Noname manuscript No.
(will be inserted by the editor)

Inference and Learning with Hierarchical Shape Models

Iasonas Kokkinos · Alan Yuille

Received: date / Accepted: date

Abstract In this work we introduce a hierarchical representation for object detection. We represent an object in terms of parts composed of contours corresponding to object boundaries and symmetry axes; these are in turn related to edge and ridge features that are extracted from the image.

We propose a coarse-to-fine algorithm for efficient detection which exploits the hierarchical nature of the model. This provides a tractable framework to combine bottom-up and top-down computation. We learn our models from training images where only the bounding box of the object is provided. We automate the decomposition of an object category into parts and contours, and discriminatively learn the cost function that drives the matching of the object to the image using Multiple Instance Learning.

Using shape-based information, we obtain state-of-the-art localization results on the UIUC and ETHZ datasets.

Keywords Inference · learning · hierarchy · contours · grouping · deformable models · shape · parsing.

1 Introduction

Object recognition has recently made major progresses using part-based models. These decompose the problem of detection into the detection of simpler, individual parts which are then combined to detect the whole object. This approach owes its success to its ability to cope with deformations, occlusions and variations in scale, while exploiting sparse image features for efficiency. However, it is still not clear what a part is and what is the best way of composing an object from its parts. During the last decade, parts (or 'visual words') have been defined in terms of image patches [1][2][3], descriptors extracted around interest points [4] [5] [6], edge contours [7–9] or regions [10] [11]. In our understanding, these structures are often not semantically meaningful and should be considered as equivalent to 'letters', instead of 'words'; treating them as parts of objects is to some extent unjustified. For example, modeling a horse in terms of corners, blobs, or junctions deviates from what we perceive as the parts of a horse, namely its torso, head, neck, and feet. It is more natural to define object parts in terms of -potentially recursive- compositions of such simpler structures. Hierarchical representations should thus link the objects with the image information extracted by front-end processing via more abstract, intermediate structures. In our modes we use shape-based parts, which can correspond to semantically meaningful structures such as wheels, handles, or necks. These parts are built by composing several contours extracted from the image.

Hierarchical representations can be supported on at least three different grounds. First, they can expand the representational power of current object models by allowing for structure variation using And-Or graphs [12] and by encoding complex dependencies among object parts with context-sensitive relations [13, 12]. Second, the sharing of parts among several categories is feasible in a hierarchical setting as demonstrated in [14] and can in theory allow vision algorithms to deal with hundreds, or thousands of object categories, with computation demands that scale-up sublinearly in the number of categories. Third, hierarchical models can deal with the problem of combining bottom-up and top-down computation in a tractable manner. The last point is our main claim

Iasonas Kokkinos
Department of Applied Mathematics, Ecole Centrale Paris
and Equipe Galen, INRIA-Saclay
Tel.: +33-01-41131827
E-mail: iasonas.kokkinos@ecp.fr

Alan Yuille
Department of Statistics and Computer Science
University of California at Los Angeles
Tel.: +1-310-2675383
E-mail: yuille@stat.ucla.edu

(a) Object Model (b) Parse Tree (c) Image Tokens (d) Parsed Object

Fig. 1: Object Parsing Task: Our goal is to compose objects using simple tokens (straight edge and ridge segments) extracted from the image. This amounts to building a *parse tree*, that indicates how image tokens are composed to form objects.

in this paper. In our earlier works [15,16] we have been using bottom-up 'proposals', such as the response of a boundary or face detector, to drive the top-down fitting of more complex probabilistic models. Here we use instead *a single model* for both detection and validation; this model is expressive enough to account for the whole object, but thanks to its hierarchical nature can be simplified to deliver efficiently a small set of 'proposals'. These proposals are now part of a single, principled inference algorithm. In the end, using a hierarchical representation allows us to deal with an object having tens of parts in reasonable time.

Our *contributions* in this work are threefold: first, we introduce a hierarchical object representation and a set of grouping rules to recursively compose an object from simpler image structures –a task we refer to as 'parsing'. Second, we present an efficient inference algorithm to perform the parsing task. Third, we describe a method to learn a hierarchical model from minimally annotated images.

Regarding our first contribution, in Sec. 3 we introduce a hierarchical compositional representation for object categories. At the highest level of the hierarchy we have the object and at lowest level the image, which is represented by a sparse set of straight edge and ridge segments. We call these structures 'tokens' to highlight their atomic nature. Our model describes how an object can generate a set of tokens by recursively generating simpler structures. Each object is seen as a small 'grammar', which can explain a part of the image, while detecting an object can be phrased as 'object parsing'. During parsing we create increasingly complex structures by composing simpler ones, starting from the image tokens, as shown in Fig. 1. Apart from localizing an object, we thereby also segment it, as we identify the image structures that it consists of. In Sec. 4.1 we formally describe parsing and present a simple bottom-up parsing algorithm for composing an object with minimal cost.

Our second contribution addresses the problem of efficiently finding good object parses. In Sec. 4.2 we describe an inference algorithm that deals with the huge number of candidate compositions. For this, we exploit the hierarchical representation to devise a coarse-to-fine detection algorithm that was inspired by the Generalized A*algorithm of [17]. Our algorithm first uses a simplified version of the representation for a quick initial detection, and then refines the search at a few promising locations. This scheme integrates the bottom-up information extracted from the image with the top-down guidance of the object model. which results in a substantial speedup in detection time.

Our third contribution, described in Sec. 5 consists in learning an hierarchical model from training images where only the bounding box of the object is known. We break this task into three subproblems: initially we recover the object contours by bringing the training images into registration, using an automatically learned deformable model. Then, we group contours into object parts by combining perceptual grouping with Affinity Propagation. Finally, we learn the parameters of the cost function that drives parsing using Multiple Instance Learning. As demonstrated in Sec. 6, this yields state-of-the-art shape-based detection results.

2 Previous Work

We first give a brief overview of the current state-of-the-art on part-based models for object detection, and then focus on research that is most closely related to our work on: (a) representation, learning and inference with hierarchical and compositional models, (b) efficient optimization for detection and (c) contour-based image and object representations.

2.1 Part-based models

Based on the image structures used to represent parts most approaches can be classified as interest point & descriptor-based [18][4][5][19][20][21], patch- or filter-based [1][22][23], contour-based [7][8][9][24][25] or region-based [10] [11] [26] [27]. The constraints among the relative locations of parts are commonly expressed using graphical models, e.g. star graphs [7][9][23][28], trees [1], k-fans [29,22] or fully-connected

graphs [5]. Other works, for example the Implicit Shape Model [30] or boosting-based approaches [8][25] include these constraints implicitly, while 'bag-of-words' models discard location and form a histogram of parts [19][20][21] or allow each model part to match several image parts [31].

The works above involve a diverse set of techniques for image and object representation, training and inference; for lack of space, we now present at some further extent only works to which our research is more closely related. Some technical differences will be clarified during the presentation of our approach in the following sections.

2.2 Grouping and Hierarchical Models

The syntactical/compositional approach to recognition has its roots in the works of Fu and coworkers [32], while hierarchies also underly the approach to recognition proposed by Marr [33]. However, grouping-based approaches to recognition were primarily applied to rigid objects [34] and were hindered by the limited feature extraction and statistical modeling tools available at the time. Therefore, with the exception of research on recognition from binary shapes [35][36][37] or articulated person detection [38], the ideas of compositionality and grammars had long been ignored. However recent influential works such as [15,12,13] and recognition systems such as [39,40] which can simultaneously deal with multiple categories have resulted in increased interest.

The work of S.C. Zhu and coworkers [12,41] aims at accurately modeling complex object categories by relying on And-Or graphs to account for structure variation and by using context-sensitive relations among parts. Learning extends the FRAME model [42] to include relations among parts and to deal with attributes such as position, scale and orientation, while inference relies on the bottom-up/top-down scheme developed in [15]. There, discriminative techniques such as Adaboost [15] or RANSAC [43] provide proposals, which are then refined by generative models in a top-down manner. Instead, we use the same model to suggest object locations during coarse-level search and for validation at a finer level, in an integrated optimization algorithm.

Jin and Geman [13] use a perturbation of a context-free model that incorporates context-sensitive relations among parts through attribute functions. Their model involves discrete variables that encode 'part-of' relationships at different levels of the object hierarchy and they resort to greedy optimization to deal with the NP-hard problem emerging from the context-sensitive relations. The authors argue that using continuous attributes is hard in a Markov system; however our inference scheme allows us to incorporate the continuous variables of location and scale in our model.

Todorovic and Ahuja use segmentation for both learning and detection [26] and to exploit the shared structures among similar categories [39,44]. During training, object models are discovered by computing the maximal common subtree from a set of hierarchical image segmentations. During testing, the hierarchical segmentation of a new image is matched to the learned tree, which allows for the simultaneous detection and segmentation of objects. This method deals with changes in scale, orientation and multiple categories, but heavily relies on hierarchical segmentation, while it still lacks a clear probabilistic interpretation.

In the work of Fidler and Leonardis [40,14] a hierarchical representation of objects is learned using edge-based information. For this a Gabor filterbank is used to capture the image information at the lowest level, and the intermediate layers are obtained by hierarchical clustering. This leads to increasingly complex structures until in the end forming the whole object. Still, this work does not clearly optimize a criterion either during training or detection.

Related work by L. Zhu et. al. [45] gives a probabilistic formulation for leaning a hierarchical representation by learning probability models of substructures which are composed together to form larger structures. This gives good performance on detection tasks.

Finally, in the same way that Conditional Random Fields (CRFs) have replaced Markov Random Fields in low-level image labelling, recently discriminative training of high-level models has gained ground, as it allows to 'tune' the model for the task at hand, e.g. parsing or classification. Initially the work of [46] used CRF training to increase the likelihood of the ground-truth body poses under a graphical model. In [47] an algorithm was proposed to train CRFs with hidden layers so as to maximize the likelihood of the class labels. In the work of L. Zhu et. al. algorithms developed for language parsing, namely Max-Margin parsing [48] and structure perceptron [49] were used for body parsing [50] and deformable model training [51] respectively. Finally, in [28] latent SVMs were proposed for training a star-graph model for detection, which as we describe in Sec. 5.3 is closely related to our Multiple Instance Learning approach.

2.3 Efficient Optimization for Object Detection

Even though efficient detection algorithms exist for detection using global object models, e.g. [52,53], the problem becomes harder when part matching gets involved. The combinatorics of matching have been extensively studied for rigid objects [34], while [54] used A* for detecting object instances. In [21] branch-and-bound is used for efficient detection with a bag-of-words model, while [55] combine Graph-Cuts with branch-and-bound for object segmentation.

An efficient algorithm for detection of a single closed contour is presented in [56]. In [17] the detection of geometric structures, such as salient open and convex curves is

formulated as a parsing problem. In our work we extend this approach to deal with high-level structures, i.e. objects with many parts and of potentially different types.

In the work of [57] a pruned version of dynamic programming is used to efficiently detect deformable objects. This involves a rough initial detection which is then refined in a top-down manner. At a high-level this is similar to our method, but our work has been based on the A*algorithm which has guaranteed optimality properties. In the more recent work of [58] a Steiner tree formulation is introduced for learning and performing inference on an hierarchical model. The authors use approximate optimization to identify the optimal manner of putting together low- and intermediate-level structures within images of an object category. In our work we focus on the more limited problem of performing inference with a predetermined hierarchical model, which allows us to perform exact optimization of a cost function defined on a fixed tree-structured graph.

2.4 Contour-Based Representations

Edges are largely invariant to intra-class appearance variations and were therefore used early on for object recognition [59,60,34]. However, boundary detection is still an open problem, due to occlusions, noise, etc., while describing and matching contours is challenging. Therefore starting from [6,61] most recent works use point-based image representations, which are easier to extract, model, and match.

A revival of interest in contour-based representations has been observed lately however, due to the increasingly better performance of boundary detection methods on ground truth datasets [62], and the understanding that contours are better suited to capture shape information than points. The technique that is currently most commonly used for object detection using contours is that of forming codebooks [8,9, 63,24]; there 'contour templates' are formed during training by clustering, and during testing the observed contours are matched to these templates. However, as observed in [9],[64],[65] this lets the contour segmentation problem creep in the model representation: this results in several different codebook entries encoding essentially the same structure. Therefore, in this work we cater for the fragmentation problem by developing a simple and efficient algorithm for matching broken curves, detailed in Sec. 3.2 and App. A.

Another approach that relies on contours for detection is that of [66]. There long, smooth contours are put together by phrasing their grouping into objects as an optimization problem that involves the context of each contour within the object. This context-sensitive approach avoids the problems of forming a codebook and is shown to give results of high precision, but involves solving a computationally demanding optimization problem. Our approach is context-free, which facilitates our efficient detection algorithm.

3 Hierarchical Object Representation

In this section we introduce our hierarchical representation for objects. At the highest level of our object hierarchy lies the whole object. One level below are the object parts; these should intuitively correspond to semantically meaningful parts, such as fingers, wheels, or legs. We use the term 'object part' for structures at this level in the hierarchy, while 'part' on its own will have the broader meaning used in vision. The object parts are decomposed into (potentially) long and curved contours. At the lowest level we have simple image structures, namely straight edge and ridge segments ('tokens').

We phrase object detection as the interpretation of some of the image tokens in terms of an object; in quantifying the quality of this interpretation we start in this section from a probabilistic point, and describe what the interpretation cost would be under a suitable generative model. In Sec. 5.3 however we will see how to 'tune' this cost in a discriminative setting, so as to optimize detection performance.

3.1 Object Model

Our object representation consists of a graph structure with nodes $i \in V$ and edges $(i,j) \in E$; the nodes correspond to the structures in the hierarchy and the edges to the part-subpart relations. Each node i lies at a certain level of the hierarchy, and is connected to a single parent node $pa(i)$ at the level above, and several children nodes $ch(i)$ at the level below. By $ch^*(i)$ we denote all descendants of node i. The graph has three levels – root node V_r, object parts V_p, and object contours V_c. The parent of the root node is empty, while the children of the contour nodes are the edge and ridge tokens; these are our observations $\mathbf{I}$.

Each node i is associated with a *pose*, namely a continuous, vector-valued state variable $\mathbf{s}_i = (\mathbf{x}_i, \log\sigma_i)$ describing its position $\mathbf{x} = (x_1, x_2)$ and scale σ. By position we mean the coordinates of the part's center, while scale is understood as the difference in size between the observed and the corresponding, 'nominal' structure. The probability of an object configuration $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_N)$ can be expressed by a Bayes net $P(\mathbf{S}) = \prod_{i\in V} P(\mathbf{s}_i|\mathbf{s}_{pa(i)})$, or by a more general exponential form:

$$P(\mathbf{S}) = \frac{1}{Z[\lambda]} \exp\left(-\sum_{i\in V} \phi_i(\mathbf{s}_i, \mathbf{s}_{pa(i)})\right), \qquad (1)$$
$$\phi_i(\mathbf{s}_1, \mathbf{s}_2) = -\lambda_i \log P(\mathbf{s}_1|\mathbf{s}_2)$$

where for a Bayes net $\lambda_i = 1,\ \forall i$ and $Z = 1$.

