Recursive Compositional Models for Vision: description and review of recent work

Long (Leo) Zhu [3], Yuanhao Chen [1], and Alan Yuille [1,2],
[1] Department of Statistics
University of California at Los Angeles
Los Angeles, CA 90095
yuille@stat.ucla.edu
[2] Department of Brain and Cognitive Engineering,
Korea University
Seoul, Korea
[3] Department of Computer Science
New York University
New York, New York

Abstract

This paper describes and reviews a class of hierarchical probabilistic models of images and objects. Visual structures are represented in a hierarchical form where complex structures are composed of more elementary structures following a design principle of recursive composition. Probabilities are defined over these structures which exploit properties of the hierarchy – e.g. long range spatial relationships can be represented by local potentials at the upper levels of the hierarchy. The compositional nature of this representation enables efficient learning and inference algorithms. In particular, parts can be shared between different object models. Overall the architecture of Recursive Compositional Models (RCMs) provides a balance between statistical and computational complexity.

The goal of this paper is to describe the basic ideas and common themes of RCMs, to illustrate their success on a range of vision tasks, and to gives pointers to the literature. In particular, we show that RCMs generally give state of the art results when applied to a range of different vision tasks and evaluated on the leading benchmarked datasets.

I. Introduction

There has recently been considerable progress in design probability models specified over structured representations such as graphs and grammars [1],[2],[3],[4],[5]. Such models have considerable representational power, but there is a tradeoff between representation and computational efficiency. For example, natural language researchers have defined stochastic context free grammars (SCFGs) [2] by putting probability distributions over context free grammars which are able to capture important properties of language – such as the hierarchy of nouns, noun phrases, and so on – and which enable efficient inference and learning algorithms by exploiting the statistical independence between different parts of the SCFG. More sophisticated models [2] are better at representing language but require more advanced inference and learning algorithms.

It is attractive to formulate vision as probabilistic inference on structured probability representations. This seems both a natural way in which to deal with the complexities and ambiguities of image patterns [6] [7] and also fits into a more unified framework for cognition and artificial intelligence [8]. But vision is a particularly challenging problem to formulate in this manner due to the complexity and ambiguity of the visual patterns which occur in natural images. This has motivated recent work which uses principles such as compositionality [9] to build stochastic grammar models of images and objects [5].

This paper describes a class of probabilistic models of objects and images which are motivated both by the complexity of images and by the need to have efficient inference and learning algorithms. The key

design principle is *recursive compositionality* and so we refer to our models as Recursive Compositional Models (RCMs). Visual patterns are represented by RCMs in a hierarchical form where complex structures are composed of more elementary structures. Probabilities are defined over these structures exploiting properties of the hierarchy (e.g. long range spatial relationships can be represented by local potentials). This compositional structure enables efficient learning and inference algorithms. In particular, when modeling multiple objects we can share parts between different objects models and obtain enormous gains in computational efficiency. Overall architecture of RCMs provides a balance between representational and computational complexity.

We note that in this paper all the probability models, with the exception of the unsupervised learning, can be reformulated within an alternative non-probabilistic framework since inference is performed by MAP and the learning involves approximating, or bounding, the partition functions – see section (IV-C). We prefer, however, to formulate the models in probabilistic terms for the following reasons. Firstly, the probabilistic framework offers a rich conceptual framework to model all vision problems (i.e. not only those we address in this paper). Secondly, section (IV-C) describes how the learning can be obtained from the probabilistic formulation by taking bounds. Thirdly, the unsupervised learning briefly sketched in section (VI) can only be justified within the probabilistic framework.

The goal of this paper is to describe the basic ideas and themes of RCMs and, in particular, to bring out the commonalities between our previous work using these models [10], [11],[12],[13],[14] [15],[16],[17]. We will show that the same types of models are very effective at a range of different visual tasks – including object detection, object parsing, boundary detection, object matching, and image labeling – when tested on benchmarked datasets including the Weizmann horse dataset [18], PASCAL [19], LabelMe [20], Berkeley Baseball Players [21], a cow dataset [22], a face dataset [23], and the MSRC image label dataset [24].

Section (II) describes the main research themes which have influenced our work. Section (III) is intended to give contact by describing standard 'flat' Markov Random Field (MRF) models and how they differ from hierarchical models. Section (IV) describes the basic framework of RCMs which are common to all applications. In section (V) we give five examples which show how this framework can be applied to different problems. For completeness, section (VI) sketches our work on unsupervised learning of object models.

II. Background and Themes

This section mentions four major research themes which relate to our work on recursive compositional models. The themes are : (i) pattern theory and generative models, (ii) artificial neural networks, (iii) biologically motivated models, and (iv) probabilistic models of artificial intelligence and cognition. There is a large literature on these topics which we refer to in more detail in our papers.

Pattern Theory and generative models. One main theme starts with the work of Grenander on pattern theory [6] [7] and its recent developments which include image parsing [25][26], compositional models and stochastic grammars [9],[5],[27]. Zhu and Mumford's review paper [5] gives a discussion of this literature with many illustrations of generative grammars including AND/OR graphs. Successful results have been obtained by using active basis functions [28] to implement the generative models. Other work within this broad theme includes [29]. There is also related literature on multiscale Markov models for signal and image processing reviewed by Willsky [30]. Although our work closely relates to this literature our emphasis is more on discriminative techniques and efficient inference and learning algorithms.

Artificial neural networks. Another important theme is the development of artificial neural networks and, in particular, recent work on deep belief networks. This includes work by Hinton and his collaborators who construct hierarchical networks by combining restricted Boltzmann Machines [31]. There is closely related work by LeCun, Bengio, Ng and their groups [32][33],[34]. This work can also be described in the framework of probabilistic models on structured representations but differs by being restricted to a specific class of probabilistic models.

Biologically motivated models. Another research theme is motivated by known properties of the mammalian visual cortex. Ullman and Poggio's research groups have published extensively on these topics and

the following references provide an entry into the literature [35],[18],[36]. There is also closely related work by Thorpe [37]. Note that much of this work, as well as our own, is influenced by the classic work of Fukushima [38]. Our work differs by being based on probabilities and without any original biological motivation, yet there remains some intriguing similarities.

Probabilistic models of artificial intelligence and cognition. Our work also fits into the bigger picture of trying to model artificial intelligence and all aspects of cognition in terms of probabilistic models defined over structured representations. Early examples of this approach are Pearl's probabilistic models [1] and related work [39] probabilistic expert systems. Good reviews of this material are provided in [3] and their relation to artificial intelligence in [4]. Natural language is an area where these techniques have been very successful [2]. There have also been interesting applications of these ideas to modeling human cognition – e.g., see Tenenbaum, Griffiths and Kemp [40] This is also related to advances in machine learning which involve structured representation – which include structured perceptron [41], structure max-margin [42][43][44][45].

III. Technical Background

We now give a brief technical review of "flat" (i.e. non-hierarchical) probability models and discuss how they can be applied to the visual tasks addressed in this paper. We will consider two types of model: (i) those that describe the entire image and are applied to tasks such as image labeling or segmentation, and (ii) those which model objects and are applied to object detection, localization and description. Then we will describe the limitation of these models which motivate us to develop alternative models based on recursive composition.

The critical aspects of these models are: (i) the *representation* which consists of the graph structure of the model and the state variables, (ii) the *inference algorithm* used to estimate properties of the model such as the most probable state, and (iii) the *learning algorithm* used to learn the parameters of the model. In earlier models learning was often not used and instead models were hand designed.

We first discuss *image models* which are capable of describing the entire image and can be used for tasks such as image labeling and segmentation. We will concentrate on the image labeling task where the goal is to assign a label to every image pixel $\mu \in \mathcal{D}$, where $\mathcal{D}$ is the image lattice. The input is an image $\mathbf{I}$, where $\mathbf{I} = \{I_\mu : \mu \in \mathcal{D}\}$ specifies the intensity values $I_\mu \in \{0, 255\}$ on the lattice, and the output $\mathbf{W}$ is $\mathbf{W} = \{w_\mu : \mu \in \mathcal{D}\}$ is a set of image labels $w_\mu \in \mathcal{L}$, see figure (1). The nature of the labels will depend on the problem. For edge detection, $|\mathcal{L}| = 2$ and the labels l_1, l_2 will correspond to 'edge' and 'non-edge'. For labeling the MSRC dataset [24] $|\mathcal{L}| = 23$ and the labels $l_1, ..., l_{23}$ include 'sky', 'grass', and so on. Similar models can be applied to other vision problems such as image segmentation [46],[47] and binocular stereo [48] (by setting the input to be the images to the left and right eyes $(\mathbf{I}^L, \mathbf{I}^R)$ and setting $\mathbf{W}$ to be the disparity labels).

