HOP: Hierarchical Object Parsing

Iasonas Kokkinos
Laboratoire MAS, Ecole Centrale de Paris
Equipe GALEN, INRIA Saclay - Ile-de-France, Orsay

Alan Yuille
University of California at Los Angeles *

Abstract

In this paper we consider the problem of object parsing, namely detecting an object and its components by composing them from image observations. Apart from object localization, this involves the question of combining top-down (model-based) with bottom-up (image-based) information.

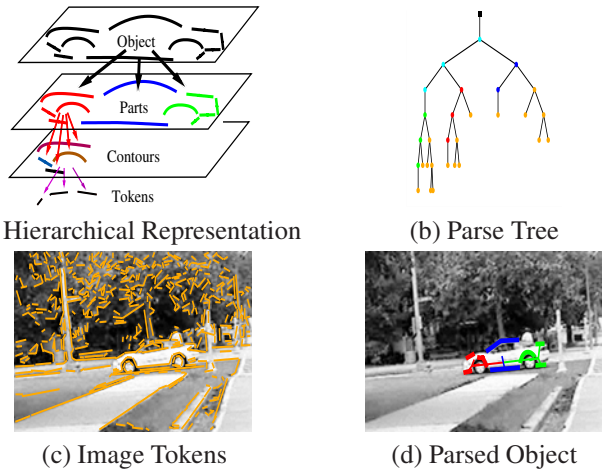
We use an hierarchical object model, that recursively decomposes an object into simple structures. Our first *contribution is the formulation of composition rules to build the object structures, while addressing problems such as contour fragmentation and missing parts. Our* second *contribution is an efficient inference method for object parsing that addresses the combinatorial complexity of the problem. For this we exploit our hierarchical object representation to efficiently compute a coarse solution to the problem, which we then use to guide search at a finer level. This rules out a large portion of futile compositions and allows us to parse complex objects in heavily cluttered scenes.*

1. Introduction

The problem we address in this work is to parse an object in a scene, as shown in Fig. 1. By parsing we mean detecting an object by composing all of its structures using a sparse representation of the image. This is a most challenging task as the object structures can deform, some may be missing, which combined with a cluttered background providing numerous tokens leads to a combinatorial explosion. However, by accurately parsing an object we can not only localize it, but also track it or segment it, without solving each problem from scratch.

For this we use an hierarchical object representation, described in Sec. 3 and shown in Fig. 1(a), which gradually decomposes an elaborate object model into simpler image structures. Specifically, objects are decomposed into parts, which in turn break into contours, which in the end produce straight edge segments ('tokens'). The latter are extracted by using the Pb edge detector [19] and line segmentation.

We phrase detection as finding an optimal sequence of compositions that start from these edge tokens and lead in the end to the whole object. This amounts to building a *parse tree*, as shown in Fig. 1(b). The leaves of this tree ('terminals') are edge tokens, and the color-coded nodes correspond to intermediate object structures; in getting closer to the root more complex structures are formed, involving more parts, while the root of the tree is the whole object structure, shown in Fig. 1(d). In Sec. 4 we explain how we can go from the hierarchical representation in Fig. 1(a) to the parse tree in Fig. 1(b)



(a) Hierarchical Representation (b) Parse Tree

(c) Image Tokens (d) Parsed Object

Figure 1: Object Parsing Task: Our goal is to compose objects using image tokens based on an hierarchical object representation. This amounts to building a *parse tree*, that indicates how image tokens are related to object structures.

There is a huge number of candidate parse trees, most of which will have low likelihood under our probabilistic model. In Sec. 6 we describe how we efficiently find those few parse trees which have high likelihood.

Our first contribution is building efficient and flexible composition rules that deal with problems such as contour fragmentation and missing parts. Our second contribution lies in controlling the combinatorial complexity of the problem, by using a parsing algorithm that exploits our object representation. We propose a 'structure coarsening' operation that simplifies the composition of structures and reduces the problem to a simpler one, allowing us to perform coarse-to-fine detection. This rules out a vast portion of the solution space in a top-down manner and makes feasible the computation of optimal parses for real images with heavy clutter, where plain bottom-up detection can get trapped in

*Work supported by NSF grants 0413214, 0613563, and 0736015.

futile compositions.

2. Previous Work

Our work builds on the the hierarchical compositional approach, e.g. [10, 14, 32, 2], which models complex structures by putting together simpler parts. Even though conceptually appealing, how to best perform inference on hierarchical models is not a solved problem.The authors of [14] perform depth-first search with a restricted representation involving discrete variables, and correct its results using a variant of reranking. Regarding using continuous attributes, as we do in this work, they mention it is 'potentially unmanageable in a Markov system'. In the work of S. C. Zhu and coworkers [12, 28] proposals are generated from discriminative models such as AdaBoost/Ransac in a bottom-up manner and are then validated by object/scene models. Instead, we use a single generative model to both suggest object locations during coarse-level search and to validate the detection results at a finer level, in the framework of an integrated optimization algorithm. In more recent work [3] a pruned version of dynamic programming is used to efficiently detect objects at an initial stage and then refine them in a top-down manner. The behavior of this method is hard to predict, while we rely on A* which has guaranteed optimality properties; further we provide results on datasets that are more challenging for localization, where the object occupies a small portion (5-10%) of the image domain.