The $P(\mathbf{s}_i|\mathbf{s}_{pa(i)})$ terms describe the distribution of a child's pose given the pose of its parent. The relative pose of the child is estimated as $\mathbf{s}_{i|pa(i)} = \mathbf{s}_i - \mathbf{s}_{pa(i)}$. We use a Normal distribution $N(\mu_{i,pa(i)}, \Sigma_{i,pa(i)})$ for the relative loca-

tion and another normal distribution $N(0,.1)$ for the scale coordinates, allowing for moderate relative scale changes. [1]

At the lowest level of the hierarchy the pose of an object contour $\mathbf{s}_i$ is related to a group of image tokens $\mathbf{h}_i$ using an observation potential $\psi_i(\mathbf{I}_{\mathbf{h_i}}, \mathbf{s}_i)$, detailed in Sec. 3.2, that compares the object contours to the image contours. This provides us with the data-fidelity term for our model, involving the poses of the contour nodes V_c, the image tokens $\mathbf{I}$, and the assignment variables $\mathbf{H} = (\mathbf{h}_i)$:

$$P(\mathbf{I}|\mathbf{S},\mathbf{H}) = \frac{1}{Z}\exp(\sum_{i\in V_c} -\psi_i(\mathbf{I}_{\mathbf{h_i}}, \mathbf{s}_i)). \quad (2)$$

We allow for missing parts at any level of the hierarchy, using a binary variable $\mathbf{y}_i$ that indicates if node i is observed. When a node is missing, i.e. $\mathbf{y}_i = 0$ we replace every summand in Eq.s 1,2 that involves either node i or a descendant $j \in ch^*(i)$ with a 'missing' potential function $\phi_j^0 = -\log P(\mathbf{y}_j = 0)$ that 'penalizes' the missing node j. So conditioned on the parent being missing, all descendants are forced to be missing; if the parent is present the probabilities of missing its children are considered independent.

Combining all terms we write:

$$P(\mathbf{I},\mathbf{S},\mathbf{H},\mathbf{y}) = P(\mathbf{I}|\mathbf{S},\mathbf{H},\mathbf{y})P(\mathbf{S},\mathbf{H},\mathbf{y}) \propto \exp(-C(I,\mathbf{S},\mathbf{H},\mathbf{y}))$$

$$C(I,\mathbf{S},\mathbf{H},\mathbf{y}) = \sum_{i\in\{V_p,V_c\}} \left(\mathbf{y}_i\phi_i(\mathbf{s}_i,\mathbf{s}_{pa(i)}) + (1-\mathbf{y}_i)\phi_i^0\right) + \sum_{i\in V_c} \mathbf{y}_i\psi_i(\mathbf{I}_{\mathbf{h_i}},\mathbf{s}_i) \quad (3)$$

$$= \sum_{i\in V_p} \mathbf{y}_i\phi_i(\mathbf{s}_i,\mathbf{s}_{V_r}) + (1-\mathbf{y}_i)\sum_{j\in\{i,ch^*(i)\}}\phi_j^0 + \sum_{i\in V_p}\mathbf{y}_i\sum_{j\in ch(i)}\mathbf{y}_j\left[\phi_j(\mathbf{s}_j,\mathbf{s}_i) + \psi_j(\mathbf{I}_{\mathbf{h_j}},\mathbf{s}_j)\right] + (1-\mathbf{y}_j)\phi_j^0 \quad (4)$$

We note that the first term in Eq. 3 would be identical to Eq. 1, if we set $\mathbf{y}_i = 1, \forall i$, i.e. if we consider all parts as being present. Now if part i is missing, $\mathbf{y}_i$ will equal zero, so instead we pay the cost ϕ_i^0 of missing part i. Using the latent variable vector y thus allows us to compactly incorporate missing parts inside our cost function. The second term in Eq. 3 corresponds to the data-fidelity term in Eq. 2.

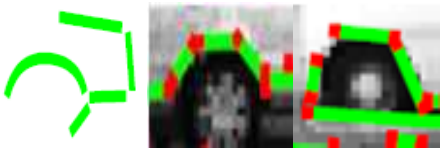
The expression in Eq. 4 is a rewrite of Eq. 3 which underlines that the cost can be computed recursively on the tree-structured graph. In specific, it breaks up the parent-child relationships into root-part and part-contour groups. If an object part i is missing the $1-y_i$ factor enforces penalties for missing i as well as all of its descendants; if a contour j is missing the $1-y_j$ factor penalizes it by ϕ_j^0.

In the rest of the paper we will be dealing with this cost function. In specific, during inference (Sec. 4) our unknowns are the pose $-\mathbf{S}$, assignment $-\mathbf{H}$ and missing part $-\mathbf{y}$ variables and our objective is to minimize Eq. 3 with respect to them. Note that due to the independence assumptions we made when formulating our model the cost function is decomposed into simpler terms that can be optimized separately. This underlies our inference algorithm.

During training (Sec. 5), we use data that contain only the bounding box of the object to learn our hierarchical representation. This includes, first, the structure of the model, namely its contours and their grouping into object parts. Second, the $\mu_{i,pa(i)}, \Sigma_{i,pa(i)}$ parameters involved in the parent-child relationships; these are estimated using maximum likelihood from registration information (Sec. 5.1). Third, the λ_i parameters related to the ϕ potentials in Eq. 3. In specific, note that the binary potentials in Eq. 1 are obtained as the product of $P(\mathbf{s}_i|\mathbf{s}_{pa(i)})$ with a parameter λ_i which can vary across i, namely across different parent-child relationships. Therefore, the distributions used here constrain only the form of our energy function (for instance, it is quadratic in $\log s_i - \log s_{pa_i}$) while its exact expression is obtained after learning the λ_i parameters (Sec. 5.3). Moreover, we also learn the missing cost potentials ϕ_i^0 used in Eq. 3, and the parameters of the observation potentials ψ_i described in the following subsection. All of these quantities are estimated discriminatively as described in Sec. 5.3.

We introduce our representation as a graphical model, but we can also think of it as a simple probabilistic grammar: the production rules start at the root node and generate the object parts, then the object contours, and finally the edge and ridge tokens. All production rules are probabilistic and involve continuous attributes. The missing part variables allow us to choose among observing or not a part, thereby implementing a simple version of the 'OR-ing' advocated in [12]. This could be extended to a mixture distribution on parts, (e.g. a chair having 4 feet versus a chair with wheels), but we leave this for future research. Structures at a certain layer of the hierarchy can only be built from structures at the layer below, so there can be e.g. no infinite recursion as is the case for language grammars. Finally, the grammar is context-free, as we assume independence among the subparts given the part at the layer above. The last three points indicate that we are exploiting only part of the grammatical formalism's potential; still, we present state-of-the-art results on the image categories we experiment with.

[1] Note that by 'scale' of a part we mean the ratio of the part's size (e.g. the radius of a disk) to the part's nominal size in the object template. The scale coordinate in the relative pose equals $\log(\sigma_i/\sigma_{pa(i)})$; in words, we first measure separately how larger the parent and the child are from their nominal scales (σ_i and $\sigma_{pa(i)}$, respectively), and constrain the ratio of these scales to be close to one. The prior of the pose tolerates moderate changes in relative scale; large scale changes can be accommodated for by modifying the root node's scale.



(a) (b) (c)

Fig. 2: Contour fragmentation problem: the model arc in (a) has to match the fragmented observations in (b) and (c).

3.2 Image-to-Object Contour Matching

We now describe how we relate the nodes at the lowest level of the hierarchy (object contours) to the image tokens via the observation potential term in Eq. 2. A design choice in our model is that we allow object contours to be long and correspond to large groups of edge or ridge points (edgels). Apart from higher discriminative power, this also deals with the *contour fragmentation* problem.

In specific, the grouping of edgels into edge tokens is difficult and ambiguous. Suppose that we want an object contour to represent the wheel of a car, shown on Fig. 2(a). As shown in Fig. 2(b),(c), the low-level grouping of edgels can give different edge tokens on relatively similar images.

For this we propose a two-step approach: first we form groups of edge and ridge contours, starting from Lindeberg's primal sketch [67] (details can be found in App. B). We then match these groups efficiently to the (long) object contours. The first step bypasses contour fragmentation by regrouping the image tokens into longer contours. This allows us to have long contours in our object representation, instead of forming codebooks of contours as e.g. in [8,9,63,24].

A crucial problem is to match these groupings with the model curves. We want to compare contours efficiently and in a way that can deal with missing parts, small deformations and changes in orientation. For this we now build a similarity criterion that accommodates these kinds of variation and can also be optimized efficiently. We describe object and image contours using their tangent function $\theta(s)$, parameterized in terms of the contour's arc-length, s. A model contour θ_M and an observed contour θ_O can then be registered using three parameters: rotation by c degrees amounts to adding a constant c to the tangent function, scaling amounts to dividing the arc-length by α, while adding/subtracting τ to the arc-length of one curve registers the two curves to reduce the length of missing parts/protrusions.

The scaling and offsetting is applied to the observed curve θ_O; the support of the transformed function depends on these parameters, and will be henceforth called S_O, temporarily assuming fixed values for α, τ; the support of the model curve is denoted by S_M. The intersection of the supports of the two curves is denoted by $S_c = S_O \cap S_M$, while their set-difference is denoted by $S_d = (S_O \cup S_M) \setminus S_c$. Our matching cost for two curves writes:

$$E_{\theta_O,\theta_M}(\alpha,\tau,c) = \int_{s\in S_c} (\gamma_1 \left[\theta_O(\frac{s}{\alpha}+\tau) - \theta_M(s) + c\right]^2 + \gamma_2 c^2) ds + \gamma_3 |S_d|. \quad (5)$$

The first term is the square norm between the observed and model angles, and penalizes differences in the tangent angle of the registered curves on their common domain; the term $\gamma_2 c^2$ acts as a penalty on wide rotations; and $|S|$ denotes the length of S, so $\gamma_3|S_d|$ equally penalizes protrusions and missing curve parts. We emphasize that in Sec. 5.3 we discriminatively estimate the degrees of flexibility $\gamma_1, \gamma_2, \gamma_3$ separately for each contour.

We evaluate the similarity of two contours θ_O, θ_M as:

$$E^*_{\theta_O,\theta_M} = \min_{\alpha,\tau,c} E_{\theta_O,\theta_M}(\alpha,\tau,c). \quad (6)$$

As detailed in App. A, since our image contours are formed by grouping piecewise straight edge segments we can evaluate the quantity in Eq. 5 in constant time, instead of linear in the length of the contours. This allows us to perform the optimization in Eq. 6 over α, τ using brute force search, while the minimum over c is obtained analytically.

Putting things together, consider that we want to find matches for the model contour i. Matching a group of image edges to the model contour i amounts to determining that contour's assignment variables $\mathbf{h}_i$. Each group $\mathbf{h}_i$ yields a different angle function θ_O, denoted as $\theta_{\mathbf{h}_i}$. As an example, if $\mathbf{h}_i$ groups tokens k and m, of lengths l_k and l_m, angles θ_k, θ_m and with in-between gap g, we would have

$$\theta_{\mathbf{h}_i}(s) = \begin{cases} \theta_k & 0 \le s \le l_k \\ \theta_m & l_k + g \le s \le l_k + l_m + g \\ \text{undefined} & \text{elsewhere} \end{cases} \quad (7)$$

Once we form the $\theta_{\mathbf{h}_i}$ function we can efficiently find the α, τ, c variables that minimize $E_{\theta_O,\theta_M}(\alpha,\tau,c)$, using the technique described in App. A. From the optimal value of α, τ, c in Eq. 5 we obtain the observation potential for the model contour i in Eq. 3:

$$\psi_i(\mathbf{I}_{\mathbf{h}_i}, \mathbf{s}^*_i) = E_{\theta_{\mathbf{h}_i},\theta_i} = \min_{\alpha,\tau,c} E(\alpha,\tau,c). \quad (8)$$

For a certain assignment $\mathbf{h}_i$ the pose $\mathbf{s}^*_i$ is obtained by setting the scale of the node equal to the estimated α and its location equal to the center of the model arc, estimated numerically based on the α, τ parameters.

A caveat of this is that we choose a single α per image group-object contour combination, namely the one that minimizes the matching cost. It may be better to have a high matching cost if it gets balanced with an agreement with the overall pose of the object. We therefore perform separate searches in different ranges of α, and perform separate detections for each such range. In specific for an object lying at scale 1 we use as search range $\alpha \in \{.7, 1, 1.3\}$. For different object scales we scale appropriately the range of α's.

The main advantage of this contour-based approach is that on the one hand we have continuous models for contours, represented as 1D functions of arc-length, and at the same time we can work with a sparse image representation. Our model can thus capture a large part of the object boundaries, while working with a small set of image structures.

4 Inference: Efficient Object Parsing

During detection we find the set of tokens and part poses that minimize the cost function in Eq. 3. Our object representation is a tree-structured graphical model, so we could use a message-passing algorithm such as Max-Product for this. However, the data likelihood terms are multi-modal and cannot be approximated with Gaussians. Performing message-passing would require either discretization, whose complexity scales in the best case linearly with the size of the image [1] or particle filtering [68] which would however require a huge number of particles if no initialization is provided.

We propose to exploit two aspects of our representation to perform inference more efficiently. First, in Sec. 4.1, we describe how to exploit its compositional nature to build up the whole object from a small set of sparse image tokens. We thus ignore the vast portion of the image where no tokens are present. Second, in Sec. 4.2 we exploit the hierarchy in our model to perform detection in a coarse-to-fine manner: we use our model to quickly identify a few promising areas, to which more computational resources are then devoted in a top-down/bottom-up computation setting inspired from the Generalized A*parsing of [28].

4.1 Bottom-Up Object Parsing

4.1.1 Recursive Structure Instantiation

Our approach to inference exploits the sparse, edge- and ridge- based representation of the image. Starting from these elementary tokens we aim at building the whole object by recursively composing its intermediate structures.