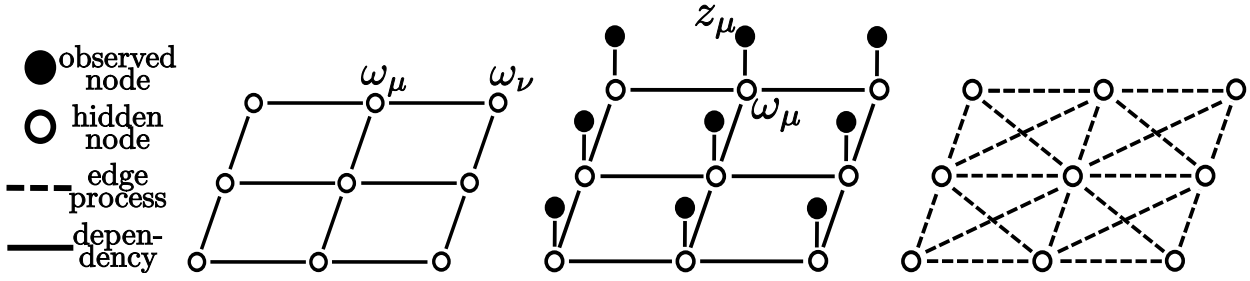

Fig. 1. GRAPHS for different MRF's. Conventions (far left), basic MRF graph (middle left), MRF graph with inputs z_μ (middle right), and graph with dense connections (far right)

We can *represent* the image labeling problem in terms of a probability distribution defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the set of nodes $\mathcal{V}$ is the set of image pixels $\mathcal{D}$ and the edges $\mathcal{E}$ are between neighboring pixels – see figure (1). The $\mathbf{W} = \{w_\mu : \mu \in \mathcal{V}\}$ are random variables specified at each node of the graph.

$P(\mathbf{W}|\mathbf{I})$ is a Gibbs distribution specified by an energy function $E(\mathbf{W}, \mathbf{I})$ which contains unary potentials $\sum_{\mu \in \mathcal{V}} \boldsymbol{\lambda}_{\mu}^{D} \cdot \boldsymbol{\phi}_{\mu}(w_{\mu}, \mathbf{I})$ and pairwise potentials $\sum_{\mu,\nu \in \mathcal{E}} \boldsymbol{\lambda}_{\mu,\nu}^{P} \cdot \boldsymbol{\psi}_{\mu,\nu}(w_{\mu}, w_{\nu})$. The unary feature functions $\boldsymbol{\phi}_{\mu}(w_{\mu}, \mathbf{I})$ depend only on the label at node/pixel μ and the input image $\mathbf{I}$. The pairwise feature functions $\boldsymbol{\psi}_{\mu,\nu}(w_{\mu}, w_{\nu})$ impose prior assumptions about the local 'context' of the labels, for example that neighboring pixels will tend to have similar labels. In many applications, the unary and binary feature functions take the same form for all nodes $\mu \in \mathcal{V}$ and edges $\mu, \nu \in \mathcal{E}$ and so these subscripts can be dropped from the feature functions. The $\boldsymbol{\lambda}_{\mu}^{D}$ and $\boldsymbol{\lambda}_{\mu,\nu}^{P}$ are parameters which are either hand-specified, or learnt from training data.

The full distribution $P(\mathbf{W}|\mathbf{I})$ is defined over the random variables $\mathbf{W} = \{w_{\mu} : \mu \in \mathcal{V}\}$ specified on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$:

$$P(\mathbf{W}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp\{-\sum_{\mu \in \mathcal{V}} \boldsymbol{\lambda}_{\mu}^{D} \cdot \boldsymbol{\phi}_{\mu}(w_{\mu}, \mathbf{I}) - \sum_{\mu,\nu \in \mathcal{E}} \boldsymbol{\lambda}_{\mu,\nu}^{P} \cdot \boldsymbol{\psi}_{\mu,\nu}(w_{\mu}, w_{\nu})\}. \tag{1}$$

The *inference task* is to assign labels to the image pixels – e.g., to label a pixel as being an "edge" or "non-edge". This is performed by specifying an inference algorithm to compute the MAP estimator:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{I}). \tag{2}$$

In general, performing inference on these graphs is difficult. Inference is straightforward if there are only unary potentials, because then it reduces to estimating $\hat{w_{\mu}} = \arg\min_{w_{\mu}} \boldsymbol{\lambda}_{\mu}^{D} \cdot \boldsymbol{\phi}_{\mu}(w_{\mu}, \mathbf{I})$, which can be done independently for each $\mu \in \mathcal{V}$. But difficulties arise due to the binary potentials which model the dependencies between the states at different pixels. A range of algorithms have been proposed but convergence guarantees are rare. Max-flow/min-cut [49] algorithms are guaranteed to converge to the optimal solution for certain classes of models if the state variables are binary-valued. If we allow the state variables $\mathbf{w}$ to take continuous values then steepest descent, and related methods, will also converge to the optimal estimate provided the energy function $\sum_{\mu \in \mathcal{V}} \boldsymbol{\lambda}_{\mu}^{D} \cdot \boldsymbol{\phi}_{\mu}(w_{\mu}, \mathbf{I}) + \sum_{\mu,\nu \in \mathcal{E}} \boldsymbol{\lambda}_{\mu,\nu}^{P} \cdot \boldsymbol{\psi}_{\mu,\nu}(w_{\mu}, w_{\nu})$ is convex in the state variables $\mathbf{W}$. Markov chain monte carlo (MCMC) methods are guaranteed to converge to good estimate of $\hat{\mathbf{W}}$, but convergence rates tend to be slow [46]. Other algorithms that empirically give good results for these types of models include variational methods [50] and belief propagation [51].

But the effectiveness of these algorithms is often restricted to models where the interactions are nearest neighbor only. This restriction reduces the representational power of the models since it only captures the local context. It is poor, for example, at capturing the regularity that image contain large regions of sky or vegetation. In general, these algorithms often become difficult, or even impractical, to implement if the edge structure $\mathcal{E}$ is complicated, see figure (1)(far right) or contains non-local interactions. There are very few convergence guarantees for these cases.

The *learning task* is to learn the model parameters $\boldsymbol{\lambda}$ from a set of supervised training examples $\{(\mathbf{I}^{i}, \mathbf{W}^{i}) : i = 1, ..., N\}$. The learning is straightforward if we only consider only the unary potentials, because we can learn the data parameters $\boldsymbol{\lambda}^{D}$ by methods such as AdaBoost [52] or simply by learning conditional probability distributions [53]. Similarly, there are techniques for learning the binary potentials separately although these are more computationally intensive [54]. Discriminative methods [55] have been used to learn the full distribution, which requires an efficient inference algorithm, so that we can compare the performance of the algorithm with its current set of parameters to the groundtruth, and then modify the parameters if necessary. This can be formulated in terms of maximum likelihood learning or conditional random fields.

We now describe *object models* using a similar formulation. Deformable template models have an extensive history in computer vision [56],[57],[58]. They can be *represented* in terms of flat probabilistic models [59][60], [61],[62], [63] as we now describe, see figure (2). The formulation is again described on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with state variables $\mathbf{W}$ defined on the nodes $\mathcal{V}$, where the state w_{μ} of node μ represents the position (and possibly orientation) of a point, or part, of the object. The unary potentials $\boldsymbol{\lambda}_{\mu}^{D} \cdot \boldsymbol{\phi}_{\mu}(w_{\mu}, \mathbf{I})$ specify how points/parts of the object relate to the image – e.g., some points on the boundary of objects

may correspond to edges in the image intensity, while others may be modeled by interest points such as corners [59]. The edges $\mathcal{E}$ specify which points/parts of the object are directly related and the binary potentials $\boldsymbol{\lambda}^P_{\mu,\nu} \cdot \boldsymbol{\psi}_{\mu,\nu}(w_\mu, w_\nu)$ model the spatial relations – e.g., capturing the overall shape of the object.

This can be expressed by a similar distribution:

$$P(\mathbf{w}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp\{-\sum_{\mu \in \mathcal{V}} \boldsymbol{\lambda}^D_\mu \cdot \boldsymbol{\phi}_\mu(w_\mu, \mathbf{I}) - \sum_{\mu,\nu \in \mathcal{E}} \boldsymbol{\lambda}^P_{\mu,\nu} \cdot \boldsymbol{\psi}_{\mu,\nu}(w_\mu, w_\nu)\}, \tag{3}$$

where the main differences are what the state variables w_μ and the graph structure, see figure (2).

