3 Recursive Compositional Models

This section describes the basic properties of RCMs as developed in recent work [28][29][30][31] [32][33][34] [36] [37] [38] [39][33] [34] [35] [40].

RCMs are hierarchical models of objects and images and are the borderline of generative and discriminative methods. They contain a hierarchy of hidden variables describing the structure of the image/object together with models for how these hidden variables relate to the image. They are specified in terms of probabilities defined over graphs. The parameters of the models, and in some cases the structure are learn. An efficient inference algorithm outputs the best estimate of the hidden variables. Because these hidden variables give a rich description of the object/image they can be used for a range of visual tasks: e.g., segmentation, detection, matching, and categorizing.

Section (3.1) describes the representation of RCMs and the forms of their distributions. In section (3.2) we describe the compositional inference algorithm. Section (3.3) describes how these models can be learnt. In section (3.4) we describe variants of RCMs.

3.1 The Representation and Form of the Probability Distribution

The object, or image, is represented by a hierarchy where parts are composed from sub-parts which, in turn, are composed of sub-sub-parts, and so on. This is formally specified by a graph whose parent-child relations specify those nodes which are connected. For some applications the graph structure is fixed – in which case the parent-child relations are referred to as AND-nodes (i.e. a part is composed of sub-part1 and sub-part2...). For other applications, the graph structure is variable and some parent-child relations are called OR-nodes (i.e. a part is composed of sub-part1 or sub-part2 or...).

The Graph Structure. We define a graph $G = (V, E)$ where V is the set of nodes (vertices) and E is the set of edges (i.e. nodes μ, ν of the graph are connected if $(\mu, \nu) \in E$). There is a parent-child structure so that $ch(\nu)$ denotes the children of node ν. We require that the graph to be tree-like so that $ch(\nu) \bigcap ch(\mu) =$ for all $\nu, \mu \in V$ (this requirement will be relaxed later). The edges are defined by the parent-child relationships and by the requirement that all the children of a node have an edge connecting them. Hence $(\nu, \mu) \in E$ provided either $\mu \in ch(\nu)$, $\nu \in ch(\mu)$, or there exist ρ s.t. $\mu, \nu \in ch(\rho)$. We define μ_R to be the root node of the graph. We let V^{LEAF} denote the leaf nodes. For any node ν, we define V_ν to be the subtree formed by the set of descendent node with ν as the root node.

There are two types of nodes. The basic type of node is an *AND node* and is used in all our examples. The intuition is that the an AND node is composed of its child nodes (e.g. the part represented by the parent node is composed of sub-parts represented by its child nodes). We represent the set of AND nodes of a graph by V^{AND}, see figure (1). For some applications [33][39], see figure (2) we supplement these with *OR nodes* [26][27]. These OR nodes act as *switch nodes* which enable us to alter the topology of the graph: if ν is an OR node then it can be related to, and only one, of its children. We represent the set of OR nodes as V^{OR}. OR enable to efficiently represent mixtures of distributions. A small graph structure with 7 levels and 40 nodes can represent 100 different topologies [33][39].

Variable defined on the graph. State variables are assigned to the graph nodes indicating properties of the parts (e.g. the subregion in the image that they correspond to). The precise nature of the state variables depends on the application and, in particular, will vary when modeling objects or modeling images. For example, the state of the top node for an object model will correspond to the orientation, scale, and center position of the object – while the state of the leaf nodes will correspond to the orientation and position of elements on the boundary of the object. States of intermediate nodes correspond to the positions, scales, and orientations of subparts of the object.

Each node $\mu \in V$ is assigned a state variable z_μ and we denote the state variables for the entire graph by $z = \{z_\mu : \mu \in V\}$. We denote $z_{ch(\mu)} = \{z_\nu : \nu \in ch(\mu)\}$ to be shorthand for the states of the child nodes of μ. The state variable indicates properties of the node and, in particular, the subregion $D(z_\mu)$ of the image domain $D \in R^2$ corresponding to the node. In several applications [37][33][39] [35] the state variable of node μ correspond to the position $\vec{x}_\mu$, orientation θ_μ, and scale s_μ of a subpart of the object: hence $z_\mu = (\vec{x}_\mu, \theta_\mu, s_\mu)$ (and $D(z_\mu)$ is calculated from these). In one application [38], the graph structure is fixed (e.g. a quadtree) so each node μ corresponds to a specific sub-domain $D(z_\mu)$ of the image domain. The rest of z_μ denotes a segmentation-recognition (s_μ, c_μ) pair which represents the segmentation of the image region $D(z_\mu)$ into subregions (s_μ is one of a set of forty segmentation templates – see figure (3) with labels c_μ into one of 23 region labels.