To formalize this, we describe a structure j at a certain level in terms of its pose and its constituent parts:

$$S^j = (\mathbf{s}^j, P^1, \dots P^N). \quad (9)$$

S^j is the description of the structure, $\mathbf{s}^j$ is its pose, described in Sec. 3.1 and $P^1 \dots P^N$ are the descriptions of the children; if a child is missing, its description is empty, $P = ()$. Otherwise it contains in turn the subpart's own pose, subparts, and so on. At the lowest level, a contour is described by its pose and the indexes of the tokens assigned to it. For example the description of an object with two parts, with the first having two and the second having one contour would have this LISP-like form:

$$S^{V_r} = (\mathbf{s}^{V_r}, (\mathbf{s}^{V_{p,1}}, (\mathbf{s}^{V_{c,1}}, \mathbf{h}_1), (\mathbf{s}^{V_{c,2}}, \mathbf{h}_2)), (\mathbf{s}^{V_{p,2}}, (\mathbf{s}^{V_{c,3}}, \mathbf{h}_3))),$$

where $\mathbf{s}^\nu$ is the pose of a structure, r, p, c stand for root, object part, and contour respectively, and $\mathbf{h}_i$ is the set of image tokens assigned to contour i. We call a set of values of the pose/assignment/missing part variables an 'instantiation' of the structure. An instantiation S^j of a structure j comes at a certain cost: for an object structure this will be the cost in Eq. 3, while for intermediate-level structures this will be their contribution to the overall cost.

Note that the cost is additive and defined on a tree structure, as can be seen from Eq. 4. We can therefore define it using the following recursion:

$$C(S^j) = \begin{cases} \sum_{i \in \{j, ch^*(j)\}} \psi_i^0, & \text{if } S^j = () \\ \psi_j(\mathbf{I}_{\mathbf{h_j}}, \mathbf{s}_j), & \text{if } j \in V_c \\ \sum_{i \in ch(j): P^i \neq ()} \phi_i(\mathbf{s}_i, \mathbf{s}_j) + \sum_{i \in ch(j)} C(P^i), & \text{else.} \end{cases}$$

In words, if a structure is missing, $S^j = ()$, its cost is equal to the cost of missing the structure and all of its descendants. Otherwise, if it is a contour, its cost is equal to its matching cost with the tokens assigned to it. And if it is a higher level structure, its cost is the sum of the spatial consistency cost $SC(S^j) = \sum_{i \in ch(j), P^i \neq ()} \phi_i(\mathbf{s}_i, \mathbf{s}_j)$ between the structure and its children and the recursively defined costs of its parts. Missing parts do not have a spatial consistency cost, and are therefore excluded from the summation for SC.

The goal of inference is to find low-cost instantiations for the whole object ('goal' structure). We refer to an instantiation of the goal structure as a *parse* for the object, namely a relation between structures extracted from the image and the object parts and subparts. There is a huge number of instantiations, corresponding to different assignments of image tokens to contours, poses of object parts, or missing parts, while our goal is to efficiently explore the small subset of these that has low cost. In this subsection we phrase the computation of instantiations as a 'bottom-up' algorithm; in the next subsection we describe how this algorithm can be sped up using 'top-down' guidance.

At iteration 1, we estimate the matching cost of each group of image tokens to each object contour. If the cost is below the penalty paid for missing the contour, we instantiate the contour with its pose estimate and the edge tokens. Otherwise it is cheaper to consider the contour as missing.

At iteration $k+1$ we instantiate structures lying at level $k+1$ of the hierarchy. The previous iteration provides possible instantiations $\mathcal{S}_{k,i} = \{(), s_1, \dots, s_{N_{k,i}}, \}, i = 1 \dots |V_k|$, for each part i at level k. Including the empty element allows for missing parts. A structure j can instantiate its N parts from $\mathcal{S}_{k,1} \dots \mathcal{S}_{k,N}$:

$$S^j = (\mathbf{s}^j, P^1, \dots, P^N), \quad P^1 \in \mathcal{S}_{k,1}, \dots P^N \in \mathcal{S}_{k,N} \quad (10)$$

and its cost is estimated recursively in terms of the part costs. The location is set to the value that minimizes the spatial consistency cost, $SC(S^j)$ given the observed children

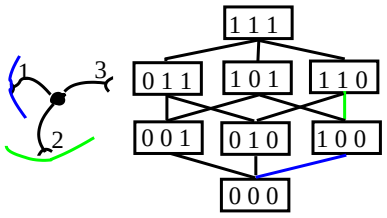


Fig. 3: Hasse diagram for a 3-part structure: structure with more constituents are closer to the top. The binary vector inside the box indicates which parts have been found.

poses:

$$\mathbf{x}^j = \left(\sum_{i:P^i \neq ()} \Sigma_{i,j}^{-1} \right)^{-1} \sum_{i:P^i \neq ()} \Sigma_{i,j}^{-1} (\mathbf{x}^i - \mu_{i,j}), \qquad (11)$$

and the log-scale coordinate is set equal to the mean of the part coordinates.

4.1.2 Ordering of Compositions

We now propose an algorithm to deal with the large number of candidate compositions. It is also useful in formulating a 'coarsening' of the parsing problem in Sec. 4.2.

The main idea is to put parts together 'one at a time': Consider a structure S with N parts $S = (\mathbf{s}, P_1, \ldots, P_N)$. Initially we set $P_i = *, \forall i$, where $*$ is a special 'dummy', non-yet instantiated part. This means that there are initially no state variables assigned to the nodes, and is not the same as having a missing part. We gradually build structures by applying composition rules of the form: $(S, \mathcal{C}) \xrightarrow{r_i} S'$, where $\mathcal{C} \in \mathcal{S}_{k,i}$ is the new constituent, attached to part i of S; r_i is the rule used to instantiate part i, and is applicable to S only if its i-th part is not-yet instantiated. E.g. for $i = 2$ we would have $S = (\mathbf{s}, S_1, *, S_3)$ and $S' = (\mathbf{s}', S_1, \mathcal{C}, S_3)$.

These rules are similar to the Greibach Normal Form [69] where production rules decompose a nonterminal into a terminal and a nonterminal. Here we compose a structure of layer $k+1$ using a structure of layer $k+1$ and a structure of layer k.

The cost of the structure is updated based on the cost of $\mathcal{C}$ and the change in the spatial-consistency cost:

$$C(S') = C(S) + C(\mathcal{C}) + SC(S') - SC(S). \qquad (12)$$

Initially, when $S = (*, \ldots, *), C(S) = 0$. Note that $\mathcal{C} \in \mathcal{S}_{k,i}$ means $\mathcal{C}$ can also be empty as $() \in \mathcal{S}_{k,i}$. This amounts to missing the i-th part; in that case $SC(S') = SC(S)$ and the cost of S' is increased by the cost of missing its part.

This way of composing structures introduces a *partial ordering* $\preceq$ among them, with $P^i \preceq P^j$ if P^j has all the parts of P^i; the ordering is partial as two structures are incomparable if e.g. each has a part that the other does not. This ordering can be visualized with a Hasse diagram [70], as shown in Fig. 3 for a 3-part structure. In this diagram boxes correspond to structures, and when two structures are connected the one lying higher has more elements than the one below it. Gradually building up structures amounts to following a path that starts from the minimum element and gradually goes to its maximum.

The number of paths that can be followed equals $\prod_{i=1}^{N} |\mathcal{S}_{k,i}|$, where N is the number of the structure parts, and $|\mathcal{S}_{k,i}|$ the cardinality of the candidate subparts. We deal with this potentially huge number with an algorithm described by the following Matlab pseudocode:

```
function cmp = Compose(parts,CostMaxPart,CostStructure)
cmp = [];  % compositions
for i =  1:length(parts),
    % compose old structures with i-th part
    cmp_new             = Compose(cmp,parts(i));
    % upgrade i-th part (parts 1:i-1 are missing)
    upgraded_part       = Upgrade(parts(i));
    cost_up             = sum(CostPart(1:i-1));
    upgraded_part.cost  = upgraded_part.cost + cost_up;
    % penalize missing i-th part in old structures
    cmp.cost            = cmp.cost + CostPart(i);
    cmp                 = [cmp,cmp_new,upgraded_part];
    below               = find(cmp.cost < CostStructure);
    cmp                 = cmp(below);
    cmp                 = nonminimum_suppress(cmp);
end
```

Before each iteration i the structures formed so far ('cmp') have only parts $1 \ldots i-1$ instantiated, while parts $i \ldots N$ are in the 'dummy', non-instantiated state. At the i-th iteration, we combine these with structures that can correspond to their i-th part ('parts(i)'). Then we allow each part to directly form a structure at the higher level (Upgrade function); but this implies that all of the parts $1 \ldots i-1$ will be missing, so we penalize missing them. We also keep the old structures in our pool of structures, but augment their cost to account for missing part i.

We finally merge the upgraded parts with the compositions and then compare the composition costs to the cost of missing the whole structure. Those that have higher cost are rejected: their cost can only increase by the further application of rules, so it will be cheaper to treat the whole structure as missing instead. We order parts so that the ones with higher missing cost come first. This quickly rules out weak compositions that do not include them.

Finally we keep only the cheapest composition within a small neighborhood (nonminimum suppression), to avoid dealing with multiple candidates with higher costs. These two steps can drastically reduce the number of utilized compositions. In the next subsection we show how to further limit their number using top-down information.

4.1.3 Caveats

We mention two caveats of our method. First, we assume that for a given set of subparts, the pose of the part can only take one value, the one for which the probability of its subparts is maximized. This ignores all other possible part poses, which could potentially lead to an overall lower cost.

This would be the case if a different pose estimate turns out to be in better accord with the parent's pose. We take this shortcut as it avoids exploring all possible locations for the pose of a part. In practice, the provided poses estimates look reasonable. Further, we expect that during training we can account for the systematically larger spatial consistency costs that are potentially introduced by this process.

Second, when performing suppression we ignore some of the alternative structures which have high cost, but could potentially lead to cheaper overall compositions based on their better spatial configuration. This becomes prominent if a large suppression neighborhood is used, but is negligible for a small neighborhood. In specific, the difference in the location cost is a quadratic function in the difference between the original coordinates and the ones resulting from suppression - therefore for smaller neighborhoods this cost becomes negligible. As we describe in Sec. 4.2.4, we keep track of the suppressed nodes so that we initially perform suppression at a large scale, but then reexamine at a smaller suppression scale the interesting image regions.

4.2 Hierarchical Object Parsing

The inference scheme we have described so far is entirely bottom-up, i.e. forming first all low-level structures and then using them to assemble higher-level ones. This however can be 'myopic', as it forms numerous compositions of a single object part, before checking if they can be used to build the whole object. For example if we cannot form the trunk and the cabin of a car somewhere in the image, we should quit forming compositions of the engine part there.

Our strategy for dealing with this consists in first performing a quick detection of the object by composing only roughly certain layers of the hierarchy. We then use these results to guide detection at a finer level. The first step quickly rules out a big portion of the image, and helps devote resources to promising areas. This augments the 'bottom-up' computation with 'top-down' guidance. Recent advances in A*Lightest Derivation (Parsing) [17] provide us with the tools to formalize this scheme.

After briefly introducing A*in Sec. 4.2.1 and its adaptation to parsing in Sec. 4.2.2, we describe how we apply A*to object parsing in Sec.s 4.2.3,4.2.4. Our detection method was initially based on a priority queue implementation of A*parsing that we describe in Sec. 4.2.2, but we have switched to the simpler coarse-to-fine scheme described in Sec. 4.2.4.

4.2.1 Search: Dijkstra's Algorithm vs. A*

Consider an agent who wants to move from the start of the maze of Fig. 4 to the exit using the path of shortest length, say L. We can use Dijkstra's algorithm until the distance from the start to the exit is found and then get the optimal path by backtracking. This algorithm explores all states ν with cost $C(\nu) \leq L$.

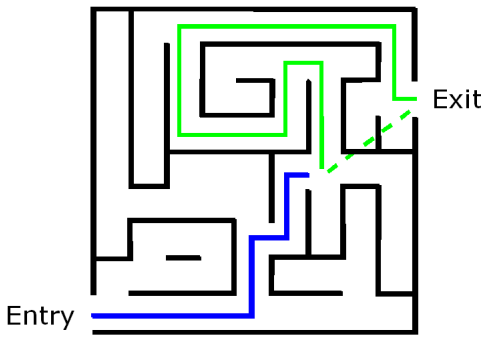


Fig. 4: A*combines the cost-so-far (dark line) with a heuristic estimate (dashed line) of the cost-to-go (green line).

The priority by which states are explored in Dijkstra's algorihm is equal to their distance from the start, or the 'cost-so-far' traveled by the agent. A*[71, 72] is a search algorithm that instead combines the 'cost so far' with an easily computable *estimate* of the 'cost to go', called a *heuristic*. The priority by which state ν is explored is now equal to $C(\nu) + h(\nu)$, where C is the distance from the start and h is the heuristic. When this estimate is a lower bound of the cost to go, it is called an *admissible heuristic*. A lower bound can be obtained by relaxing some of the problem constraints, which is called *problem abstraction*. This could be for example the Manhattan distance, as it ignores the walls of the maze, or the Euclidean distance which also allows the agent to move diagonally.

A*is guaranteed to lead to the optimal solution if it uses admissible heuristics; further it finds it by exploring only those states ν for which $C(\nu) + h(\nu) \leq L$. If $h(\nu)$ is a tight lower bound these can be substantially fewer than those explored by Dijkstra's method. This reduction in the number of explored nodes comes at the cost of computing h.

As the lower bounds computed by problem abstractions can be loose, another option is to use instead *expected costs*. This can speed up A*but results in non-admissible heuristics, which can lead to suboptimal solutions [71].

In summary A*keeps search focused towards the goal, by favoring partial solutions that seem to be getting closer to the goal. This is intuitively similar to the saying 'keep your feet on the ground and your eyes on the stars'. We do not only want to have a good partial solution; we also want it to lead us to the full solution with low cost.

4.2.2 Parsing: KLD vs. Generalized A*

To describe A*parsing we use an analogy between search and parsing. In parsing our 'exit state' is a 'goal' structure, i.e. a structure at the highest level of the hierarchy; and the

'path length' we are minimizing is the cost of the structure's instantiation. To get to our goal structure we need to first instantiate structures lying in intermediate levels of the hierarchy; similar to passing through intermediate states in search. The 'cost-so-far' for an instantiation is its cost, computed recursively as described in Sec. 4.1. And the 'cost-to-go' is the additional cost that will be paid until we form an instantiation of a whole object, while starting from this structure.

In the same way that Dijkstra's algorithm prioritizes intermediate states based on their distance from the start ('cost-so-far'), Knuth's Lightest Derivation (KLD) [17] prioritizes intermediate structures based on their instantiation cost. KLD maintains a list of minimal-cost structures (similar to visited nodes in Dijkstra) and a priority queue of structures that can be composed from structures in this list (similar to the paths in Dijkstra's queue). At each step KLD removes the cheapest structure from the queue, and if it is not already in the list it forms compositions that use this structure. KLD stops when it generates the goal statement, and is guaranteed to find the one of minimal cost, L. However, it first needs to consider all structures with cost less than L.