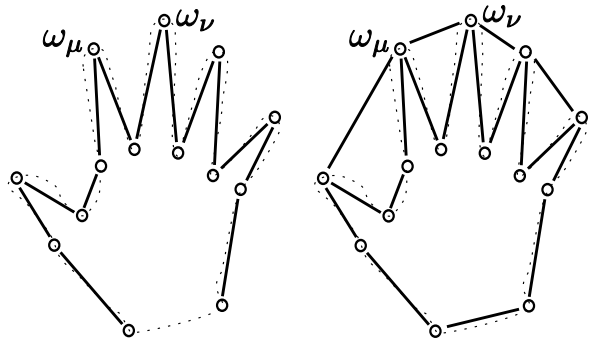

Fig. 2. A deformable template model of a hand without closed loops (left) and with closed loops (right).

Inference is different for these type of models. Firstly, the state variables can typically take a far larger set of values – i.e. the set of possible positions in an image is very large. Secondly, the types of graph structure are different. If the object has a chain-like structure – i.e., without closed loops – then dynamic programming can be to perform inference and detect the object independent [59] but pruning is necessary to ensure that the algorithm converges quickly. The computations required by dynamic programming can be sped up using various techniques [62]. By choosing more complex image features, such as shape context, [61] it is possible to perform good local matches by ignoring the binary potentials and then get better matches by use of the hungarian algorithm. If there are good estimates of the initial configuration, then variational methods can be effective for inference [60]. In addition, it is also possible to combine shape context and variational methods effectively [64]. But, once again, the inference algorithms become less effective if we start introducing longer range edges to capture the spatial regularities of objects, see figure (2)(right).

Learning is possible for these models provided groundtruth data has been specified. The unary terms are straightforward, since they can be learnt independently, but the binary terms are more difficult. In practice, many of the original models were hand specified. Coughlan et al. [59] learnt the unary potentials directly and adjusted the binary potentials interactively by stochastic sampling from the model and adjusting parameters until the samples looked like realistic hands. More advanced learning methods are practical provided there is an efficient inference algorithm.

In summary, flat models for image labeling and object detection can be effective if only simple graph models are used with local neighbor interactions. But it is difficult to extend these models to include longer-range interactions, to model the non-local context that often occurs in images, because of the difficulties of finding effective inference and learning algorithms. Moreover, it is unclear how these class of models can efficiently exploit the similarities between different objects and share parts between them.

Recent work in the computer vision literature has attempted to overcome these limitations by introducing additional layers of unobserved latent variables. This gives a class of hierarchical models [65][66][67],[68] which are typically limited to two layers, but with exceptions such as Ahuja and Todorovic [69]. Segmentation by weighted aggregation [70] is an alternative approach which allows for an arbitrary number of layers and which emphasizes algorithmic efficiency, but is not formulated probabilistically. Kokkinos

[71] developed a class of hierarchical models which are closely related to this paper. Also hierarchical models occur naturally in the literature of pattern theory, artificial neural networks, biologically motivated models, as described in section (II).

IV. Recursive Compositional Models (RCMs)

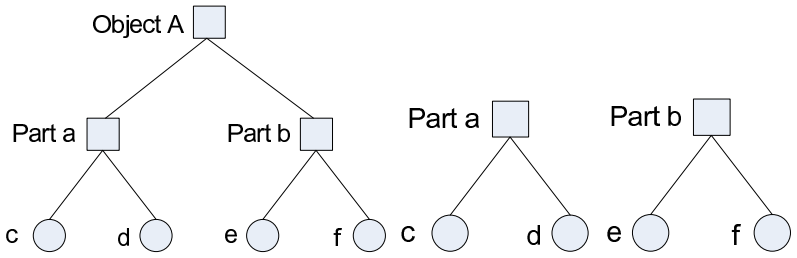

Fig. 3. Basic RCMs. An object A can be represented by an RCM (left) which is specified in terms of parts a, b which are more elementary RCMs (middle and right). This is a schematic and RCMs will typically involve more layers, more children, and sideways edges liking the child nodes.

This section introduces recursive compositional models (RCMs). These are hierarchical probability distributions which, intuitively, are built as compositions of elementary components as shown in figure (3). We will consider two different types of RCMs in this paper. In both types of RCMs each child node has only a single parent. The first types of RCMs only have edges linking the parent and child nodes and hence their graph structure is a tree (i.e. it contains no closed loops), see figure (3). The second type of RCMs also have edges between the child nodes of each parent, and hence do contain closed loops, see figure (4). But there are restrictions on the number of children of each parent node (e.g., less than six) and so the number of closed loops is not large. The restriction that each node has a single parent will be relaxed when we consider families of RCMs in section (V-E).

RCMs are designed to overcome the limitations of flat MRF models discussed in the previous section. Their *hierarchical nature* enables them to model object and image structures that occur at different scales. Essentially they can represent long-range dependencies by short-range vertical connections (i.e., between scales) and avoid the need for long-range horizontal connections. Moreover, their *recursive compositional structure* means that they can represent multiple objects efficiently, or single objects with multiple poses, by sharing elementary components recursively. This advantage will be shown for modeling baseball players from multiple poses and for multiple objects in later sections (V-B,V-E).

The compositional nature of RCMs enables efficient inference and learning algorithms. Inference is performed intuitively by searching for probable states of the sub-parts and using them to propose states for the parts. Similarly, learning also exploit the recursive structure by first learning sub-parts and then learning ways to combine them to form larger parts. Figure (4) shows an RCM for a horse, with closed loops, which illustrates the recursive compositional structure of RCMs.

A. RCM components

We now define RCMs more formally to illustrate their common aspects and then we will give five explicit examples in section (V). The fifth example is a families of RCMs and needs some additional concepts which will be introduced later.

An RCM is specified by a sextuplet $(\mathcal{V}, \mathcal{E}, \boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\lambda}^P, \boldsymbol{\lambda}^D)$, which specifies its graph structure $(\mathcal{V}, \mathcal{E})$, its feature functions $\boldsymbol{\psi}, \boldsymbol{\phi}$, and its parameters $\boldsymbol{\lambda}^P, \boldsymbol{\lambda}^D)$ (the dot products between the parameters and the feature functions give the potentials). We now describe these in turn.

Graph Structure: An RCM with root node $\mathcal{R}$ is specified by a graph $(\mathcal{V}_\mathcal{R}, \mathcal{E}_\mathcal{R})$ where $\mathcal{V}_\mathcal{R}$ denotes the graph nodes and $\mathcal{E}_\mathcal{R}$ the graph edges. These edges always include parent-child relations, see figure (3), and we use $ch(\mu)$ to denote the set of child nodes of μ. In this paper each node is restricted to having a single parent – i.e. $ch(\nu) \bigcap ch(\mu) = \emptyset$ for all $\nu, \mu \in \mathcal{V}_\mathcal{R}$ – except for the family of RCMs in section (V-E).

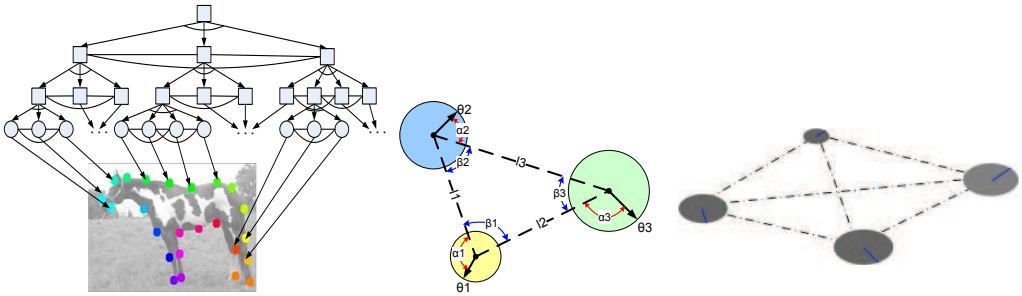

Fig. 4. The left panel illustrates the graph structure for an RCM representation of a horse. The state variables of the leaf nodes of the graph correspond to the poses (e.g., positions and orientations) of points on the boundary of the horse. The state variables of higher-level nodes represent the poses of parts, which are composed of subparts represented by lower-level nodes. Following the summarization principle, the state variables only represent coarse information about the part (e.g., its overall pose) and the lower-level nodes provide more detailed information about it. Note that this RCM includes horizontal edges between child nodes and hence the graph structure has closed loops. These horizontal edges are defined over triplets of child nodes (center), and impose constraints (which will be learnt) on the relative poses of the subparts represented by the child nodes. If a parent node has four child nodes, then we impose constraints on all four triplets (right). Triplets are used because we can impose the constraints over properties of the child nodes which are invariant to rotation and scaling. All parent nodes are restricted to having a maximum of five child nodes, so the number of closed loops in the graph is limited.

