The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects

John Winn
Microsoft Research Cambridge
Cambridge, UK
jwinn@microsoft.com

Jamie Shotton
Department of Engineering
University of Cambridge, UK
jdjs2@cam.ac.uk

Abstract

This paper addresses the problem of detecting and segmenting partially occluded objects of a known category. We first define a part labelling which densely covers the object. Our Layout Consistent Random Field (LayoutCRF) model then imposes asymmetric local spatial constraints on these labels to ensure the consistent layout of parts whilst allowing for object deformation. Arbitrary occlusions of the object are handled by avoiding the assumption that the whole object is visible. The resulting system is both efficient to train and to apply to novel images, due to a novel annealed layout-consistent expansion move algorithm paired with a randomised decision tree classifier. We apply our technique to images of cars and faces and demonstrate state-of-the-art detection and segmentation performance even in the presence of partial occlusion.

1. Introduction

This paper addresses the problem of detecting and segmenting both clean and partially occluded deformable objects of a known category. The approach uses a part labelling which densely covers the object and models the label distribution using an enhanced Conditional Random Field which we call the Layout Consistent Random Field (LayoutCRF).

The use of parts has several advantages. First, recognising parts of an object allows for object detection under partial occlusion. Second, there are local spatial interactions between parts that can help with detection; for example, we expect to find the nose just above the mouth on a face. Hence, we can exploit local part interactions to exclude invalid hypotheses at a local level. Third, knowing the location of one part highly constrains the locations of other more distant parts. For example, knowing the locations of wheels of a car constrains where the rest of the car can be detected. Thus, we can improve object detection by incorporating long range spatial constraints on the parts. Finally, by inferring a part labelling for the training data, we can accurately assess the variability in the appearance of each part, giving better part detection and so better object detection.

A key aspect of the model is the use of asymmetric pairwise potentials to capture the spatial ordering of parts, e.g. car wheels must be below car body, not vice-versa. These asymmetric potentials allow propagation of long-range spatial dependencies using only local interactions.

The pairwise potentials are carefully constructed to distinguish between various types of occlusion, such as object occluding background[1], background occluding object, and object occluding object. The model is capable of representing multiple object instances which inter-occlude, and infers a pairwise depth ordering.

1.1. Related Work

There have been a number of recent parts-based approaches to segmentation and detection of categories of objects, though few have specifically dealt with partial occlusion. It is possible to pre-select which parts are used as in [4], though this requires considerable human effort for each new object class. Alternatively, parts can be learned by clustering visually similar image patches [1, 11] but this approach does not exploit the spatial layout of the parts in the training images. There has been work with generative models that do learn spatially coherent parts in an unsupervised manner. For example, the constellation models of Fergus et al. [5] learn parts which occur in a particular spatial arrangement. However, the parts correspond to sparsely detected interest points and so parts are limited in size, cannot represent untextured regions and do not provide a segmentation of the image. More recently, Winn and Jojic [18] used

[1] Note that throughout the paper, 'background' is used to mean pixels not belonging to an identified object class and 'foreground' is used to mean pixels that do belong to the class. Hence it is possible to have background objects in front of foreground ones, as illustrated by a person (background) occluding a car (foreground) in figure 2.

a dense generative model to learn a partitioning of the object into parts, along with an unsupervised segmentation of the object. Their method does not learn a model of object appearance (only of object shape) and so cannot be used for object detection in cluttered images.

As well as unsupervised methods, there are a range of supervised methods for segmentation and detection. Ullman and Borenstein [2] use a fragment-based method for segmentation, but do not provide detection results. Shotton et al. [15] use a partially-supervised boosting method based on image contours for detection, but this does not lead to a segmentation. A number of methods use a Conditional Random Field (CRF) [10] to achieve segmentation [9] or sparse part-based detection [14]. The OBJ CUT work of Kumar et al. [8] uses a discriminative model for detection and a separate generative model for segmentation but requires that the parts are learned in advance from video.

None of the above methods deals explicitly with partial occlusion, although OBJ CUT does allow for self occlusion. The Variational Ising Classifier of Williams et al. [17] presents a detection scheme which explicitly models partial occlusions for cropped face images, but it is unclear whether this technique would scale to cluttered images containing multiple object instances. The LayoutCRF presented here extends previous work on the LHRF [6] to allow for multiple object instances and inter-object occlusion.

2. Layout Consistent Random Field Model

Our aim is to take an image $\mathbf{x}$ and infer a labelling for each pixel indicating both the class of object and which instance of that class the pixel belongs to. We denote the set of all image pixels as V and for each pixel $i \in V$ define a instance label $y_i \in \{0, 1, \ldots, M\}$ where the background label is indicated by $y_i = 0$, and M foreground instances by $y_i \in \{1, \ldots, M\}$. We will describe the case where only one non-background class is considered at a time, though preliminary results for multiple classes are also given, in which case y_i labels pairs of (class, instance).