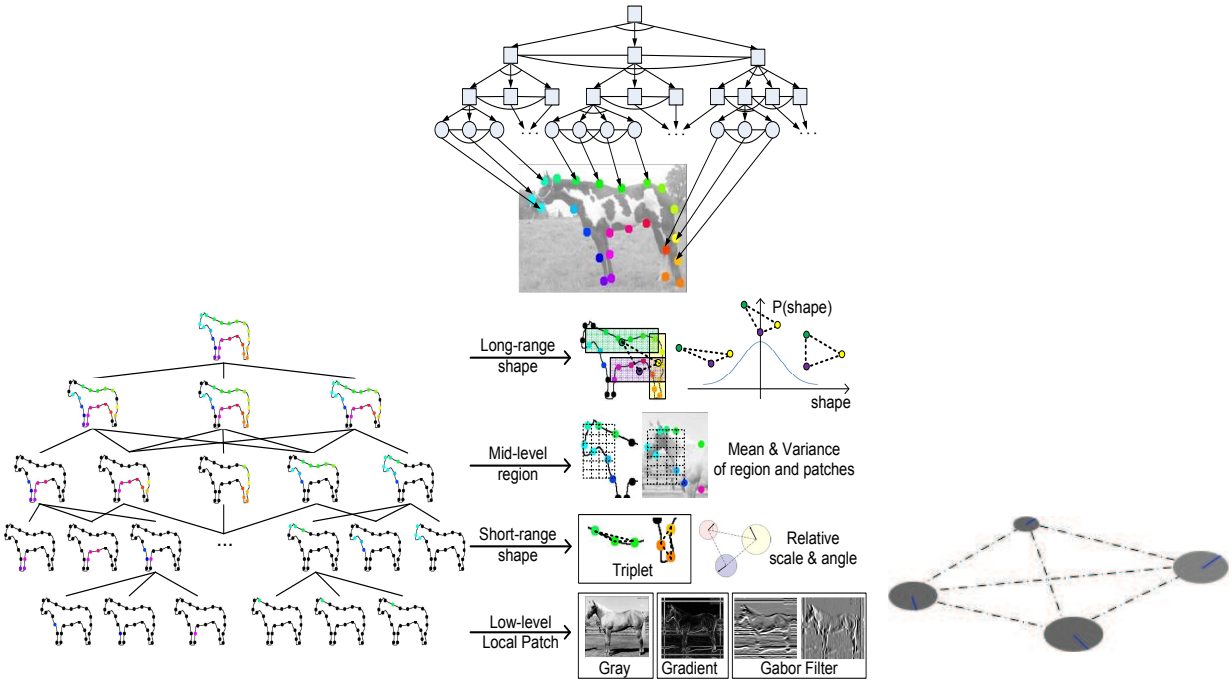

Figure 1: Figure to illustrate the graph structure, constructing the hierarchy, and the triplets. Left and middle panel: the leaf nodes of the hierarchy correspond to the positions of the boundary of the horse, the higher-level nodes correspond to positions of the parts of the horse. Right panel: a Hierarchical Log-Linear Model [35] has a hierarchical representation of a large variety of different images cues and spatial interactions at a range of scales. The bottom panel shows how subparts are combined to build bigger parts.

For applications with OR nodes, we supplement the state variables z_μ by a binary variable $t_\mu \in \{0, 1\}$ indicating whether or not the node is selected. This selection is non-local and depends which nodes are selected at higher levels in the graph. If a parent node is selected then either: (a) it is an AND node so all its child nodes are also selected, or (b) it is an OR node and only one of its child nodes are selected. A node can only be selected if its parent node is selected (except for the root node which is always selected). Formally, we impose consistency constraints so that: (i) if ν is an AND node with $t_\nu = 1$ then $t_\rho = 1$ for all $\rho \in ch(\nu)$, (ii) if ν is an OR node and $t_\nu = 1$ then $t_\rho = 1$ for one, and only one, $\rho \in ch(\nu)$, and (iii) and if $t_\nu = 0$, then $t_\rho = 0$ for all $\rho \in ch(\nu)$.).

Probability distributions defined on over the state variables on the graph. We now define probability distributions on the state variables defined on the graph. These distributions are of exponential form defined in terms of potentials $\phi(.)$ which are weighted by parameters α. The specify, for example, the probability distributions for the relative states of the hidden variables and the imaging terms.

There are two types of terms: (i) "prior potentials" defined over the cliques of the graph $\vec{\phi}(z_\mu, \{z_\nu : \nu \in ch(\mu)\})$, for all $\mu \in V$, which are independent of the image $\mathbf{I}$, and (ii) "data potentials" of form $\vec{\phi}_D(\mathbf{I}, D(z_\mu))$, for all $\mu \in V$, which depend on measurements of the image $\mathbf{I}$ within the domain $D(z_\mu)$. (Note: in some papers [33][39][35] the "prior" terms are decomposed into "vertical" and "horizontal" terms but this loses the generality that we aim for in this overview). The potentials will have coefficients $\vec{\alpha}_\mu, \vec{\alpha}_\mu^D$ respectively for all $\mu \in V$. We use α to denote $\{\vec{\alpha}_\mu, \vec{\alpha}_\mu^D\}$. These potentials depend on the application and on the type of learning.