The leaf nodes $\mathcal{V}_{\mathcal{R}}^{leaf}$ are those nodes which have no children– i.e. $\mu \in \mathcal{V}_{\mathcal{R}}^{leaf}$ if $\mu \in \mathcal{V}_{\mathcal{R}}$ and $ch(\mu) = \emptyset$. The *level* l of an RCM is the number of generations in the graph minus one. Hence leaf nodes can be assigned to level 0, their parent nodes to level 1, and so on.

We illustrate RCMs by figures (3,4). Figure (3) shows a two-level RCM representing object A, which is composed from two 1-level RCMs represented by the parts a and b which, in turn, are represented into terms of 0-level (leaf nodes) RCMs c, d, e, f. In this example there are no edges linking the child nodes, so the graph contains no closed loops. Figure (4) shows a three-level RCM representing a horse, composed of three two-level RCMs representing the front, middle, and back of the horse. These RCMs have edges joining their child nodes and hence the graph has a limited number of closed loops.

State Variables: We define state variables w_μ at each node $\mu \in \mathcal{V}_R$. We set $\mathbf{w}_{ch(\mu)}$ to be the states of the child nodes of μ – i.e. $\mathbf{w}_{ch(\mu)} = (w_{\mu_1}, ..., w_{\mu_m})$, where $(\mu_1, ..., \mu_m) = ch(\mu)$. We use the terminology $\mathbf{W}_\mu$ to denote the states of the tree $\mathcal{V}_\mu$ with root node μ — e.g. $\mathbf{W}_A = (w_A, w_a, w_b, w_c, w_d, w_e, w_f)$ and $\mathbf{W}_a = (w_a, w_c, w_d)$ for the RCM for Object A in figure (3).

The state variables are required to be of the same form for all nodes of the model. For example, the state variables of an RCM for an object represent the poses of parts of the object. This means that a parent node represents summary information about the state of a part while the child nodes represent information about the state of its subparts. This implies that that upper levels of the hierarchy specify a coarse, or 'executive summary', representation of the object while the lower level nodes give more fine scale detail. We call this the *summarization principle*. It ensures that the complexity of the representation is the same at all levels of the hierarchy, which is important for efficient representation and inference.

Potential Functions and Model Parameters: The probability distribution for an RCM is specified by potentials $\mathbf{\Phi}(.)$ and parameters $\boldsymbol{\lambda}$ defined over the state variables and their cliques – recall that a clique is a subset of nodes such that all nodes are connected by edges (i.e. $C = \{\mu_1, ..., \mu_m\}$ is a clique if $(\mu_i, \mu_j) \in \mathcal{E}, \forall i \neq j \in \{1, ..., m\}$). If the child nodes are not linked by edges, then the cliques will consist of pairs of parents and children – i.e. $\{(\mu, \nu) : \nu \in ch(\mu)\}$. If the child nodes are linked by edges, the the cliques will consist of parents and all their children $\{\mu, ch(\mu)\}$.

There are two types of potential. The first type are *data potentials* of form $\boldsymbol{\lambda}_\mu^D \cdot \boldsymbol{\phi}_\mu(w_\mu, \mathbf{I})$ which relate the state w_μ of a node μ to the image $\mathbf{I}$. These dataterms will depend on the specific problem, see section (V), and can be defined at all levels of the hierarchy (for some RCMs the data potentials are defined only for leaf nodes). The second type are *prior potentials* which are of form $\boldsymbol{\lambda}_\mu^P \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ or $\boldsymbol{\lambda}_{\mu,\nu}^P \cdot \boldsymbol{\psi}_{\mu,\nu}(w_\mu, w_\nu)$, depending on whether the children are linked by edges or not. These prior potentials impose statistical constraints on the states of the nodes in the cliques – e.g., the probable relative poses of sub-parts of an object.

For example, in figure (3), there are prior potentials of form $\boldsymbol{\lambda}_{A,a}^P \cdot \boldsymbol{\psi}(w_A, w_a)$ and $\boldsymbol{\lambda}_{a,c}^P \cdot \boldsymbol{\psi}(w_a, w_c)$. We

will always have data potentials at the leaf nodes – e.g., $\boldsymbol{\lambda}_c^D \cdot \boldsymbol{\phi}(w_c, \mathbf{I})$, – and, depending on the application, we may also have dataterms at non-leaf nodes – e.g., $\boldsymbol{\lambda}_a^D \cdot \boldsymbol{\phi}(w_a, \mathbf{I})$.

In our applications, there are two types of prior feature functions $\boldsymbol{\psi}_{\mu,ch(\mu)}(.,.)$. The first type are represented by $\boldsymbol{\psi}^{AND}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ and correspond to *AND-potentials*. These are used when the parent node is a composition of the child nodes – see figure (4) where the root node represents the entire horse and is a composition of the front, middle, and back of the horse. The second type are *OR-potentials*, which use feature functions denotes by $\boldsymbol{\psi}^{OR}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$. These are used if the parent node selects between different child nodes, see figure (5)(right) where the root node is either object A or object B. In general, all AND-potentials and OR-potentials have the same form for all cliques (except that the size of the cliques, and the parameters, can vary). Similarly in our applications the data potentials $\boldsymbol{\phi}_\mu$ are typically the same at all levels of the RCM except at the leaf nodes.

Probability distributions: The probability distribution over the state variables $\mathbf{W}_\mathcal{R}$ is a Gibbs distribution whose energy is the sum of the data and prior potentials. More specifically, for a graph $\mathcal{V}_\mathcal{R}$ with root node $\mathcal{R}$ we define the conditional distribution $P(\mathbf{W}_\mathcal{R}|\mathbf{I})$:

$$P(\mathbf{W}_\mathcal{R}|\mathbf{I}) = \frac{1}{Z}\exp\{-E(\mathbf{W}_\mathcal{R}, \mathbf{I})\}, \text{ where } E(\mathbf{W}_\mathcal{R}, \mathbf{I}) = \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}_\mathcal{R}, \mathbf{I}), \tag{4}$$

with

$$\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}_\mathcal{R}, \mathbf{I}) = \sum_{\mu \in \mathcal{V}_\mathcal{R}/\mathcal{V}_\mathcal{R}^{leaf}} \boldsymbol{\lambda}^P_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)}) + \sum_{\mu \in \mathcal{V}_\nu} \boldsymbol{\lambda}^D_\nu \cdot \boldsymbol{\phi}_\nu(w_\nu, \mathbf{I}). \tag{5}$$

Recursive Formulation of the Energy: A fundamental property of RCMs (with or without closed loops) is that the energy can be computed recursively as follows:

$$E_\mu(\mathbf{W}_\mu, \mathbf{I}) = \boldsymbol{\lambda}^P_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)}) + \boldsymbol{\lambda}^D_\mu \cdot \boldsymbol{\phi}_\mu(w_\mu, \mathbf{I}) + \sum_{\rho \in ch(\mu)} E_\rho(\mathbf{W}_\rho, \mathbf{I}). \tag{6}$$

This equation can be derived from equations (4,5). It shows that the energy $E_\mu(\mathbf{W}_\mu, \mathbf{I})$ for a subtree with root node w_μ can be computed recursively in terms of the energies of its descendants $\{\mathbf{W}_\rho\}$ for $\rho \in ch(\mu)$. More precisely, the energy is computed from the energy of its components at the previous level $\{E_\rho(\mathbf{W}_\rho) : \rho \in ch(\mu)\}$, together with prior potentials $\boldsymbol{\lambda}^D_\mu \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, w_{ch(\mu)})$ which couples the parent node to its children, and an input directly from the image $\boldsymbol{\lambda}^D_\mu \cdot \boldsymbol{\phi}_\mu(w_\mu, \mathbf{I})$. The formulation is recursive and is initialized by $E_\rho(\mathbf{W}_\rho, \mathbf{I}) = \boldsymbol{\lambda}^D_\rho \cdot \boldsymbol{\phi}_\rho(w_\rho, \mathbf{I})$ for the leaf nodes $\rho \in \mathcal{V}^{leaf}$.

Equation (6) is the *fundamental equation* of RCMs. It combines the key elements of RCMs – recursion and composition – in a single equation. There is nothing analogous for the "flat models" described in section (III). In the next sections we will show how to exploit the fundamental equation to obtain efficient inference and learning algorithms.

B. RCM Inference Algorithm

The goal of inference is to estimate the most probable state $\hat{\mathbf{W}}_\mathcal{R} = \arg\max P(\mathbf{W}_\mathcal{R}|\mathbf{I})$ and, more generally, states $\mathbf{W}$ which have high probability.