Additionally, a hidden layer of *part labels* h_i is used as introduced in the Hidden Random Field (HRF) model of [16]. Each object instance has a separate set of H part labels so that $h_i \in \{0, 1, \ldots, H \times M\}$. These hidden variables represent the assignment of pixels to parts and are not observed during training. Parts are learned so as to densely cover the object in a coarse deformable grid (see figure 6).

Our model, the Layout Consistent Random Field (LayoutCRF), is an HRF with asymmetric pairwise potentials, extended with a set of discrete valued instance transformations $\{T_1, \ldots, T_M\}$. Each transformation T represents the translation and left/right flip of an object instance, by indexing all possible integer pixel translations for each flip orientation. Each of these transformation variables is linked to every part label h_i. This aspect of the model extends the work of [6] to cope with multiple object instances.

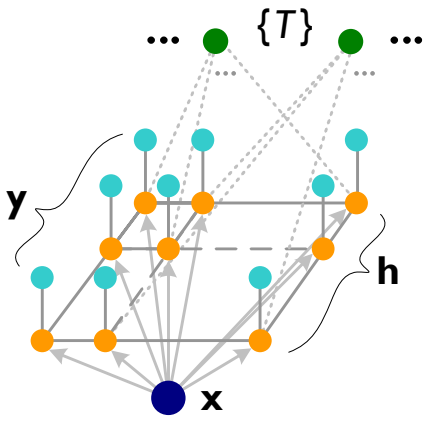

Figure 1. **The Layout Consistent Random Field.** All part label nodes $\mathbf{h}$ (orange) are conditioned on the image $\mathbf{x}$ (dark blue), and connected 4-wise with their neighbours. These pairwise potentials are asymmetric in our model. A deterministic mapping links $\mathbf{h}$ to class labels $\mathbf{y}$ (light blue). A set of instance transformations $\{T\}$ (green) are each connected to all of the part labels.

Fig. 1 shows the graphical model corresponding to the LayoutCRF. Note that the local dependencies captured are between parts rather than between instance labels. The edges from part labels h_i to instance labels y_i represents the unique deterministic mapping from part labels to instance labels, which we denote as $y_i = y(h_i)$.

The conditional distribution for the label image $\mathbf{y}$ and part image $\mathbf{h}$ is defined as:

$$P(\mathbf{y}, \mathbf{h}, \{T\} \,|\, \mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta}, \mathbf{x})} \prod_{i \in V} \phi_i(h_i, \mathbf{x}; \boldsymbol{\theta})\, \delta(y_i = y(h_i))\, \lambda_i(h_i, \{T\}; \boldsymbol{\theta}) \prod_{(i,j) \in E} \psi_{ij}(h_i, h_j, \mathbf{x}; \boldsymbol{\theta}). \quad (1)$$

where $\boldsymbol{\theta}$ represents the learned parameters of the model, and E is the set of all 4-wise neighbours between pairs of part labels. The *unary* potentials $\phi_i(h_i, \mathbf{x}; \boldsymbol{\theta})$ use only local image information, and, as described below, take the form of randomised decision trees. The *asymmetric pairwise* potentials $\psi_{ij}(h_i, h_j, \mathbf{x}; \boldsymbol{\theta})$ encourage local and, to a certain extent, long-range compatibility between the part labels. The *instance* potentials $\lambda_i(h_i, \{T\}; \boldsymbol{\theta})$ encourage the correct long-range spatial layout of parts for each object instance. Finally the potentials $\delta(y_i = y(h_i))$ enforce the deterministic mapping from part labels to instance labels.

2.1. Layout Consistent Pairwise Potentials

An important contribution of this paper is the form of the pairwise potentials ψ_{ij}. A common choice in CRF mod-

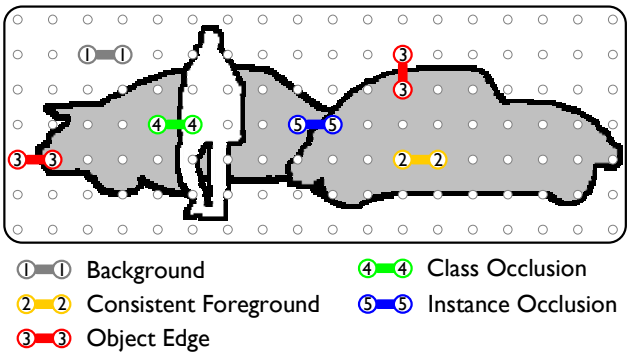

Figure 2. **Distinguished transitions.** The distinguished types of transitions between neighbouring part label nodes in the graph. See text for definitions. Note that a pairwise depth ordering is implicit in these types.

els [16, 6] is to use only symmetric pairwise potentials, whereas ours are *asymmetric*. The use of asymmetric potentials allows the relative layout (above/below/left/right) of parts to be modelled, whilst also propagating long-range spatial constraints using only local pairwise interactions.

