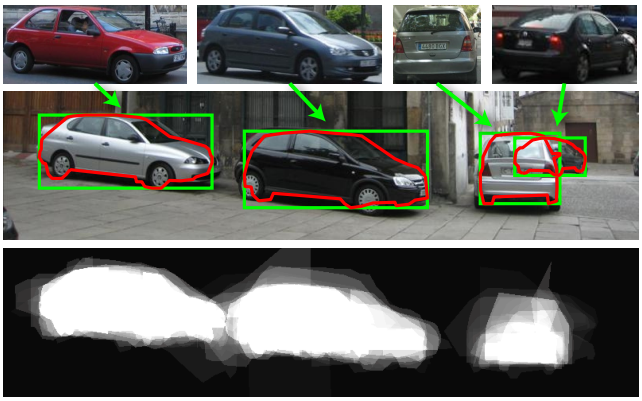

Fig. 3 Computation of the detector-based data term. For each positive detection (green bounding box) in the test image (middle row) we transfer the mask (red polygon) from the associated exemplar (top) into the test image. The data term for "car" (bottom) is obtained by summing all the masks weighted by their detector responses.

images as in Section 3.1. Then we run the detectors associated with the first k instances of each class in that retrieval set (the instances are ranked in decreasing order of the similarity of their image to the test image, and different instances in the same image are ranked arbitrarily). We restrict k purely to reduce computation; all our experiments use $k = 100$. Next, we take all detections that are above a given threshold t_d (we use the negative margin or $t_d = -1$ as suggested in [24]). For each detection we project the associated object mask into the detected bounding box (Figure 3). To compute the *detector-based data term* E_{D} for a class c and pixel $\mathbf{p}$, we simply take the sum of all detection masks from that class weighted by their detection scores:

$$E_{\mathrm{D}}(\mathbf{p}, c) = \sum_{d \in D_{\mathbf{p},c}} (w_d - t_d)\,, \tag{3}$$

where $D_{\mathbf{p},c}$ is the set of all detections for class c whose transferred mask overlaps pixel $\mathbf{p}$ and w_d is the detection score of d. Refer back to Figure 1(e) for examples of detector-based data terms for the test image of Figure 1(a).

Note that the full training framework of [24] includes calibration and contextual pooling procedures that are meant to make scores of different per-exemplar detectors more consistent. These procedures are very computationally intensive, as they require coupling detectors that are otherwise trained independently. In our implementation, we have found these steps to be unnecessary, as they are at least partially superseded by the combined region- and detector-based inference scheme described in Section 3.3.

Even without calibration, training the per-exemplar SVMs in parallel on a cluster of 512 nodes takes approximately four days for our largest dataset (see Section 5.3 for more details on running times). To reduce this training time, we have also experimented with the approach of Hariharan et al. [14] that replaces the SVM with a classifier obtained via linear discriminant analysis (LDA). In this framework, given an exemplar represented by a HOG feature vector $\mathbf{h}$, the weight vector $\mathbf{w}$ of the corresponding classifier is given by

$$\mathbf{w} = \Sigma^{-1}\,(\mathbf{h} - \mu_0)\,, \tag{4}$$

where Σ is the covariance matrix of the data and and μ_0 is the mean for windows of the same size as $\mathbf{h}$ across the training set (i.e., the background mean). At first it would seem that we would need to estimate a separate Σ and μ_0 for every possible window size, but Hariharan et al. [14] show how to compute a single mean and covariance, from which we can then extract the mean and covariance for any window size.

We can use the LDA classifiers in place of the SVMs to produce the response value w_d in eq. (3). The only modification needed is to find a new threshold t_d for our LDA setup; a grid search over the training set has yielded the value of 0.25, which we use in all subsequent experiments involving LDA. A comparison of performance of per-exemplar SVMs and LDA classifiers will be given in Section 5.1.

3.3 SVM Combination

Once we run the parsing systems of Sections 3.1 and 3.2 on a test image, for each pixel $\mathbf{p}$ and each class c we end up with two data terms, $E_{\mathrm{R}}(\mathbf{p}, c)$ and $E_{\mathrm{D}}(\mathbf{p}, c)$, as defined by eqs. (2) and (3). For a dataset with C classes, concatenating these values gives us a $2C$-dimensional feature vector at each pixel. Next, we train C one-vs-all SVMs, each of which takes as input the $2C$-dimensional feature vectors and returns final per-pixel scores for a given class c.

Training data for each SVM is generated by running region- and detector-based parsing on the entire training set using a leave-one-out method: for each training image a retrieval set of similar training images is obtained, regions are matched to generate $E_{\mathrm{R}}(\mathbf{p}, c)$, and the per-exemplar detectors from the retrieval set are run to generate $E_{\mathrm{D}}(\mathbf{p}, c)$. Unfortunately, the resulting amount of data is huge: our largest dataset has over nine billion pixels, which would require 30 terabytes of storage. To make SVM training feasible, we must subsample the data – a tricky task given the unbalanced class frequencies in our many-category datasets.

We could subsample the data uniformly (i.e., reduce the number of points by a fixed factor without

regard to class labels). This preserves the relative class frequencies, but in practice it heavily biases the classifier towards the more common classes. Conversely, sub-sampling the data so that each class has a roughly equal number of points produces a bias towards the rare classes. We have found that combining these two schemes in a 2:1 ratio achieves a good trade-off on all our datasets. That is, we obtain 67% of the training data by uniform sampling and 33% by even per-class sampling. We train all SVMs on 250,000 points – using more did not significantly improve performance for any of our setups.

For training one-vs-all SVMs, we normalize each feature dimension by its standard deviation and use fivefold cross-validation to find the regularization constant. Another important aspect of the implementation is the choice of the SVM kernel. As will be shown in Section 5.1, the linear kernel already does quite well, but we can obtain further improvements with the RBF kernel. Since it is infeasible to train a nonlinear SVM with the RBF kernel on our largest dataset, we approximate it by training a linear SVM on top of the random Fourier feature embedding [26]. We set the dimensionality of the embedding to 4,000 and find the kernel bandwidth using fivefold cross-validation. Experiments on the LMO dataset in Section 5.1 (Table 3) will confirm the quality of the approximation.

The resulting SVMs produce C responses at each pixel. Let $E_{\text{SVM}}(\mathbf{p}_i, c)$ denote the response of the SVM at pixel $\mathbf{p}_i$ for class c. To obtain the final labeling, we could simply take the highest-scoring class at each pixel, but this is well known to produce noisy results. Instead, we smooth the labels using a CRF objective function, as described next.

3.4 CRF Smoothing

We compute a global pixel labeling of the entire image by minimizing a CRF objective function defined over the field of pixel labels $\mathbf{c}$:

$$E(\mathbf{c}) = \sum_i \underbrace{\psi_{\text{u}}(\mathbf{p}_i, c_i)}_{\text{unary (eq.6)}} + \alpha \sum_i \underbrace{\psi_{\text{sp}}(\mathbf{p}_i, c_i)}_{\text{spatial prior (eq.7)}} + \sum_{(i,j)\in\mathcal{N}} \underbrace{\psi_{\text{sm}}(c_i, c_j)}_{\text{smoothing (eq.8 or 9)}} \tag{5}$$

where i indexes the image pixels and $\mathcal{N}$ is a set of edges determining pairwise clique relationships (we have experimented with two different neighborhood structures, as described below).

The unary potential $\psi_{\text{u}}(\mathbf{p}_i, c_i)$ is obtained from the SVM response for pixel $\mathbf{p}_i$ and class c_i:

$$\psi_{\text{u}}(\mathbf{p}_i, c_i) = -\log \sigma(E_{\text{SVM}}(\mathbf{p}_i, c_i)), \tag{6}$$

where $\sigma(t) = 1/(1+e^{-t})$ is the sigmoid function turning the raw SVM output into a probability-like value.

The spatial term $\psi_{\text{sp}}(\mathbf{p}_i, c_i)$, similar to the one from [23], represents the prior probability that a class c_i occurs at pixel $\mathbf{p}_i$, estimated from counts of labels at each location across the training set:

$$\psi_{\text{sp}}(\mathbf{p}_i, c_i) = -\log \text{hist}_{c_i}(\mathbf{p}_i) \tag{7}$$

where hist_{c_i} is the spatial histogram for class c_i over the training set, normalized to sum to one at each pixel. We have found that including the spatial prior improves our per-pixel labeling accuracy slightly (typically by less than 1%) without smoothing away any of the smaller or rarer classes. The constant α is set to 0.1 by grid search over the LMO dataset.

For the pairwise spatial smoothing terms ψ_{sm} we experiment with two different CRF graph structures: an 8-connected graph and a fully connected graph. For the 8-connected one, only adjacent pairs of pixels i, j are connected by edges and we define the smoothing penalty similarly to [23,30]:

$$\psi_{\text{sm}}(c_i, c_j) = \lambda \left(1_{\{c_i \neq c_j\}}\right) \exp\left(-\gamma ||\mathbf{I}_i - \mathbf{I}_j||^2\right) \tag{8}$$

where $1_{\{c_i \neq c_j\}}$ is 0 when $c_i = c_j$ and 1 otherwise; $\mathbf{I}_i$ and $\mathbf{I}_j$ are the RGB color values for pixels $\mathbf{p}_i$ and $\mathbf{p}_j$ and $\gamma = (2\langle ||\mathbf{I}_i - \mathbf{I}_j||^2 \rangle)^{-1}$, where $\langle\cdot\rangle$ denotes the average over the training images as suggested by [27]. The constant λ controls the amount of smoothing; we set $\lambda = 16$ by grid search. Inference is performed using α-expansion [3,20]. This was the setup used in our original paper [32] and works quite well but does exhibit some metrication artifacts [2]: namely, the boundaries between different classes tend to step between vertical, horizontal, and 45-degree angles as they attempt to follow the graph structure (Figure 4).