Coming to works that are more closely related to our method at the technical level, in [5] inference with an hierarchical model was presented, but limiting the object representation to a single contour. In [23] the detection of geometric structures, such as salient open and convex curves was 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, while applying it in a Coarse-to-Fine manner instead of using Best-First-Search. The combinatorics of matching have been studied for rigid objects [11], and [20] also used A*for detecting object *instances*, while in [16] branch-and-bound is used for efficient detection with a bag-of-words model.

Finally, as in recent works for object detection [26, 22, 8, 31] and earlier ones on grouping [18, 11, 13], we use a contour-based object representation. We argue that as contours cover a larger portion of the object than interest points [1, 6], they can be more easily used in conjunction with other tasks, like tracking or segmentation.

3. Hierarchical Object Representation

We use a three-layered hierarchy for our model, as shown in Fig. 1 for a car; at the highest layer we have the whole 'object' structure that is decomposed into three parts in the second layer. Each of these parts is then decomposed into a set of contours in the first layer, which in turn generate edge tokens, lying at the 'zeroth' level.

The structures at each layer are described by their pose vectors, $\mathbf{p} = (\mathbf{x}, \log\sigma, \theta)^T$ that contain information about location- $\mathbf{x}$, scale- σ and orientation- θ. These are random variables, and the dependencies among structure and part poses are modeled by probability distributions, whose parameters are automatically estimated, as described in 7.1.

For tractability we consider that the parts are independent conditioned on their parent structure, so we have:

$$\log P_r(\mathbf{p}_{S_r}|\mathbf{p}_r) = \sum_{i\in S_r} \log P_{i,r}(\mathbf{p}_{i|r}). \quad (1)$$

P_r relates the pose $\mathbf{p}_r$ of structure r to the poses $\mathbf{p}_{S_r}$ of its constituents S_r. Each $P_{i,r}$ models the relative pose of part i to that of structure r, given by:

$$\mathbf{p}_{i|r} = ([\mathbf{x}_i - \mathbf{x}_r]\, R(\theta_r), \log(\sigma_i) - \log(\sigma_r), \theta_i - \theta_r)^T, \quad (2)$$

where $R(\theta_r)$ is a rotation matrix. The location and scale components of $\mathbf{p}_{i|r}$ are modeled with Gaussians and orientation with a von Mises distribution. At the lowest level we compare the r-th model contour, having pose $\mathbf{p}_r$, with contours formed by grouping subsets of tokens, $\mathcal{T}_{S_r}$. We use a discrepancy measure among contours $D(\mathcal{T}_{S_r}|\mathbf{p}_r)$ described in Sec. 4, and consider $P(\mathcal{T}_{S_r}|\mathbf{p}_r) \propto \exp(D(\mathcal{T}_{S_r}|\mathbf{p}_r))$.

If we know the poses of the object structures at the three levels of the hierarchy $\mathcal{P}_{1:3}$, and the token-to-contour associations $\mathcal{S}_1$ at the first level, we can express the likelihood of an image represented as a set of tokens $\mathcal{T}_{1:C}$ as:

$$\log P(\mathcal{T}_{1:C}|\mathcal{S}_1, \mathcal{P}_{1:3}) = \log P_O(\mathbf{p}_O) + \sum_{p\in S_O} \underbrace{\log P_{p,O}(\mathbf{p}_{p|O})}_{\text{object}\to\text{part}}$$
$$+ \sum_{p\in P}\sum_{c\in S_p} \underbrace{\log P_{c,p}(\mathbf{p}_{c|p})}_{\text{part}\to\text{contour}} \sum_{c\in C} \underbrace{\log P(\mathcal{T}_{S_c}|\mathbf{p}_c)}_{\text{contour}\to\text{tokens}} + c(\mathcal{T}_{\mathcal{B}}) \quad (3)$$

where S_O is the object's support, i.e. its parts and S_p is the support of part p, i.e. its contours. C is the set of model contours and $c(\mathcal{T}_{\mathcal{B}}) = \sum_{i\in\mathcal{B}} \log P_B(\mathcal{T}_i)$ is the log likelihood of the unexplained tokens under a background model.

This representation is essentially an hierarchical, tree-structured graphical model: the relations among structures and parts define a tree-structured graph with pairwise cliques among parents and children, and the observation potentials at the lowest nodes are defined by the cost of matching the model contours to image tokens. However, there are two essential differences in our inference algorithm: *First*, we use sparse information extracted from the bottom-up to indicate node states with nontrivial likelihood. Inference in graphical networks using Belief Propagation, e.g. [4] considers all node states, most of which have almost zero likelihood. *Second*, the graph has an hierarchy. We exploit this to quickly rule out unpromising solutions instead of propagating information around all nodes in a flat graph. In the

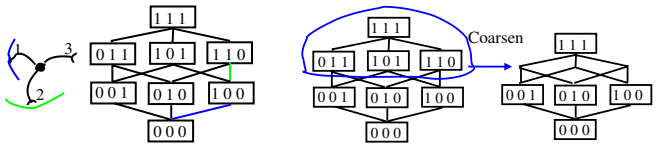


Figure 2: Left: Hasse diagram showing for a 3-part structure: as a structure acquires its constituents, it gradually climbs to the top of the diagram. The binary vector inside the box indicates the parts that have been found. Intuitively, we view each structure as having bonds that gradually pick up parts. Right: Structure coarsening: the problem of assembling a structure is simplified by considering as completed any structure the contains a single part.

following two sections we describe how these tie in with our compositional inference approach.