These probability distributions can be specified either as *generative or discriminative*. A *generative model* can be decomposed into a prior model $P(z)$ on the state z together with a likelihood function $P(\mathbf{I}|z)$ for the observed image, where:

$$P(\mathbf{I}|z) = \frac{1}{Z(\alpha)} \exp\{-\sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)})\},$$

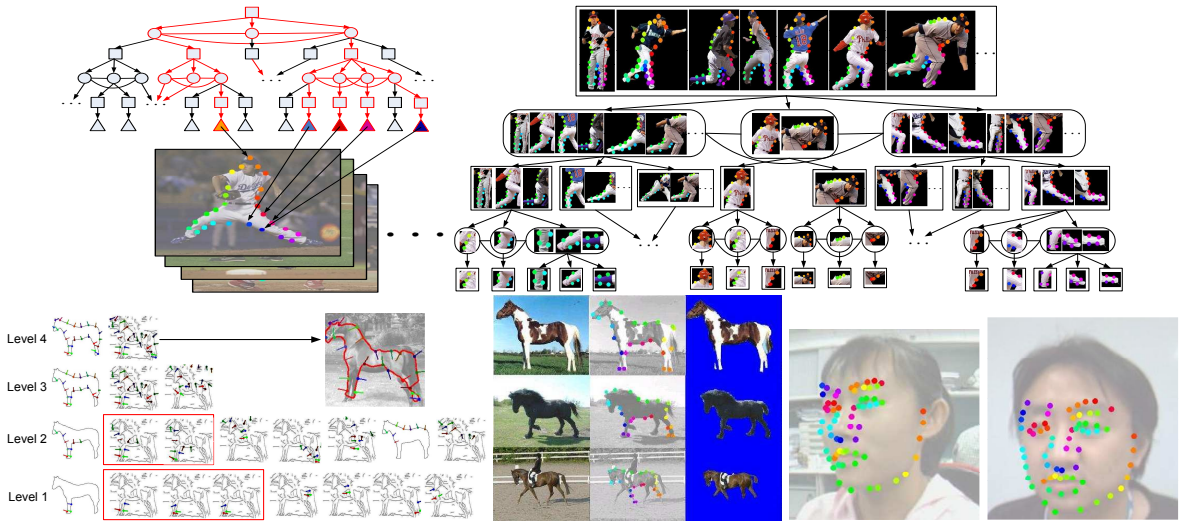

Figure 2: Upper Left Panel: The AND/OR representation allows us to model many poses of the object [33][39]. A parse graph, which is an instantiation of the AND/OR graph, represent a specific pose of the human body. The nodes and edges with red boundary indicate one parse graph. In this model, there are 98 poses corresponding to different topologies of the AND/OR graph. Upper Right Panel: The AND/OR graph is an efficient way to represent different appearances of an object. The bottom level of the graph indicates points along the boundary of human body. The higher levels indicating combinations of elementary substructures. The graph that we used contains eight levels (three lower levels are not shown here due to lack of space). Color points distinguish different body parts. Lower Left Panel: Snapshot of compositional inference. Lower Centre Panel: illustration of the hierarchical models for segmentation. Lower Right Panel: Multi-view face alignment.

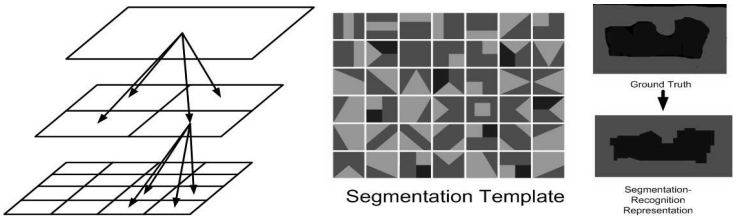

Figure 3: The left panel shows the structure of the Hierarchical Image Model [38]. The grey circles are the nodes of the hierarchy. All nodes, except the top node, have only one parent nodes. All nodes except the leafs are connected to four child nodes. The middle panel shows a dictionary of 30 segmentation templates. The color of the sub-parts of each template indicates the object class. Different sub-parts may share the same label. For example, three sub-parts may have only two distinct labels. The last panel shows that the ground truth pixel labels (upper right panel) can be well approximated by composing a set of labeled segmentation templates (bottom right panel).

$$P(z) = \frac{1}{Z(\alpha^D)} \exp\{-\sum_{\mu \in V} \vec{\alpha}_\mu^D \cdot \vec{\phi}_\mu^D(\mathbf{I}, D(z_\mu))\}. \tag{1}$$

A *discriminative model* will attempt to directly model the posterior distribution $P(z|\mathbf{I})$:

$$P(z|\mathbf{I}) = \frac{1}{Z(\alpha, \mathbf{I})} \exp\{-\sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)}) - \sum_{\mu \in V} \vec{\alpha}_\mu^D \cdot \vec{\phi}_\mu^D(\mathbf{I}, D(z_\mu))\}. \tag{2}$$