To produce a smoothed labeling without the metrication artifacts, we have experimented with an alternative fully connected, or dense, CRF of Krahenbuhl and Koltun [21]. For this setup, we use the contrast-sensitive two-kernel potentials from [21]:

$$\begin{aligned} \psi_{\text{sm}}(c_i, c_j) &= 1_{\{c_i \neq c_j\}} \left(k^{(1)}(\mathbf{p}_i, \mathbf{p}_j) + k^{(2)}(\mathbf{p}_i, \mathbf{p}_j)\right), \\ k^{(1)}(\mathbf{p}_i, \mathbf{p}_j) &= w^{(1)} \exp\left(-\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\theta_\alpha^2} - \frac{||\mathbf{I}_i - \mathbf{I}_j||^2}{2\theta_\beta^2}\right), \\ k^{(2)}(\mathbf{p}_i, \mathbf{p}_j) &= w^{(2)} \exp\left(-\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\theta_\gamma^2}\right), \end{aligned} \tag{9}$$

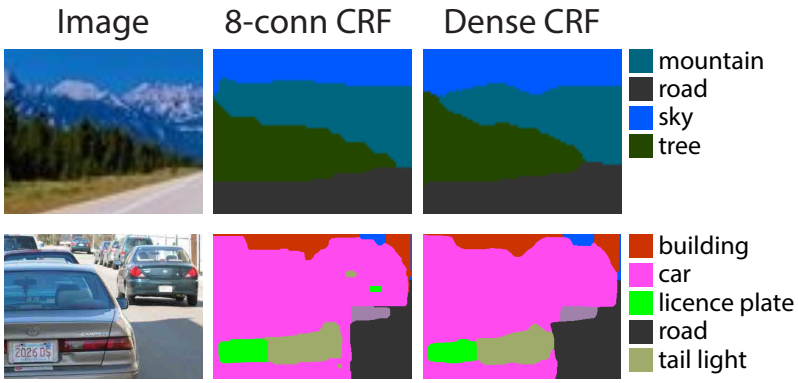

Fig. 4 Qualitative comparison of 8-connected and dense pairwise potentials in our CRF. Notice how the edges produced by 8-connected CRF step between vertical, horizontal, and 45-degree angles as they attempt to follow the graph structure, especially in the top photo. This phenomenon is known as metrication and is avoided by the dense CRF.

where $||\mathbf{p}_i - \mathbf{p}_j||$ is the Euclidean distance between pixel locations $\mathbf{p}_i$ and $\mathbf{p}_j$. We find the values of the parameters by grid search on the training set of the LMO dataset: $w^{(1)} = w^{(2)} = 4$, $\theta_\alpha = 15$, $\theta_\beta = 20$ and $\theta_\gamma = 3$ for all experiments. To perform CRF inference on the fully connected graph, we use the efficient message passing code of [21], which is at least as fast as the α-expansion that we use for the 8-connected CRF.

A comparative evaluation of the 8-connected and fully connected CRF models will be presented in Section 5.1, Table 4. In the end, despite the qualitatively superior results of the fully connected CRF, we could not find a set of parameters that could always outperform the 8-connected CRF in terms of pixel accuracy. This is likely due to the tendency of the dense CRF to smooth away small objects like the license plate and tail light of the right car in Figure 4.

4 Instance-Level Inference

We now wish to find a set of object instance masks and their occlusion ordering that can provide a richer interpretation of the scene. For an overview of this process, refer back to Figure 2. The steps for performing instance-level inference on a test image are as follows:

1. Generate a set of candidate instance masks based on per-exemplar detections and two variants of our pixel labeling (Section 4.1).
2. For each candidate instance, define a score indicating its "quality" or degree of agreement with the pixel labeling. For each pair of overlapping instances, determine whether the overlap relationship is valid based on learned co-occurrence statistics (Section 4.2).
3. Solve a quadratic integer program to select a subset of candidates that produce the highest total score while maintaining valid pairwise overlap relationships (Section 4.3).
4. Use a simple graph-based algorithm to recover a depth or occlusion ordering for the selected instances (Section 4.4).
5. Based on the inferred occlusion ordering, define an object potential that augments the CRF model of eq. (5) and recompute a pixel labeling that better agrees with the instance-level image interpretation. Alternate between using the updated pixel labeling to further refine the instance inference and vice versa (Section 4.5).

4.1 Candidate Instance Generation

This section explains how we generate object instance hypotheses using the pixel-level parsing framework of Section 3. First, it is worth pointing out the different nature of hypotheses for "thing" and "stuff" classes. For "things" like cars, people, and fire hydrants, instances are discrete and well-defined. It is highly desirable to correctly separate multiple "thing" instances even when (or *especially* when) they are close together or overlapping. On the other hand, for "stuff" classes such as buliding, roads, trees, and sky, the notion of instances is a lot looser. In some cases, it may be possible to identify separate instances of buildings and trees, but most of the time we are content to treat areas occupied by these classes as undifferentiated masses. Accordingly, we manually separate all classes in our datasets into "things" and "stuff" and use different procedures to generate candidates for each (the full split is available on our project website).

For "things," we get candidates from per-exemplar masks transferred during the computation of the detector-based data term (Section 3.2). As shown in Figure 5,

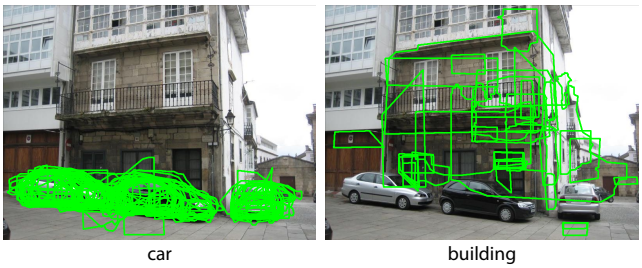

Fig. 5 Candidate instance masks obtained from per-exemplar detections. There are many good "car" masks but no good "building" ones. Accordingly, we take the per-exemplar masks as our candidate "thing" instances but use a different procedure for "stuff" (see text).

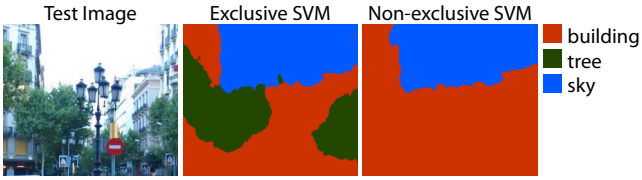

Fig. 6 "Stuff" parsing using both exclusive and non-exclusive SVMs. The exclusive SVM generates good "tree" candidates, while the non-exclusive one generates a "building" candidate without holes. The connected components of both parses give us our "stuff" hypotheses (see text).

these masks make fairly good candidates for "things" such as cars, but for "stuff" such as buildings, they tend to be overly fragmented and poorly localized.

To get "stuff" hypotheses that better fit the boundaries in the image, one might think of simply taking the connected components of the corresponding class labels from the initial pixel-level parse. However, the resulting masks may have artifacts due to occlusions. For example, a scene may have a building with trees in front of it (Figure 6), and if we simply pick out the "building" pixels from the initial parse, we will end up with a hypothesis that has tree-shaped holes in it. Instead, we would like to compute a layered representation of the image reflecting the fact that both "building" and "tree" coexist in the same location, with the latter being in front of the former. Fortunately, our datasets come from LabelMe where the ground truth consists of overlapping object polygons, and it allows us to learn about such relationships [28].

The initial pixel-level parse is determined primarily by the combination SVM data term E_{SVM} (Section 3.3). By default, the combination SVM is trained to assign to each pixel only the visible or front-most object label. For example, if a training pixel is contained within "headlight," "car," and "building" ground-truth polygons, it is used as training data only for the front-most "headlight" label. To generate better "stuff" hypotheses, we want a high response to background (occluded) classes as well. So we also train a second *non-exclusive* combination SVM where each pixel is considered a positive example of *every* ground-truth polygon that contains it. We denote the output of this combination SVM as $E_{\text{NXSVM}}(\mathbf{p}_i, c_i)$ and the output of the original exclusive one as $E_{\text{XSVM}}(\mathbf{p}_i, c_i)$. A comparison of the two is shown in Figure 7.

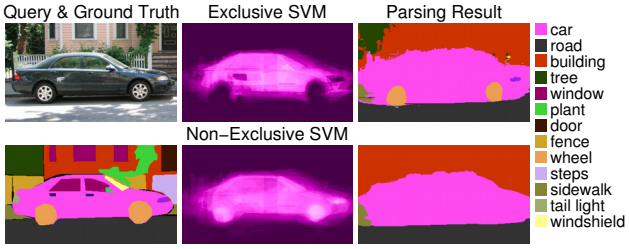

Fig. 7 Exclusive vs. non-exclusive SVM output. The exclusive SVM has a very low response for areas of the car that have attached objects, like the wheels. This would penalize candidate object masks that include the wheel pixels. By contrast, the non-exclusive SVM has a high response for all parts of the car, and the resulting pixel labeling favors classes with larger area, which typically correspond to background or occluded classes.

We produce two pixel-level parses using $E_{\text{XSVM}}(\mathbf{p}_i, c_i)$ and $E_{\text{NXSVM}}(\mathbf{p}_i, c_i)$ in the unary of eq. (6), respectively, and restricting the labels to "stuff." The connected components of these parses give us our candidate "stuff" masks. Taking out the "things" ensures that they do not occlude the "stuff" hypotheses. The exclusive parse favors the foreground "stuff" objects, while the non-exclusive one favors the larger background "stuff" objects as the smaller occluding or attached objects tend to have equal or lower SVM responses and hence get smoothed away.

The "stuff" hypotheses from the two parses are slightly redundant – in particular, in the example of Figure 6 we get two "building" hypotheses, one with a hole where it is occluded by a tree and one without. In the next section, we will describe a scoring scheme that will enable us to select the appropriate combination of hypotheses – the building with no hole and the tree in front.

4.2 Instance Scores and Overlap Constraints

In order to perform instance-level inference (Section 4.3), we need to assign a score to each candidate instance indicating its "quality" or degree of agreement with the initial pixel labeling (with a higher score indicating a better "quality"), as well as to define overlap constraints between pairs of instances. These components will be described in the current section.

We have found a unified scoring scheme that works equally well for both "things" and "stuff." First, given instance m with class label c_m and segmentation mask O_m, we define an *unweighted* score $\hat{s}_m$ as the sum of (shifted) non-exclusive SVM scores for its class label and every pixel $\mathbf{p}_i$ inside its segmentation mask:

$$\hat{s}_m = \sum_{\mathbf{p}_i \in O_m} (1 + E_{\text{NXSVM}}(\mathbf{p}_i, c_m)) \,. \tag{10}$$

Since any object with a negative score will be rejected by the optimization framework of Section 4.3, we move the effective decision boundary of the SVM to the negative margin so that as many reasonable candidate objects are considered as possible. It is important to use the *non-exclusive* SVM because we do not want to penalize an instance due to other classes that may occlude it. In the example of Figure 7, if we were to use the exclusive SVM to score a candidate car mask, the wheel regions would be penalized and a worse mask without the wheels would likely be preferred. Note, however, that we will still need the exclusive SVM to update pixel-level labels based on the inferred instances (Section 4.5), since the non-exclusive SVM tends to prefer the larger and more common classes as explained earlier.

So far, the scores defined by eq. (10) depend only on the SVM data term, which is computed once in the beginning and never modified. To allow the selected instances to iteratively modify the pixel labeling and vice versa, the score also needs to depend on the current labeling. To this end, we weight the score of each instance by the proportion of its area that is in agreement with this labeling. Given an instance m with class label c_m and pixel $\mathbf{p}_i$ with label c_i inside the instance mask, we define a *pixel-level agreement flag* as follows:

$$V_{\text{p}}(c_m, c_i) = \begin{cases} 1 & \text{if the instance label } c_m \text{ can appear} \\ & \textit{behind} \text{ the pixel label } c_i; \\ 0 & \text{otherwise.} \end{cases}$$

For example, a candidate "car" instance is in agreement with "headlight," "window," and "license plate" pixels because all these are often seen in front of "car"; it is not in agreement with "building" or "road" pixels because cars occur *in front of* these classes. On the other hand, note that a "building" or "road" instance *is* in agreement with a "car" pixel. Thus, the V_{p} flag reflects not just attachment, but more general overlap or occlusion relationships.