As in A*search, it is therefore beneficial to have an estimate of the 'cost-to-go' and use it prioritize the compositions. To articulate this, Felzenszwalb and McAllester propose in [17] the concept of 'context'. Loosely stated, the context $Con(S)$ of structure S is what S needs to get to the goal; for a example, for a car the context of a wheel structure would be the engine and trunk structures.

Formally, $Con(S)$ is an instantiation of other structures which, combined with S, lead to a goal statement. So if structure S can be combined with structure P and lead to a goal structure S', S is P's context and vice versa. Further, the cost of these contexts will be $C(Con(P)) = C(S') - C(P)$ and $C(Con(S)) = C(S') - C(S)$, so that $C(S) + C(Con(S)) = C(S')$. Contexts can be defined recursively; if S and P lead to a non-goal structure S', the context of S is what it needs to get to S' (P) plus what S' needs to get to the goal. The context of S will thus have cost $C(Con(S)) = C(Con(S')) + C(S') - C(P)$. Obviously, we are only interested in the context with minimal cost.

Contexts are however hard to compute; implicit in their recursive definition is that we knew how to compute the goal statement, which means that we have already solved the parsing problem. However, A*requires only a lower bound of the cost-to-go. So to prioritize a structure S we only need to lower bound the cost of $Con(S)$. This can be performed quickly by computing $Con(S)$ in a simplified setting (problem abstraction), and then using its cost as an estimate of the 'cost-to-go'. Formally, if $abs(S)$ is the mapping of S to the abstracted problem domain, the cost of $Con(abs(S))$ is used as a heuristic to determine the priority of S at the fine level. This is in brief the algorithm proposed in [17].

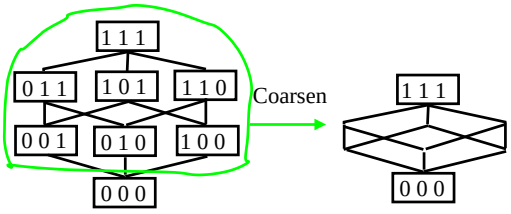


Fig. 5: Structure coarsening: a structure is considered completed if it contains a single part.

4.2.3 Heuristics for Objects

Having shown how A*can use heuristics to speed up computation without sacrificing optimality, we describe the heuristics for our problem, namely object parsing. The Generalized A*algorithm was applied by [17] to finding the most salient curve in an image and was demonstrated to deal with the large number of groupings formed by aggregating local image features into longer structures. Here we consider detecting objects by composing multiple parts; we demonstrate that A*applies to our problem based on the composition mechanism described in Sec. 4.1.

We note that in our implementation we use a Coarse-to-Fine scheme which we found to be better suited to the problem of detection, as described in detail in Sec. 4.2.4. This uses the ideas described in this subsection, which are applicable to both the A*parsing and coarse-to-fine methods.

To compose a structure using for example three parts we need to climb to the top of a Hasse diagram, as shown earlier in Fig. 3. For each move we pay the cost of the new part, plus the change in the composition cost. Adding up these costs along the path to the top gives the cost of synthesizing the whole structure. The main computational burden comes from the large number of possible compositions: in the diagram we show only one path from one substructure to the one above it; in practice there are as many different paths as structure instantiations, while the number of possible top nodes equals the product of the candidate part instantiations.

However, if we can construct a lower bound for all parts lying above the first level, we can find only the first part (the one at the lowest level), and then 'fill in' the other parts with dummy structures having costs equal to the lower bounds of the part costs. This gives us a lower bound for the cost of composing the whole structure that is rapidly computable.

This amounts to what we call a *structure coarsening*. It consists in collapsing several of the nodes of a Hasse diagram into a single one as shown e.g. in Fig. 5, and considering as identical structures that have one, two or three parts instantiated, as long as they have one part in common.

The cost of acquiring part P^i is $\phi_i(\mathbf{s}^i, \mathbf{s}^{pa(i)}) + C(P^i)$, i.e. the location cost plus the part cost; a readily computable

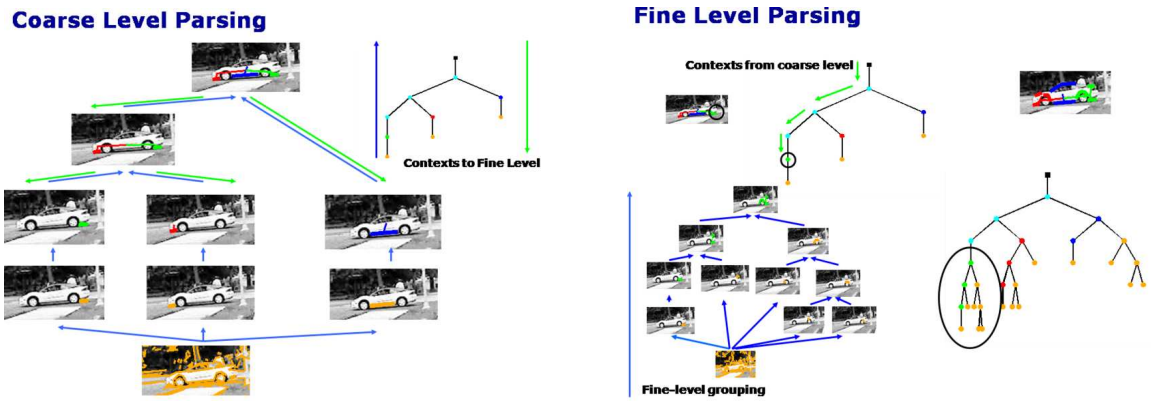


Fig. 6: A*Parsing of a car: Initially (left) the car is coarsely parsed using Knuth's Lightest Derivation. The coarsening consists in considering that each car part needs only a single contour. Once a car is thus parsed its parsing is 'rolled back', and generates contexts for parsing at the finer level. This blows-up a single coarse-level node into a full-blown subtree at the fine level. The parse trees indicate the number of low-, mid-, and high- level structures (orange,blue/green/red,black) that are involved in the coarse- and fine-level composition procedure.

lower bound is thus $\min_{\mathbf{s}^i} \phi_i(\mathbf{s}^i, \mathbf{s}^{pa(i)}) + 0 = \lambda_i \frac{1}{2} \log |\Sigma_i|$, as $\phi_i = -\lambda_i \log P_i(\mathbf{s}^i - \mathbf{s}^{pa(i)})$ and $P_i \sim N(\mu_i, \Sigma_i)$.

Our composition algorithm should thus be modified for coarsened parsing as follows:

```
..
nparts      = length(parts);
for added_part =  1:nparts,
    upgraded_part = Upgrade(parts(added_part));
    other_parts   = setdiff([1:nparts],added_part);
    bound_others  = sum(CostBound(other_parts));
    cost_bound    = upgraded_part.cost + bound_others;
    upgraded_part.cost = cost_bound;
    cmp           = [cmp,upgraded_part];
...
end
```

So we no longer form compositions containing multiple parts; we only upgrade parts from the previous level and add to their cost the lower bound of the other part costs. This drastically reduces the number of explored compositions.

Apart from structure coarsening we can also perform location coarsening, by considering as identical structures whose parts are in a common image neighborhood. This also reduces the number of explored parts, while we are able to go back and mend the inaccuracies of the first coarse-level parsing by the simple book-keeping described in Sec. 4.2.4.

4.2.4 Coarse-to-Fine parsing

The original description of A* implies that we form a composition, compute its heuristic cost, and then insert it in the priority queue. However, computing the heuristic cost 'on demand' is computationally impractical. Instead, as in [17] we consider the opposite strategy of first computing the heuristic and then using it to trigger the formation of a structures.

In specific, we first solve the parsing problem at the abstract level, by coarsening the object-part layer of the hierarchy. At the end of this stage a set of object instantiations is computed whose cost is a lower bound of the actual object's cost. We reject object structures whose cost is higher than a conservative threshold and use the rest for fine-level search.

For this, we identify the low-level structures that can lead to the retained object instantiations. This requires some book-keeping during the coarse detection, where for each structure we keep track of the structures it suppresses. We can then backtrack from each object to all low-level structures that can be used to build it at the fine level.

We thus use a simplified version of A* that does not require a priority queue, and is substantially easier to implement: the process of 'identifying' the low level structures described above amounts to providing them with a heuristic cost equal to zero. The rest of the structures which were not identified obtain a large heuristic cost which removes them from consideration. This quantized heuristic cost thus acts as a 'top-down' signal, indicating if it is worth trying to compose structures towards building an object. This is exactly what we were asking for initially, namely some indication about where it is useful to search for complex compositions.

This method seems to us more natural for detection, as we generate all instantiations that have a cost below a threshold instead of focusing on the cheapest one; further it is substantially faster in Matlab as it is vectorizable, and we have found it to be significantly easier to implement.

4.2.5 Object Parsing Demonstrations

We demonstrate how the algorithm works for an individual object in Fig. 6: Consider detecting a car structure composed of an engine, cabin and trunk structures, which are in turn composed of multiple contours. Initially we simplify the parsing problem by coarsening the object part level of

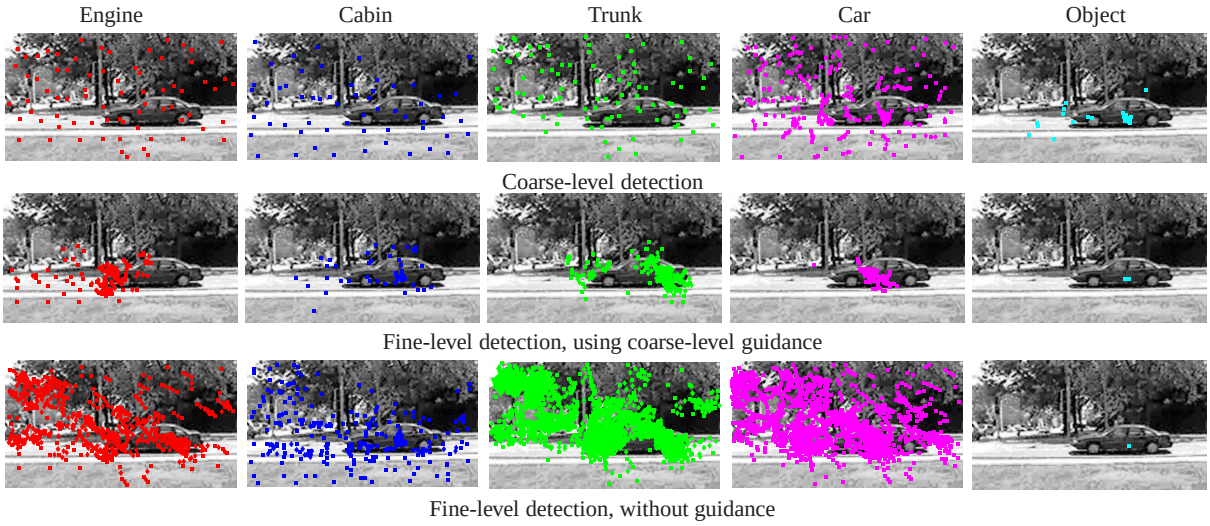


Fig. 7: The top-two rows demonstrate the Coarse-to-Fine detection scheme, which is contrasted to the plain, Fine-Level detection scheme of the bottom row. At the coarse level a small set of candidate object locations is quickly identified; these locations guide search at the fine level, acting like top-down guidance. Instead, when doing Fine-level Parsing without guidance (bottom row) a detailed parse of the object's parts is performed in several background locations, e.g. around trees. This wastes computation resources on locations which end up being useless.

the hierarchy. For example for the engine structure, we consider it is complete when we have found one of its contours, e.g. the hood; the same applies to the cabin and trunk parts. In this way we first compute a coarse parse of the object, where each part is composed from one contour.

At that point, we backtrack and find the mid- and low-level structure instantiations that were suppressed during the coarse-level detection. These structures now have a 'context' at the coarse level, namely a lower bound of their cost-to-go. These are then used for a more detailed parse at the fine level, where all part contours are required to be present, or else penalized. For example, as shown on the right of Fig. 6, the context for the 'back' structure initiates a detailed composition of the structure at the fine level, blowing up a single node of the coarse parse tree to a whole subtree. We can thus build complete parse trees while avoiding elaborate search in background clutter.

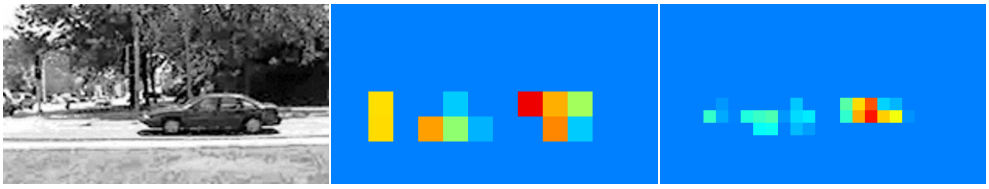
In Fig. 7 we demonstrate the computational gain due to the Coarse-to-Fine scheme for an entire image. At the coarse level we perform structure coarsening and spatial suppression, so we find a few structures for each part - we show their centers as dots. Contrary to the object-part level, we do not coarsen the object level; namely we require all object parts to be present. As all three object parts can be accidentally detected only in few locations, this reduces the candidate objects as shown in the top left corner of Fig. 7.

The object structures whose cost is above threshold are shown on the right column. Starting from these, we backtrack to the lower levels and focus on the image areas likely to contain an object. We now penalize missing contours and build full-blown object parts with multiple contours, which leads to a single object instantiation being above threshold.

By contrast, as shown in the bottom row of Fig. 7 purely bottom-up fine-level detection is 'short sighted'. By trying to form all object parts at full detail from the beginning, it wastes computational resources. This is evident from the large number of individual object parts formed on the background, avoided by the Coarse-to-Fine approach.

In Fig. 8 we show as a heat map the cost function at the coarse and fine levels. A single 'goal' structure resides within each box-shaped neighborhood after location suppression, and neighborhoods where the cost is lower are more red. At the coarse level instead of admissible heuristics we use expected costs, estimated during training. As we use both structure and location coarsening, the domain of the cost function is very coarsely quantized initially; this allows us to rapidly focus on the interesting locations, and then refine our detection.

For example, the car model, learned as described in Sec. 5, contains 65 contours. Plain fine-level detection with this model takes more than 50 seconds for an image with approximately 300 image tokens. Instead, coarse-level detection can be done in less than 10 seconds; following this, fine-level detection can take place in less than 15 seconds, so the sum is typically less than half the original computation time. These measurements do not include the average cost of boundary detection ($\simeq$10 seconds), curve linking ($\simeq$3 seconds) and matching of contours to object boundaries ($\simeq$3 seconds) which are common for both methods. The gain in efficiency varies per image, and is most prominent for images with heavy clutter.