But it is *important to realize* that this type of discriminative model includes an *explicit* prior on the state z given by the $\sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)})$ term in the exponent in equation (2). This is obtained by applying Bayes rule $P(z|\mathbf{I}) = P(\mathbf{I}|z)P(z)/P(\mathbf{I})$ and identifying the components of $P(z|\mathbf{I})$ which depend on $\mathbf{I}$ as $P(\mathbf{I}|z)/P(\mathbf{I})$ and those which depend only on z as $P(z)$ (up to an unknown normalization constant). In most of our applications we use distributions of discriminate form in equation (2). This is for three main reasons: (i) it is far easier to learn discriminative models for intensities rather than generative ones (e.g. we can use AdaBoost to discriminate between the interiors and backgrounds of cows and horses, but there are no generative models that can realistically synthesize the intensity properties of cows and horses [172], (ii) it is easier to learn discriminative models than generative ones (because of the difficulties of dealing with the normalization factors $Z(\alpha)$), (iii) our discriminative models are sophisticated and contain hidden variables z and priors, so they can deal with large variations of the data.

Recursive Formulation: (required for inference and learning). Both distributions, generative and discriminative, are of Gibbs form and have an energy function: $E(z|\mathbf{I}) = \sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)}) + \sum_{\mu \in V} \vec{\alpha}^D_\mu \cdot \vec{\phi}^D_i(\mathbf{I}, D(z_\mu))$. We exploit the tree-structure and express this energy function recursively by defining an energy function $E_\nu(z_{des(\nu)}|\mathbf{I})$ over the subtree with root node ν in terms of the state variables $z_{des(\nu)}$ of the subtree where $des(\nu)$ stands for the set of descendent nodes of ν – i.e. $z_{des(\nu)} = \{z_\mu : \mu \in V_\nu\}$.

This gives $E_\nu(z_{des(\nu)})|\mathbf{I}) = \sum_{\mu \in V_\nu} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\nu)}) + \sum_{\mu \in V_\nu} \vec{\alpha}^D_\mu \cdot \vec{\phi}_D(\mathbf{I}, D(z_\mu))$, which can be computed recursively by $E_\nu(z_{des(\nu)}|\mathbf{I}) = \sum_{\rho \in ch(\nu)} E_\rho(z_{des(\rho)}|\mathbf{I}) + \vec{\alpha}_\nu \cdot \vec{\phi}(z_\nu, z_{ch(\nu)}) + \vec{\alpha}^D_\nu \cdot \vec{\phi}^D(\mathbf{I}, D(z_\nu))$, so that the full energy $E(z|\mathbf{I})$ is obtained by evaluating $E_\nu(.)$ at the root node μ_R.

We define corresponding Gibbs distributions over the subtrees (which will be useful during unsupervised learning). These distributions are either generative or discriminative depending on the normalization constant $P_\nu(z_{des(\nu)}|\mathbf{I}) = \frac{1}{Z(\alpha,\mathbf{I})} \exp\{-E_\nu(z_{des(\nu)}|\mathbf{I})\}$, $P_\nu(z_{des(\nu)}, \mathbf{I}) = \frac{1}{Z(\alpha)} \exp\{-E_\nu(z_{des(\nu)}|\mathbf{I})\}$. This illustrates the recursive form of the RCM. The state variables are of the same form at all levels of the hierarchy, the parent-child structures are similar, and so are the forms of the prior potentials. The energy functions, and probability distributions, can be calculated recursively.

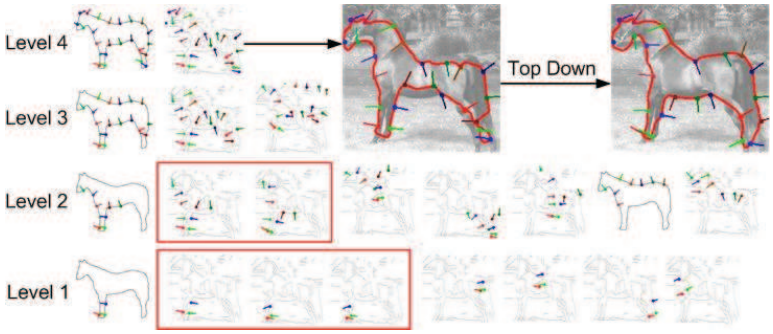

Figure 4: This snapshot illustrates the compositional algorithm [29][**?**]. The bottom-up process starts from level 1 and constructs proposals from children nodes. Only proposals above threshold are kept. Similar proposals in the same local window defined over space, orientation and scale are grouped together. See the proposals inside the windows. The proposal with the highest score within this cluster is kept to propagate to upper level. The top-down process starts by dynamic programming to fill in the boundary curve across the points fixed by the proposals. The solution will be further refined by changing the proposals. Changes appear in the body and right leg. See the two boundary images.

3.2 Inference Algorithm

The purpose of the inference algorithm is to estimate the most probable state $z^* = \arg\min_z E(z|\mathbf{I})$ or a set of highly probable (or low energy) states.