Figure 2 illustrates the types of transitions which we distinguish between in the pairwise potentials. To describe these types, we must first define the concept of *layout-consistency*. A label is layout-consistent with itself, and with those labels that are adjacent in the grid ordering as defined in figure 3. We assume that neighbouring pixels whose labels are not layout-consistent are not part of the same object. Hence, for a pair of neighbouring labels h_i and h_j, we define transitions to be one of the following types:

Background Both h_i and h_j are background labels.

Consistent Foreground Both h_i and h_j are *layout-consistent* foreground labels. Note the asymmetry here: for example, if $h_i = a$ to the left of $h_j = b$ is layout-consistent then (assuming $a \neq b$) $h_i = b$ to the left of $h_j = a$ is not layout-consistent (see figure 3).

Object Edge One label is background, the other is a part label *that lies on the object edge*. Treating this type of transition specially allows to encourage object-background transitions at the true object boundary.

Class Occlusion One label is an *interior* foreground label, the other is the background label. This represents the case where the 'background' occludes the object.

Instance Occlusion Both are foreground labels but are not layout-consistent, with at least one label being an object edge. This represents the case where one instance of an object occludes another instance.

Inconsistent Interior Foreground Both labels are interior foreground labels which are not layout-consistent. This can only occur due to transparency or self-occlusion, both of which are considered to be rare and hence this case is penalised more heavily.

The value of the pairwise potential varies according to transition type as follows:

$$-\log \psi_{ij}(h_i, h_j, \mathbf{x}; \boldsymbol{\theta}) = \begin{cases} \beta_{\text{bg}} & \text{Background} \\ 0 & \text{Consistent Foreground} \\ \beta_{\text{oe}}.e_{ij} & \text{Object Edge} \\ \beta_{\text{co}}.e_{ij} & \text{Class Occlusion} \\ \beta_{\text{io}}.e_{ij} & \text{Instance Occlusion} \\ \beta_{\text{iif}} & \text{Inconsistent Interior Foreground} \end{cases} \quad (2)$$

where cost e_{ij} is an image-based edge cost to encourage object edges to align with image boundaries, and is set to $e_{ij} = e_0 + \exp(\gamma \|x_i - x_j\|^2)$. The contrast term γ is estimated separately for each image as $(2 < \|x_i - x_j\|^2 >)^{-1}$ where $<>$ denotes a mean over all neighbouring pairs of pixels.

2.2. Instance Potentials

The instance potentials are look-up tables

$$\lambda_i(h_i, \{T_1, \ldots, T_M\}; \boldsymbol{\theta}) = \widetilde{P}(h_i | \text{loc}(T_{y(h_i)}, i))^{\nu} \quad (3)$$

where $\text{loc}(T_m, i)$ returns position i inverse-transformed by the transformation T_m, and ν is a parameter to weight the strength of the potential. This potential encourages the correct spatial layout of parts for each object instance by gravitating parts towards their expected positions, given the transformation $T_{y(h_i)}$ of the instance.

3. Inference

We use an annealed layout-consistent expansion move algorithm to infer the part labellings and hence (deterministically) instance labellings, as described below. First, however, we describe our three-step algorithm for inferring the number of object instances and their locations.

Step 1 Initially we have no knowledge of the number of objects and so do not distinguish between different instances. We therefore collapse all the part labels together across instances, so that we have $h_i \in \{0, 1, \ldots, H\}$. Additionally we merge all the instance labels together so that $y_i \in \{0, 1\}$, and remove the links from the instance transformation nodes $\{T\}$ to the part labels $\mathbf{h}$. MAP inference is performed on this simplified model, resulting in a part labelling image $\mathbf{h}^*$.