4. Bottom-Up Composition Mechanisms

Having formulated our model in a top-down fashion, where we start from the object structure and decompose it until generating image contours, we now invert this formulation, and describe how we can compose the object in a bottom-up manner.

4.1. Composition Rule Formulation

In order to detect objects in a compositional framework on needs certain composition rules that are applied iteratively; for example, to assemble a 'box' structure one could , as proposed in [32], use custom rules, e.g. grouping the two pairs of parallel strokes and then joining them based on perpendicularity, or combining corners. However this requires new hand-crafted rules for new structures, e.g. an hexagon, or learned structures, while one needs to devise special rules for the case of missing parts.

Our approach addresses these problems by allowing each structure to acquire its constituent parts, one at a time. Consider building in this way a structure S having N constituents $S = (S_1, \ldots, S_N)$. Initially we set $S_i = *, \forall i$, where $*$ is a special 'dummy', non-instantiated part. Gradually building up the structures amounts to recursively applying binary compositions rules of the form:

$$(S, C) \rightarrow S', \quad S'_i = C \tag{4}$$

where C is the new constituent, attached to part i of S; the rule is applicable only if $S_i = *$, e.g. for $i = 2$ we would have $S = (S_1, *, S_3)$ and $S' = (S_1, C, S_3)$.

This way of composing structures introduces a *partial ordering* $\preceq$ among them, with $S^i \preceq S^j$ if S^j has all the parts of S^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, as shown in Fig. 2 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 in a Hasse diagram that starts from the minimum element and gradually goes to its maximum.

Further, we introduce the idea of *structure coarsening*, i.e. collapsing several of the nodes of the Hasse diagram into a single one as shown e.g. in Fig. 2. This means that we consider identical structures that have one, two or three parts instantiated, as long as they have a part in common. This reduces the number of distinct structures and helps us simplify the parsing problem, as described later in Sec. 6.3.

4.2. Weighted Composition Rules

As our model is probabilistic, we can quantify how good a structure is in terms of its *composition cost*, which is equal to minus the log-likelihood of the observations generated from it. Further, as our model has a hierarchical tree structure, we recursively estimate the composition cost of a structure using only the costs of its constituents.

This is done with weighted composition rules, that relate the costs of the precedents with that of the antecedent; each binary composition rule is written as:

$$(S = w_1), (C = w_2) \rightarrow (S' = w_3), \tag{5}$$
$$w_3 = w_1 + w_2 + w_r, \quad w_r = \log P(\mathbf{p}_C | \mathbf{p}_S) + C_{seal}(C)$$

where S and C have composition costs, w_1, w_2 and S' pays an extra w_r for binding the two structures. The first term of w_r, $\log P(\mathbf{p}_C | \mathbf{p}_S)$ is the log-likelihood of part C conditioned on structure S, and comes naturally from Eq. 3. $C_{seal}(C)$ is a 'sealing cost' that implements our mechanism for dealing with missing parts. Specifically, each structure C does not pay for non-instantiated parts until its gets merged with a larger structure. At that point, all such parts are declared missing, so C has to pay a penalty for these missing constituents. For contours, as described in the next section, we use the missing length penalty in Eq. 7, γL_m, and for complex structures we estimate the cost recursively. In this way we can generate structures with *missing parts*, which however come at a cost when the structures move higher up in the hierarchy.

5. Integral Angles for Contour Matching

The interface between our model and the image is at its first level, where model contours are associated with edge tokens. A problem we need to address there is contour fragmentation: as shown in Fig. 3 an object contour like the arc of the wheel can give rise to different image tokens. As we do not want our model to account for all possible fragmentations, we need an efficient way to reconcile tokens and contours. For this we use a post-processing step that first groups tokens into larger contours and then matches them to the model. Once they are matched, they are used to generate more complex structures, and eventually the whole object,

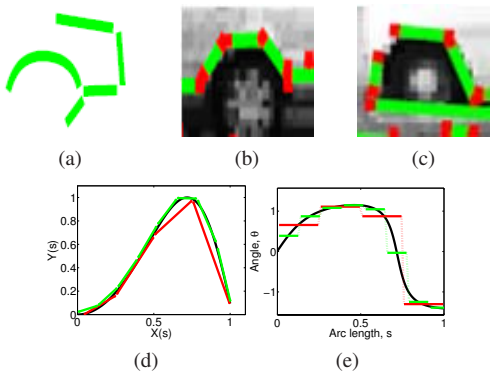


Figure 3: Matching image tokens to model structures: the model arc in (a) has to be matched to fragmented observations as in (b) and (c). Using the angle-based representation of the contours in (d), the fragmented lines (red/green) become piecewise constant signals (e). This allows using 'integral angles' for efficiency.

as we describe in the next subsection. In this subsection we address the technical problem of efficiently matching token groupings to object contours. We do this using a variation of the integral image technique.