The algorithm is called *compositional inference* [29][31], see table [**?**]. It is a pruned form of dynamic programming (DP) that exploit the independence structure of the graph model. It is run bottom-up starting by estimating possible states for the leaf nodes and proceeding to estimate possible states for the nodes higher up the tree (a top-down stage is sometimes run as a variant). Although DP guaranteed to be polynomial in the relevant quantities (no. layers, no. graph nodes, state space of z) full DP is too slow because of the large size of the state space (i.e. range of values that z_μ can take for each node μ). The pruning reduces the allowable states of a node μ to a set of *proposals* (borrowing terminology from the MCMC literature). These proposals are chosen are selected by two mechanisms: (i) *energy pruning* - to remove proposals corresponding to large energy, and (ii) *surround suppression* - to remove proposals which are too similar to each other (e.g, similar to non-maximal suppression – or techniques for keeping samples apart in particle filters)

More precisely, we proceed as follows. *Initialization*: at each leaf node $\nu \in V^{LEAF}$ we calculate the states $\{p_{\nu,b}\}$ (b indexes the proposal) such that $E_\nu(p_{\nu,b}|\mathbf{I}) < T$ (*energy pruning* with threshold T) and $E_\nu(p_{\nu,b}|\mathbf{I}) \leq E_\nu(p_\nu|\mathbf{I})$ for all $z_\nu \in W(p_{\nu,b})$ (*surround suppression* where $W(p_{\nu,b})$ is a window centered on $p_{\nu,b}$). We refer to the $\{p_{\nu,b}\}$ as proposals for the state z_ν and store them with their energies $E_\nu(p_{\nu,b}|\mathbf{I})$. *Recursion for AND nodes*: to obtain

Input: $\{MP^1_{\nu^1}\}$. Output:$\{MP^L_{\nu^L}\}$. $\oplus$ denotes the operation of combining two proposals.
Loop : $l = 1$ to L, for each node ν at level l

- IF ν is an OR node
  1. Union: $\{MP^l_{\nu,b}\} = \bigcup_{\rho \in T_\nu, a=1,...,M^{l-1}_\rho} MP^{l-1}_{\rho,a}$
- IF ν is an AND node
  1. Composition: $\{P^l_{\nu,b}\} = \oplus_{\rho \in T_\nu, a=1,...,M^{l-1}_\rho} MP^{l-1}_{\rho,a}$
  2. Pruning: $\{P^l_{\nu,a}\} = \{P^l_{\nu,a} | E(\Lambda^l_{\nu,a}) > K_l\}$
  3. Local Maximum: $\{(MP^l_{\nu,a}, CL^l_{\nu,a})\} = LocalMaximum(\{P^l_{\nu,a}\}, \epsilon_W)$ where ϵ_W is the size of the window W^l_ν defined in space, orientation, and scale.

Figure 5: The compositional inference algorithm [31]. Proposals $\{MP\}$ are composed to form proposals P at higher levels, which are pruned by energy thresholding and surround suppression, implemented by LocalMaximum. The inference algorithm is applicable to all RCMS. (The OR step is dropped if there are no OR nodes.

the proposals for an AND node μ at a higher level of the graph $\mu \in V/V^{LEAF}$ we first access the proposals for all its child nodes $\{p_{\mu i,b_i}\}$ where $\{\mu i : i = 1, ..., |ch(\mu)|\}$ denotes the set of child nodes of μ and their energies $\{E_{\mu i}(p_{\mu i,b_i}|\mathbf{I}) : i = 1, ..., |ch(\mu)|\}$. Then we compute the states $\{p_{\mu,b}\}$ such that $E_\mu(p_{\mu,b}|\mathbf{I}) \leq E_\mu(p_\mu|\mathbf{I})$ for all $z_\mu \in W(p_{\mu,b})$ where $E_\mu(p_{\mu,b}|\mathbf{I}) = min_{\{bi\}}\{\sum_{i=1}^{|ch(\mu)|} E_{\mu i}(z_{des(\mu i,b_i)}|\mathbf{I}) + \vec{\alpha}_\mu \cdot \vec{\phi}(p_\mu, \{p_{\mu i,b_i}\}) + \vec{\alpha}^D_\mu \cdot \vec{\phi}^D(\mathbf{I}, D(p_\mu))\}$. *Recursion for OR nodes*: to obtain proposals for an OR node μ we access the proposals and energies for the child nodes as above. Then we compute the states $\{p_{\mu,b}\}$ such that $E_\mu(p_{\mu,b}|\mathbf{I}) \leq E_\mu(p_\mu|\mathbf{I})$ for all $z_\mu \in W(p_{\mu,b})$ where $E_\mu(p_{\mu,b}|\mathbf{I}) = min_i min_{bi}\{E_{\mu i}(z_{des(\mu i,b_i)}|\mathbf{I}) + \vec{\alpha}_\mu \cdot \vec{\phi}(p_\mu, p_{\mu i,b_i})\} + \vec{\alpha}^D_\mu \cdot \vec{\phi}^D(\mathbf{I}, D(p_\mu))$.