Step 2 We determine the number of layout-consistent regions in the labelling $\mathbf{h}^*$ using connected component analysis, where two pixels are considered connected if

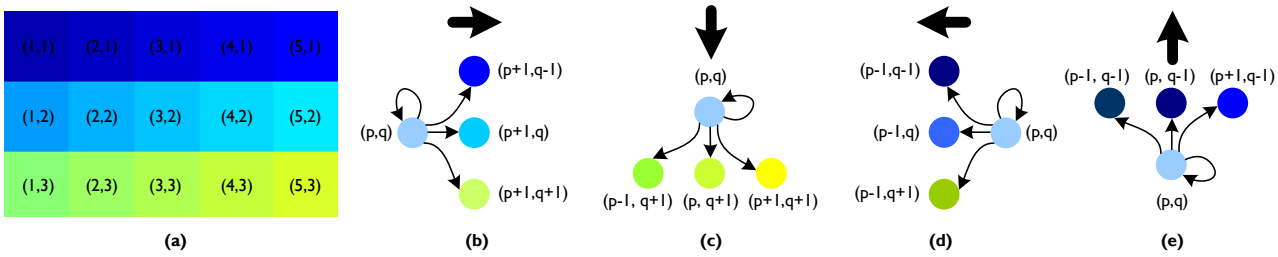

Figure 3. **Layout-consistency.** Colours represent part labels. (a) A sub-section of the regular grid with label numbers overlaid as pairs (p, q). (b-e) Layout-consistent label pairs for pairwise links of each of the four orientations (left-to-right, top-to-bottom, right-to-left, bottom-to-top). Note that slight deformations from the regular grid are still considered layout-consistent.

they have layout-consistent part labels. This yields an initial estimate of the number of object instances M, and an initial instance labelling.

It is then necessary to estimate the transformations $T_1 \dots T_M$ for each instance label. These are estimated as $\arg\max_{\{T\}} \prod_i \lambda_i(h_i, \{T\}; \boldsymbol{\theta})$, which can be computed separably for each instance label. To capture two possible modes corresponding to left/right flips of the object, we choose to create two instance labels for each connected component. When estimating T for each label, the first is constrained to have T facing left, the second has T facing right. Thus, M is equal to twice the number of connected components.

Step 3 We are now able to use the full model with the label set $h_i \in \{0, 1, \dots, H \times M\}$, $y_i \in \{0, 1, \dots M\}$ and including the links from $\{T\}$ to $\mathbf{h}$. Using this model, we re-run the MAP inference, obtaining $\hat{\mathbf{h}}$ which now distinguishes between different object instances. Typically, $\hat{\mathbf{h}}$ contains part labels for only a subset of the instances. For example, normally only one of each pair of left-facing and right-facing instances is retained.

If desired, steps 2 and 3 can be iterated to refine the instance transformation and the instance labelling, but this was not performed in our experiments.

3.1. Expansion Move Algorithm

An annealed expansion move algorithm is used for approximate MAP inference of the part labels. We first describe the standard expansion move algorithm, followed by our extensions.

The idea of the expansion move algorithm is to reduce the problem of maximizing a function $f(\mathbf{h})$ with multiply-valued labels $\mathbf{h}$ to a sequence of binary-valued maximization problems. These sub-problems are called α-*expansions*, and for *regular* energies can be efficiently solved using graph cuts (see [3, 7] for details).

Suppose that we have a current configuration (set of labels) $\mathbf{h}$ and a fixed label $\alpha \in U$ where U is the set of possible label values. In the α-expansion operation, each pixel i makes a binary decision: it can either keep its old label or switch to label α. A binary vector $\mathbf{s} \in \{0, 1\}^V$ defines the auxiliary configuration $\mathbf{h}[\mathbf{s}]$ as follows for all i:

$$h_i[\mathbf{s}] = \begin{cases} h_i & \text{if } \mathbf{s}_i = 0 \\ \alpha & \text{if } \mathbf{s}_i = 1 \end{cases} \tag{4}$$

This auxiliary configuration $\mathbf{h}[\mathbf{s}]$ has therefore transformed the function f with multiple labels into a function of binary variables $f'(\mathbf{s}) = f(\mathbf{h}[\mathbf{s}])$. The global maximum of this binary function can be found using a graph cut.

The expansion move algorithm starts with an initial configuration $\mathbf{h}^0$. Then it computes optimal α-expansion moves for labels α in some order, accepting the moves only if they increase the objective function.

The algorithm is guaranteed to converge. Its output is a strong local minimum characterised by the property that no α-expansion can increase the function f.

Annealed Expansion Move Algorithm

For our problem we wish to encourage the discovery of contiguous regions of part labels that are layout-consistent. Since any part of a regular grid is guaranteed to be layout-consistent, we choose our expansion move to be to a repeating grid of labels at a fixed offset (see figure 4). The total set of expansion moves is the set of possible offsets of this repeating grid (though for efficiency these are quantised to be only every 3×3 pixels). At each iteration, any of the pixels can choose to adopt this new labelling and a region that does so will form a local rigid grid structure. Deformations in the grid can be handled over a number of expansion moves by using labels at nearby offsets. The resultant regions will be layout-consistent and will form a deformed, rather than rigid, grid. This process is illustrated in figure 4 which shows two expansion moves with slightly different offsets being used to label a car with a deformed grid.