The inference algorithm exploits the recursive energy formulation given by equation (6). Inference is performed by dynamic programming in a bottom-up and top-down pass. For object RCMs, such as the horse in figure (4), we called it the *compositional inference algorithm* [72], [73] because it proceeds intuitively by constructing the object by composing it from subparts. Pruning is sometimes required because the state space of the state variables can be very large for some applications (thresholds used for pruning are selected by evaluating performance on the datasets). Dynamic programming can deal with the closed loops introduced by horizontal edges by using junction trees techniques [39] (there are only a limited number of closed because parent nodes are restricted to have a small number of children, e.g., six or less). Dynamic programming is quadratic in the state space (with adjustments for the maximal size

of the cliques). Inference times are reported in section (V) and are typically of the order of seconds to minutes.

The *bottom-up pass* is defined as follows.

Initialization: at each leaf node $\nu \in \mathcal{V}^{leaf}$ we calculate the set of states $\{w_{\nu,b} : b = 1, ..., N_\nu\}$ such that $E_\nu(w_{\nu,b}) = \boldsymbol{\lambda}_\nu^D \cdot \boldsymbol{\phi}_\nu(w_{\nu,b}, \mathbf{I}) < T$ (*energy pruning* with threshold T). We refer to the $\{w_{\nu,b}\}$ as *proposals* for the state w_ν and store them with their *minimum energies*, which are defined to be $E_\nu^{\min}(w_{\nu,b}, \mathbf{I}) = E_\nu(w_{\nu,b}, \mathbf{I})$. Note that, for the leaf nodes, each $w_{\nu,b}$ gives a proposal for the states $\mathbf{W}_{\nu,b}$ of the sub-trees with root node ν and with energy $E_\nu(\mathbf{W}_{\nu,b}, \mathbf{I}) = E_\nu^{min}(w_{\nu,b}, \mathbf{I})$. This gives the initialization for performing inference exploiting equation (6).

There will minor variations in pruning depending on the specific application as discussed in section (V). For example, the energy pruning may accept a fixed percentage of proposals, or we may use *surround suppression* to prune out proposals which are too similar.

Recursion: the input is the set of proposals $\{w_{\mu_i,b_i}\}$ for the states of the children μ_i of node $\mu \in \mathcal{V}/\mathcal{V}^{leaf}$ together with minimum energies $E_{\mu_i}^{\min}(w_{\mu_i,b_i}, \mathbf{I})$. These minimum energies are obtained recursively, as described below, and are the lowest energies $E_{\mu_i}(\mathbf{W}_{\mu i}, \mathbf{I})$ over all possible states $\mathbf{W}_{\mu_i}$ of the subtree whose root node μ_i takes state w_{μ_i}. Note that we do not need to know the lowest energy state $\hat{\mathbf{W}}_{\mu_i}$ during the bottom-up pass because we can compute it later in a top-down pass (as is standard for dynamic programming), but we do need to know its energy. Here $\{b_i\}$ is the index set for the state variables $\{w_{\mu_i,b_i}\}$ for node μ_i.

Then for each state $w_{\mu,b}$ of node μ, we compute:

$$E_\mu^{\min}(w_{\mu,b}, \mathbf{I}) = \min_{\{b_i\}} \{\boldsymbol{\lambda}_{\mu,ch(\mu)}^P \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_{\mu,b}, \{w_{\mu_i,b_i}\}) + \boldsymbol{\lambda}_\mu^D \cdot \boldsymbol{\phi}_\mu^D(w_{\mu,b}, \mathbf{I}) + \sum_{j=1}^{|ch(\mu)|} E_{\mu_j}^{min}(w_{\mu_j,b_j}, \mathbf{I})\}. \quad (7)$$

This requires minimizing over all compositions of proposals of the child nodes which are consistent with state $w_{\mu,b}$ (i.e. over all possible $b_1, ..., b_i, ...$, see section (V) for examples of consistency).

Here $E_{\mu_i}^{min}(w_{\mu_i,b_i}, \mathbf{I})$ is the smallest energy of the subgraph $\mathcal{V}_{\mu_i}$ with root node μ_i taking state w_{μ_i,b_i}, and is computed recursively by equation (7) with boundary conditions provided by the initialization. Observe that equation (7) exploits the fundamental equation (6).

This gives a set of proposals $\{w_{\mu,b}\}$ for each node μ together with their minimal energies $E_\mu^{min}(w_{\mu,b}, \mathbf{I})$. Pruning is used to remove some of the proposals – e.g., energy pruning removes proposals $w_{\mu,b}$ for which $E_\mu^{min}(w_{\mu,b}, \mathbf{I})$ is below a threshold. The output is the set of proposals $\{w_{\mu,b}, E_\mu^{min}(w_{\mu,b}, \mathbf{I})\}$ which survive the pruning.

The procedure stops when we reach the root node $\mathcal{V}$ of the graph. The output is a set of proposals $\{w_{\mathcal{R},b}\}$ for the state of the root node $\mathcal{R}$ and their minimum energies $\{E_\mathcal{R}^{min}(w_{\mathcal{R},b}, \mathbf{I})\}$.

The *top-down pass* is used to find the state configurations $\{\mathbf{W}_{\mathcal{R},b}\}$ of the graph nodes $\mathcal{V}_\mathcal{R}$ which correspond to the proposals $\{w_{\mathcal{R},b}\}$ for the root nodes. By construction, the energies of these configurations is equal to the minimum energies at the root nodes, see equation (7). This top-down pass recursively inverts equation (7) to obtain the states of the child nodes that yield the minimum energy – i.e., it solves:

$$\{b_i^*\} = \boldsymbol{\lambda}_{\mu,ch(\mu)}^P \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_{\mu,b}, \{w_{\mu_i,b_i}\}) + \boldsymbol{\lambda}_\mu^D \cdot \boldsymbol{\phi}_\mu^D(w_{\mu,b}, \mathbf{I}) + \arg\min_{\{b_j\}} \{\sum_{j=1}^{|ch(\mu)|} E_{\mu_j}^{\min}(w_{\mu_j,b_j}, \mathbf{I})\}. \quad (8)$$

We then set the optimal solution to be:

$$\hat{\mathbf{W}}_\mathcal{R} = \mathbf{W}_{\mathcal{R},b^*}, \quad \text{where } b^* = \arg\min_b E_\mathcal{R}^{\min}(w_{\mathcal{R},b}, \mathbf{I}). \quad (9)$$

Beyond dynamic programming. The algorithms described above can be easily extended to RCMs with a little number of closed loops (as occurs when there are a limited number of edges between the child nodes). But the compositional inference algorithm can also be applied to graphs with many closed loops, as described in [73]. The strategy is to perform the bottom-up pass on a simplified graph model which

has many of its edges removed. The edges, and their constraints are then imposed during the top-down process. This has interesting relationships to belief propagation [1]. Our empirical studies [74] [75] show that this procedure works well.

C. Learning the Parameters by Structure Learning

This section describes how we learn the parameters $\boldsymbol{\lambda}$ of the RCM from supervised training data provided the graph structure $\mathcal{V}, \mathcal{E}$ is known. We learn the model parameters $\boldsymbol{\lambda}^D, \boldsymbol{\lambda}^P$. To simply notation, in this section we refer to the models parameters as $\boldsymbol{\lambda}$ and the potentials as $\boldsymbol{\Phi}$ (i.e. we make no distinction between the data and the prior potentials and parameters). Similarly we drop the subscript $_{mR}$ from the state variables $\mathbf{W}$ because we will always be dealing with all the state variables of the graph.

The input is a set of training images $\{\mathbf{I}^i\}$ with groundtruth $\{\mathbf{W}^i\}$, or partial groundtruth $\{\mathbf{y}^i\}$, specified:

$$\{(\mathbf{W}^i, \mathbf{I}^i) : i = 1, ..., N\} \text{ groundtruth } \{(\mathbf{y}^i, \mathbf{I}^i) : i = 1, ..., N\} \text{ partial groundtruth} \tag{10}$$

For some datasets the groundtruth is only specified for the state variables at the leaf nodes of graph but the nature of the problem enables us to determine the state variables for the rest of the graph, see sections (V-A,V-C). For other examples, see section (V-D), the partial groundtruth $\{\mathbf{y}^i\}$ only specifies if an object is present $\mathbf{y}^i = 1$, or not present $\mathbf{y}^i = -1$, in an image $\mathbf{I}^i$. In this case, the state $\mathbf{W}^i$ of the RCM are hidden/latent variables and so must be estimated during learning.

We learn the parameters of the distributions using machine learning methods which can be derived as approximations/bounds to the more standard parameter estimation techniques such as maximum likelihood. These machine learning techniques are computationally simpler than more traditional methods and arguably more effective for discriminative tasks.