We learn the pixel-level agreement flags from the co-occurrence statistics of labels in the training set. As stated earlier, our training data comes from the LabelMe database, whose annotations are in the form of possibly overlapping instance polygons labeled with their categories. These overlap relationships are a surprisingly rich and consistent source of information that can be used to infer a variety of 3D relationships and even to reconstruct a scene [28]. For example, two partially overlapping polygons usually indicate that one occludes the other, and small polygons contained within large ones indicate attached objects [29]. To obtain the matrix of V_{p} values from these annotations, for each pair of class labels c and c', we count the number of pixels in the training set for which a polygon of class c is behind[2] a polygon of class c', normalize each count by the total count for the less common class, and set $V_{\text{p}}(c, c')$ to 1 if the resulting value is above a very low threshold (0.001 in the implementation).

The final score for an instance m is given by weighting $\hat{s}_m$, the total SVM score for its class c_m (eq. 10), by the proportion of the pixels in its mask O_m whose labels are in agreement with c_m:

$$s_m = w_m \hat{s}_m \,, \quad \text{where} \quad w_m = \frac{1}{|O_m|} \sum_{\mathbf{p}_i \in O_m} V_{\text{p}}(c_m, c_i) \,. \tag{11}$$

Our instance-level inference, which will be described next in Section 4.3, will select a combination of instance hypotheses that maximize the sum of the s_m scores while requiring that every pair of selected instances have a valid overlap relationship. For this inference, the overlap relationship needs to be defined differently than the pixel-level agreement flag V_{p} above. V_{p} only determines the compatibility between the class labels of an instance mask and a pixel inside that mask for the purpose of individual instance scoring. What we need now is an indicator that tells us whether two *instance masks* with given class labels are allowed to overlap. In order to get sufficiently expressive constraints, in addition to looking at the two labels, it is important to look at their amount of overlap, because only certain kinds of overlap make sense for certain pairs of classes: for example, two different "car" instances should not be almost coincident, while a "headlight" instance must be contained almost entirely within a "car" instance. Given two instances m and n with classes c_m and c_n and masks O_m and O_n, we quantify their overlap by the score

$$os_{mn} = \frac{|O_m \cap O_n|}{\min(|O_m|, |O_n|)} \,. \tag{12}$$

We use this definition instead of the more standard intersection-over-union (I/U) in order to have a con-

[2] To determine depth ordering from polygon annotations, we use the *LMsortlayers* function from the LabelMe toolbox, which takes a collection of polygons and returns their depth ordering.

sistent score for attached objects. If we use I/U, a large window hypothesis partially overlapping with a building can have the same score as a small window hypothesis with full overlap. Now we can define an *object-* or *instance-level* agreement flag between m and n as follows:

$$V_{\text{o}}(m,n) = \begin{cases} 1 & \text{if } c_m \text{ can appear } \textit{either behind or} \\ & \textit{in front of } c_n \text{ with an overlap score} \\ & \text{of } os_{mn}; \\ 0 & \text{otherwise.} \end{cases}$$

To compute these flags, for all pairs of labels (c, c'), we use the training set to build a histogram $H(c, c', os)$ giving the probability for an instance with class c to be *behind* an instance with class c' with overlap score of os (quantized into ten bins). Given candidate instances m and n from the test image, we then determine whether their relationship is valid by thresholding the maximum of the histogram entries corresponding to both orderings:

$$V_{\text{o}}(m,n) = 1_{[\max(H(c_m,c_n,os_{mn}),H(c_n,c_m,os_{mn}))>t]}\,. \quad (13)$$

Again we set a fairly conservative threshold of $t = 0.001$.

The reader may have noticed that V_{o} is symmetric in the two instances m and n, while H is not. The reason is that V_{o} is needed for the instance-level inference formulation of the next section, where the focus is only on selecting a subset of candidate instances with valid overlaps, without the added complexity of ordering them. For this, symmetric overlap constraints suffice. As will be described in Section 4.4, occlusion ordering will then be done as a post-process on the output of instance-level inference, and at that point, the directional histograms H will be used.

4.3 Instance-level Inference

This section shows how to select a subset of candidate instances that has the highest total score while maintaining valid overlap relationships. Let $\mathbf{x}$ denote a binary vector each of whose entries x_m indicates whether the mth candidate should be selected as part of our scene interpretation. We infer $\mathbf{x}$ by optimizing the following objective function:

$$\begin{aligned} &\arg\max_{\mathbf{x}} \sum_m s_m x_m - \sum_{n\neq m} 1_{\{c_m=c_n\}} s_{mn} x_m x_n \\ &\text{s.t. } \forall(x_m=1, x_n=1, n\neq m) \;\; V_{\text{o}}(m,n) = 1\,, \end{aligned} \quad (14)$$

where c_m and c_n are the class labels of m and n, s_m is the instance score from eq. (11), and s_{mn} is is defined for any two overlapping objects of the same class exactly the same as s_m, but over the intersection of the respective masks. By adding the s_{mn} term for any two selected instances of the same class, we avoid double-counting scores and ensure that any selected instance has sufficient evidence outside of regions overlapping with other instances. Without this term, we tend to get additional incorrect instances selected around the borders of correct ones.

Eq. (14) can be rearranged into an integer quadratic program:

$$\arg\min_{\mathbf{x}} \mathbf{x}^T Q\mathbf{x} - \mathbf{s}\mathbf{x} \quad \text{s.t.} \quad A\mathbf{x} < \mathbf{b}\,, \quad (15)$$

where $\mathbf{s}$ is a vector of all instance scores s_m and $Q(m,n) = 1_{\{c_m=c_n\}} s_{mn}$. The constraint matrix A is constructed by adding a row for each zero entry $V_{\text{o}}(m,n)$ (that is, each pair of instances with invalid overlap). This row consists of all zeros except for elements m and n which are set to 1. Finally, $\mathbf{b}$ is a vector of ones with the same number of rows as A. With this encoding, we cannot select two candidate objects with invalid overlap, as it would result in a row of $A\mathbf{x}$ being larger than 1.

To infer the labels $\mathbf{x}$ we adopt two methods. The first is greedy inference that works by adding one candidate instance at a time. In each round, it searches for the instance whose addition will produce the lowest energy (eq. 15) while still respecting all overlap constraints. This continues until each remaining instance either increases the energy or violates a constraint. This method is very fast and works well. The second method is to use the integer quadratic programming solver CPlex [16]. It consistently gives instance configurations with lower energies, but is much slower than greedy inference and cannot handle as many candidates. Figure 8 compares the two solutions on fragments of two test images; all the subsequent figures will show only the output of CPlex as it achieves a slightly higher accuracy overall (see Section 5.2 for the experimental evaluation). Finally, note that prior to running the instance-level inference, we reject any instance less than 10% of whose pixels have the same label as the current pixel parse $\mathbf{c}$.

4.4 Occlusion Ordering

The instances selected by the above optimization process are guaranteed to have valid overlap relationships, but for each overlapping pair we do not yet know which one occludes the other. To determine this ordering, we build a graph with a node for each selected mask O_m. For each pair of overlapping masks O_m and O_n we add a pair of edges: one from m to n with edge weight equal to $H(c_m, c_n, os_{mn})$ and one from n to m with weight

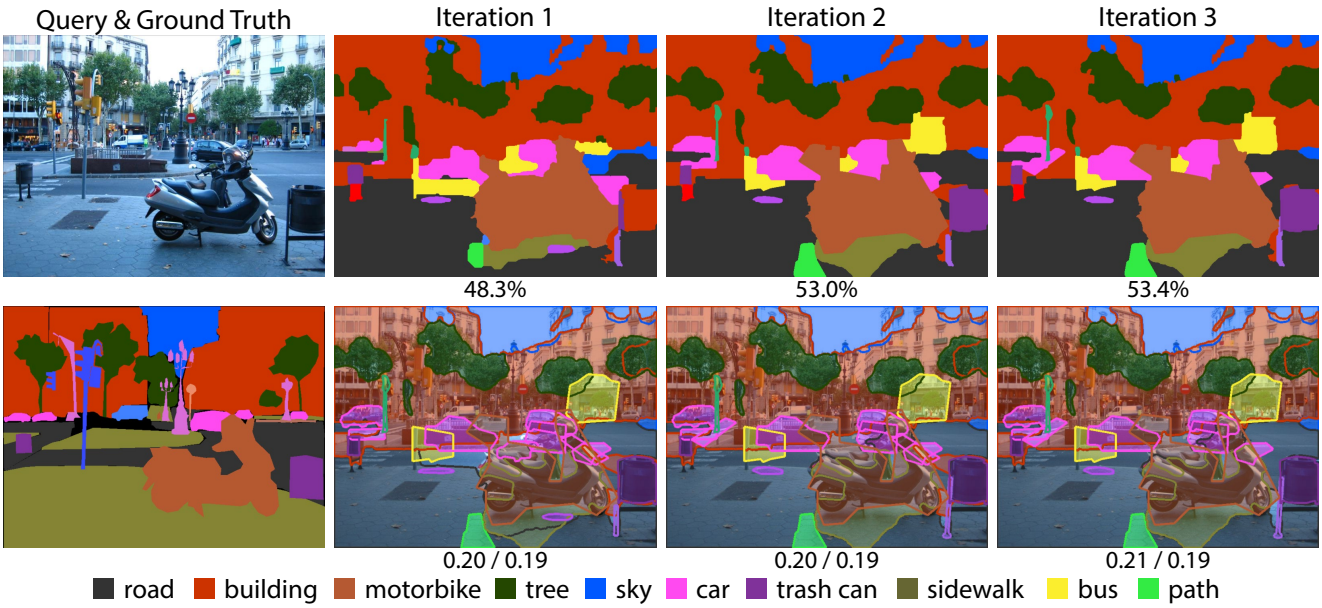

Fig. 9 Output of pixel-level and instance-level inference at each iteration. For pixel labeling (top row) we show the per-pixel rate underneath each image and for instance predictions (bottom row) we show Object P/R (Section 5). From iteration 1 to 2, we correct the pixel labeling of the trashcan on the right; from iteration 2 to 3 minor details in the background get refined. Based on the initial pixel labeling, two bus instances get predicted, and these unfortunately stay.

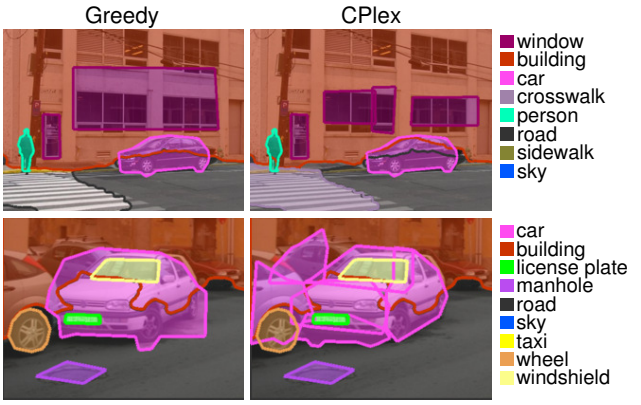

Fig. 8 A comparison of our two instance-level inference methods. In the first row, greedy inference combines six windows into one. In the second row, quadratic programming over-segments the foreground car, but greedy inference misses the two occluded cars in the background.