The set of expansion moves corresponding to all grid offsets (typically several hundred offsets) is applied in a random order. Additionally, these expansion moves are interspersed with standard α-expansion moves for changing to

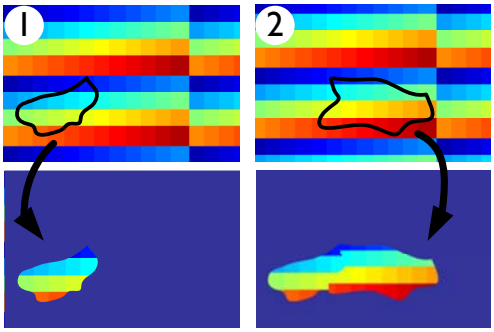

Figure 4. **Expansion move algorithm.** During inference, at each iteration of the expansion move algorithm, a repeated regular grid re-labelling is permitted. Hence, an object can be labelled with a deformed grid over several iterations. Note that the offsets of the grid are different in the two iterations illustrated.

the background label. Our pairwise potentials are not guaranteed to be regular as defined in [3], but in the rare cases where they are not regular (typically $< 0.5\%$ of cases) the potential is truncated to the closest regular potential.

Despite this careful choice of expansion move, the algorithm is vulnerable to getting stuck in local minima due to the strong interaction terms that are present in the model. Highly deformed objects are particularly affected as it takes more expansion moves to reach the optimal labelling. To ameliorate this, an annealing schedule is used: during early rounds of the expansion move algorithm the pairwise potential is weakened (by raising to a power less than one). Experimentally, we have found that for fairly rigid classes (such as cars, faces) the annealing gives a minor but noticeable improvement in performance. However as we move towards evaluating the technique on more deformable objects, such as horses, we anticipate that annealing will become more important.

4. Learning

We learn the potentials in this model using a supervised algorithm which requires a foreground/background segmentation for each training image, but not part labellings.

4.1. Unary Potentials

For the unary potentials, we use randomised decision trees [12] which are both straightforward to implement and very efficient. Using a set of decision trees, each trained on a random subset of the data, increases the efficiency of learning and improves generalization performance over using a single decision tree. For position i in image $\mathbf{x}$, decision tree t_k returns a distribution over the part labels, $\phi_i^k(y_i, \mathbf{x}; \theta)$. The set of K such decision trees are combined by simply averaging these distributions:

$$\phi_i(y_i, \mathbf{x}; \theta) = \frac{1}{K} \sum_{k=1}^{K} \phi_i^k(y_i, \mathbf{x}; \theta). \qquad (5)$$

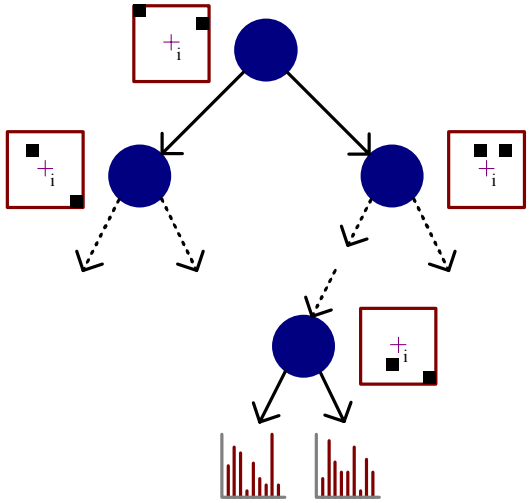

Figure 5. **Decision tree.** Each non-terminal node in the binary tree evaluates an intensity difference or absolute intensity difference between a learned pair of pixels (indicated as black squares), relative to the position of pixel i being classified, and compares this to a learned threshold. The terminal nodes represent distributions over part labels, learned by histogramming all training examples that reach this node.

Each decision tree t_k is a binary tree (illustrated in figure 5), where each non-terminal node evaluates a binary test based on one image feature. We employ two types of feature, chosen for speed: pixel intensity differences, and absolute pixel intensity differences. Each are evaluated relative to the position of pixel i being classified. Both features are constrained to only use pixel information within a box of side D, centered on pixel i. We found that a small D value was critical in achieving good recognition results of occluded objects, since this ensures invariance to occlusions which are further than $D/2$ pixels away. Having two types of feature allows the classifier to detect both image edges and smooth image regions. The intensity difference is compared to a learned threshold, and the left or right branch of the node is taken accordingly. At each terminal node, a distribution over part labels is learned as the histogram of all the training image pixels which have reached that node. Inferring $\phi_i^k(y_i, \mathbf{x}; \theta)$ simply involves traversing the tree, evaluating features relative to position i in image $\mathbf{x}$, and taking the learned distribution at the terminal node reached.