This algorithm outputs a set of proposals $\{p_b\}$ for the entire graph together with their energies $E(p_b|\mathbf{I})$. The proposal with smallest energy is taken as the MAP estimate: i.e. we estimate $z^* = \arg\max P(z|\mathbf{I})$ by $p_{b^{ast}} = \arg min_b E(p_b|\mathbf{I})$. The use of surround suppression ensures that these proposals are highly separated.

Observe that intuitively the algorithm is similar to constraint satisfaction. We have a set of proposals for sub-parts which are composed to form proposals for parts by ensuring that these compositions obey the appropriate probabilistic constraints (as expressed by the prior potential terms and the data potential term).

3.3 Learning

Learning is performed always for the parameters of the probabilistic models and in some cases for the harder task of learning the graph structure. The nature of the learning depends on the amount of supervision required – e.g. do we specify the name of the object and also its precise boundary? or just the name? or whether the image contains one of several types of object or even no object at al? If the structure of the graph is known, we have successfully applied discriminative machine learning algorithms – such as structure perceptron and structure max-margin – to estimate the parameters of the distribution (and in other related work – PGMM – we have successfully used the EM algorithm). For some examples, we have successfully learnt the structure by one shot learning (with a labeled example of the object boundary) and in others we have learnt the structure hierarchically with minimal supervision (ECCV paper).

Partially supervised learning. Projects on hierarchical log-linear models [35] AND/OR graphs [33] [39], and hierarchical image models [38] are examples where we used discriminative methods to learn discriminative models with partial supervision. For the hierarchical log-linear model and the AND/OR graph the boundary of the object is specified in the training data. For the hierarchical image model pixels are labeled into 23 classes. The graph structure for the hierarchical log-linear model is learnt by clustering in one shot [35], while the structure for the AND/OR graph was hand specified [33] [39].

After learning, or specifying, the structure we must estimate the parameters $\{\alpha, \alpha^D\}$. We used a dataset $\{(\mathbf{I}_a, S_a)\}$ where information S_a about the state of the leaf nodes are provided for image $\mathbf{I}_a$. From this, we can automatically estimate the ground truth states z^T_a for the model (see papers for details). This gives $\{(\mathbf{I}_a, z^T_a)\}$.

We have used two different discriminative learning algorithms for estimating the parameters. The *structure perceptron* algorithm [80][81][82] and *structure max-margin* [83][84][85][86][87]. Both algorithms are alternatives to the

standard maximum likelihood (ML) method which is difficult to apply to this problem since it involves computing the partition functions (i.e. $Z(\alpha)$). Structure perceptron is simpler to implement but has max-margin has better performance guarantees and enables us to use the kernel trick. In both cases, we specify a dictionary of potentials ϕ, ϕ^D. Cross-validation and related techniques are used to prevent over-generalization.

Structure Perceptron Learning. The structure perceptron algorithm is an iterative process which compares the energy evaluated for the ground-truth (as estimated automatically from the labeled silhouettes) to the energy obtained by using the inference algorithm to determine the most probable state. Formally the algorithm is specified by the update rules:$\vec{\alpha}_\mu^{t+1} = \vec{\alpha}_\mu^t + \vec{\phi}(z_\mu^T, z_{ch(\mu)}^T) - \vec{\phi}(z_\mu^*, z_{ch(\mu)}^*)$, $\vec{\alpha}_\mu^{D,t+1} = \vec{\alpha}_\mu^{D,t} + \vec{\phi}^D(\mathbf{I}, D(z_\mu^T)) - \vec{\phi}_i^D(\mathbf{I}, D(z_\mu^*))$,where z^* is the best estimate provided by the inference algorithm (with current settings of α). See table (6).

Input: A set of training images with ground truth (x^i, y^i) for $i = 1..N$. Initialize parameter vector $\alpha = 0$. For $t = 1..T, i = 1..N$

- find the best state of the model on the i'th training image with current parameter setting, i.e., $y^* = \arg\max_y \psi(x^i, y) \cdot \alpha$
- Update the parameters: $\alpha = \alpha + \psi(x^i, y^i) - \psi(x^i, y^*)$
- Store: $\alpha^{t,i} = \alpha$

Output: Parameters $\gamma = \sum_{t,i} \alpha^{t,i}/NT$

Figure 6: Structure-perceptron learning. This is used to train the HLLM

Structure perceptron has the desirable property, empirically verified for these applications, that many of the weights remain close to zero (with weights initialized at zero). Therefore it acts like a selection process which selects which potentials to use from a dictionary by awarding them large weights.