$H(c_n, c_m, os_{mn})$. These respective edges represent the situation that object m is behind object n and vice versa. For objects from the same class we weight the edges by the object size so that larger objects are favored to be in front of smaller ones.

To infer the occlusion order we now need to remove one edge from each pair to generate a directed acyclic graph with the highest edge weights possible. To do so, we remove the edge with the smaller weight for each pair, and check whether there are cycles in the resulting graph. If there are, we pick a random cycle and swap the edge pair with the smallest difference in weight. We continue this until there are no more cycles. Finally, we perform a topological sort of the directed acyclic graph to assign a depth plane to each object, which is useful for visualization, like in Figure 2(e).

4.5 Pixel Label Inference with Object Potentials

Once we have the selected object hypotheses and their occlusion ordering, we can close the loop between instance-level and pixel-level inference. Specifically, we obtain an updated pixel-level labeling by augmenting the CRF of eq. (5) with *object potentials* ψ_o:

$$E(\mathbf{c}) = \sum_i \underbrace{\psi_u(\mathbf{p}_i, c_i)}_{\text{unary}} + \sum_i \underbrace{\psi_o(\mathbf{p}_i, c_i, \mathbf{x})}_{\text{object potential}} + \alpha \sum_i \underbrace{\psi_{sp}(\mathbf{p}_i, c_i)}_{\text{spatial prior}} + \sum_{(i,j)\in\mathcal{N}} \underbrace{\psi_{sm}(c_i, c_j)}_{\text{smoothing}} . \quad (16)$$

The object potential is defined as follows:

$$\psi_o(\mathbf{p}_i, c_i, \mathbf{x}) = \begin{cases} 0 & \text{if no selected instance in } \mathbf{x} \text{ contains } \mathbf{p}_i; \\ -\log(0.5+\beta) & \text{if the front-most instance containing } \mathbf{p}_i \text{ has class } c_i; \\ -\log(0.5-\beta) & \text{otherwise.} \end{cases}$$

The constant β determines the amount by which the object potentials modulate the unaries, and it is set to

0.1 in our implementation. This formulation is similar to that of [19].

We alternate between inferring object instances (eq. 15) and pixel labels (eq. 16) until neither one changes. This process tends to converge in three rounds and after the first round the results are already close to final. Figure 9 shows three rounds of alternating inference for a fairly complex scene.

5 Evaluation

Section 5.1 will present an evaluation of the pixel-level labeling system introduced in Section 3, and Section 5.2 will present an evaluation of the instance-level inference introduced in Section 4. Section 5.3 will conclude by analyzing the training- and test-time requirements of our systems.

We evaluate our approaches on two subsets of LabelMe [29]. The first dataset, **LMO** [23], consists of outdoor scenes. It has 2,488 training images, 200 test images, and 33 classes. For this dataset, we use retrieval set size of 200. Our second dataset, **LM+SUN** [33], was collected from the SUN dataset [36] and LabelMe [29]. It contains 45,676 images (21,182 indoor and 24,494 outdoor) and 232 classes. We use the split of [33], which consists of 45,176 training and 500 test images. Since this is a much larger dataset, we set the retrieval set size to 800.

5.1 Pixel-Level Inference Evaluation

In this section, we evaluate our hybrid region- and detector-based pixel labeling approach (Section 3). As in our previous work [33], we evaluate pixel labeling accuracy by reporting the overall per-pixel rate (percent of test set pixels correctly labeled) and the average of per-class rates. The two measures complement each other, as the former one is dominated by the more common "stuff" classes, while the latter one is dominated by the rarer "thing" classes.

We begin by systematically examining the impact of individual implementation components, including region- and detector-based data terms, per-exemplar detectors (exemplar-SVM vs. LDA), different kernels for the combination SVM, and variants of the CRF objective function.

First, one may wonder whether the power of our approach truly lies in combining region- and detector-based cues, or whether most of our performance gain comes from the extra layers of combination SVM (Section 3.3) and CRF inference (Section 3.4). Table 1 shows the performance of various combinations of region- and detector-based data terms with and without training of SVMs on top of them, with and without CRF smoothing. The region-based data term obtains higher per-pixel accuracy than the detector-based one on both datasets, and higher per-class accuracy on LMO. On the LM+SUN dataset, which has the largest number of rare "thing" classes, the detector-based data term actually obtains higher per-class accuracy than the region-based one. While the combination SVM can sometimes improve performance when applied to the individual data terms, applying it to their combination gives by far the biggest and most consistent gains. CRF inference on top of any data term further raises the per-pixel rate, but often lowers the per-class rate by smoothing away some of the smaller objects (this is consistent with our earlier observations [33]).

Because part of our motivation for incorporating detectors is to improve performance on the "thing" classes, we want to know what will happen if we train detectors only on them – if detectors are completely inappropriate for "stuff" classes, then this strategy may improve accuracy, not to mention speed up the system considerably. The "Region + Thing" section of Table 1 shows the performance of the combination SVM trained on the full region-based data term and the subset of the detector-based data term corresponding only to "things" (the "things" vs. "stuff" splits are specified manually and are available on our results website). Interestingly, the results for this setup are weaker than those of the full combined system. Thus, even the relatively noisy "stuff" detector responses still provide useful information to the pixel-level inference task, though for instance-level inference, we could not derive any useful candidates from the individual "stuff" detections, as discussed in Section 4.1.

Figures 10 and 11 show the per-class rates of our system on the most common classes in the LMO and LM+SUN datasets for region-based, detector-based, and combined parsing. We can see that adding detectors significantly improves many "thing" classes (including car, sign, and balcony), but also some "stuff" classes (road, sea, sidewalk, fence).

Next, Table 2 looks at the effect of replacing the exemplar-SVM detector training framework by the faster LDA-based one. We can see that the LDA detectors are fairly comparable to the exemplar-SVMs in terms of accuracy, which is promising since they reduce the training time by a factor of a hundred (see Section 5.3). The most significant disadvantage of the LDA detectors seems to be on the rare classes, especially on the LM+SUN dataset, where they miss many of the less common classes the exemplar-SVM detectors do get. This is likely because the covariance computed across

	LMO		LM+SUN	
	Per-Pixel	Per-Class	Per-Pixel	Per-Class
Detector ML	65.1	25.8	33.0	14.1
Detector SVM	62.5	25.4	46.1	12.0
Detector SVM CRF	71.1	26.7	52.5	11.3
Region ML	74.1	30.2	51.5	7.5
Region SVM	75.0	35.9	56.3	6.7
Region SVM CRF	77.7	32.8	58.3	5.9
Region + Thing SVM	74.4	36.9	58.5	14.1
Region + Thing SVM CRF	77.5	35.7	60.0	12.9
Region + Detector SVM	75.6	**41.1**	59.6	**15.5**
Region + Detector SVM CRF	**78.6**	39.2	**61.4**	15.2

Table 1 Comparison of different data terms in the pixel labeling system of Section 3. All results here are obtained using exemplar-SVM detectors, combination SVMs with the approximate RBF embedding, and 8-connected CRFs. "Detector ML" and "Region ML" simply assign to each pixel $\mathbf{p}_i$ the labels c that maximize $E_{\mathrm{D}}(\mathbf{p}_i, c)$ and $E_{\mathrm{R}}(\mathbf{p}_i, c)$, respectively. "Detector SVM" and "Region SVM" obtain the E_{SVM} score by training one-vs-all SVMs on top of E_{D} and E_{R} individually (instead of concatenating them). "Region + Thing SVM" trains the combination SVM on the full E_{R} concatenated with the subset of E_{D} corresponding to only the "thing" classes. Finally, "Region + Detector SVM" trains the SVM on the full concatenation of E_{R} and E_{D}, as described in Section 3.3.

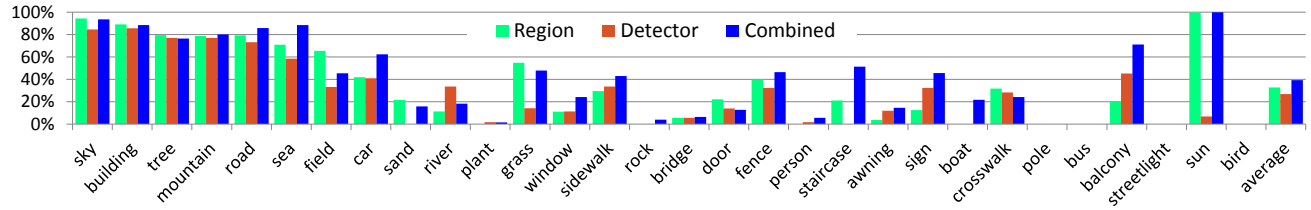

Fig. 10 Classification rates of individual classes (ordered from most to least frequent) on the LMO dataset for region-based, detector-based, and combined parsing. All results use approximate RBF SVM and 8-connected CRF smoothing.

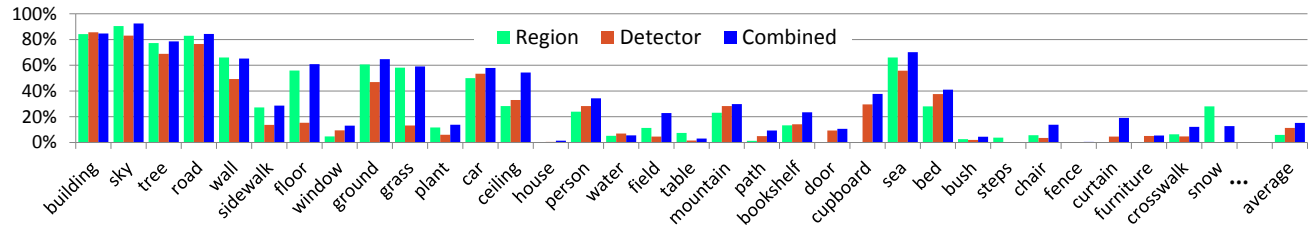

Fig. 11 Classification rates of the most common individual classes (ordered from most to least frequent) on the LM+SUN dataset for region-based, detector-based, and combined parsing. All results use approximate RBF SVM and 8-connected CRF smoothing.

the full training set is a reasonable approximation for the common classes but not for the rare ones. In all subsequent experiments, we stick with exemplar-SVMs due to their higher average per-class accuracy on the LM+SUN dataset.

Table 3 compares different kernels for the combination SVM: linear kernel, approximate RBF (using the embedding of [26]), and exact nonlinear RBF. The linear SVM may be a better choice if speed is a concern (see Section 5.3 for a discussion of running times), but the approximate and exact RBF are able to boost performance by 1-2%. Also note that the approximate RBF embedding obtains similar results to the exact nonlinear RBF. All subsequent experiments will report only the SVM results with the approximate RBF.