We use three different machine learning methods. The first, the *structure-perceptron algorithm* [41], is an approximation/bound to maximum likelihood learning. The second, *structure max-margin* [42][43][44][45] is an approximation/bound to parameter estimation with a prior and a loss function. The third, *latent structural SVM* [68],[76], is an approximation/bound to parameter estimation with a prior, loss function, and hidden/latent variables. Hence the difference between these methods, and the selection of which to use, is determined by whether there is a prior, a loss function, and/or hidden/latent variables. Structure perceptron is the simplest algorithm to use, followed by structure max-margin, and finally latent structural SVM. All these algorithms need an efficient inference algorithm as a pre-requisite.

We have also worked on novel unsupervised learning algorithms which are capable of learning the graph structure. There is not sufficient space to describe them in this paper, but we will briefly sketch them in section (VI).

Structure Perceptron Learning.

This algorithm can be obtained as an approximation to maximum likelihood (ML) estimation of the parameters $\boldsymbol{\lambda}$ of an exponential model $P(\mathbf{W}|\mathbf{I}, \boldsymbol{\lambda}) = \frac{1}{Z[\mathbf{I}, \boldsymbol{\lambda}]} \exp\{-\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}, \mathbf{I})\}$ by replacing the ML criterion $F_{ML}(\boldsymbol{\lambda}) = -\sum_{i=1}^{N} \log P(\mathbf{W}_i|\mathbf{I}_i, \boldsymbol{\lambda})$ by the bound:

$$F_{ML,approx}(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \boldsymbol{\lambda} \cdot \{\boldsymbol{\Phi}(\mathbf{W}_i, \mathbf{I}_i) - \boldsymbol{\Phi}(\hat{\mathbf{W}}(\mathbf{I}_i, \boldsymbol{\lambda}), \mathbf{I}_i)\}, \tag{11}$$

$$\text{where} \quad \hat{\mathbf{W}}(\mathbf{I}_i, \boldsymbol{\lambda}) = \arg\min_{\mathbf{W}} \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}, \mathbf{I}_i). \tag{12}$$

The approximation is obtained by approximating $Z[\mathbf{I}, \boldsymbol{\lambda}]$ by its dominant term $\exp\{\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\hat{\mathbf{W}}(\mathbf{I}, \boldsymbol{\phi}), \mathbf{I})\}$. Recall that $\hat{\mathbf{W}}(\mathbf{I}_i, \boldsymbol{\lambda}) = \arg\min_{\mathbf{W}} \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}, \mathbf{I}_i)$ is also the MAP estimate of $\mathbf{W}$ from $P(\mathbf{W}|\mathbf{I}, \boldsymbol{\lambda})$, and can be obtained by the inference algorithm described above.

The structure perceptron algorithm performs online learning for $F_{ML,approx}(\boldsymbol{\lambda})$ by selecting an element $\mathbf{W}_i, \mathbf{I}_i$ from the training data, see equation (10), and performing an iteration of gradient descent on $\boldsymbol{\lambda} \cdot \{\boldsymbol{\Phi}(\mathbf{W}_i, \mathbf{I}_i) - \boldsymbol{\Phi}(\hat{\mathbf{W}}(\mathbf{I}_i, \boldsymbol{\lambda}), \mathbf{I}_i)\}$:

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t - \boldsymbol{\Phi}(\mathbf{W}_i, \mathbf{I}_i) + \boldsymbol{\Phi}(\mathbf{W}^*(\mathbf{I}_i, \boldsymbol{\lambda}^t), \mathbf{I}_i), \tag{13}$$

where an additional approximation is made by ignoring the dependence of $\hat{W}(\mathbf{I}, \boldsymbol{\lambda})$ on $\boldsymbol{\lambda}$ when the derivative is taken.

Structure perceptron has the desirable property, empirically verified for these applications, that many of the weights $\boldsymbol{\lambda}$ remain close to zero (the weights are initialized at zero). This enables a selection process where we specify many feature functions – i.e. make $\boldsymbol{\Phi}(\mathbf{W}, \mathbf{I})$ a high dimensional vector – and reject feature functions if the corresponding term in $\boldsymbol{\lambda}$ is small (using a threshold). Better selection procedures can be obtained by adding a spareness prior – e.g., $P(\boldsymbol{\lambda}) \approx \exp\{-|\boldsymbol{\lambda}|\}$ – and modifying $F_{ML,approx}(\boldsymbol{\lambda})$ by adding $|\boldsymbol{\lambda}|$.

Structure Max-Margin:

We modify the ML criterion $F_{ML}(\boldsymbol{\lambda})$ by introducing a loss function $l(\mathbf{W}, \mathbf{W}_i)$, which gives a penalty for making decision $\mathbf{W}$ for input $\mathbf{I}_i$ when the groundtruth is $\mathbf{W}_i$, and a prior $P(\mathbf{W})$. This requires minimizing a criterion $F_{loss}(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \sum_{\mathbf{W}} \exp\{L(\mathbf{W}, \mathbf{W}_i)\} P(\mathbf{W}|\mathbf{I}_i, \boldsymbol{\lambda}) - \log P(\boldsymbol{\lambda})$ which we approximate by:

$$F_{loss,approx}(\boldsymbol{\lambda}) = \frac{1}{2}|\boldsymbol{\lambda}|^2 + C \sum_i \max_{\mathbf{W}} \{\boldsymbol{\lambda} \cdot \{\boldsymbol{\Phi}(\mathbf{W}_i, \mathbf{I}_i) - \boldsymbol{\Phi}(\mathbf{W}, \mathbf{I}_i)\} + L(\mathbf{W}, \mathbf{W}_i)\}, \tag{14}$$

where $P(\boldsymbol{\lambda})$ is a Gaussian prior, the summation of over $\mathbf{W}$ is approximated by the dominant term $\max_{\mathbf{W}} \exp\{L(\mathbf{W}, \mathbf{W}_i) - \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}, \mathbf{I}_i)\}$, and the $Z[\mathbf{I}_i, \boldsymbol{\lambda}]$ term by $\exp\{-\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{W}_i, \mathbf{I}_i)\}$. This gives a convex quadratic criterion $F_{loss,approx}(\boldsymbol{\lambda})$. From the perspective of machine learning, the prior term $|\boldsymbol{\lambda}|^2$ is interpreted as a margin term and the goal is to classify the data while maximizing the margin. Like other max-margin criteria it can be expressed as a primal-dual problem and its solution can be expressed in terms of the support vectors:

$$\boldsymbol{\lambda}^* = C \sum_{i,\mathbf{W}} \alpha^*_{i,\mathbf{W}} \{-\boldsymbol{\Phi}(\mathbf{W}_i, \mathbf{I}_i) + \Phi(\mathbf{W}, \mathbf{I}_i)\}, \tag{15}$$

where the $\{\alpha_{i,\mathbf{W}}\}$ are solutions to the dual problem, and only take non-zero values for the small number of datapoints that lie on the margin. They can be efficiently solved for using the working set algorithm [44], [45], which sequentially creates a nested working set of tighter relaxations using cutting plane methods.

Latent Structural Support Vector Machines

For some applications only part of the groundtruth is specified. Instead the training data only gives a set of images $\{\mathbf{I}^i : i = 1, ..., N\}$ and a binary variable $\{y^i : i = 1, .., N\}$, where $y^i = 1$ if the object is present in $\mathbf{I}^i$ and $y^i = 1$ otherwise. This requires extending the probability model to a distribution over y and the states $\mathbf{W}$ of an RCM:

$$P(y, \mathbf{W}|\mathbf{I}, \boldsymbol{\lambda}) = \frac{1}{Z[\mathbf{I}, \boldsymbol{\lambda}]} \exp\{-\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y, \mathbf{W}, \mathbf{I})\}. \tag{16}$$

The state variables $\{\mathbf{W}^i\}$ of the RCM are not specified in the training dataset and hence are hidden/latent variables. In some applications, see section (V-D), the object is modeled by a mixture of RCMs so the mixture variables are also hidden/latent variables. In this case we use $\mathbf{W}$ in the equations below, to represent $\{V, \mathbf{W}_1, \mathbf{W}_2\}$ where $\mathbf{W}_1, \mathbf{W}_2$ are the states of two RCMs and V is an index variable specifying which RCM is selected.