The trees are built in a simple, greedy fashion, where non-terminal node tests are chosen from a set of candidate features together with a set of candidate thresholds to maximise the expected gain in information. This process is halted when the best expected gain in information falls below a threshold ϵ. The time taken to learn the decision trees is dominated by feature calculations and hence almost independent of the number of labels. This will become more important as we move to more classes and labels in future.

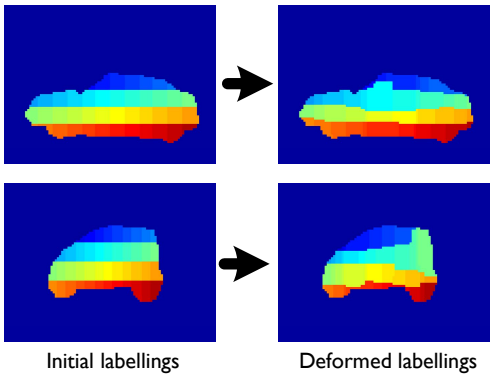

Figure 6. **Deforming the part labellings of training data. Left:** two examples of initial labellings, based on a tight-fitting regular grid. Different colours represent different part labels; since the initial labelling is scan-line ordered, the colours vary slowly along rows and jump between rows. **Right:** the resulting labellings after deformation. Note that corresponding parts (e.g. wheels, rear) are now given corresponding labellings, and hence the unary classifier will learn a tighter appearance model for each part, as desired.

Deformed Labellings

In order to build the unary classifier, a part labelling is required for each training image. We want the unary classifier to learn consistent appearance models for each part, but with deformable objects the part positions vary. To avoid requiring hand-labelled part positions we propose an iterative learning scheme as follows.

The part labelling for the training images is initialised based on a dense regular grid that is sized to tightly fit the bounding box of the object and then masked with the given object segmentation (see figure 6 left). The dense grid is spatially quantised such that a part covers several pixels (on average an 8×8 pixel square). The unary classifiers are learned as described above, after which a new labelling is inferred for all the training images, illustrated in figure 6 right. The deformed labelling is then used to re-learn the unary classifier which can now learn a much tighter appearance distributions for each part. Two iterations of this process were found to be sufficient for good results.

4.2. Pairwise Potentials

The parameters for the pairwise potentials, β_{bg}, β_{oe}, β_{co}, β_{io}, β_{iif}, ν and e_0 are learned using cross-validation, by a search over a sensible range of positive values. The size of our dataset made gradient-based maximum likelihood learning of the parameters too slow to be used in practice. In future we would like to investigate other more efficient means of learning these parameters.

4.3. Instance Potentials

The instance potential look-up tables $\widetilde{P}(h|w)$ for label h at position w are learned as follows. The deformed part labellings of all training images are aligned on their segmentation mask centroids. A bounding box is placed relative to the centroid around the part labellings, just large enough to include all non-background labels. For each pixel within the bounding box, the distribution over part labels is learned by simply histogramming the deformed training image labels at that pixel. A count of one (corresponding to a weak Dirichlet prior) is added to ensure non-zero probabilities.

5. Evaluation

We have evaluated our technique on the UIUC car database [11] for both detection and segmentation performance, and on the Caltech and AR face databases [5, 13] for tolerance to partial occlusion.

5.1. UIUC Car Database

We train on 46 segmented images from the TU Darmstadt database [11] (for the purposes of this paper the car windows were labelled as part of the car), and also a subset of 20 images from the UIUC car database [11], containing one completely visible car instance, which were segmented by hand. To learn the pairwise potential parameters by cross validation, the training set was divided into two halves, and the parameters hand-optimised against one half. The unary potentials were then retrained on the entire training set. The final parameters used were: $\beta_{\text{oe}} = 6$, $\beta_{\text{co}} = 12$, $\beta_{\text{io}} = 12$, $\beta_{\text{iif}} = 30$, $\nu = 0.2$, $e_0 = 0.2$ and $D = 15$.

Detection Accuracy: For testing detection accuracy, the system was evaluated on the remaining 150 previously unseen images. These contain cars under considerable occlusions and rotations, facing both left and right.