The next implementation aspect we examine is CRF smoothing (Section 3.4). Table 4 compares the performance of 8-connected and dense CRFs. While the dense CRF results are generally more visually pleasing, the dense CRF seems to more easily smooth over small objects. This was illustrated in Figure 4, where the dense CRF smoothed away the smaller license plate when the 8-connected CRF did not. Because this over-smoothing tendency can appreciably lower the average per-class rate (as it does on the LM+SUN dataset), for

	LMO	LM+SUN
E-SVM	75.6 (41.1)	59.6 (15.5)
E-SVM CRF	78.6 (39.2)	61.4 (15.2)
LDA	77.1 (41.5)	59.1 (14.3)
LDA CRF	78.4 (40.9)	59.7 (14.3)

Table 2 Comparison of exemplar-SVM and LDA-based detectors. The first number is the per-pixel rate and the second number in parentheses is the per-class rate. All results are for the full combined region- and detector-based system, with the approximate RBF used for the combination SVM. "E-SVM" (resp. "LDA") is the version of the system using the exemplar-SVM (resp. LDA) detectors before smoothing with the CRF. Also shown are results with the 8-connected CRF. Notice that for the LMO dataset the LDA detectors do slightly better, but on LM+SUN they are slightly worse. Interestingly, on both datasets smoothing with the CRF has less of an effect on the LDA-based detector terms likely because they are already smooth.

	LMO	LM+SUN
Linear	75.4 (40.0)	57.2 (16.6)
Linear CRF	77.5 (40.2)	59.5 (15.9)
Approx. RBF	75.6 (41.1)	59.6 (15.5)
Approx. RBF CRF	78.6 (39.2)	61.4 (15.2)
Exact RBF	75.4 (41.6)	N/A
Exact RBF CRF	77.6 (42.0)	N/A

Table 3 Comparison of different kernels used to train the combination SVM. All results here are obtained using exemplar-SVMs and 8-connected CRF. The RBF kernel has a slight edge over the linear kernel, and the approximate RBF embedding of [26] has comparable performance to the exact nonlinear RBF. Note that training the exact RBF on the largest LM+SUN dataset was computationally infeasible.

	LMO	LM+SUN
Combined SVM	75.6 (41.1)	59.6 (15.5)
Combined SVM CRF 8-conn	78.6 (39.2)	61.4 (15.2)
Combined SVM Dense CRF	78.0 (39.8)	61.4 (14.5)

Table 4 Comparison of performance of 8-connected and dense CRF formulations. The first number is the per-pixel rate and the second one in parentheses is the average per-class rate. "Combined SVM" is the result of the region+detector SVM data term before smoothing with the CRF. The dense CRF actually does slightly worse on the LM+SUN dataset and about the same on the LMO dataset. We have found that often the dense CRF would oversmooth small objects even when the unary has strong support for the object.

the instance-level inference evaluation of Section 5.2, we will go with the 8-connected CRF.

Finally, Table 5 shows the effect of the spatial prior. It tends to produce smoother results without hurting the average per-class rate, and will be included in the evaluation of Section 5.2.

Table 6 compares our combined pixel-level parsing system to a number of state-of-the-art approaches on the LMO dataset. We outperform them, in many cases

	LMO	LM+SUN
Combined SVM	75.6 (41.1)	59.6 (15.5)
Combined SVM CRF	78.6 (39.2)	61.4 (15.2)
Combined SVM + SP	76.2 (41.0)	60.4 (15.5)
Combined SVM + SP CRF	78.6 (39.4)	62.0 (15.4)

Table 5 Comparison of performance with and without the spatial prior. "Combined SVM" is the result of the combined SVM data term before smoothing with the CRF. "Combined SVM + SP" is the same but with the spatial prior term added. Also shown are results with the 8-connected CRF.

	LMO
Ours: Combined CRF	78.6 (39.4)
Tighe and Lazebnik [33]	77.0 (30.1)
Liu et al. [23]	76.7 (N/A)
Farabet et al. [9]	78.5 (29.6)
Farabet et al. [9] balanced	74.2 (46.0)
Eigen and Fergus [7]	77.1 (32.5)
Myeong et al. [25]	77.1 (32.3)

Table 6 Comparison to state of the art on the LMO dataset. The "Combined CRF" result for our system are obtained with the following settings: exemplar-SVM detectors, approximate RBF SVM, 8-connected CRF and a spatial prior (see last line of Table 5).

	LM+SUN
Ours: Combined CRF	62.0 (15.4)
Outdoor Images	66.0 (15.5)
Indoor Images	46.9 (12.4)
Tighe and Lazebnik [33]	54.9 (7.1)
Outdoor Images	60.8 (7.7)
Indoor Images	32.1 (4.8)

Table 7 Comparison to [33] on the LM+SUN dataset with results broken down by outdoor and indoor test images. Our "Combined CRF" result is obtained with exemplar-SVM detectors, approximate RBF SVM, 8-connected CRF and a spatial prior (see last line of Table 5).

beating the average per-class rate by up to 10% while maintaining or exceeding the per-pixel rates. The one exception is the system of Farabet et al. [9] when tuned for balanced per-class rates, but their per-pixel rate is much lower than ours in this case. When their system is tuned to a per-pixel rate similar to ours, their average per-class rate drops significantly below ours.

On LM+SUN, which has an order of magnitude more images and labels than LMO, the only previously reported results at the time of development were from our earlier region-based system [33]. As Table 7 shows, by augmenting the region-based term with the new detector-based data term, we are able to raise the per-pixel rate from 54.9% to 62.0% and the per-class rate from 7.1% to 15.4%.

Finally, Figure 12 gives a close-up look at our performance on many small object categories, and Figures

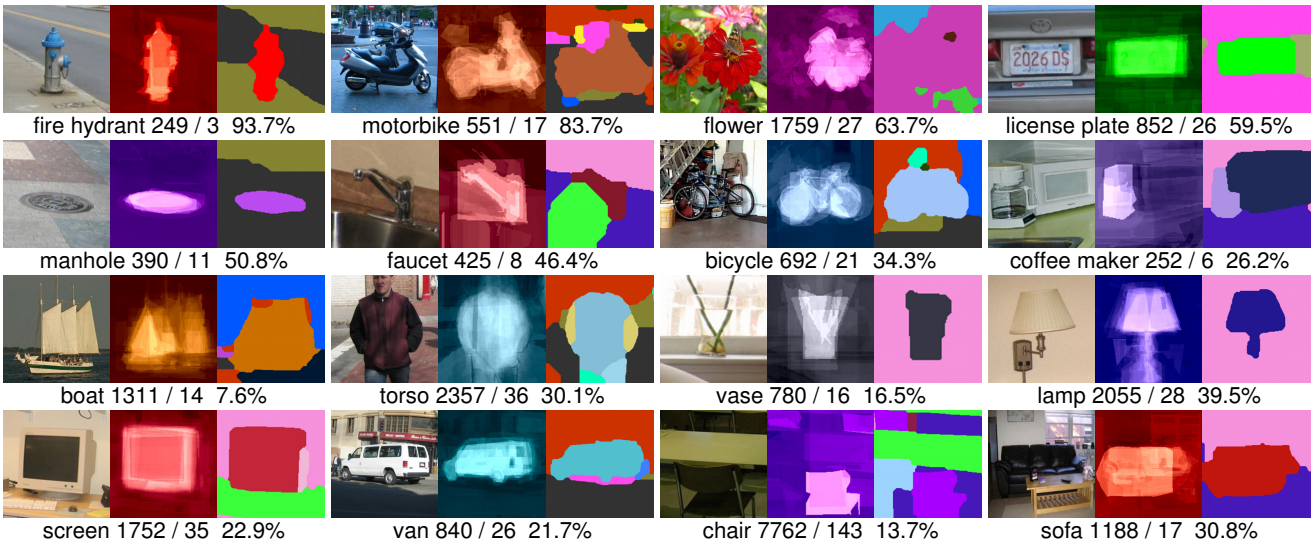

Fig. 12 Examples of "thing" classes found by our system in LM+SUN. For each class we show a crop of an image, the combination SVM output, and the smoothed final result. The caption for each class shows: (# of training instances of that class) / (# of test instances) (per-pixel rate on the test set)%. Best viewed in color.

13 and 14 show several parsing examples on LMO and LM+SUN. Later in Section 5.2 we will show the output of our layered instance inference on the same examples, and it will be interesting to see, for example, whether we will be able to separate the blob of "car" labels from Figure 13(b) or the blob of "people" labels from Figure 13(c) into individual instances.

5.2 Instance-Level Inference Evaluation

Now we move on to evaluating the instance-level inference method of Section 4. For pixel-level CRF inference with object potentials (eq. 16), this section uses exemplar-SVMs, approximate RBF SVM data terms, 8-connected CRF smoothing, and a spatial prior. Just as in Section 5.1, we evaluate the accuracy of pixel-level labeling by reporting overall and average per-class pixel rates. Additionally, in order to assess the quality of inferred object instances and their depth ordering, we introduce two other performance measures.

The first, referred to as *Object P/R*, measures precision and recall of predicted instances. We define a correctly predicted instance to be one that has an intersection over union (I/U) score greater than 0.5 with at least one ground truth polygon of the same class. If two predicted objects both have an I/U score over 0.5 with the same ground truth polygon, only one is considered correct. This is similar to the definition used in PASCAL [8] but relies on object masks rather than bounding boxes. Then precision is computed as the number of correctly predicted objects divided by the total number of predicted objects, while recall is the number of correctly predicted objects divided by the number of ground truth objects.

The second measure, *Pixel P/R*, evaluates the pixel labeling generated by compositing the predicted instances back to front, with any pixels not contained within any instance remaining unlabeled. Precision for this measure is the number of correct pixel labels divided by the total number of predicted pixels, and recall is the number of correct pixel labels divided by the total number of pixels in the test images. Note that Pixel P/R is affected by occlusion order inference, while Object P/R is not. Also, Pixel P/R tells us how well our instance predictions can produce a pixel parse without relying on CRF unaries or smoothing.

To help validate our instance-level inference formulation, we have implemented two simpler baselines to compare against. Both are used to replace the quadratic programming step of Section 4.3; all the other steps stay the same, including the estimation of occlusion ordering (Section 4.4), which is needed to compute Pixel P/R.