Structure Max-Margin Learning. For each training example i, we have the ground-truth $z^T(i)$ and we can use the compositional inference algorithm to estimate the state $\hat{z}(i)$. We define an error measure $\Delta(z_\mu^T(i), z_\mu(i)) = 1$ if $dist(z_\mu^T(i), z_\mu(i)) \geq \sigma$ and $\Delta(.,.) = 0$ otherwise (where σ is a constant). We define an error measure for the entire parse $L(z^T(i), z(i)) = \sum_{\nu \in V^{AND}} \Delta(z_\mu^T(i), z_\mu(i)) + \sum_{\nu \in V^{LEAF}} \Delta(z_\mu^T(i), z_\mu(i))$. Structure Max-Margin requires us to find values of the weights $\{\alpha, \alpha^D\}$ to minimize the criterion: $\frac{1}{2}|\alpha|^2 + C\sum_i \zeta_i$ s.t. $\{\vec{\alpha}_\mu \cdot \phi(z^T) + \vec{\alpha}^D \cdot (\vec{\phi}(\mathbf{I}, z^T(i))\}$ $-\{\vec{\alpha}_\mu \cdot \phi(z) + \vec{\alpha}^D \cdot (\vec{\phi}(\mathbf{I}, z(i))\} \geq L(z^T(i), z(i)) - \zeta_i, \ \forall i, z$.

This is a generalization of the standard criterion used for learning Support Vector Machines for binary classification. It attempts to find values of the parameters α so that the energies are smallest for the states z closest to the ground truth z^T. The minimization can be done, as for standard SVM's, by using the dual formulation. In addition, we use the working set method [86] to deal with the exponential number of constraints (due to the exponential number of z for each image i). We use the kernel trick [76] to replace the dot products with a kernel. We use the radial basis function kernel [76] (for details, see paper [33][39]).

Unsupervised Structure Learning. We also performed [37] unsupervised learning (the structure and parameters of the distributions are unknown – an object, or one of several objects, is present in an image with significant and varied background. See related work by [50].

This is a challenging problem. We assume, initially, that the image can be represented by a set of attributed features (e.g. edgelets or interest points). But it is unclear which features in the image correspond to the object (and to which part of the object) and which to the background.

The procedure used is hierarchical clustering. We define a basic dictionary of oriented edgelets quantized to have orientations at $0, 45, 90, 135$ degrees. Then we use the principle of suspicious coincidences [149] to perform clustering to determine frequently occurring (hence suspicious) compositions (triplets) of the dictionary elements. These serve as initial estimates of triplet subgraphs $P(\{z_{ch(\nu)}\}|z_\nu)P(z_\nu)$. This gives us a *level 1 dictionary*, see figure (7), where (it is hoped) some of the dictionary elements correspond to low-level graphs of the object model and the rest correspond to background (each element of this dictionary is a distribution $P(\{z_{ch(\nu)}\}|z_\nu)P(z_\nu)$ whose means are shown in figure (7) (we keep the acceptance threshold low because we cannot tolerate false negatives – i.e., we must have models for the low-level parts of the object – but false positives will be removed automatically at the higher level composition/clustering stages). We now detect instances of the level 1 dictionary in the image (using the compositional