As shown in Fig. 3 (e) we use an angle-based representation of contours [11], mapping the arclength s to the tangent angle θ. The registration between a model contour θ_M and an observed contour θ_O can be described in terms of three parameters: rotation amounts to an additive constant c, scaling is a dilation by a factor α and constant differences in the arc-length argument are accounted for by shifting one of the functions by τ. The matching quality is then expressed as:

$$E_{\theta_O,\theta_M}(\alpha,\tau,c) = \int_{s=0}^{L} \left[\theta_O(\frac{s}{\alpha}+\tau) - \theta_M(s) + c\right]^2 ds. \quad (6)$$

A square norm is used to make the the distance between the two functions efficiently computable; the optimal value of c is obtained as the mean difference of the registered functions, but α and τ are nonlinearly related to E.

This leaves exhaustive evaluation as the only choice, so we need to speed up the computation of the integral in Eq. 6. To do this we exploit working with piecewise constant contours: If a contour is formed by linking N line tokens T_n, each having length l_n and orientation θ_n, the criterion used for matching writes:

$$E(\alpha,\tau,c) = \sum_{n=1}^{N} \int_{s=0}^{\frac{l_n}{\alpha}} \left[\theta_n - \theta_M(s+\tau_n) + c\right]^2 + \gamma L_m, \quad (7)$$

where $\tau_n = \tau_{n-1} + \frac{l_{n-1}+g_{n-1}}{\alpha}$. Above g_n is the gap between tokens T_{n-1}, T_n, and τ_n is the model coordinate system where the match with the n-th line segment begins, and $\tau_0 = \tau$. $L_m = L - \sum_n \frac{l_n}{\alpha}$ is the length of model contour that is not observed, due to gaps or because the image contour is smaller than the model contour. We symmetrize this cost to equally penalize larger contours. As θ_n is constant, the integrals in Eq. 7 write:

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

The only integral that remains involves the model angle function θ_M and its square. We reduce the complexity of computing these from $O(L)$ to $O(1)$ using the precomputed 'integral angle' functions $\Theta_M(s) = \int_0^s \theta_s$ as follows:

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

This 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 100 points per contour. We then use $D(\theta_{\mathcal{T}_{S_r}}, \theta_{M_r}) = \min_{\alpha,\tau,c} E_{\theta_{\mathcal{T}_{S_r}},\theta_{M_r}}(\alpha,\tau,c)$ as our data-fidelity term that links the model prediction to the image observations, as mentioned earlier in Sec. 3.

We can thus have an object model with a few long contours, while dealing with all possible contour fragmentation scenarios. This combines the tractability of a continuous contour model with the efficiency of working with a sparse image representation.

6. Efficient Object Parsing

Having described our hierarchical object representation and the composition rules used to assemble it, we now address the *efficient composition* of the most likely object, given a set of edge tokens extracted from the image. As composing an object is a sequential task, we can prioritize the order in which we explore compositions. One way to do this is using a purely bottom-up method that starts from the tokens and builds increasingly complex structures, until in the end forming the whole object. A priority rule that can be applied in this setting is to consider first structures that have small composition cost, or high likelihood; this strategy, amounts to Knuth's Lightest Derivation (KLD) in parsing [23]. A problem with this approach is that large groupings are harder to form than small ones, since their cost is the sum of more positive terms. Therefore, in cluttered scenes many intermediate structures are formed on the background, perpetually postponing the construction of more complex ones.

Intuitively, we would like to use our model to guide us in a top-down manner to areas where there is complementary evidence for an object structure. For example if we know that we cannot find the mid- and back- parts of a car in an image region, we should quit searching there for bottom-up compositions of the front part.

This can be accomplished in a principled way using the A*algorithm [25]. As shown in Fig. 4, A*prioritizes search

not only based on the cost of a path traveled so far, but also on an estimate of the cost to get to the goal, called 'heuristic'. This keeps search focused towards the goal, by favoring solutions that seem to be getting closer to it. In [23] the A*algorithm was generalized to the problem of hierarchically computing lightest derivations (parsings). This addresses the problem of composing a set of observations to build a goal structure by using a set of weighted rules that have minimal net cost. As in A*search, the formed structures are prioritized based on both the composition cost and the remaining cost required to get to the goal statement. This was demonstrated to deal with the enormous number of possible curve and convex groupings formed by aggregating local image features into longer structures.

In our case we use the 'syntax' of our object model and consider building up objects having multiple parts, without making special assumptions about the object structures. As we show, applying A*to our problem is feasible using the composition mechanisms of Sec. 4 and the structure coarsening described in Sec. 6.3. We then extend this idea to performing Coarse-to-Fine detection, thereby avoiding the Best-First approach of A*which may not be ideal for object detection. In the next two subsections we describe how [23] apply A*to the general parsing problem and how it relates to our problem, while in the last two subsections we further elaborate on how we apply it to our problem.

6.1. A*for parsing

During parsing we have a priority queue Q, where all composed structures are stored and a set $\mathcal{S}$ of structures which have been produced with the lowest possible cost. Initially $\mathcal{S}$ is empty and Q contains the edge tokens extracted from the image.