We extend the criteria $F_{loss}(\boldsymbol{\lambda})$ to include hidden variables giving $F_{latent}(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \sum_{y,\mathbf{W}} \exp\{L(y, y_i)\}$ $P(y, \mathbf{W}|\mathbf{I}_i, \boldsymbol{\lambda}) - \log P(\boldsymbol{\lambda})$ which can be approximated/bounded by:

$$F_{latent,approx}(\boldsymbol{\lambda}) = \frac{1}{2}|\boldsymbol{\lambda}|^2 + C \sum_i \{\max_{y,\mathbf{W}} \{L(y, y^i) - \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y, \mathbf{W}, \mathbf{I}^i)\} + \max_{\mathbf{W}} \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y^i, \mathbf{W}, \mathbf{I}^i), \tag{17}$$

where we approximate the summation over $y, \mathbf{W}$ by the dominant term, and the normalizing term $Z[\mathbf{I}, \boldsymbol{\lambda}] = \sum_{y,\mathbf{W}} \exp\{-\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y, \mathbf{W}, \mathbf{I})\}$ by $\max_{\mathbf{W}} \exp\{-\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y_i, \mathbf{W}, \mathbf{I})\}$.

The criterion $F_{latent,approx}(\boldsymbol{\lambda})$ is a non-convex function of $\boldsymbol{\lambda}$ and there is no algorithm that can be guaranteed to converge to the global minimum. Instead Yu and Joachims [76] proposed an algorithm based on the CCCP procedure [77] by decomposing $F_{latent,approx}(\boldsymbol{\lambda})$ into a convex and concave part.

This reduces to an iterative algorithm which performs two steps in alternation:

$$\boldsymbol{\lambda}^{t+1} = \arg\min \frac{1}{2}|\boldsymbol{\lambda}|^2 + C\sum_{i=1}^{N} \min_{y,\mathbf{W}} \{\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y, \mathbf{W}, \mathbf{I}) + L(y, y_i)\} - C\sum_{i=1}^{N} \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\mathbf{I}_i, y_i, \mathbf{W}_i^t), \quad (18)$$

$$\mathbf{W}_i^{t+1} = \arg\min_{\mathbf{W}} \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y_i, \mathbf{W}, \mathbf{I}_i). \quad (19)$$

The first step requires solving a standard structure learning problem – i.e. minimizing a criteria like $F_{loss,approx}(\boldsymbol{\lambda})$ – and can be solved by the methods described above in the Structure Max-Margin section. The second step can be performed by the compositional inference algorithm. Observe that this algorithm is analogous to the standard EM algorithm. In [16] we describe a variant of this approach, called iCCCP, which converges significantly faster.

D. Hierarchical Dictionaries of RCMs

Now suppose we want to model a large number of objects seen from different viewpoints and with different poses. How can we use RCMs to represent these objects/viewpoints compactly so as to enable efficient learning and inference? One solution is to represent objects by *hierarchical dictionaries of RCMs* [16], which consists of a set of dictionaries of RCMs at each level together with rules for how dictionary elements at level-l are composed – by AND-ing or OR-ing – from dictionary elements at level-$(l-1)$.

We illustrate the basic idea in figure (5), where objects are built recursively from more elementary RCMs which can be shared between different objects/viewpoints. This is formalized by having hierarchical dictionaries of RCMs $\mathcal{T}$'s, where $\mathcal{T}^0$ are elementary RCMs of level-0 (single nodes in this example), $\mathcal{T}^1$ denotes RCMs of level-1 which are composed of elements of $\mathcal{T}^0$, and $\mathcal{T}^2$ are composed of elements of $\mathcal{T}^1$. In this example $\mathcal{T}^2$ represents the two objects A and B. We can perform inference on these models efficiently by performing *inference on the dictionaries* – i.e. detecting instances of $\mathcal{T}^0$, using them to detect instances of $\mathcal{T}^1$ and so on. This is a simple generalization of the compositional inference algorithm and is also dynamic programming in this example (with some pruning). Moreover, we can also learn the dictionaries – and hence the graph structure of the models – automatically as described in [16]. For reasons of space we will not describe this learning algorithm in detail, but we will sketch it in section (VI).

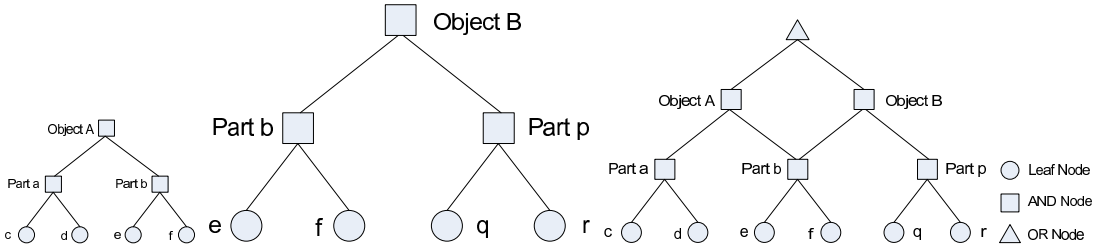

Fig. 5. Hierarchical Dictionaries of RCMs. We can represent objects compactly by constructing them hierarchically from elementary components, which allows part sharing and is very efficient for inference and learning. In this example, there are three dictionaries $\mathcal{T}^2 = \{A, B\}$, $\mathcal{T}^1 = \{a, b, p\}$, $\mathcal{T}^0 = \{c, d, e, f, q, r\}$ at levels 2,1 and 0 respectively. The higher level dictionaries are constructed from lower level dictionaries, in this example by composition (i.e. AND-ing)– e.g., in the left panel the level-1 RCM a is a composition of two level-0 RCMs c and d, and the level-2 RCM A is a composition of two level-one RCMs a and b. The full RCM (right) is an OR combination of two level-two RCMs for A and B (left and center). This representation shows the advantages of part-sharing – we only need to search for part b once, since it shared between A and B. This makes representation, inference, and learning much more efficient for multiple objects, viewpoints, and poses.

More formally, a hierarchical dictionary of RCMs is specified by $\mathcal{T} = \cup_{l=1}^{L} \mathcal{T}^l$, where each $\mathcal{T}^l$ is a dictionary of RCMs with l levels. A dictionary is of form $\mathcal{T}^l = \{t_a^l : a = 1, ..., n^l\}$ where there are

n^l RCMs t_a^l at level-l. In addition, an RCM t_a^l at level l has a set of child RCMs $\{t_{i(a)}^{l-1}\}$ at levels $(l-1)$. Child RCMs can have one or more 'parent RCMs'. Hence the full graph structure is specified by the dictionaries and the parent-child relationships. These relationships will be either AND-ing (i.e. compositional) or OR-ing, as we will discuss.

Suppose that a level-l RCM t_a^l has m child level-$(l-1)$ RCMs $\{t_{i(a)}^{l-1} : \ a = 1, ..., m\}$. Express all these level-$(l-1)$ RCMs by their sextuplets $(\mathcal{V}_{i(a)}^{l-1}, \mathcal{E}_{i(a)}^{l-1}, \boldsymbol{\psi}_{i(a)}^{l-1}, \boldsymbol{\phi}_{i(a)}^{l-1}, \boldsymbol{\lambda}_{i(a)}^{P,l}, \boldsymbol{\lambda}_{i(a)}^{D,l})$. Then we construct the sextuplet $(\mathcal{V}_a^l, \mathcal{E}_a^l, \boldsymbol{\psi}_a^l, \boldsymbol{\phi}_a^l, \boldsymbol{\lambda}_a^{P,l}, \boldsymbol{\lambda}_a^{D,l})$ for t_a^l as follows:

$$\begin{aligned}
\mathcal{V}_a^l &= (\mathcal{R}_a^l) \cup_{i(a)=1}^m \mathcal{V}_{i(a)}^{l-1} \\
\mathcal{E}_a^l &= (\mathcal{R}_a^l, \{\mathcal{R}_{i(a)}^{l-1}\}) \cup_{i(a)=1}^m \mathcal{E}_{i(a)}^{l-1}, \\
\boldsymbol{\psi}_a^l &= (\boldsymbol{\psi}_{\mathcal{R}_a^l, \{\mathcal{R}_{i(a)}^{l-1}\}}, \{\boldsymbol{\psi}_{i(a)}^{l-1} : \ i(a) = 1, ..., m\}) \\
\boldsymbol{\lambda}_a^{P,l} &= (\boldsymbol{\lambda}_{\mathcal{R}_a^l, \{\mathcal{R}_{i(a)}^{l-1}\}}^P, \{\boldsymbol{\lambda}_{\mathcal{R}_{i(a)}^{l-1}}^P : \ i(a) = 1, ..., m\}), \\
\boldsymbol{\phi}_a^{P,l} &= (\boldsymbol{\phi}_{\mathcal{R}_a^l}, \{\boldsymbol{\phi}_{i(a)}^{l-1} : \ i(a) = 1, ..., m\}) \\
\boldsymbol{\lambda}_a^{D,l} &= (\boldsymbol{\lambda}_a^{D,l}, \{\boldsymbol{\lambda}_{i(a)}^{D,l-1} : \ i(a) = 1, ..., m\}).
\end{aligned} \tag{20}$$

The choice of the potential $\boldsymbol{\lambda}_{\mathcal{R}_a^l, \{\mathcal{R}_{i(a)}^{l-1}\}}^P \cdot \boldsymbol{\psi}_{\mathcal{R}_a^l, \{\mathcal{R}_{i(a)}^{l-1}\}}$ – i.e. whether it is an AND-potential or an OR-potential – determines whether this is an AND composition, see figure(5)(left and center), or an OR combination as shown by combining A and B at the root node in figure(5)(right).