Figure 8 shows example detections (and the simultaneously achieved segmentations) achieved by our method. The LayoutCRF detects multiple instances jointly and does not involve any ad-hoc detection mechanisms such as sliding windows, thresholding scores and post-processing

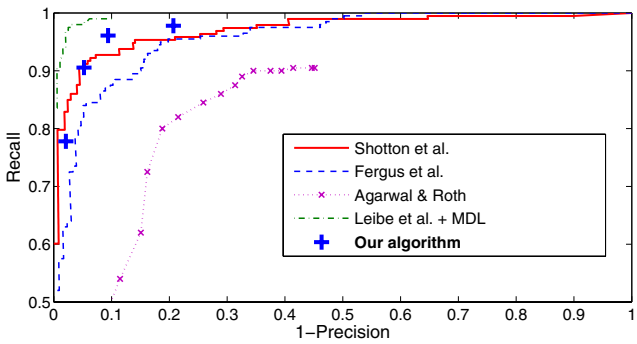

Figure 7. **Recall-precision curves for the UIUC Database.** Note that our unified algorithm does not have a simple threshold that can be varied to generate the whole curve efficiently. Hence we show only four points on the curve for comparison.

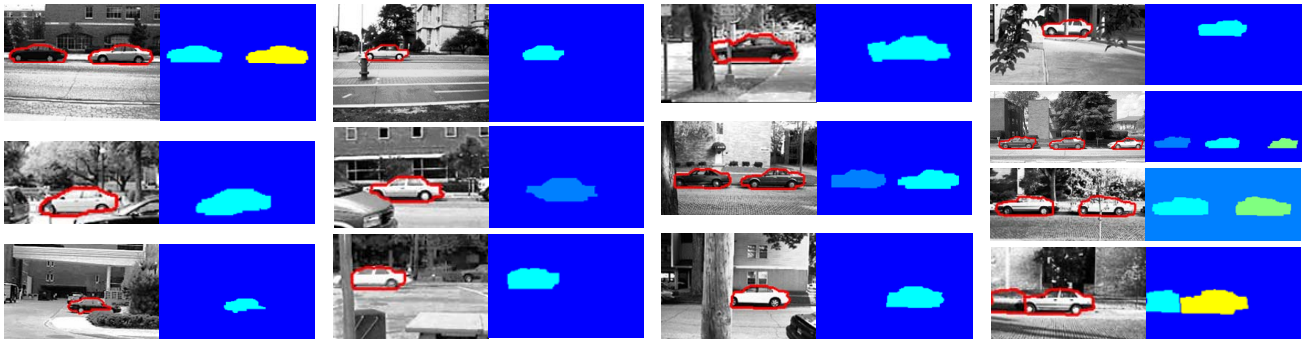

Figure 8. **Example detections and segmentations on the UIUC Database.** In each pair, the left shows the test image with detected object segmentations illustrated in red, the right shows the corresponding instance labelling for each pixel, where colours denote different instances. Note tolerance to partial occlusions, detection of multiple instances, detections facing both left and right, and very accurate segmentations. The bottom right result is an example failure case where a false positive has been detected.

merging steps. However, the disadvantage of our unified method is that generating a full recall-precision curve is extremely slow as the entire inference procedure must be re-run for each point on the curve, using a different value of the background prior β_{bg}. Instead, we show the results of our method as four points (at $\beta_{\text{bg}} \in \{1.65, 1.70, 1.75, 1.80\}$) on the recall-precision axes alongside competing curves for comparison (figure 7). The point closest to equal error rate on the recall-precision curve we achieve is recall=96.1% at precision=89.6%. Note that these figures are not directly comparable with the other techniques as the method was tested on only 150 of the 170 test images; moreover the 20 images removed from the test set and used for training were clean, single-instance images, and so our performance over the 150 remaining test images would be therefore expected to be slightly lower than for the whole database. Also, we discount detected cars instances with fewer than 70% of parts, as the supplied ground truth data does not include labellings of partially occluded cars.

Ignoring these differences, our detection performance exceeds or is highly competitive with that of all competing methods, bar that of [11] with their minimum-description length criterion (without this our recall-precision points lies above their performance). We are currently investigating how our algorithm could be improved in an analogous but more integrated way by imposing a prior on the number of object instances.

Note also that our technique solves a harder problem than the compared methods: allowing occluded cars gives more degrees of freedom and hence a higher error rate than if the model were constrained to detect only unoccluded cars. For example, the false positive in figure 8 is unlikely to have been detected had we been able to assume that there was no occlusion.

Segmentation Accuracy: We evaluated the segmentation accuracy on a randomly chosen subset of 20 of the UIUC test images, containing a total of 34 car instances. These were segmented by hand to produce ground-truth instance labels. Averaged across all image pixels, we achieve a per-pixel figure-ground segmentation accuracy of 96.5%. To get a measure of the segmentation accuracy per instance, we also computed the ratio of the intersection to the union of the detected and ground-truth segmentations areas for each car instance. The average of this measure across all 34 instances was 0.67. Some example segmentations are shown in figure 8. Note that our algorithm produces segmentations that accurately delineate the true object boundary and cope with occlusions correctly.