At each step a structure is popped from Q, it is examined if it is are already in $\mathcal{S}$ and discarded if so. Otherwise it is put in $\mathcal{S}$ and all its potential combinations with elements already in $\mathcal{S}$ are formed. These are inserted in Q, and wait to be popped in turn, according to their priority. If we only want the single optimal parse we stop when popping the first 'goal' statement from Q, otherwise we stop when we exceed a cost threshold.

Depending on the rule used to compute the priorities, we have different search strategies. As mentioned, Knuth's Lightest Derivation (KLD) assigns priorities equal to the production costs. Instead, A*obtains the priority $\pi(S)$ of a structure S by adding to its production cost $g(S)$ a heuristic term $h(S)$ that assesses the cost to get to the goal starting from S, i.e. $\pi(S) = g(S) + h(S)$. If $h(S)$ is a lower bound of the actual cost it is an *admissible heuristic* function, allowing us to get to an optimal solution.

By using $h(S)$ we rule out structures that have a small production cost g but are far away from a goal. This focuses search in directions that can prove promising in the long run. To apply A*to the parsing problem we need to construct and estimate the heuristic $h(S)$; this brings us to contexts and abstractions.

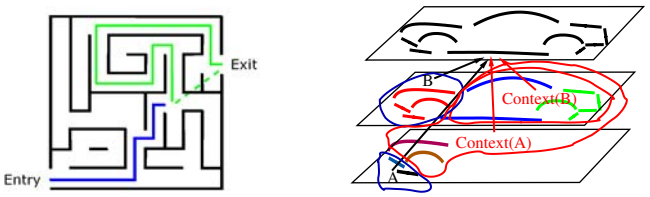


Figure 4: Heuristics and parsing. Left: In search, A*combines the cost-so-far (dark line) with a heuristic (dashed line) that approximates the cost-to-go (green line). This helps direct search towards the goal. Right: In parsing, heuristics can be computed using contexts; the context of each structure consists of the set of structures that are required to lead to the 'goal', object structure.

6.2. Contexts and Abstractions

The context of a structure S, $Con(S)$ provides us with the means to construct a heuristic function. Loosely stated, $Con(S)$ is the complement of S, and tells S how much more it has to pay to get the goal (Fig. 4). Formally, it is an instantiation of other structures which, combined with S, lead to a goal statement. So for a composition

$$(A_1 = w_1), (A_2 = w_2) \rightarrow (goal = w_3)$$

we can write $(Con(A_1) = w_3 - w_2)$ and $(Con(A_2) = w_3 - w_1)$. Contexts are defined recursively for the intermediate structures, starting from the highest-level contexts and going down to contexts for terminals. Intuitively, with reference to Fig. 4, this means that the context of A is what A needs to get to B, plus what B needs to get to the goal.

Contexts are difficult to compute, too: if for a structure we know what else we need to get to the goal, this means we have solved the parsing problem already. However, A*requires only *a lower bound of the cost* of getting to the goal. We can thus use *problem abstractions*: the abstraction of a problem is obtained by eliminating some of its aspects that make it hard to solve, so that the computed solution's cost underestimates that of the actual solution. As this can be done by relaxing problem constraints, it requires no 'hand crafted' lower bounds.

So for parsing, if $abs(S)$ is the mapping of a structure to the abstracted problem domain, the cost of $Con(abs(S))$ is used as a heuristic to determine the priority of S at the concrete level. That is, we solve the rest of the parsing problem in a simplified setting, and use the estimated cost as a lower bound of the actual 'cost-to-go'. Composition rules of the form of Eq. 5 are thus rewritten as:

$$(A_1 = w_1), (A_2 = w_2), (Con(abs(S)) = w_c) \overset{\pi(S)}{\rightarrow} (S = w_3),$$