(a) Input (b) Cost, coarsened problem (c) Cost, fine-level problem

Fig. 8: Cost function for the coarsened problem (left) and the original problem (right); low costs are red and high costs are blue. We efficiently compute the cost on the left, which provides us with a lower bound for the cost on the right. We then refine the computation at those locations where the coarse cost falls below a conservative threshold. *Please see in color.*

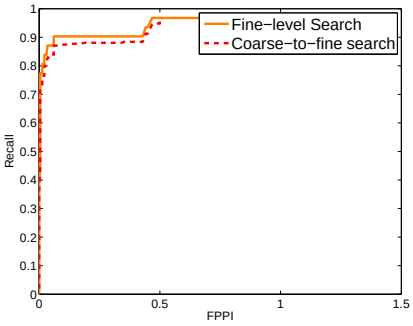


Fig. 9: Performance curves for a detector using fine-level parsing (orange-solid line) and a detector using our coarse-to-fine algorithm.

We note that we have experimentally found expected costs to be more effective at reducing the parsing time than admissible heuristics. As mentioned in Sec. 4.2.1 on the one hand this may result in suboptimal solutions, but the heuristic is substantially closer to the actual cost and results in substantial reductions of the computation cost. Details about the computation of the expected costs are provided in App. B. A quantitative demonstration of the gain in performance as well as the associated loss in accuracy due to the non-admissible heuristics is demonstrated in Fig. 9. There we plot the recall as a function of the false-positive-per-image rate for the bottle category of the ETHZ dataset. The orange-solid curve corresponds to fine-level search, while the red-dashed curve corresponds to the performance of the detector when optimization takes place in a coarse-to-fine manner. The loss in performance observed is due to false negatives (lower recall), which demonstrates that our non-admissible heuristics can reject some useful compositions prematurely. On the other hand, parsing 231 images took 270 minutes using fine-level search and 160 minutes using coarse-to-fine, almost cutting by half the computation time.

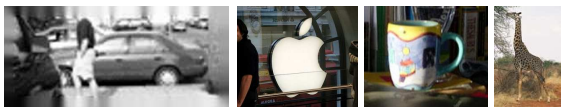


Fig. 10: Typical Images from our training set .

5 Model Learning

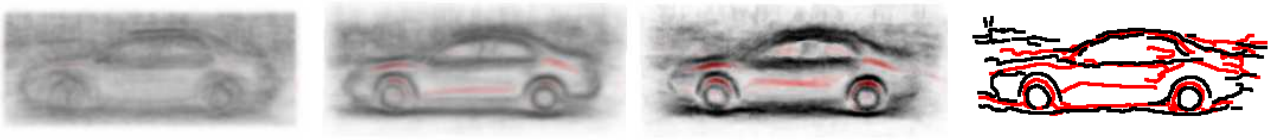
We now turn to the problem of learning our object model. We assume that we are provided with a small set of images (in the range of 20 to 50) that contain only the bounding box of the object. We do not require any manually-segmented features, while the images we work with have heavy occlusions, noise, illumination variations etc; examples of images in our training sets are shown in Fig. 10.

We decompose the learning task into three stages: First we recover the *object contours* by registering the training set using unsupervised Active Appearance Model training (Sec. 5.1). We then find the *object parts* by estimating the geometric affinity of contours and clustering them with Affinity Propagation (Sec. 5.2). Finally we learn the *parsing cost* discriminatively via Multiple Instance Learning so as to optimize the detection performance (Sec. 5.3).

We demonstrate learning results for cars from the UIUC dataset and the ETHZ shape classes (apples, bottles, giraffes, mugs and swans). The testing evaluation of the last 5 categories uses a protocol where the training images are chosen at random from a larger set of images 5 times; we therefore demonstrate 5 different learning results per category. We use common parameters throughout.

5.1 From Images to Contours

The contours used at the lowest level of the object representation are obtained from the edge and ridge contours of the images in our training set. On individual images these contours are noisy but when aggregated over the training set they become substantially more robust.



(a) No registration (b) Affine registration (c) AAM registration (d) Contours

Fig. 11: We obtain the contours at the lowest level of our object representation via deformable model learning: The car template in (a) is computed by averaging the edges (black) and ridges (red) of the training set before registration; it is therefore has few clear structures. The template in (b) is computed using only affine registration and is still missing several structures. The template in (c), computed using AAMs, is sharper as a category-specific deformation model is learned and used. The object contours, shown in (d) are extracted from (c) by non-maximum suppression followed by hysteresis thresholding.

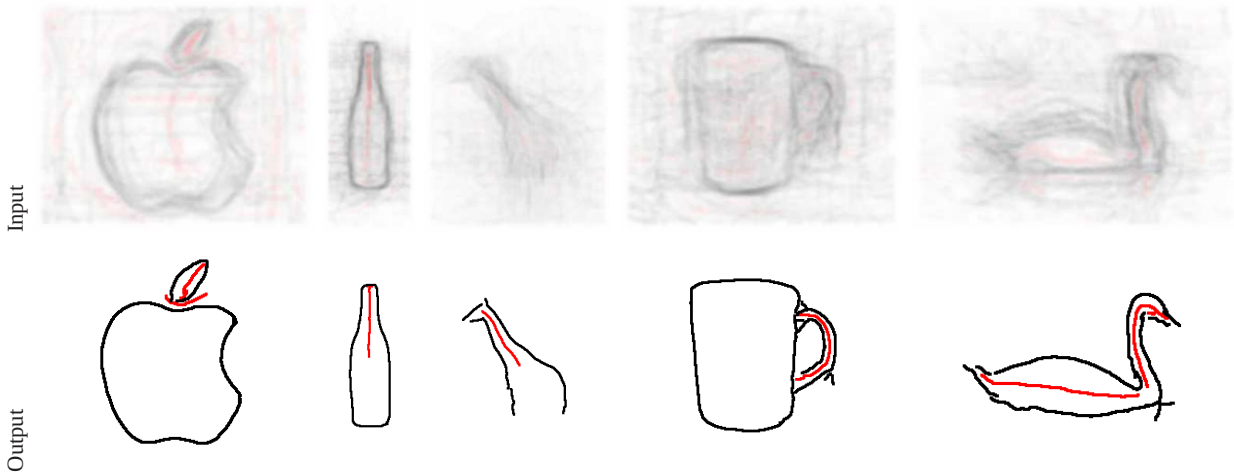


Fig. 12: Initial (top) and final (bottom) templates for the five categories of the ETHZ dataset, learned as described in Fig. 11.

This idea underlies several other recent works that either explicitly [65, 73] or implicitly [23] register a training set. We rely on our work on unsupervised Active Appearance Model (AAM) learning in [74], which we only briefly describe below for completeness, as we intend to present it in a different publication in the future.

In [74] we learn a deformable template model for an object category from images that do not contain landmarks. The template is a dense model of edge and ridge maps, which are largely invariant to changes in appearance. We therefore use the mean of these features as the model prediction, and do not account for appearance variation. The deformation model accounts for shape variation using a linear basis to synthesize the deformation field applied to the template. This basis is category-specific, and is combined with global affine transformations to register the training images.

We learn the mean template and the deformation basis with EM. In the E-step we register the training set to the template and in the M-step we update the deformation model and the template's appearance so as to better register the images in the next step. The output of this process is (a) an Active Appearance Model (template and deformation basis) and (b) a registration of the training images.

The improvement in registration can be seen by comparing the images in Fig. 11(b) with Fig. 11(c). Each image shows the average ridge and edge maps of the whole training set, obtained by taking the mean of the training images. The learned model aligns the training images and gives clean contours after averaging. The object contours shown in Fig. 11(d) row are obtained using nonmaximum suppression suppression followed by hysteresis thresholding. In Fig. 12 we show indicative results for one trial of each category, to indicate that starting from a set of unregistered images we can accurately extract boundary and symmetry information for the whole category.

5.2 From Contours to Object Parts

Having obtained a set of long contours that capture the boundaries and symmetry axes of the object, we want to find the intermediate level structures that will connect these contours into coherent object parts. This is similar to learning the

(a) (b)

Fig. 13: Pairwise clustering is used to discover object parts: first the object contours are broken into straight segments, which are seen as nodes on a graph. Ridges are shown as ellipses whose width is proportional to the scale of the ridge. Nodes are connected with edges based on continuity and parallelism. The affinity among nodes is estimated using statistical and geometric information. The object parts shown on the left are obtained using Affinity Propagation.

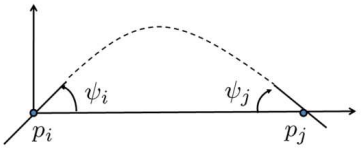


Fig. 14: The geometrical affinity among pairs of nodes is expressed using the Elastica completion cost.

structure of a graphical model [75, 76]; but our problem is harder as we want to 'invent' hidden nodes with continuous variables for the object parts.

We therefore use a data-driven approach that exploits the geometric nature of our problem. We turn the contours extracted in the previous step into a set of straight edge and ridge segments as shown in Fig. 13. Each of these segments is treated as a node in a weighted undirected graph, whose weights quantify the affinity among nodes. We then use pairwise clustering, specifically Affinity Propagation, [77] to partition this graph into coherent components.

Affinity Propagation is a non-parametric clustering algorithm, that allows clusters to be formed based on the pairwise affinities of observations. Central to this algorithm is the identification of 'examplars', i.e. observations that have high affinity to a subset of the data; these are identified in a distributed manner, similar to inference on a graphical model. Apart from the affinities themselves, which will be discussed below, the only crucial parameter of this algorithm is the 'preference', p for each point, which determines how likely it is going to be picked as an exemplar. This quantity determines the number of clusters and is set to $p = .1$ in all of our experiments. If less than three object parts emerge, we perform bisection search over p until we get at least three.

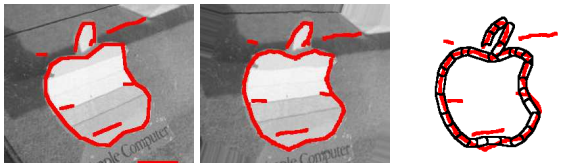


Fig. 15: As pre-processing to estimate the statistical affinity of two nodes, every image in the training set (left) is backward-wrapped to the template coordinate system (middle) using the deformation estimated during AAM training. There, we measure the extent to which each node (black cell) is covered by an image contour (red curve) to determine if the node is present in the training image.

The affinity among two nodes on the graph is computed based on both statistical and geometric cues. First, we compute the geometric affinity among every pair i, j of edges or ridges, based on contour continuity. For this we quantify their grouping cost using the analytic approximation to the Elastica functional [78] provided by [79]. In specific, consider two edge segments i, j, with i ending at p_i with angle ψ_i and j beginning at p_j with angle ψ_j as shown in Fig. 14. It has been demonstrated in [79] that the quantity $E_{i,j} = 4(\psi_i^2\psi_j^2 - \psi_i\psi_j)$ provides an accurate approximate to the scale-normalized integral of the Elastica energy [78] of the illusory contour grouping these edge segments. We set $w_{i,j}^{shape} = e^{-E_{i,j}}$. For adjacent contours we set $w_{i,j}^{shape} = \max\left(1/2, e^{-E_{i,j}}\right)$ to allow for corners, which are otherwise severely penalized by $E_{i,j}$.

Second, to exploit symmetry during grouping we also connect edges to ridges. Based on the scale of each ridge we estimate the set of locations in its periphery where it would expect to find edges; we then set its connection weight to the edges nodes to be equal to the proportion of the edge that is contained in the bounding box.

Third, we incorporate statistical information by measuring how often two tokens appear together in the training images. This strengthens the bonds among parts that have common appearance patterns and isolates parts that appear rarely in the dataset. We compute the function $I_{i,k}$ that indicates whether line i was (partially) observed in image k and set the affinity among lines i and j equal to $w_{i,j}^{occlusion} = \frac{1}{K}\sum_{k=1}^{K} I_{i,k}I_{j,k}$.

To compute $I_{i,k}$ we use the deformations estimated during AAM training, as illustrated in Fig. 15. For this, we first backward warp the edges found on training image i (left) to the template coordinate system (middle), using the deformation estimated during AAM learning. On the template coordinate system we measure the length of the edge contour passing through the 'cell' associated to node k (right). These cells are obtained by dilating the support of each template edge, in order to tolerate small misalignments by the

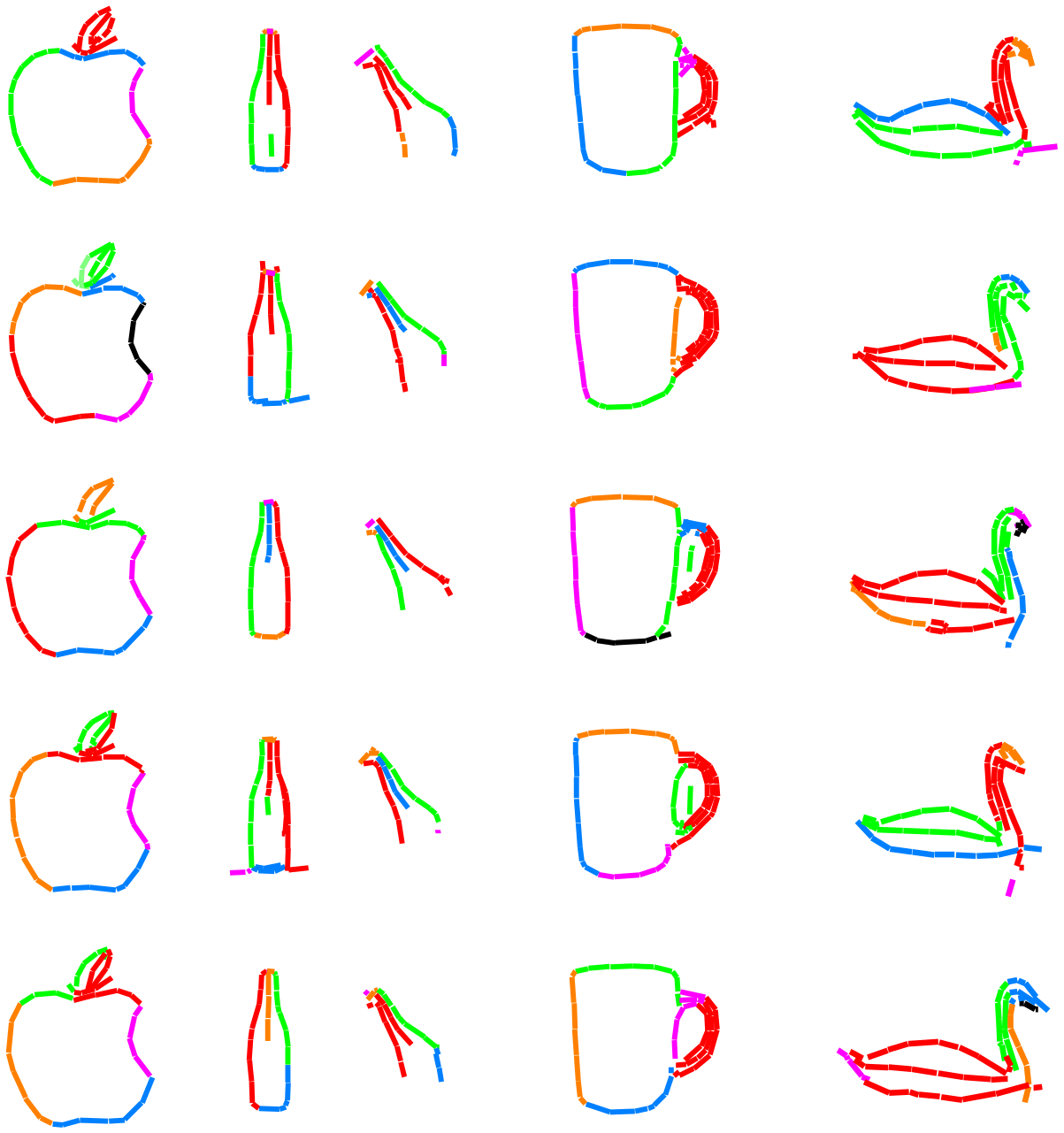


Fig. 16: Object parts delivered by our method for five different trials for the categories of apples, bottles, giraffes, mugs, and swans: boundaries and symmetry axes belonging to the same object part are shown with the same color. *Please see in color*.