Our first baseline, named *NMS Detector*, minimizes the reliance on the pixel parsing when inferring objects. It takes per-exemplar detection masks as candidate instances for both "things" and "stuff" and scores each candidate with the detector responses. To prevent the output from being overrun by false positives, we restrict the candidate instances to classes that appear in the pixel labeling, but this is the only way in which the pixel labeling is used. Simple greedy non-maximal suppression is performed on each class individually with a

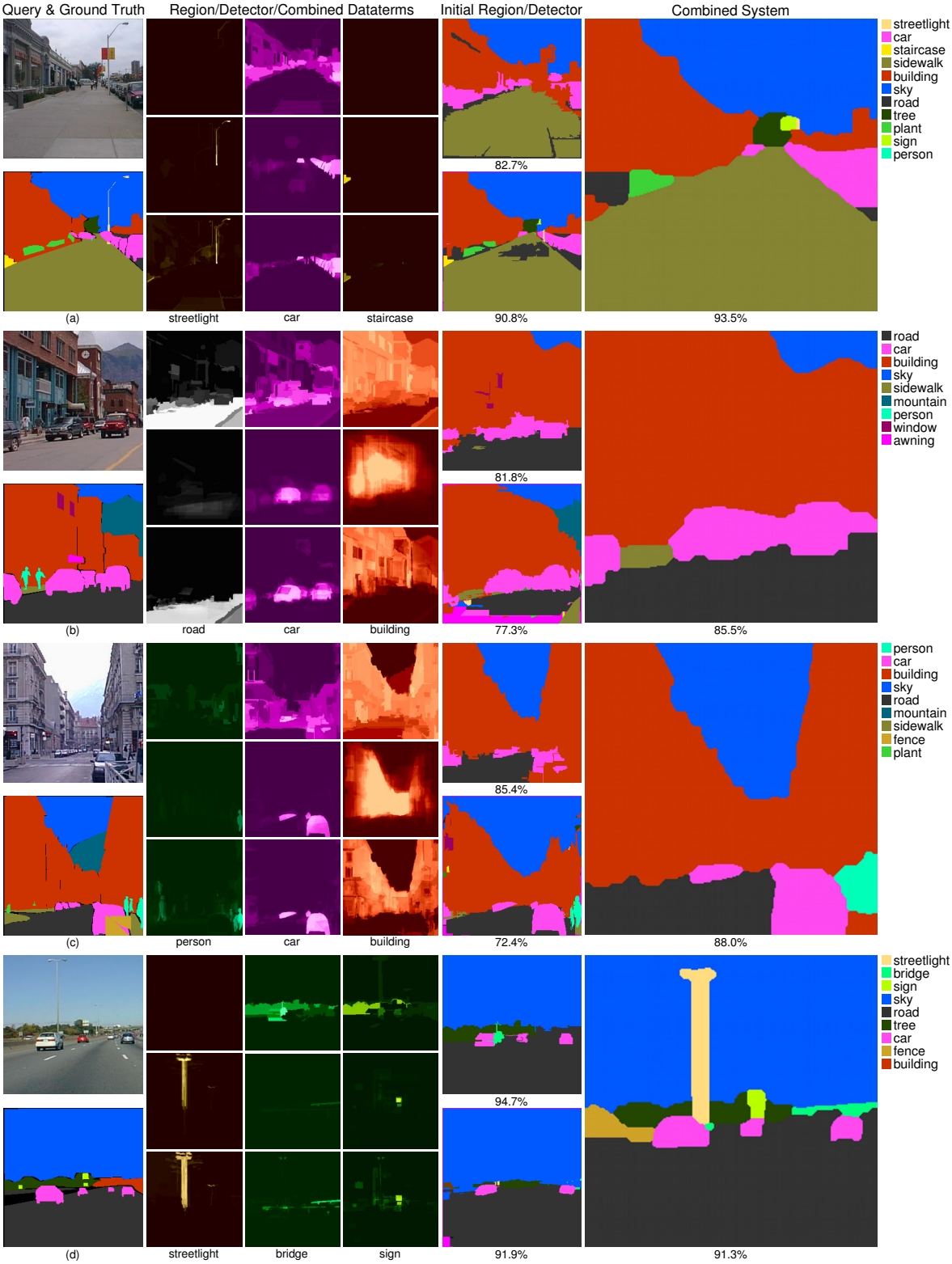

Fig. 13 Example results on the LMO dataset (best viewed in color). First column: query image (top) and ground truth (bottom). Second through fourth columns: region-based data term (top), detector-based data term (middle), and SVM combination (bottom) for three selected class labels. Fifth column: region-based parsing results (top) and detector-based parsing results (bottom) without SVM or CRF smoothing. Right-most column: smoothed combined output. In all example images the combined system does well on cars as well as the "stuff" classes. In (a) the system correctly labels the plants and the detectors find the streetlight and staircase, though these do not survive in the final labeling. In (c) the system finds the people and in (d) the streetlight is correctly labeled.

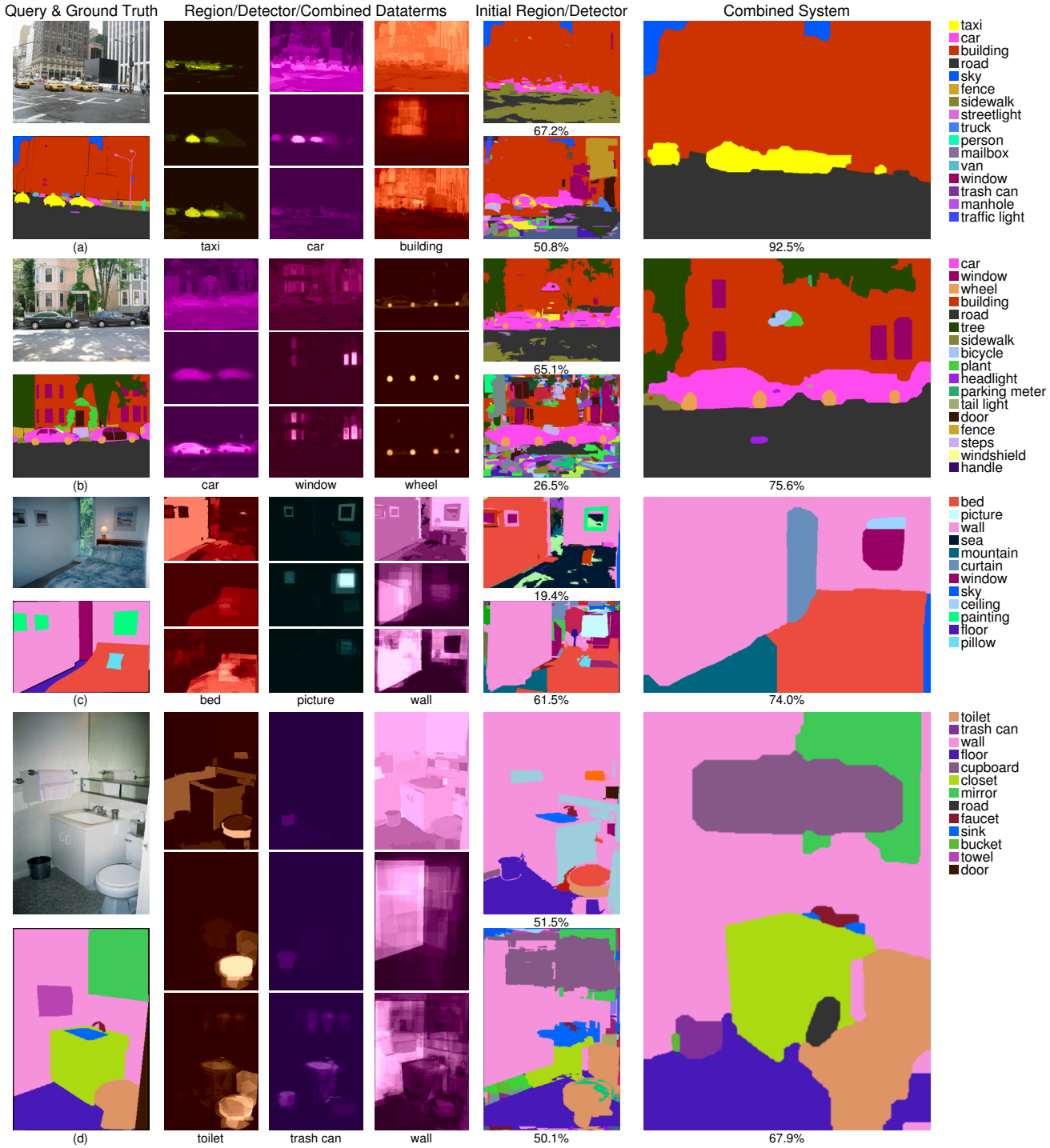

Fig. 14 Example results on the LM+SUN dataset (best viewed in color). First column: query image (top) and ground truth (bottom). Second through fourth columns: region-based data term (top), detector-based data term (middle), and SVM combination (bottom) for three selected class labels. Fifth column: region-based parsing results (top) and detector-based parsing results (bottom) without SVM or CRF smoothing. Right-most column: smoothed combined output. The example in (a) has strong detector responses for both "car" and "taxi," and the combination SVM suppresses the former in favor of the latter. In (b), the system correctly identifies the wheels of the cars and the headlight of the left car. In (c), the detectors correctly identify the wall and most of the bed. Note that the region-based parser alone mislabels most of the bed as "sea"; the detector-based parser does much better but still mislabels part of the bed as "mountain." In this example, the detector-based parser also finds two pictures and a lamp that do not survive in the final output. In (d), our system successfully finds the toilet, faucet, and trash can (which, incidentally, is present in the image though not in the ground truth).

	Instances	Object P/R	Pixel P/R	Pixel Parse
	LMO			
Initial Pixel Parse				**78.6 (39.4)**
NMS Detector	13734	3.1 / 21.4	58.0 / 50.5	77.8 (39.1)
NMS SVM	3236	11.8 / 18.4	75.7 / 62.0	78.1 (38.8)
Greedy	912	**44.4** / 20.0	**75.4** / 71.8	78.4 (38.5)
CPlex QP	990	42.8 / **21.1**	**75.4** / **71.9**	78.5 (38.7)
	LM+SUN			
Initial Pixel Parse				62.0 (15.4)
NMS Detector	146101	0.8 / 12.2	27.8 / 25.1	60.9 (15.1)
NMS SVM	12839	9.3 / 12.9	53.3 / 52.8	61.8 (15.9)
Greedy	4902	**24.7** / 13.1	60.8 / 59.8	62.2 (16.1)
CPlex QP	5425	22.5 / **13.4**	**61.0** / **60.0**	**62.3 (16.3)**

Table 8 Comparison of instance-level inference baselines (NMS Detector, NMS SVM) and our proposed quadratic programming inference (Greedy, CPlex QP). The "Instances" column lists the total number of instances predicted by each method on the test set. Below, the accuracies for the four instance-level inference methods are shown after three iterations of alternating pixel- and instance-level inference. The "Pixel Parse" column reports the per-pixel and average per-class rates for pixel-level CRF inference, which uses approximate RBF SVM combination potentials, 8-connected CRF, a spatial prior, and object potentials derived from the respective instance inference methods. "Initial Pixel Parse" is the result of CRF inference without object potentials and matches the values from Tables 6 and 7. See text for definition of the Object P/R and Pixel P/R measures.

threshold of 50% overlap in the same manner as [10] to infer the final objects. Table 8 shows that this setup produces far more object instance predictions and thus has a far lower object precision. The object recall is fairly high but if we look at the Pixel P/R scores we can see that it fails to produce accurate pixel predictions.