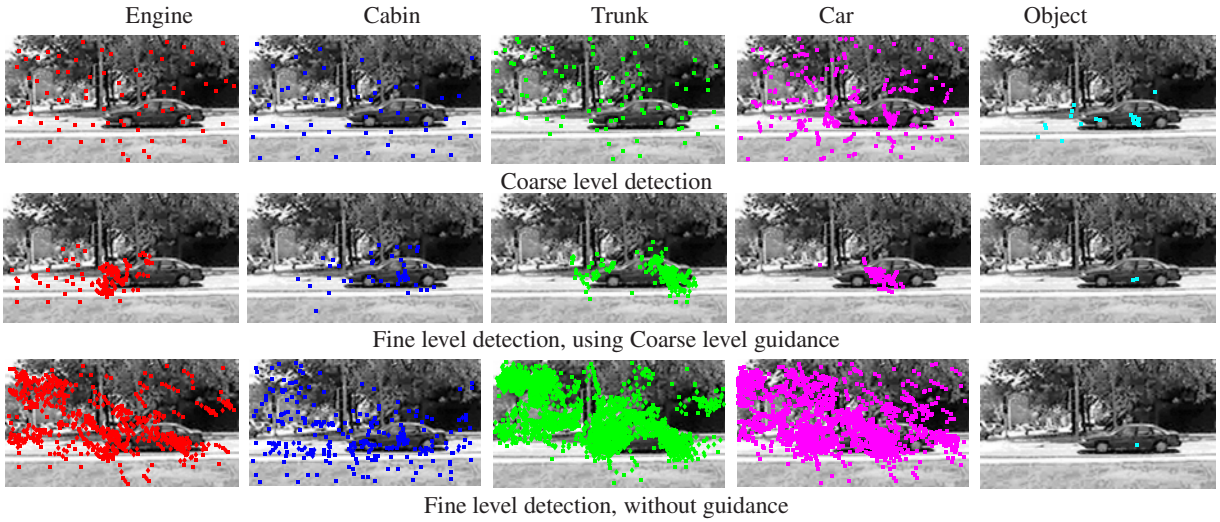


Figure 5: Coarse-to-Fine vs. Fine-level Parsing: At the coarse level a small set of candidate object locations is quickly identified; these are then used to guide search at the fine level, acting like top-down guidance. Instead, when doing Fine-level Parsing without guidance a detailed parse of the object is attempted in futile directions (see text for details). *Please see in color on screen.*

where $\pi(S) = w_3 + w_c$, the priority of S, is determined by its composition cost w_3 and the heuristic cost $w_c = Con(abs(S))$.

6.3. Heuristics via Structure Coarsenings

Having laid out the abstract setting for A*parsing, we now describe how we compute heuristics for our problem, namely object parsing. For this, we use the structure coarsening described in Sec. 4.1 to compute heuristics.

As shown in Fig. 2, composing a structure amounts to climbing to the top of a Hasse diagram, where each upward move comes with a cost for the newly acquired part. Adding up these costs along the path to the top gives the cost of synthesizing the whole structure. We can thus lower bound this cost by bounding from below the costs of certain arcs, e.g. those lying above the first level. This is the idea behind structure coarsening, which we discussed in 4.1. We consider for example the composition of the object structure 'O'; the cost of acquiring its part p, if that has zero cost on its own, will equal:

$$-\log P(\mathbf{p}_{p|O}) = \frac{1}{2}\left[\log((2\pi)^n|\Sigma|) + [\mathbf{p}_{p|O}]^T\Sigma^{-1}[\mathbf{p}_{p|O}]\right]$$

where $P(\mathbf{p}_{p|O})$ is the conditional probability of the pose of part p given the pose of the structure. This quantity is the cost of an arc in the Hasse diagram for that structure, and can be lower bounded by $\frac{1}{2}\log((2\pi)^n|\Sigma|)$. We thereby assume that we will find a part p whose pose is identical to that predicted by the object structure, i.e. $\mathbf{p}_{p|O} = \mathbf{0}$ and has zero cost on its own. The cost of composing the object structure in this simplified setting is thus a lower bound of its actual cost; we can therefore use this cost as an admissible heuristic for A*.

As this bound is too optimistic, we build a tighter one by observing that since we have a prioritized search strategy, we always pick first the part that has lowest cost, say C_1. So all other parts of the structure will have cost at least equal to that of the first part. We can thus lower bound the cost of the other constituents with $\max(\frac{1}{2}\log((2\pi)^n|\Sigma|), C_1)$.

Another option would be to replace minimum costs with *expected costs*; This is known to significantly speed up A*but results in non-admissible heuristics, which can lead fine-level search to suboptimal solutions [24]. As we do not want to introduce uncontrolled problems in our evaluation we leave this for future work.

6.4. Coarse-to-Fine Parsing

In order to apply the ideas described above to object detection, we do not follow the Best-First approach described so far. As we do not want the single best object, but may want to have multiple detections, it is more natural to use Coarse-to-Fine detection in a 'parallel' manner, instead of the 'serial' Best-First approach. In specific, we use the simplification of the parsing problem described above as a coarse-level detection; for this we coarsen the part-contour level of the hierarchy, and identify parts composed from multiple contours. For an 'engine' structure this means it can be composed from one contour, e.g. the wheel, and is then considered to be complete.

This results in parts found in many image location, but can be performed efficiently, as we do not form parts with more than one contour, as seen by comparing the top and bottom rows of Fig. 5. We use these coarse parts to build object structures and finally generate 'goal' structures from these objects, i.e. parse tree roots Fig. 1(b). When turning objects into goals, we penalize missing parts as described in 4.2. Even though at this stage any object part needs to have only one contour, this still results in many poor object structures falling below threshold.

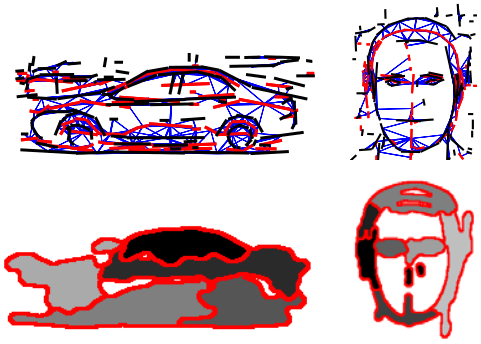


Figure 6: Top Row: Adjacency graphs among edge (black) and ridge (red) lines for cars and faces. The arcs of the graph are shown in blue and their width indicates the affinity between the lines. Bottom Row: Object parts found using spectral clustering of the adjacency graphs.(Gray-level encodes part label)

The 'goal' structures that are above threshold then propagate contexts to the fine level along the lines of the previous section, thereby focusing the detection to a few locations in the image where an object is likely to reside. At the fine level a more detailed parsing is performed, leading to a highly accurate detection of the object. Coarse-level detection thereby focuses on the few image locations where all object parts can be simultaneously present, and brings 'top-down' guidance into detection at the fine level.