AAM. If the length of the intersection of the image curves with the cell is above half the length of the template edge we consider that the corresponding cell is present, i.e. we set $I_{i,k} = 1$.

Finally, the affinity is expressed as $w_{i,j} = w_{i,j}^{shape} w_{i,j}^{occlusion}$. This affinity only partially captures the coherence of two nodes. Two edges lying on opposite sides of a ridge may both be strongly connected to the ridge, but weakly connected with each other according to the continuity cue. Intuitively we would like these two edges to be easy to group together. We therefore consider the product of the pairwise affinities among all possible paths connecting two nodes and use as affinity the maximum of this product among all paths; the product can easily be computed using matrix multiplications.

In order to obtain a clustering from these affinities we have found Affinity Propagation to be particularly effective. We had initially experimented with several alternatives of

Normalized Cuts, but finding automatically the number of object parts turned out to be problematic. Affinity Propagation instead gave visually appealing results after little experimentation. Most of the object parts are visually plausible and correspond to clear structures, e.g. handles, heads, wheels, etc.

Once groups of tokens are identified by the process above we form all possible groupings of these tokens into larger convex arcs. For instance the green contour corresponding to the apple on the top-left figure can be decomposed in several different ways into smaller contours. These are included in what is shown as the green part, but during learning (detailed below) most of them are eventually ignored automatically.

5.3 Multiple Instance Learning for Discriminative Parsing

In Sec. 3 we have expressed our cost function as the likelihood of the image under a probabilistic model. To derive it we have assumed that the child nodes are independent conditioned on their parent, that the reconstruction errors of the tangent function are independent in Eq. 5, while other terms were derived from some prior, e.g. on the probability of having missing contour parts or non-overlapping image and object contours. In practice the independence assumptions do not hold, and we do not have annotated data to construct the priors. Moreover, using generative models is not necessarily optimal for object detection, since our task is to discriminate among objects and background images.

We therefore turn to a discriminative method to estimate the parameters in our cost function. We view our cost function as the output of a linear classifier that uses the potentials computed from an instantiation as features. The decision is formed by taking a weighted sum of these features and rejecting instantiations whose cost is above a threshold.

In specific, the features F used by our classifier are formed by concatenating: (i) The location features, equal to the minus-log probability of the relative poses involved in the part-child relationships,

$$F_1 = \{-\log P(\mathbf{s}_\nu|\mathbf{s}_{pa(\nu)}), \nu \in V \setminus V_r\}. \quad (13)$$

The λ_i parameters in Eq. 1 are the weights of these features. (ii) The contour match features F_2, namely the mismatch cost $\int_{s\in S_c}\left[\theta_O(\frac{s}{\alpha}+\tau)-\theta_M(s)+c\right]^2 ds$, the turning penalty $c^2|S_c|$, and the missing length $|S_d|$ of each object contour $\nu \in V_c$. The $\gamma_1, \gamma_2, \gamma_3$ parameters in Eq. 5 are the weights for these features; each contour has its own weights. (iii) The missing node features F_3, computed from the indicator function h as:

$$F_3 = \{h'(\nu) = 1 - h(\nu), \nu \in V \setminus V_r\}. \quad (14)$$

The missing part potentials ϕ^0_ν in Eq. 3 are the weights of these features. For a missing part we set to zero the location and contour match features of both it and its descendants, while the corresponding missing-node features, y are set to one.

Each instantiation thus gives us $2(|V|-1)+3|V_c|$ features. Our goal is to learn how to weigh these features for the purpose of classification. Our training problem is non-standard, as we do not know the correct object configurations; so we do not know the features that our classifier should be using. We only know that for a positive image at least one configuration will be positive; and all configurations composed from a negative image should be negative. Essentially we want to train a classifier with a high-dimensional hidden variable, namely the correct object configuration.

This is a problem that can be addressed using Multiple Instance Learning (MIL) [80]. Before describing in detail MIL and how we apply it to our case, we note that MIL has been used in computer vision in [81–83], but not for hierarchical models as in our case. In [28] a latent SVM was proposed to train a deformable part model discriminatively, with the poses of parts treated as missing variables. Their algorithm alternates between extracting features by minimizing the cost function w.r.t to the hidden variables, and estimating the feature weighting that optimally separates positive from negative images. At each round this algorithm uses a single hidden variable per positive image, which can cause instabilities, as the authors mention. In more recent work [84] that was developed independently from ours [85], the authors acknowledge that their algorithm is an instance of the MIL framework, but still use a single instance for each positive image, which requires a careful initialization of their algorithm. Instead, we entertain multiple instantiations for each positive image and let the algorithm decide which is most appropriate.

MIL is a general framework to deal with missing data in classification, accommodating for instance Large-Margin training [86] and Boosting [81]. Typical learning assumes training samples come in feature-label pairs. Instead, MIL takes as a training sample a 'bag' of features and its label. Each bag is a set of features, and the classifier is applied to each feature separately. A bag should be labeled positive if at least one of its features is classified as positive, and negative otherwise. The hidden information is the identity of the positive feature(s) in the positive bags. This is exactly our problem, too: from each image we can form a bag of features, corresponding to all possible object instantiations. We want to train a classifier that will label at least one as positive for a positive image, and all as negative for a negative image.

We rely on the Deterministic Annealing approach of [87] to MIL, which resolves some problems in the original work of [86] on Large-Margin MIL. In specific for the i-th training image we form a bag-label pair B_i, Y_i, where B_i is a set of features computed from J_i candidate instantiations,

Fig. 17: Positive and negative bags for the car class: for each training image we compute a set of instantiations, visualized as a part-level labelling of the image tokens. The goal of training is to learn a cost function such that for each positive image at least one instantiation has low cost, while all instantiations from a negative training image have high cost. This is solved using Multiple-Instance Learning.

i.e. $B_i = \{F_{i,1}, \dots F_{i,J_i}\}$. $Y_i = 1$ if i is a positive bag and $Y_i = -1$ for a negative bag. Consider now a hidden variable vector h_i for each bag i, such that $h_{i,j} = 1$ if $F_{i,j}$ corresponds to the correct instantiation for a positive image, and the most object-like instantiation of a negative image.

If we know the hidden variable vector, training the large margin classifier could be accomplished by minimizing the following cost:

$$C(W,b) = c\, W \cdot W + \sum_{i=1}^{N} \sum_{j=1}^{J_i} h_{i,j} l[-Y_i(F_{i,j} \cdot W + b)] \quad (15)$$

$$l[x] = \begin{cases} 1-x, & x < 1 \\ 0, & x \geq 0 \end{cases}, \quad (16)$$

where W is the weight vector, $cW \cdot W$ penalizes the classifier's complexity, b is the classifier's offset and $l[x]$ penalizes examples on the wrong side of the margin. Note that a small value of the cost $F_i \cdot W + b$ implies i is positive, so we need to flip the labels Y_i to turn our problem into the standard SVM training setup.

In practice we do not know the binary hidden variables $h_{i,j}$. However, we could compute them if we knew the classifier parameters (W,b); this suggests an iterative strategy, similar to EM. Following [87] we consider a distribution on the instances of each bag, and replace the binary hidden variables with probabilities $p_{i,j}$. We now consider a new optimization problem that involves both (W,b) and p:

$$C(W,b,p) = cW \cdot W + \sum_{i=1}^{N} \sum_{j=1}^{J_i} p_{i,j} l[-Y_i(F_{i,j} \cdot W + b)] + \nu \sum_{i,j} p_{i,j} \log p_{i,j} \quad (17)$$

$$s.t. \quad \sum_{j=1}^{J_i} p_{i,j} = 1, \quad \forall i \quad (18)$$

$$0 \leq p_{i,j} \leq 1, \quad \forall i,j \quad (19)$$

The last two constraints guarantee that p will be a distribution on the instances of each bag; and $\sum_{i,j} p_{i,j} \log p_{i,j}$ is the sum of the negative entropies of these distributions.

The entropy term is necessary to avoid the local minima of the optimization problem: the optimized function is convex in (W,b) for fixed p and convex in p for fixed (W,b). However it is not convex in W, b, p, due to the misclassification term, which contains negative products of the p and W variables. Deterministic annealing is therefore used to avoid local minima: ν is set initially to a high value, 'convexifying' the optimization problem. For that ν, alternating optimization w.r.t to (W,b) and p is used to optimize the cost function. At the next iteration, ν is decreased and the optimization w.r.t. to (W,b) and p is initialized using the solution computed at the previous step. This process is repeated for decreasing values of ν, which amounts to making the distribution p on each instance increasingly peaked around the instance that has the lowest cost. We have used the code provided by [87] and have found this scheme to provide systematically better solutions that those obtained without annealing.

Adapting this approach to our case requires addressing three technical points. First, the weight vector W should be positive: for a negative weight a structure's contribution to the cost would turn negative, while worse parts would result in lower costs, which is counterintuitive. Moreover it would no longer be possible to lower bound the cost of a part. Therefore, during the optimization w.r.t. (W,b) we need to keep W positive. At timestep t consider that we have found the solution (W^t, b^t); we want to find the increment to W^t, b^t that maximally decreases the cost, while keeping W positive. For this, we first form a second order approximation to the cost around W^t, b^t, i.e.

$$C'(h,f) = C(W^t + h, b^t + f) \simeq \frac{1}{2}[h,f]^T H [h,f] + [h,f]^T J + C(W^t, b^t), \quad (20)$$

where H and J are the Hessian and Jacobian of C respectively at the current estimate of W, p, b. At each t we then recover the optimal update to W^t, b^t by solving the follow-

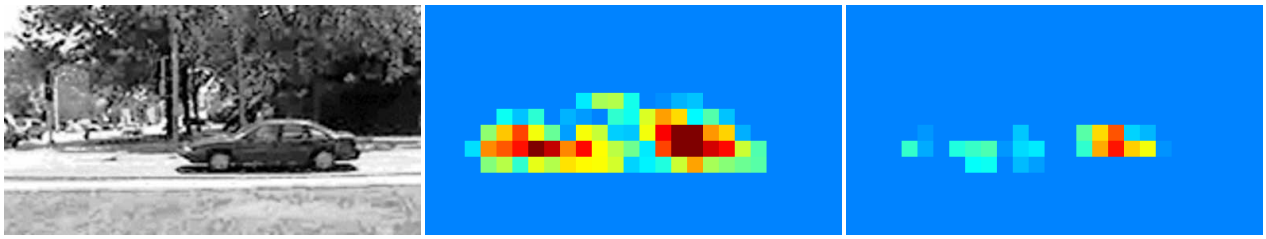


Fig. 18: Improvement of the parsing cost function for cars: initially (middle) our model mistakes parallel structures for cars, giving low cost to the street to the left of the car. After six iterations (right) of MIL training, the cost function indicates more sharply the location of the car.

ing quadratic program with inequality constraints:

$$\min_{h,f} \ \frac{1}{2}[h,f]^T H[h,f] + [h,f]J \tag{21}$$

$$\text{s.t.} \ \ h_i + W_i^t > 0 \quad \forall i \in \{1,\ldots,|h|\} \tag{22}$$

Second, it is impractical to consider all instantiations for each image, as the optimization problem becomes intractable. Instead we alternate between estimating (W, b) using the annealing algorithm, and parsing our training set. For each positive and negative image we keep the five best scoring instantiations at each round, and add them to bag computed for that image from the previous rounds. We have found this to be more robust than keeping only the instantiations from the current round, as it allows the algorithm to 'backtrack' on occasion and give lower cost to instantiations computed earlier. The initial instances are obtained by optimizing the cost function that would correspond to the generative model formulation, i.e. by setting the λ_i variables equal to 1 in Eq. 1 and the missing cost potentials equal to minus the log probability of missing a part.

Third, changing the weight vector potentially has an effect on the features. To see this, consider that the cost of a part's instantiation is higher than the cost of missing that part. As it will be cheaper for a higher- level structure to consider that part as missing, such instantiations are rejected by our composition algorithm. As during training W is changing, some of the instantiations become impossible: if the cost of missing one part decreases, all structures having a more expensive instantiation of that part could no longer be generated. Instead, they should be replaced with structures having that part missing. This means setting to zero their location and contour match features and to one their missing part features.

Therefore, a change (h, f) in (W, b) proposed from the minimization above can potentially result in an increase in the optimized function as the features will change at $(W + h, b + f)$. We deal with this by taking the increment (h, f) computed at each iteration as a direction in which the weight vector should be changed. Then we estimate the length in which this direction should be followed using line search: for each value of the length we modify the feature set as described above, and compute the new cost function. This procedure helps drastically reduce the number of rounds required by the parsing/training algorithm: in practice after 5 iterations the training algorithm converges.

In Fig. 18 we show how the training process gives a cost function that is better suited for detection, by learning to discriminate among car and car-like structures from the background. We note that after several iterations the cost function is more sharply peaked around the actual location of the car; this indicates that we learn a cost function that is better tuned for the detection task.