For our second baseline, named *NMS SVM*, we use the same candidate objects as in Section 4.1 and score them as in Section 4.2. However, instead of the inter-class overlap constraints defined in Section 4.2 we again use greedy non-maximal suppression. Thus, when compared to our proposed inference method, this baseline shows the effectiveness of overlap constraints. As can be seen in Table 8, NMS SVM produces two to three times the number of instance predictions over our proposed method, and has lower Object P/R and Pixel P/R performance.

In addition to the two baselines, Table 8 also compares greedy inference and the CPlex solver [16] for solving the objective function of eq. (15). The greedy inference tends to infer fewer object instances than CPlex. Concretely, on the LM+SUN dataset, both start with an average of just over 300 candidate instances per test image, out of which greedy selects about 11 instances while CPlex selects about 12. Greedy inference has a higher object precision, but it does miss some objects and thus has a lower object recall.

Table 9 shows the effect of alternating pixel- and instance-level inference. Somewhat disappointingly, instance-level inference does not give us any significant gains in pixel accuracy over the system of Section 3. By comparing the initial pixel parse numbers in the top row of Table 8 to the final ones for the greedy and CPlex inference, we can see that our overall and average per-class rates drop slightly for the LMO dataset and increase slightly for LM+SUN. As we have learned from our experiments, alternating between pixel-level and instance-level inference only tends to improve performance in complex scenes where overlap relationships can help correct errors (e.g., in Figure 9, some wrong "building" labels get corrected to "trashcan"). Because most images in the LMO dataset are simplistic, with few overlapping instances, our iterative method does not produce any improvement. What is worse, in some cases, enforcing sensible overlap constraints can actually hurt the pixel-level accuracy. For example, in the LMO test image shown in Figure 15, the water on the bottom initially gets incorrectly but confidently labeled as "road." Then, though there is a decent initial "boat" labeling as well as good candidate "boat" masks, these masks cannot be selected as they have an inconsistent overlap with "road." Because there are so few boats in the test images, this brings down the "boat" class rate for the entire test set from 14.8% to 2.3%, and is almost single-handedly responsible for the drop in average per-class rate from the initial to the final pixel labeling ac-

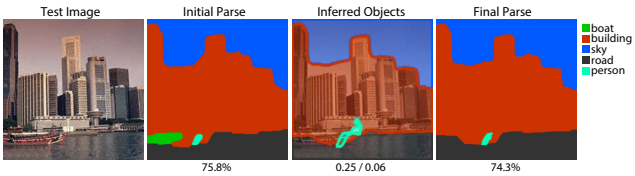

Fig. 15 LMO test example where instance-level inference ends up hurting the pixel-level labeling. The initial pixel labeling for the boat is fairly good, but the water gets mislabeled as "road." Since "boat" cannot overlap "road," a boat instance is never turned on, which in turn decreases the pixel labeling performance.

		LMO			LM+SUN		
	Iteration	1	2	3	1	2	3
Greedy:	Object P/R	43.9 / 20.1	44.1 / 20.1	44.4 / 20.0	24.2 / 13.1	24.4 / 13.1	24.7 / 13.1
	Pixel P/R	75.6 / 71.8	75.4 / 71.8	75.4 / 71.8	60.7 / 59.7	60.8 / 59.7	60.8 / 59.8
	Pixel Parse	78.6 (39.4)	78.5 (38.7)	78.4 (38.5)	62.0 (15.4)	62.0 (16.1)	62.2 (16.1)
CPlex QP:	Object P/R	42.8 / 21.0	42.7 / 21.0	42.8 / 21.1	22.4 / 13.3	22.2 / 13.3	22.5 / 13.4
	Pixel P/R	75.4 / 71.5	75.3 / 71.6	75.4 / 71.9	61.0 / 60.0	60.8 / 59.8	61.0 / 60.0
	Pixel Parse	78.6 (39.4)	78.4 (38.6)	78.5 (38.7)	62.0 (15.4)	62.0 (16.1)	62.3 (16.3)

Table 9 Evaluation of iterative refinement. See text and caption of Table 8 for definition of performance measures.

curacy on LMO. As for LM+SUN, it has more complex and varied scenes, and we do see small gains from initial to final pixel labeling accuracy on that dataset.

Here we can also compare our results to that of Isola et al. [17] who also predict object instances by transfering ground truth masks from the training set. They report a per-pixel performance of 70.0% on the LMO dataset. If we just use our predicted objects to produce a pixel labeling we would achieve a pixel accuracy of 71.9% (the recall from our Pixel P/R measure) and our pixel parsing result has a per-pixel accuracy of 78.5%, clearly outperforming Isola et al. [17].

Finally, Figures 16 and 17 show the output of our instance-level inference on the same test images as Figures 13 and 14. It is satisfying to observe, for example, that our layered representation is able to correctly separate the mass of "taxi" labels from Figure 14(a) into four different instances.

5.3 Running Times

To conclude our evaluation, we examine the computational requirements of our system. Unless stated otherwise, the following timings are given for our MATLAB implementation (feature extraction and file I/O excluded) on the LM+SUN dataset on a six-core 3.4 GHz Intel Xeon workstation with 48 GB of RAM.

First, let us discuss training time requirements. There are a total of 354,592 objects in the LM+SUN training set, and we have to train a per-exemplar detector for each of them. For exemplar-SVMs, the average training time per detector is 472.3 seconds; training all of them would require 1,938 days on a single core, but we do it on a 512-node cluster in approximately four days. For the LDA-based per-exemplar detectors, the training time is dominated by the covariance computation, which is an online process requiring 4.82 seconds per image, or 60.5 hours for the entire LM+SUN dataset on a single core. This can also be trivially parallelized across our 512-node cluster, to bring the training time down to roughly 20 minutes. Leave-one-out parsing of the training set (see below for average region- and detector-based parsing times per image) takes 939 hours on a single core, or about two hours on the cluster. Next, training a set of 232 one-vs-all SVMs takes a total of one hour on a single core for the linear SVM and ten hours for the approximate RBF. Note that the respective feature dimensionalities are 464 and 4,000; this nearly tenfold dimensionality increase accounts for the tenfold increase in running time. Tuning the SVM parameters by fivefold cross-validation on the cluster only increases the training time by a factor of two.

Next, we look at the computational requirements of our pixel-level and object-level inference methods at test time. For pixel labeling, the computation of the region-based data term takes an average of 27.5 seconds per image. The computation of the detector-based data term involves running an average of 4,842 detectors per image in 47.4 seconds total. This step of our system could be speeded up by implementing the hashing technique of Dean et al. [6]. Computation of the combined SVM data term takes an average of 8.9 seconds for the linear kernel and 124 seconds for the approximate RBF (once again, the tenfold increase in feature dimensionality and the overhead of computing the embedding account for the increase in running time). CRF inference takes an average of 6.8 seconds per image for the 8-connected α-expansion setup and 4.2 seconds for the fully connected CRF.

For instance-level inference, generating the "stuff" candidate instances amounts to two CRF inference computations taking to an average of 2.3 seconds each (4.6 seconds total). Combining the "stuff" and "thing" candidate instances produces on average 316.9 candidates per image. Computing the scores and overlap penalties for all candidate instances takes on average 47.3 seconds and is typically our most expensive step. Finally, inference using our greedy method and CPlex takes an average of 0.026 and 2.8 seconds, respectively. Both methods exhibit exponential growth in runtime as a function of the number of candidate instances, but for images with roughly 1000 candidate instances CPlex took about 73.9 seconds while our greedy inference took 0.71 seconds. Occlusion order inference takes less than 0.002 seconds and is thus negligible. One iteration of combining the instance inference with the CRF pixel la-

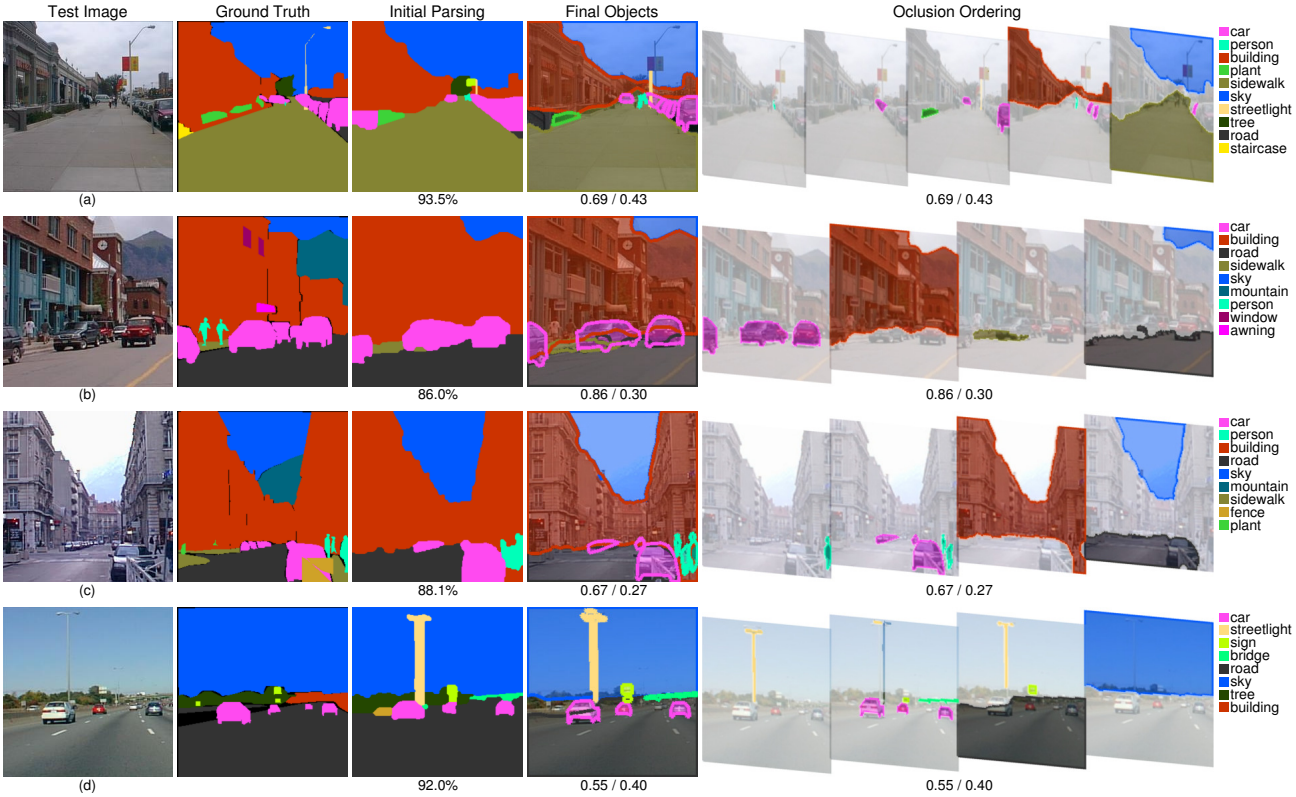

Fig. 16 Layered inference results on the same LMO test images as in Figure 13. For pixel labeling we report the per-pixel rate and for instance prediction we report Object P/R underneath each image. In (a) and (b), the blob of cars is split up into individual instances, though not all the cars are found in (b). In (c), three people are correctly identified on the right side. In (d), the streetlight, missing from ground truth, is found, though three instances are predicted because there turns out to be very little overlap between the three thin masks and thus they form a valid configuration.

bel inference takes an average of 4.3 seconds for greedy and 7.1 seconds for CPlex. Note this does not include the "stuff" candidate generation or the instance score computation as we compute these once per image, not at every iteration.