Instead, as shown in the bottom row of Fig. 5 *plain* fine-level detection is 'short sighted', and tries to form all parts at full detail from the beginning, thereby wasting computational resources. This is evident from the large number of individual parts formed on the background, which the Coarse-to-Fine approach manages to avoid.

7. Application to Object Detection

We validate our method using the UIUC car [1], Caltech face [6] and ETHZ apple and bottle [8] datasets.

7.1. Model Construction

For apples and bottles we use the provided object boundaries and manually determine their hierarchical structure. For all of the parsed car images shown we use a hand-crafted model, for ease of communication; for the benchmark results reported on cars and faces we use however automatically learned models. These are constructed using the method briefly described below; we emphasize however that the topic of this paper is efficient inference, while learning will be detailed in a longer version of this paper, and is reported for completeness.

Initially, we extract a set of contours that frequently occur in the object category using the method of [15] to automatically establish correspondences from 50 unregistered training images. Edge and ridge line segments are computed from the registered feature maps using the grouping method of [15], and as shown in Fig. 6 an adjacency graph is formed based on the geometric configuration of these lines: edge-to-edge and ridge-to-ridge connections are established based on continuity, while an edge is connected to a ridge if it is almost parallel and its distance is approximately equal to the width of the ridge. Further, to favor making a single part out of lines that move together we introduce weights between each connected pair of lines based on their joint deformation statistics.

We group these line segments using spectral clustering [29], and after pruning small clusters, we obtain the object parts shown in the bottom row of Fig. 6. Each of these parts contains certain contours, whose parametric distributions are estimated afterwards from the established correspondences. Other methods have been recently introduced to combine boundaries, or segments of multiple images into object parts [30, 9, 2], but we argue that our approach is simpler, as parts emerge naturally from the interactions among geometric primitives. The parameters in the cost function are estimated in an EM-manner, by iteratively detecting objects in fore- and back-ground images from training set, and then updating the cost parameters so as to minimize classification errors.

7.2. Detection Results

In the top rows of Fig. 7 we demonstrate parsing results on images from [1]; there we observed that our algorithm can deal with real images containing substantial clutter, typically containing 300-500 edge segments. There, simple fine-level search can take around 7-10 seconds, while with our coarse-to-fine detection we reduce this to less than 1 second; in particular coarse-level detection typically takes around one tenth of a second. These measurements do not include the average cost of boundary detection (10 seconds), curve linking (3 seconds) and matching of contours to object boundaries (3 seconds) which are common for both methods; timings are in Matlab on a 1.8Gh PC.

In Fig. 7(a) we compare detection performance using parses computed at the coarse and fine levels. At the fine level performance improves but at the cost of longer computation. The hierarchical parsing strategy can thus allow us to choose among increasingly elaborate models, depending on the level of accuracy required. Further, we experimented with learning how to combine the different terms in Eq. 3, by learning different weights for each summand type (part/contour/token). For this we train a sigmoidal classifier using parses from fore- and back-ground images, which yields another improvement in performance. In Fig. 7(b) we compare our results to some classical works using sparse image representations. Our system performs comparably despite using no appearance information. Better results have recently appeared in the literature, e.g. [9, 16], but we note that we have not tuned our system for detection, since our main concern has been to develop an efficient inference algorithm.