6 Application to Object Detection

We validate our method using the UIUC car [4] and the ETHZ shape classes [24]. The car dataset is the most challenging in terms of image quality; the images there are of low resolution, while in many cases the image contrast is so low that whole parts of the object are missed by the contour-detection front-end. The ETHZ shape classes have been introduced more recently and are used to benchmark detection algorithms that deal with deformations, occlusions etc., while relying on the shape cue that is dominant for these classes.

Our models have been automatically learned using the procedure described in Sec. 5, while for all classes we use common settings during both training and detection. For cars we use 50 images to learn the contours and object parts, and 300 positive and negative images for discriminative training. For the ETHZ datasets we use the common evaluation protocol: for each category we use half of its images for training, and the remaining images from the category, and all images of other categories, for validation; we present results averaged over 5 different trials. As negatives we use 300 images from the Caltech background images.

In Fig. 19 we demonstrate parsing results on these datasets. We show parsing at the object part-level, where the color encodes the object part to which a token is assigned; our algorithm actually works at a finer level, as each token is assigned to a specific contour of the object. We observe that

Fig. 19: Parsing results: For each image we show object instantiations that are classified as positive by our inference algorithm. We show the parse results at the object part-level, using color to indicate the object part to which a token is assigned; but our algorithm establishes a finer, contour-level relation.

our algorithm can deal with real images containing substantial clutter; for example, in the car images only a small fraction of the image tokens is used to build the object. Fig. 19 demonstrates that our algorithm is able to perform simultaneously the localization and parsing tasks.

In Fig. 20 we report results on these benchmarks. On the top-left plot we compare on the UIUC dataset our results to those of other works using sparse image representations. Our system outperforms these works despite not using appearance information. Our Recall at Equal-Error-Rate (when precision equals recall) is 98% percent, equal to the best reported so far by Fidler et. al. in [40] with a sparse representation.

In the following plots we report results on the ETHZ dataset, and compare to the boundary-based works of Ferrari et. al. [88, 89] and the region-based works Gu et. al. [27].

We plot the recall of our detection algorithm (ratio of detected objects) versus the number of false-positives-per-image (FPPI), averaged over the whole dataset and averaged over the 5 trials. We use the strict PASCAL evaluation criterion that considers as correct a bounding box for an object if its intersection with a ground truth bounding box is larger

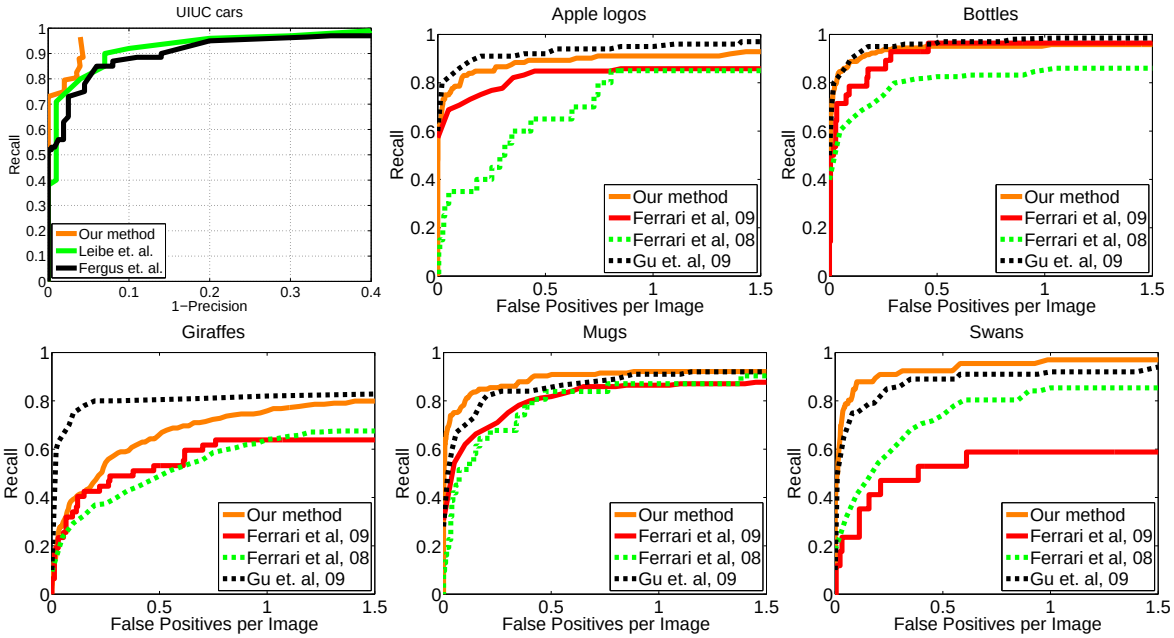


Fig. 20: First plot: benchmark results on the UIUC dataset; we compare to the sparse, part-based approaches of Fergus [7] et. al. and Leibe et. al. [30]. Next five plots: Benchmark results on the ETHZ classes: comparisons with Ferrari et. al. (shape-based), and Gu et. al. (region-based).

than 50% of the union of their areas. All other detections are counted as false-positives.

Our method systematically outperforms both shape-based methods developed by Ferrari et. al.; the first [88] uses a discriminatively trained codebook-based detector, while the second [89] uses voting for bottom-up detection and graph-matching for validation. Apart from delivering better results, our method is unified: we use a single model for bottom-up and top-down detection, and discriminatively learn the cost function that is used for detection.

Comparing to the region-based method of Gu et. al. [27], we observe that our method performs better on mugs and swans, equally well on bottles, slightly worse on apples and systematically worse only on giraffes. The results are not directly comparable, as the authors use an entirely different approach, involving hierarchical image segmentation, boundary descriptors and exemplar-based detection, while a substantially better boundary detection system is used. However, this difference in performance is to some extent intuitive: our method can accurately model the outline of objects, which is distinctive for the categories where it performs well, while the regional cues used in [27] can more naturally capture the texture of giraffes. Further, the edge and ridge maps for these images are particularly noisy, which further challenges our approach for the giraffe category.

This brings us to the limitations of our approach, which can be exemplified by the failure cases in Fig. 21. On the top row we show the tokens computed by our front-end: several object boundaries and symmetry axes are missing, due to the poor image contrast; almost the whole upper part of the car is considered to be missing, while there are very few edge segments aligned with the giraffe's neck. It is therefore hard to form a long contour during grouping. On the bottom row, the coincidental bottle-like configuration of tokens from different objects leads to a false positive. The failure case on the top row can be addressed in at least three different ways (i) learning better boundary and symmetry detectors (ii) incorporating different sparse features, e.g. blobs, corners, junctions that can more easily be detected than contours and (iii) using regional cues. There is substantial work on all of these three research directions that we intend to explore in future work. The failure cases on the bottom can be addressed in three different ways: (i) incorporating appearance information to refine the set of contours that are used to detect an object [64] (ii) using more elaborate grouping to form more elaborate low-level structures than tokens [66] and (iii) using context-sensitive relations [13, 12], potentially in a second, re-ranking stage, to capture 'horizontal' spatial relations among the object contours.

7 Conclusions and Future Work

In this paper we have introduced, first, a hierarchical object representation, second, a principled and efficient inference algorithm, and third, a learning method that only uses the bounding box of the object to learn the model. Our results

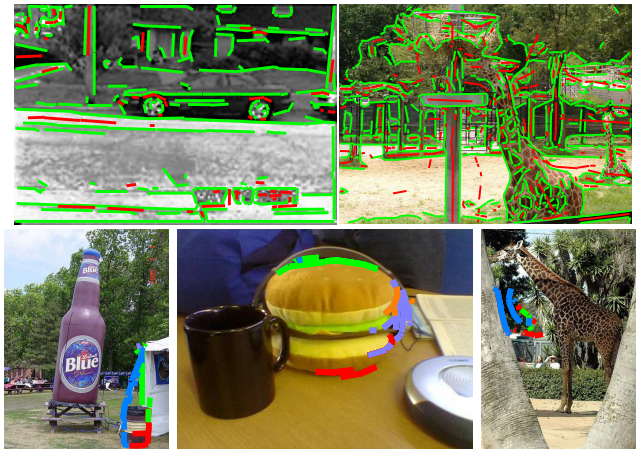
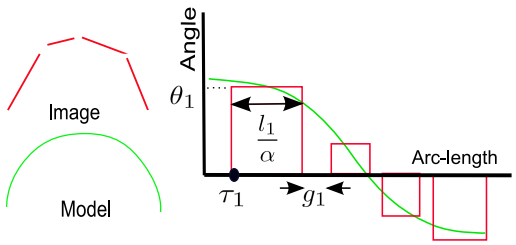


Fig. 21: Modes of failure of our approach: on the top row, the poor image representation delivered by the front-end results in a false negative. On the bottom-row, the object-like configuration of some background edges results in a false positive. Please see text for details.

demonstrate the practical applicability of our approach in challenging images containing substantial background clutter. There a two-fold improvement in detection efficiency is achieved, while the detection performance compares favorably to the current state-of-the-art.

One of the main results of this paper is that with a hierarchical model one can perform efficient detection. This is performed based on a tightly coupled bottom-up/top-down inference scheme, with a small loss in detection accuracy. This makes more elaborate, multi-part models affordable, thereby allowing us to both detect and approximately segment an object.

There are several directions in which we intend to extend this research. The first is to incorporate regional information from segmentation, which has recently been shown to yield excellent results [26,27]. The second is to complement our shape-based features with appearance information, thereby capturing the context around each contour; this is something that we have started working on in [64]. The third is to allow our models to choose among different parts, thereby implementing an OR-ing mechanism as in [12,84,50]. This is straightforward during inference (we already use an OR for missing parts), but the challenge lies in automatically learning the part-mixture components during training. Finally, in the long run, we would like to automatically learn the parts that are shared among multiple categories and are reusable for modeling and detection. This would allow us to simultaneously detect multiple categories in time sublinear in the number of categories, thereby allowing our algorithm to scale up to the detection of tens, or hundreds of objects.



(a)

Fig. 22: Matching image tokens to model structures: Using an angle-based representation of contours, the fragmented lines of the image become piecewise constant signals.

Acknowledgements This work was supported by the W.M. Keck Foundation and NSF Grant 613563. We thank the Reviewers for their constructive feedback, which helped improve the presentation of the paper.

A Integral Angles

In this appendix we describe an efficient method to match image and object contours, generalizing the method of [90] which was used to compare polygonal curves. We use an idea inspired from integral images [52] to compare an arbitrary continuous curve (the model contour) to an image curve, while also dealing with missing parts.

The cost function we use in Sec. 3.2 to quantify the match of two contours is:

$$E_{\theta_O,\theta_M}(\alpha,\tau,c) = \int_{s\in S_c} (\gamma_1 \left[\theta_O(\frac{s}{\alpha}+\tau) - \theta_M(s) + c\right]^2 + \gamma_2 c^2) ds + \gamma_3 |S_d| \quad (23)$$

For fixed α, τ the optimal value of c can be obtained in closed form as:

$$c^* = \frac{\int_{s\in S_c} \theta_M(s) - \theta_O(\frac{s}{\alpha}+\tau) ds}{|S_c|(1+\frac{\gamma_2}{\gamma_1})}. \quad (24)$$

However, E is nonlinear and nonconvex in α and τ. This leaves exhaustive evaluation as the only choice, so we need to speed up the computation of the integral in Eq. 5. For this we exploit working with piecewise straight contours: If a contour is formed by linking N line tokens T_n, each having length l_n and orientation θ_n, the matching criterion writes:

$$E(\alpha,\tau) = \sum_{n=1}^{N} \int_{s=0}^{\frac{l_n}{\alpha}} \left(\gamma_1 \left[\theta_n - \theta_M(s+\tau_n) + c^*\right]^2 + \gamma_2 (c^*)^2\right) ds + \gamma_3 |S_d|,$$
$$\tau_n = \tau_{n-1} + \frac{l_{n-1}+g_{n-1}}{\alpha}, \quad \tau_1 = \tau. \quad (25)$$

Above g_n is the gap between tokens T_{n-1}, T_n, and τ_n is the coordinate where the the n-th line segment begins (see Fig. 22). As θ_n is constant, the integral above writes:

$$(\theta_n + c^*)^2 \frac{l_n}{\alpha} - 2[\theta_n + c^*] \int_{s=0}^{\frac{l_n}{\alpha}} \theta_M(s+\tau_n) + \int_{s=0}^{\frac{l_n}{\alpha}} \theta_M^2(s+\tau_n) ds$$

The only integrals that remain involve the model angle function θ_M and its square. Using the precomputed 'integral angle' functions $\Theta_M(s) = \int_0^s \theta_s$ we can reduce the complexity of computing these from $O(L)$ to $O(1)$ as follows:

$$\int_{s=0}^{\frac{l_n}{\alpha}} \theta_M(s+\tau_n) = \Theta_M(s+\tau_n+\frac{l_n}{\alpha}) - \Theta_M(s+\tau_n).$$

Finally $|S_d|$ can be computed for α, τ as follows: θ_M is defined in the domain $D_m = [0, L]$, while θ_O is defined in $D_c = [\tau_1, \tau_n + \frac{l_n}{\alpha}]$.

Considering the case were the two sets are not disjoint, we have $|D_c \cup D_m| = \max(\tau_n + \frac{l_n}{\alpha}, L) - \min(\tau_1, 0)$, $|D_c \cap D_m| = \min(\tau_n + \frac{l_n}{\alpha}, 0) - \max(0, \tau_1)$. Denoting by G the length of the gaps that fall within $D_c \cap D_m$, we then have $|S_d| = |D_c \cup D_m| - |D_c \cap D_m| + G$.

The 'integral angle' technique described above allows brute force search within a range of values for α, τ with a small computation cost, while using densely sampled contour models - we use 50 points per contour.

B Implementation Details

Below we provide implementation details for several parts of the paper that have been presented at a high level. As AAM learning with be the subject of a future publication, we focus on the rest of the paper.

Section 3.1: Contour grouping: We first extract a set of straight edge/ridge segments, using the contour fragmentation method of [60] and the code of [9]. Edges are obtained from the Berkeley boundary detector [62] and ridges from Lindeberg's primal sketch [67].

We subsequently form all possible combinations of such segments into convex arcs by combining the method of Jacobs [91] with the intervening contour idea of [92]. The technique of Jacobs guarantees that all groupings returned will be convex arcs. On top of that, we do not allow arcs linking two tokens if there is a roughly perpendicular edge intersecting the line connecting their endpoints. To determine whether these are perpendicular we use the condition $|\cos(\theta_1 - \theta_2)| < \cos(2\pi/3)$, where θ_1 is the angle of the line connecting the two tokens and θ_2 is the angle of the intervening contour. The intervening contour cue is used if the response of the boundary detector is above .1.