5.2. Face Databases

We also investigated the performance of the LayoutCRF on the Caltech face database [5] under artificial partial occlusion, using 20 segmented face images for training, and evaluated the same trained model on images from the AR

Figure 9. **Detection and segmentations on the Caltech face database with artificial occlusions (top three rows) and the AR face database with real occlusions (bottom row).** Notice that faces are detected correctly even when significant features (such as eyes) are occluded.

face database [13] containing real occlusions. We show in figure 9 some example results of detection and segmentation where randomly the top, bottom, left, or right half of images were occluded by a uniform grey rectangle. Note excellent detection and segmentation despite significant occlusion; many sparse feature based approaches (which rely heavily, say, on the presence of both eyes) would fail under such occlusion.

6. Conclusions and Future Work

We have presented a novel discriminative model for the detection and segmentation of partially occluded and deformable objects, and shown state-of-the-art performance on the UIUC database. We have also demonstrated excellent tolerance to occlusion on artificially occluded images from the Caltech face database and naturally occluded images from the AR face database. Our approach is efficient: in our unoptimised Matlab implementation, learning takes approximately 30 minutes to learn from 66 images, dominated heavily by the re-labelling process. Testing takes approximately 45 seconds per image to both detect the number of object instances and segment each of these.

Currently the model assumes objects are at a fixed scale, although the deformable part labellings tolerate a range of scales around this fixed scale. For large changes in scale, it would be necessary to learn scale invariant unary potentials and extend the set of transformations T to include a scaling.

One limitation of the model as stated is that there is no incentive for layout-consistent disconnected regions to belong to the same instance. This could be achieved by incorporating a prior on the number of instances and hence favouring hypotheses where consistent disconnected regions correspond to a single occluded object. The addition of such a prior would also improve detection performance by removing false positives where multiple parts of the same car are given different instance labels.

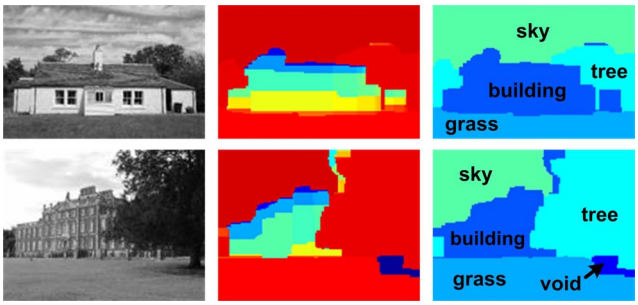

Figure 10. **Examples of multi-class detection and segmentation.** One structured class (building) and three unstructured classes (sky, tree, grass) are accurately detected and segmented using a multi-class LayoutCRF. **Left:** input image. **Middle:** inferred parts labellings. Unstructured classes are assigned only one part label (shown as different shades of red), while structured classes have a grid of part labels. **Right:** output segmentations given by deterministic mapping from the part labellings, with inferred class labels superimposed.

This paper has presented a first investigation into the capabilities of the LayoutCRF model. We plan to further investigate the model performance, in terms of its tolerance to scale, rotation, occlusion, and object deformation or articulation and the effect of varying the numbers of parts. This model can also be applied to multiple classes, both structured (e.g. cars, people) and unstructured (e.g. grass, road); results from a preliminary experiment are given in figure 10. In the multi-class case, the pairwise potential could be enhanced to model local object context (e.g. car appears above road) while also handling inter-object occlusions.

References

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, 2002.

[2] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *Proceedings IEEE workshop on Perceptual Organization in Computer Vision, CVPR 2004*, 2004.

[3] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *Proc. of IEEE ICCV.*, 2001.

[4] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, 2005.

[5] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition*, 2003.

[6] A. Kapoor and J. Winn. Located hidden random fields learning discriminative parts for object detection. In *European Conference on Computer Vision*, 2006.

[7] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, February 2004.

[8] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005.

[9] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *ICCV*, 2003.

[10] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[11] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV'04 Workshop om Statistical Learning in Computer Vision*, May 2004.

[12] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR05*, pages II: 775–781, 2005.

[13] A. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, June 1998.

[14] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS*, 2004.

[15] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *ICCV*, 2005.

[16] M. Szummer. Learning diagram parts with hidden random fields. In *International Conference on Document Analysis and Recognition*, 2005.

[17] O. M. C. Williams, A. Blake, and R. Cipolla. The Variational Ising Classifier (VIC) algorithm for coherently contaminated data. In *Advances in Neural Information Processing Systems 17*, pages 1497–1504. MIT Press, Cambridge, MA, 2005.

[18] J. Winn and N. Jojic. LOCUS: Learning Object Classes with Unsupervised Segmentation. In *ICCV*, 2005.