In the middle three rows of Fig. 7 we show parsing re-

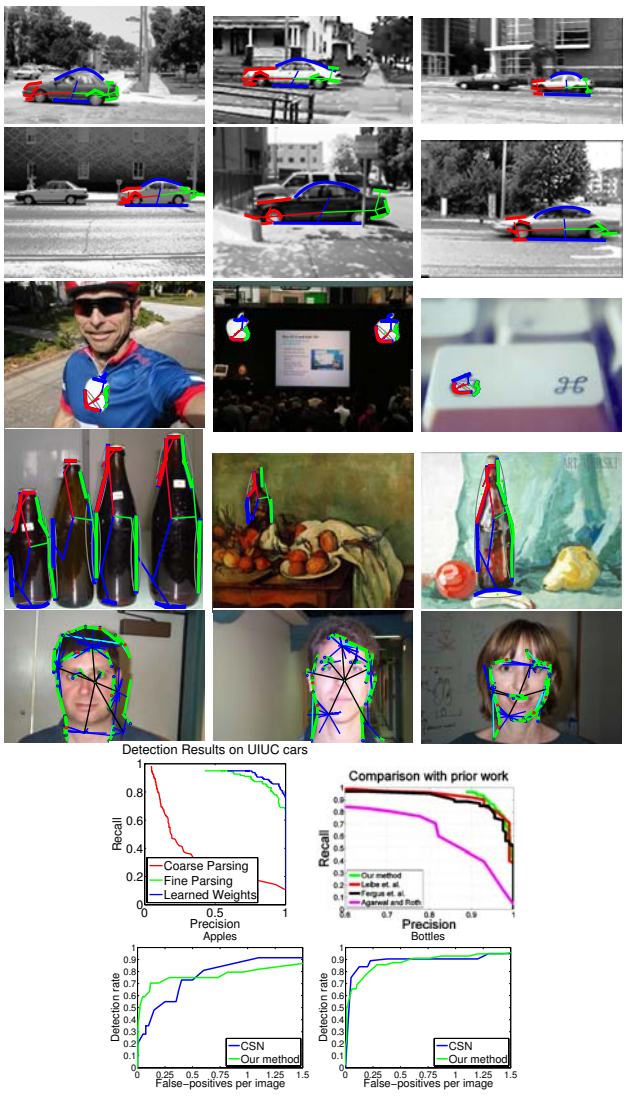


Figure 7: Top two rows: Parsing results on images from [1]. Next three rows: Parsing results on apples and bottles from [8] and faces from [6]. Thick lines are contours, thin lines are part-structure connectors. Next row: Left: Precision-Recall curves on UIUC cars using parsing at the coarser level, parsing at the fine level (all parts considered) and the learned cost parameters. Right: Comparison to the methods of [1, 6, 17], Bottom Row: comparison to [7].

sults on apples and bottles from [8] and faces from [6]. and in Fig.s (c) and (d) we report quantitative results. Using the same performance measures as in the references, for faces we have an EER of 93.4, while for apples and bottles we perform comparably to [8]. Overall, we observe that our system can cope with multiple scales and missing parts, while performing well for several of categories.

8. Conclusions

In this paper we develop a principled and efficient inference method for hierarchical object representations. Our results demonstrate the practical applicability of our approach in real images containing substantial clutter, where a ten-fold improvement in performance is attained.

In future work we intend to further explore how recent works such as [27, 2, 9, 30, 21] for learning hierarchical compositional models can be combined with our inference approach.

References

[1] S. Agrawal and D. Roth. Learning a Sparse Representation for Object Detection. In *ECCV*, 2002.

[2] N. Ahuja and S. Todorovic. Learning the taxonomy and models of categories present in arbitrary images. In *ICCV*, 2007.

[3] Y. Chen, L. Zhu, C. Lin, A. L. Yuille, and H. Zhang. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In *NIPS*, 2007.

[4] P. Felzenszwalb and D. Huttenlocher. Pictorial Structures for Object Recognition. *IJCV*, 2005.

[5] P. Felzenszwalb and J. Schwartz. Hierarchical Matching of Deformable Shapes. In *CVPR*, 2007.

[6] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *CVPR*, 2003.

[7] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. Technical report, INRIA, 2006.

[8] V. Ferrari, T. Tuytelaars, and L. V. Gool. Object Detection by Contour Segment Networks. In *ECCV*, 2006.

[9] S. Fidler and A. Leonardis. Towards Scalable Representations of Object Categories. In *CVPR*, 2006.

[10] K. S. Fu. *Syntactic Pattern Recognition*. Prentice-Hall, 1974.

[11] E. Grimson. *Object Recognition by Computer*. MIT Press, 1991.

[12] F. Han and S. C. Zhu. Bottom-Up/Top-Down Image Parsing by Attribute Graph Grammar. In *ICCV*, 2005.

[13] D. Jacobs and M. Lindenbaum. Special Issue of Perceptual Organization in Computer Vision. *IEEE T. PAMI*, 25(4), 2003.

[14] Y. Jin and S. Geman. Context and Hierarchy in a Probabilistic Image Model. In *CVPR*, 2006.

[15] I. Kokkinos and A. Yuille. Unsupervised Learning of Object Deformation Models. In *ICCV*, 2007.

[16] C. H. Lampert, M. B.Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[17] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *ECCV*, 2004.

[18] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer, 1984.

[19] D. Martin, C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries. *IEEE T. PAMI*, 26(5):530–549, 2004.

[20] P. Moreels, M. Maire, and P. Perona. Recognition by probabilistic hypothesis construction. In *ECCV*, page 55, 2004.

[21] B. Ommer and J. M. Buhmann. Learning the Compositional Nature of Visual Objects. In *CVPR*, 2007.

[22] A. Opelt, A. Pinz, and A. Zisserman. Boundary-fragment-model for object detection. In *CVPR*, 2006.

[23] P. Felzenszwalb and A. McAllester. The generalized A* Architecture. *Journal of Artificial Intelligence Research*, 2007.

[24] J. Pearl. *Heuristics*. Addison-Wesley, 1984.

[25] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

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

[27] A. Storkey and C. Williams. Image modelling with position-encoding dynamic trees. *IEEE T. PAMI*, 25(7), 2003.

[28] Z. Tu, X. Chen, A. Yuille, and S. Zhu. Image Parsing: Unifying Segmentation, Detection, and Recognition. *IJCV*, 63(2):113–140, 2005.

[29] L. Zelnik and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2005.

[30] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. Yuille. Unsupervised Structure Learning: Hierarchical Recursive Composition, Suspicious Coincidence and Competitive Exclusion. In *ECCV*, 2008.

[31] Q. Zhu, L. Wang, Y. Wu, and J. Shi. Contour Context Selection for Object Detection: A Set-to-Set Contour Matching Approach. In *ECCV*, 2008.

[32] S. C. Zhu and D. Mumford. Quest for a Stochastic Grammar of Images. *Foundations and Trends in Comp. Gr. and Vis.*, 2007.