We form all groupings composed of up to four tokens. We then reject all groupings for which the sum of the gap lengths is more than half the length of the grouped edgels. This discards groupings of faraway edges, which simply happen to lie on a convex arc.

Section 3.2: Contour Matching: We do a preliminary screening before matching a contour group with a model curve to reduce the number of matchings. We denote the length of the contour group by l_g and its orientation by θ_g; orientation is defined as the angle of the line connecting its two endpoints. We denote the corresponding quantities for the model curve as l_m and θ_m; note that as we consider separate ranges of α, l_m is obtained by the ratio of the nominal length of the curve over the median scale considered at a time (so that we can perform a scale-invariant match). Our screening consists in rejecting matches when $|\cos(\theta_m - \theta_g)| < \cos(\pi/4)$ and when $|\log(l_g/l_m)| < .6$. We thereby avoid checking for very large rotations and for contour matches which would imply large missing parts.

Section 4.2.4 a) Inadmissible Heuristics: For structure coarsening instead of composing all parts of a multi-partite structure we compose only one, and replace the costs of the remaining ones with predictions of their costs. Forming an admissible heuristic would require using a lower bound of the cost of each part. Unfortunately, these bounds can often be loose. Instead we use inadmissible heuristics, which may no longer be lower bounds, but are more efficient at ruling out unpromising search directions.

For this, we keep track of each part's cost over the whole positive set during training; in specific, we work with the instances identified as witnesses for the positive bags: for each image i we identify $j^* = \text{argmax}_j p_{i,j}$, where $p_{i,j}$ is the distribution over instances entertained by the MIL-SVM algorithm of [87] in Eq. 18. Denoting by $c_{i,k}$ the cost for part k in instance j^* of image i, our prediction for the cost of part i is the 20-th percentile of the set $\{c_{1,k}, \ldots, c_{N,k}\}$ where N is the number of training images. Even though this is obviously not a lower bound of the cost, it is a relatively conservative estimate of the cost that can guide the heuristic search. Note that if the part is not present in any of the 'witness' instances, the cost estimate will be equal to the cost of missing the part.

Section 4.2.4 b) Location Coarsening: Apart from structure coarsening we can also use location coarsening. This amounts to quantizing the location coordinates with a fixed grid. In specific, we use two levels of coarsening; at the coarse level we quantize the the location coordinates into boxes of 60x60 pixels, and at the fine level we quantize into boxes of 10x10 pixels. For efficiency, we use kd-trees to quickly identify the box into which every candidate composition should contribute to. For each box we keep only the composition falling into it that has minimal cost. If no composition enters a box, its cost is set to infinity. This coarsening is performed separately for each part. We can form compositions of more complex structures by putting together these fewer representatives. This reduces the number of compositions.

Section 4.2.4 c) Structure and Location Coarsening: We solve the parsing problem at three levels of abstraction. At the highest (most abstract) level, we perform structure coarsening at the level of mid-level parts together with location coarsening with a grid of boxes sized 60x60. At the second level we perform only location coarsening with a grid of boxes sized 60x60. At the finest level we use a grid of boxes sized 10x10; this last level provides us with our detector results.

References

1. Felzenszwalb, P., Huttenlocher, D.: Pictorial Structures for Object Recognition. IJCV **61** (2005) 55–79
2. Moosmann, F., Triggs, B., Jurie, F.: Randomized clustering forests for building fast and discriminative visual vocabularies. In: NIPS. (2006)
3. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: ECCV. (2006)
4. Agrawal, S., Roth, D.: Learning a Sparse Representation for Object Detection. In: ECCV. (2002)
5. Fergus, R., Perona, P., Zisserman, A.: Object Class Recognition by Unsupervised Scale-Invariant Learning. In: CVPR. (2003)
6. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV **60** (2004) 91–110
7. Fergus, R., Perona, P., Zisserman, A.: A sparse object category model for efficient learning and exhaustive recognition. In: CVPR. (2005)
8. Shotton, J., Blake, A., Cipolla, R.: Contour-based learning for object recognition. In: ICCV. (2005)
9. Kokkinos, I., Maragos, P., Yuille, A.: Bottom-Up and Top-Down Object Detection Using Primal Sketch Features and Graphical Models. In: CVPR. (2006)
10. Borenstein, E., Ullman, S.: Class-Specific, Top-Down Segmentation. In: ECCV. (2002)
11. Russell, B.C., Efros, A.A., Sivic, J., Freeman, W.T., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: CVPR. (2006)
12. Zhu, S.C., Mumford, D.: Quest for a Stochastic Grammar of Images. Foundations and Trends in Computer Graphics and Vision **2** (2007) 259–362
13. Jin, Y., Geman, S.: Context and Hierarchy in a Probabilistic Image Model. In: CVPR. (2006)
14. Fidler, S., Boben, M., Leonardis, A.: Similarity-based cross-layered hierarchical representation for object categorization. In: CVPR. (2008)
15. Tu, Z., Chen, X., Yuille, A., Zhu, S.: Image Parsing: Unifying Segmentation, Detection, and Recognition. IJCV **63** (2005) 113–140
16. Kokkinos, I., Maragos, P.: Synergy Between Image Segmentation and Object Recognition Using the Expectation Maximization Algorithm. IEEE T. PAMI **31** (2009) 1486–1501
17. P. Felzenszwalb and A. McAllester: The generalized A* Architecture. Journal of Artificial Intelligence Research **29** (2007) 153–190

18. Welling, M., Weber, M., Perona, P.: Unsupervised Learning of Models for Recognition. In: ECCV. (2000)
19. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual Categorization with Bags of Keypoints. In: ECCV. (2004)
20. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering Object Categories in Image Collections. In: ICCV. (2005)
21. Lampert, C., Blaschko, M., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In: CVPR. (2008)
22. Crandall, D., Felzenszwalb, P., Huttenlocher, D.: Spatial Priors for Part-Based Recognition using Statistical Models. In: CVPR. (2005)
23. Wu, Y., Shi, Z., Fleming, C., Zhu, S.C.: Deformable Template As Active Basis. In: ICCV. (2007)
24. Ferrari, V., Tuytelaars, T., Gool, L.V.: Object Detection by Contour Segment Networks. In: ECCV. (2006)
25. Opelt, A., Pinz, A., Zisserman, A.: Incremental learning of object detectors using a visual shape alphabet. In: CVPR. (2006)
26. Todorovic, S., Ahuja, N.: Extracting subimages of an unknown category from a set of images. In: CVPR. (2006)
27. Gu, C., Lim, J.J., Arbelaez, P., Malik, J.: Recognition using Regions. In: CVPR. (2009)
28. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR. (2008)
29. Amit, Y., Kong, A.: Graphical templates for model registration. IEEE T. PAMI **18** (1996) 225–236
30. Leibe, B., Leonardis, A., Schiele, B.: Combined Object Categorization and Segmentation with an Implicit Shape Model. In: ECCV. (2004) SLCV workshop.
31. Sudderth, E., Torralba, A., Freeman, W., Willsky, A.: Learning Hierarchical Models of Scenes, Objects, and Parts. In: ICCV. (2005)
32. Fu, K.S.: Syntactic Pattern Recognition. Prentice-Hall (1974)
33. Marr, D.: Vision. W.H. Freeman (1982)
34. Grimson, E.: Object Recognition by Computer. MIT Press (1991)
35. Siddiqi, K., Kimia, B.: Parts of Visual Form: Computational Aspects. IEEE T. PAMI **17** (1995) 239–251
36. Zhu, S.C., Yuille, A.: Region Competition: Unifying Snakes, Region Growing and Bayes/MDL for Multiband Image Segmentation. IEEE T. PAMI **18** (1996) 884–900
37. Keselman, Y., Dickinson, S.: Generic model abstraction from examples. IEEE T. PAMI **27** (2001) 1141–1156
38. Ioffe, S., Forsyth, D.A.: Probabilistic Methods for Finding People. IJCV **43** (2001) 45–68
39. Ahuja, N., Todorovic, S.: Learning the taxonomy and models of categories present in arbitrary images. In: ICCV. (2007)
40. Fidler, S., Leonardis, A.: Towards Scalable Representations of Object Categories:Learning a Hierarchy of Parts. In: CVPR. (2007)
41. J. Porway and B. Yao and S.C. Zhu: Learning compositional models for object categories from small sample sets. In: Object Categorization: Computer and Human Vision Perspectives. Cambridge Press (2008)
42. Zhu, S.C., Mumford, D.: GRADE- Gibbs Reaction and Diffusion Equation. IEEE T. PAMI (1997)
43. Han, F., Zhu, S.C.: Bottom-Up/Top-Down Image Parsing by Attribute Graph Grammar. In: ICCV. (2005)
44. Todorovic, S., Ahuja, N.: Learning subcategory relevances for category recognition. In: CVPR. (2008)
45. Zhu, L., Lin, C., Huang, H., Chen, Y., Yuille, A.: Unsupervised Structure Learning: Hierarchical Recursive Composition, Suspicious Coincidence and Competitive Exclusion. In: ECCV. (2008)
46. Ramanan, D., Sminchisescu, C.: Training Deformable Models for Localization. In: CVPR. (2006)
47. Quattoni, A., Wang, S., Morency, L.P., Collins, M., Darrell, T.: Hidden conditional random fields. IEEE T. PAMI **29** (2007) 1848–1852
48. Taskar, B., Klein, D., Collins, M., Koller, D., Manning., C.: Max-Margin Parsing. In: EMNLP04. (2004)
49. Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In: EMNLP. (2002)
50. Zhu, L., Chen, Y., Lu, Y., Lin, C., Yuille, A.: Max Margin AND/OR Graph Learning for Parsing the Human Body. In: CVPR. (2008)
51. Zhu, L., Chen, Y., Ye, X., Yuille, A.: Structure-Perceptron Learning of a Hierarchical Log-Linear Model. In: CVPR. (2008)
52. Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. In: CVPR. (2001)
53. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. Volume 2. (2005) 886–893
54. Moreels, P., Maire, M., Perona, P.: Recognition by probabilistic hypothesis construction. In: ECCV. (2004) 55
55. Lempitsky, V., Blake, A., Rother, C.: Image segmentation by branch-and-mincut. In: ECCV. (2008)
56. Felzenszwalb, P., Schwartz, J.: Hierarchical Matching of Deformable Shapes. In: CVPR. (2007)
57. Chen, Y., Zhu, L., Lin, C., Yuille, A.L., Zhang, H.: Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In: NIPS. (2007)
58. Parikh, D., Zitnick, L., Chen, T.: Unsupervised Learning of Hierarchical Spatial Structures in Images. In: CVPR. (2009)
59. Russell, G., Brooks, R., Binford, T.: The ACRONYM model-based vision system. In: IJCAI. (1979)
60. Lowe, D.: Perceptual Organization and Visual Recognition. Kluwer (1984)
61. Schmid, C., Mohr, R.: Local grayvalue invariants for object retrieval. IEEE T. PAMI **19** (1997) 530534
62. Martin, D., Fowlkes, C., Malik, J.: Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. IEEE T. PAMI **26** (2004) 530–549
63. Opelt, A., Pinz, A., Zisserman, A.: Boundary-fragment-model for object detection. In: CVPR. (2006)
64. Kokkinos, I., Yuille, A.: Scale Invariance without Scale Selection. In: CVPR. (2008)
65. Jiang, T., Jurie, F., Schmidt, C.: Learning shape prior models for object matching. In: CVPR. (2009)
66. Zhu, Q., Wang, L., Wu, Y., Shi, J.: Contour Context Selection for Object Detection: A Set-to-Set Contour Matching Approach. In: ECCV. (2008)
67. Lindeberg, T.: Edge Detection and Ridge Detection with Automatic Scale Selection. IJCV **30** (1998)
68. Sudderth, E., Ihler, A., Freeman, W., Willsky, A.: Nonparametric belief propagation. In: CVPR. (2003)
69. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (2006)
70. Birkhoff, G.: Lattice Theory. AMS (1967)
71. Pearl, J.: Heuristics. Addison-Wesley (1984)
72. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (2003)
73. Ferrari, V., Jurie, F., Schmid., C.: Accurate object detection with deformable shape models learnt from images. In: CVPR. (2007)
74. Kokkinos, I., Yuille, A.: Unsupervised Learning of Object Deformation Models. In: ICCV. (2007)
75. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Information Theory **14** (1968) 462–467
76. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine learning **20** (1995) 197243
77. Frey, B., Dueck, D.: Clustering by Passing Messages Between Data Points. Science **315** (2007) 972 976
78. Mumford, D.: Elastica and Computer Vision. In J., B., ed.: Algebraic Geometry and its applications. Springer Verlag (1993) 507–518

79. Sharon, E., Brandt, A., Basri, R.: Completion energies and scale. IEEE T. PAMI **22** (2000) 1117–1131
80. Dietterich, T.G., Lathrop, R.H., Lozano-perez, T.: Solving the multiple-instance problem with axis-parallel rectangles. Artificial Intelligence **89** (1997) 31–71
81. Viola, P., Platt, J.C., Zhang, C.: Multiple Instance Boosting and Object Detection. In: NIPS. (2006)
82. Dollar, P., Babenko, B., Belongie, S., Perona, P., Tu, Z.: Multiple Component Learning for Object Detection. In: ECCV. (2008)
83. Vijayanarasimhan, S., Grauman, K.: Multiple-instance learning for weakly supervised object categorization. In: CVPR. (2008)
84. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object Detection with Discriminatively Trained Part-Based Models. IEEE T. PAMI (2009) to appear.
85. Kokkinos, I., Yuille, A.: Inference and learning with hierarchical compositional models. In: Stochastic Image Grammars Workshop. (2009)
86. Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: NIPS. (2002)
87. Gehler, P., Chapelle, O.: Deterministic Annealing for Multiple Instance Learning. In: AISTATS. (2007)
88. Ferrari, V., Fevrier, L., Jurie, F., Schmid, C.: Groups of adjacent contour segments for object detection. IEEE T. PAMI **30** (2008) 36–51
89. Ferrari, V., Jurie, F., Schmid, C.: From images to shape models for object detection. IJCV (2009)
90. Arkin, M., Chew, L., Huttenlocher, D., Kedem, K., Mitchell, J.: An Efficiently Computable Metric for Comparing Polygonal Shapes. IEEE T. PAMI **13** (1991) 209–217
91. Jacobs, D.W.: Robust and efficient detection of salient convex groups. IEEE T. PAMI **18** (1996) 23–37
92. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. IJCV **43** (2001) 7–27