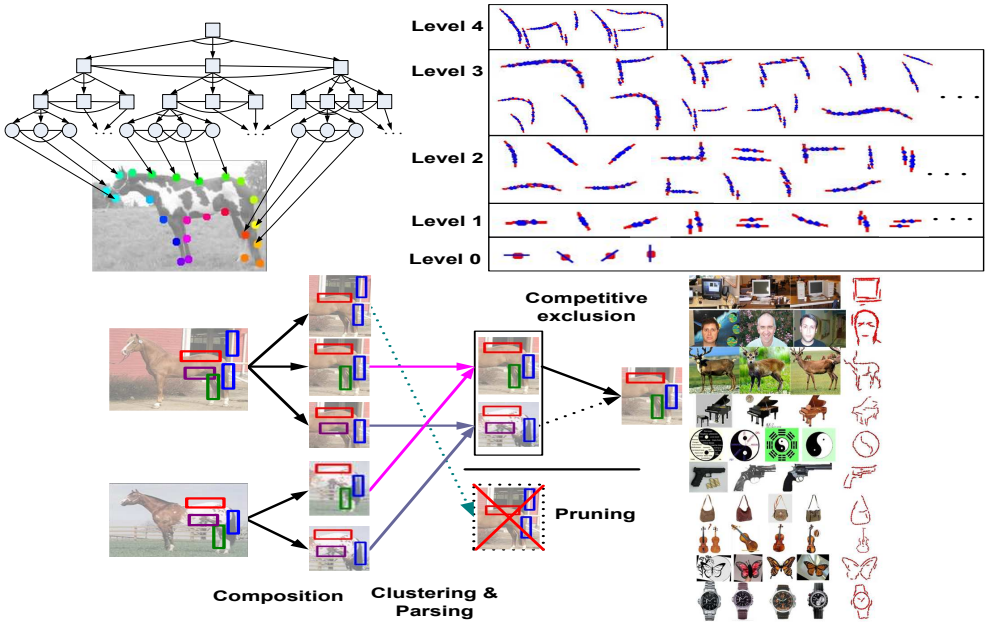

Figure 7: Upper left panel: The compositional representation of the object [37]. The boxes represent non-leaf nodes. The circles denote leaf nodes that directly relate to properties of the input image. Upper right panel: This figure shows elements of the hierarchy of schemas for horses (mean values only – i.e., we do not illustrate the shape variability of these schema). Observe how the set includes "generic" shapes at low levels, but finds horse-specific parts at the higher levels. Lower left panel: illustrates the unsupervised learning procedure. The first column shows the responses of the features from the "vocabulary" at level l. the second column shows the compositions at step 1). the third column shows the clustering at step 2. The noise(non-regular) pattern bounded by dotted line is pruned out by step 4. The last column plots the results after step 5. At step 5, the compositions, which are constructed by different components, but parse the same areas in the image, are grouped together (the maximum is kept). Lower right panel: the hierarchies learnt for other objects using the same strategy.

inference algorithm), perform clustering to determine frequently occurring compositions (triplets) of elements of the level 1 dictionary, and use this to generate a *level 2 dictionary*, see figure (7). We repeat this procedure to obtain new dictionaries at higher levels. The process stops automatically when it finds the largest structure that repeats in the images (e.g., a horse). The dictionary elements are generic, reminiscent of Gestalt grouping laws, at low levels and become more object specific at higher levels.

3.4 Alternative Composition: Flat Composition and Knowledge Propagation

For completeness, we mention two additional techniques that we have also used for composing elementary models: (i) flat composition, and (ii) knowledge propagation.

Flat composition was used in [30][**?**] to build a model by adding elementary components (triplets) to an existing model, see figure (8)(left panel). The model can be expressed as an OR of different models (e.g. mixture model) where each OR child node corresponds to a different aspect of the object. Each aspect is expressed as a composition of triplets. When growing the model, a new triplet can either be added to an existing aspect model or can be used to create a new aspect. The structure, and parameters, of the model are learnt automatically. A dictionary (see previous section) is used to propose ways to grow the model (with the assumption that triplets that occur frequently in the training data are likely to correspond to the object). New proposals for growing the model are evaluated by standard model selection techniques. The compositional structure of the model enable efficient inference, by dynamic programming, and parameter learning by the EM algorithms (where the proposals give good initial conditions for avoiding local minima). This method used interest point features as input which, because of their sparseness and rich appearance, makes this flat compositional learning practical (this method would not work for edgelets because they are indistinguishable and there are too many of them – for them, the hierarchical method is required).

Knowledge propagation [36] [40] is an alternative form of composition where the models being composed are of different forms. For example, we can compose a PGMM (or POM-IP) which represents the object by spare interest points with a POM-mask which represents the object by a mask [36]. The term *knowledge propagation* is used because the first model POM-IP is used to help learning and inference for the POM-mask by providing knowledge,

see figure (8)(center panel). In more detail, the POM-IP model learns a distribution $P(d_1\mathbf{I}|s,G)\ P(s,G)$, where $d_1()$ represents the interest points extracted from the data, s is the aspect, and G is the pose. As described above (previous paragraph) the POM-IP can be learnt in an unsupervised manner. The POM-mask is of form $P(d_2(\mathbf{I})|L,q)\ P(L|s,G)\ P(q)$ where $d_2(\mathbf{I})$ denotes features extracted from the image, L is the position of the mask, and q is a distribution on the features. The variables G, s link the POM-IP to the POM-mask. As described in [36], the POM-IP (after learning) supplies knowledge of s, G which can make learning, and performing inference, of the POM-mask model. The result is a composition of the POM-IP with the POM-mask as a single probabilistic model.

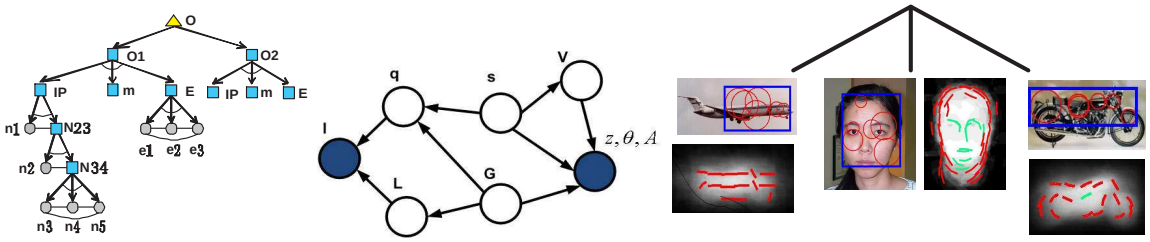

Figure 8: Left Panel: AND-OR graph for POM-IP [30][32]. Triangle: Or node. Rectangle: And node. Circle: Leaf node. IP: interest point. m: object mask. E: edgelets. Centre Panel [36][40]: Bayes net for combining POM-IP and POM-mask. Right Panel: Hybrid Model. Training images consist of faces, motorbikes and airplanes where we don't know the identity of the class for each image.