6 Discussion

This paper has presented approaches for densely labeling pixels in complex scene images, as well as predicting individual object instances and their occlusion ordering. The pixel-level parsing system of Section 3 successfully combines cues based on segmentation regions and per-exemplar detectors to achieve state-of-the-art accuracy on two challenging datasets containing up to tens of thousands of training images and hundreds of class labels. The instance-level parsing system of Section 4 is capable of interpreting scenes in richer and more structured ways, including separating multiple objects from the same class and inferring the occlusion ordering of different objects. While instance-level inference currently does not yield a significant further improvement in per-pixel parsing accuracy over the system of Section 3, it does go significantly beyond it in representational capability.

Ultimately, we would like our systems to function on *open universe* datasets, such as LabelMe [29], that are constantly evolving and do not have a pre-defined list of classes of interest. The region-based component of our system already has this property [33]. In principle, per-exemplar detectors are also compatible with the open-universe setting, since they can be trained independently as new exemplars come in. Our SVM combination scheme is the only part of the method that relies on batch offline training (including leave-one-out parsing of the entire training set). In the future, we plan to investigate online methods for this step.

To improve the accuracy of our instance-level prediction, a key direction is generating better candidate instances. Currently, we manually divide our label set into "stuff" and "thing" classes and use different methods to generate candidate objects for each. To an extent, this division is arbitrary, as many classes can sometimes appear as "things" with well-defined boundaries, and sometimes as diffuse "stuff." In the future, we plan

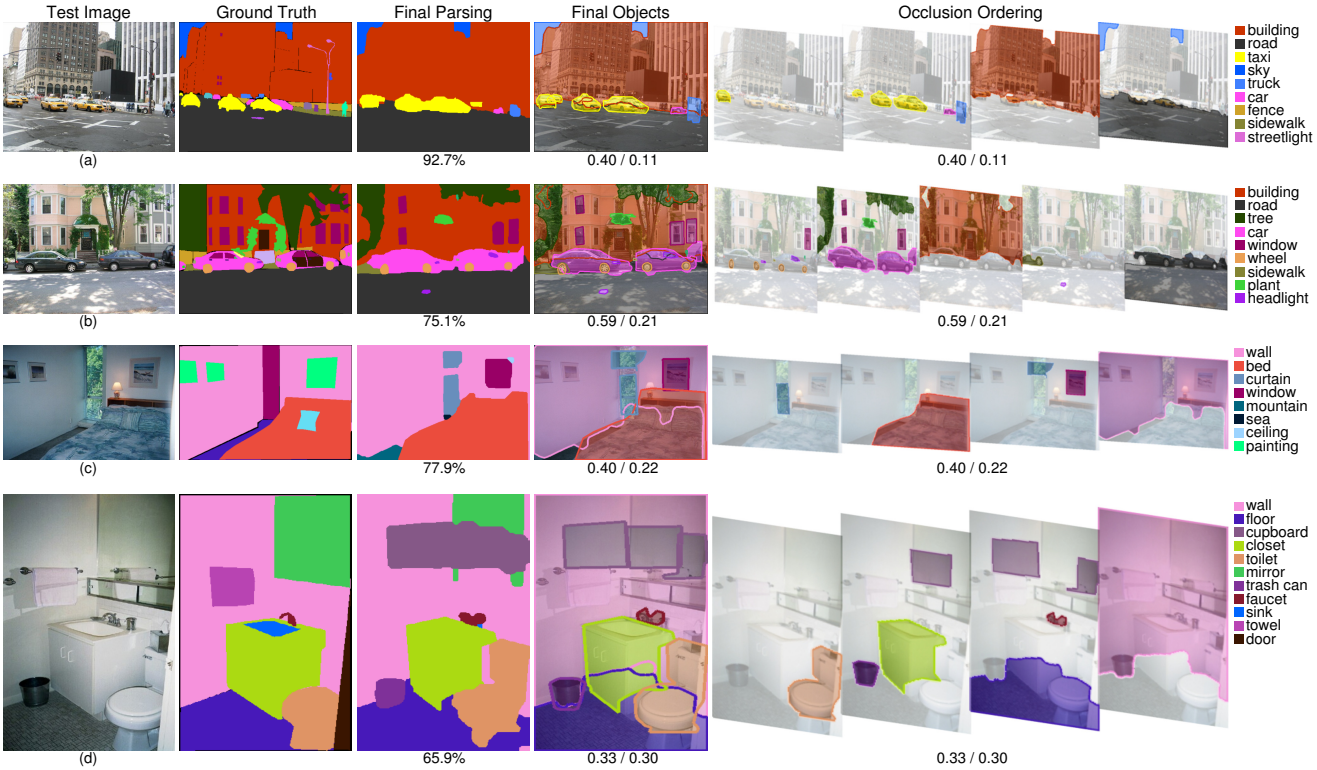

Fig. 17 Layered inference results on the same LM+SUN test images as in Figure 14. For pixel labeling we report the per-pixel rate and for instance prediction we report Object P/R underneath each image. In (a), we successfully separate the four taxi cabs, including the highly occluded one second from the left. In (b), our system finds the two central cars and is even close to identifying the cropped one on the right. In (c), we find the bed and walls of the room. In (d), we find the toilet, faucet, and trash can instances, together with some hallucinated cupboards.

to investigate ways of generating candidate object masks that would not rely on a hard "things vs. stuff" split. We also plan to develop a more principled treatment of background classes than the one afforded by our non-exclusive SVM formulation. Finally, it would be very interesting to explore more powerful inter-object constraints than the ones based on overlap – for example, constraints based on relative positions and sizes of objects.

Acknowledgements This research was supported in part by NSF grants IIS 1228082 and CIF 1302438, DARPA Computer Science Study Group (D12AP00305), Microsoft Research Faculty Fellowship, Sloan Foundation, and Xerox. We thank Arun Mallya for helping to adapt the LDA detector code of [14].

References

1. Adelson, E.H.: On seeing stuff: the perception of materials by humans and machines. In: Human Vision and Electronic Imaging, pp. 1–12 (2001)
2. Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: ICCV (2003)
3. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. PAMI **26**(9), 1124–37 (2004)
4. Brostow, G.J., Shotton, J., Fauqueur, J., Cipolla, R.: Segmentation and recognition using structure from motion point clouds. In: ECCV (2008)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
6. Dean, T., Ruzon, M.A., Segal, M., Shlens, J., Vijayanarasimhan, S., Yagnik, J.: Fast, accurate detection of 100,000 object classes on a single machine. In: CVPR (2013)
7. Eigen, D., Fergus, R.: Nonparametric image parsing using adaptive neighbor sets. In: CVPR (2012)
8. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html
9. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Scene parsing with multiscale feature learning, purity trees, and optimal covers. In: ICML (2012)
10. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI **32**(9), 1627–1645 (2010)
11. Floros, G., Rematas, K., Leibe, B.: Multi-class image labeling with top-down segmentation and generalized robust P^N potentials. In: BMVC (2011)

12. Gould, S., Fulton, R., Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: ICCV (2009)
13. Guo, R., Hoiem, D.: Beyond the line of sight: labeling the underlying surfaces. In: ECCV (2012)
14. Hariharan, B., Malik, J., Ramanan, D.: Discriminative decorrelation for clustering and classification. In: ECCV (2012)
15. Heitz, G., Koller, D.: Learning spatial context: Using stuff to find things. In: ECCV, pp. 30–43 (2008)
16. IBM: Cplex optimizer. http://www.ibm.com/software/commerce/optimization/cplex-optimizer/ (2013)
17. Isola, P., Liu, C.: Scene collaging: Analysis and synthesis of natural images with semantic layers. In: ICCV (2013)
18. Kim, B., Sun, M., Kohli, P., Savarese, S.: Relating things and stuff by high-order potential modeling. In: ECCV Workshop on Higher-Order Models and Global Constraints in Computer Vision (2012)
19. Kim, J., Grauman, K.: Shape sharing for object segmentation. In: ECCV (2012)
20. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? PAMI **26**(2), 147–59 (2004)
21. Krahenbuhl, P., Koltun, V.: Efficient inference in fully connected CRFs with Gaussian edge potentials. In: NIPS (2011)
22. Ladický, L., Sturgess, P., Alahari, K., Russell, C., Torr, P.H.: What, where & how many? combining object detectors and CRFs. In: ECCV (2010)
23. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. PAMI **33**(12), 2368–2382 (2011)
24. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-SVMs for object detection and beyond. In: ICCV (2011)
25. Myeong, H.J., Chang, Y., Lee, K.M.: Learning object relationships via graph-based context model. CVPR (2012)
26. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: NIPS (2007)
27. Rother, C., Kolmogorov, V., Blake, A.: "grabCut" – interactive foreground extraction using iterated graph cuts. SIGGRAPH (2004)
28. Russell, B.C., Torralba, A.: Building a database of 3d scenes from user annotations. In: CVPR (2009)
29. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme : a database and web-based tool for image annotation. IJCV **77**(1-3), 157–173 (2008)
30. Shotton, J., Winn, J.M., Rother, C., Criminisi, A.: Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. IJCV **81**(1), 2–23 (2009)
31. Sturgess, P., Alahari, K., Ladický, L., Torr, P.H.S.: Combining appearance and structure from motion features for road scene understanding. BMVC (2009)
32. Tighe, J., Lazebnik, S.: Finding things: Image parsing with regions and per-exemplar detectors. In: CVPR (2013)
33. Tighe, J., Lazebnik, S.: SuperParsing: Scalable nonparametric image parsing with superpixels. IJCV **101**(2), 329–349 (2013)
34. Tighe, J., Niethammer, M., Lazebnik, S.: Scene parsing with object instances and occlusion ordering. In: CVPR (2014)
35. Tu, Z., Chen, X., Yuille, A.L., Zhu, S.C.: Image parsing: Unifying segmentation, detection, and recognition. IJCV **63**(2), 113–140 (2005)
36. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: CVPR (2010)
37. Yang, Y., Hallman, S., Ramanan, D., Fowlkes, C.: Layered object models for image segmentation. PAMI **34**(9), 1731–1743 (2012)
38. Yao, J., Fidler, S., Urtasun, R.: Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In: CVPR (2012)
39. Zhang, C., Wang, L., Yang, R.: Semantic segmentation of urban scenes using dense depth maps. In: ECCV (2010)