The *compositional inference algorithm* can be applied directly to the dictionaries. For example in figure (3), when we seek to detect objects A and B, we search for probable instances for the elements $\{c, d, e, f, q, r\}$ of the first level dictionary $\mathcal{T}^1$, then compose these to find probable instances for the second level dictionary $\mathcal{T}^2$, and then proceed to find instances of the third level $\mathcal{T}^3$ – which detects the objects. By comparison, performing inference on objects A and B separately, would waste time detecting part b for each object separately.

More precisely, for each RCM t_a^1 in dictionary $\mathcal{T}^1$ we find all instances $\{W(t_a^1)\}$ whose energy is below a threshold (chosen to provide an acceptably small false negative rate) and whose states are sufficiently different by selecting the state with locally minimal energy, see [14] for more details of the inference algorithm. Next we form compositions of these to obtain instances of the RCMs from the next level directory $\mathcal{T}^2$ and proceed in this way until we reach the object-RCMs at the top level. At each stage of the algorithm we exploit the recursive equation (7) to compute the energies for the "parent" RCMs at level l in terms of their "child" RCMs at level $(l-1)$. It should be emphasized that this dictionary inference algorithm directly outputs the objects/viewpoints which have been detected as well as their parses – there is no need for a further classification stage.

Observe that there may be closed loops in the graph structures specified by the dictionaries, since child RCMs can have multiple parent RCMs. This means that the compositional inference algorithm will not, in general, be guaranteed to converge to the optimal solution. In this case, the update rules for dynamic programming can be thought of as belief propagation using message parsing [1] which is known to be a good approximation for inference on graphs with closed loops. Some of our empirical studies [74] [75] show that these approximate methods can give good results for these types of models.

Learning of these models can be performed by the learning algorithms described above if the graph structure is known. Structure induction – i.e. learning the hierarchical dictionaries – is a more challenging task and we will briefly sketch how we perform it in section (VI).

V. Five Examples

We now describe five examples of RCMs which show how they can be applied to a representative set of computer vision tasks. Three examples apply RCMs to model objects, the fourth addresses multiple objects, and the fifth addresses image labeling. We will give results on standard benchmarked datasets –

Weizmann horse dataset [18], the MSRC image label dataset [24], a Baseball player dataset [21], a cow dataset [22], a face dataset [23], PASCAL [19], LabelMe [20]– showing that our results are competitive to the state of the art. All these examples follow the basic strategy described in the previous section but the details – e.g. graph structures, state variables, pruning techniques, and learning methods – will depend on the application.

A. A Hierarchical Deformable Model

The first example describes a hierarchical model for a deformable object where the goal is to segment the object and parse it – i.e. to detect and label subparts of the object [12],[15]. We test on the Weizmann horse dataset [18], see figure (4) and on a dataset of cows constructed by Magee and Boyle [22] and studied by [78]. In these datasets, the objects are deformable but there is comparatively little variation in viewpoint or pose. Note that these datasets are defined for segmentation only and other researchers do not address parsing.

The Graph structure: This RCM uses a fixed graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ illustrated in figure (4). This graph structure was learnt by a simple hierarchical clustering algorithm, adapted from SWA [70], which used a single object boundary as input, as described in [15]. The edges $\mathcal{E}$ connect child node to their parents and child nodes to each other, see figure (4)(right). The number of children is restricted (to between 3 and 6 for each parent) so the number of closed loops is not very large. All nodes are connected to the image – i.e., there are data potentials $\boldsymbol{\phi}_\mu(w_\mu, \mathbf{I})$ defined for all nodes $\mu \in \mathcal{V}$.

The State variables: The state variables w_μ represent the poses of the parts and subparts of the object. More precisely, they are of form $w_\mu = (x_\mu, y_\mu, \theta_\mu, s_\mu)$, where (x_μ, y_μ) is the position in the image, θ_μ is the orientation, and s_μ is the size.

These state variables obey the summarization principle because they are the same at all levels of the hierarchy. The top level of the RCM specifies a crude description of the object – its position, orientation, and scale – while the lower levels specify the poses of subparts of the object. The state variables for all graph nodes gives a *parse* of the object, indicating the poses of all parts and subparts.

The Potentials:

The feature functions $\boldsymbol{\phi}_\mu(w_\mu, \mathbf{I})$ of the data potentials are defined in terms of a dictionary of image features – e.g., Gabor filters, derivatives of Gaussians, histograms of filter responses, see [12],[15] for details. For this RCM, data potentials are defined for all nodes $\mu \in \mathcal{V}$ of the graph. The data potentials for the leaf nodes $\mu \in \mathcal{V}^{leaf}$ are defined in terms of image features, such as edges, which give cues for the boundaries of the object (all leaf nodes have the same potentials). The data potentials for all other nodes $\mu \in \mathcal{V}/\mathcal{V}^{leaf}$ use features which correspond to the appearance of subregions of the object – e.g., to the appearance of the hair on a horse's body (All non-leaf nodes have the same potentials).

The prior potentials for this model are used to specify spatial relations between parts (parent nodes) and subparts (child nodes). The feature functions $\psi_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ have the same form for all nodes $\mu \in \mathcal{V}/\mathcal{V}^{leaf}$. This model uses AND-potentials only which we specify as follows. We divide them into *vertical* feature functions $\boldsymbol{\psi}^V_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$, which relate the state of the parent node μ to a summary statistic of state of its children, and *horizontal* feature functions $\boldsymbol{\psi}^H_{\mu,ch(\mu)}(\mathbf{w}_{ch(\mu)})$ which relate the relative states of the child nodes. The vertical features functions $\boldsymbol{\psi}^V_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ are simplified by requiring that the state of the parent node is a deterministic function of the states of the child nodes. We set $\boldsymbol{\psi}^V_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)}) = -\delta(w_\mu - f(\mathbf{w}_{ch(\mu)}))$, where $\delta(.)$ is a delta function, and $f(.)$ specifies the position and orientation of the parent to be the mean positions and orientations of its child nodes (and the size is obtained by the region that is covered). The vertical feature function is a hard constraint and it has no parameters to be learnt. The horizontal feature functions $\boldsymbol{\psi}^H_{ch(\mu)}(\mathbf{w}_{ch(\mu)})$ are defined over the triplets of the child nodes. They are defined to be the second order moments of the *invariant transformation vector* (ITV) $\vec{l}$ which is defined to be independent of the position, scale, and orientation of the triangle, see figure (4) and [12],[15]. These specify statistical constraints on the relative poses of the child nodes

which are independent of the orientation and scale of the object. This gives prior potentials of form:

$$\boldsymbol{\lambda}^{P}_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)}) = -\delta(w_\mu - f(\mathbf{w}_{ch(\mu)})) + \boldsymbol{\lambda}^{P}_{ch(\mu)} \cdot \boldsymbol{\psi}^{H}_{ch(\mu)}(\mathbf{w}_{ch(\mu)}). \quad (21)$$

The Inference algorithm and pruning:

We use the compositional inference algorithm, described earlier by equations (7,8). For this application, the state space of the variables $w_\mu = (x_\mu, y_\mu, \theta_\mu, s_\mu)$ is very large. Hence, like other applications of dynamic programming to computer vision [79], we restrict the size of the state space using three pruning mechanisms.

Firstly, we use *energy pruning* to remove all configurations whose minimum energy falls below threshold – $E^{\min}_{\mu}(w_{\mu,b}, \mathbf{I}) < T$.

Secondly, we use *surround suppression* to inhibit proposals which are too similar to each other and hence are redundant. More formally, surround suppression is formulated by eliminating any proposal $w_{\mu,b}$ unless $E_\mu(w_{\nu,b}, \mathbf{I}) \leq E_\mu(w_\nu, \mathbf{I})$ for all $w_\mu \in Wd(w_{\mu,b})$, where $Wd(w_{\mu,b})$ is a window centered on $w_{\mu,b}$). The size of the window specifies the degree of accuracy which we require. At leaf nodes, surround suppression is similar to non-maximal suppression as used in the Canny edge detector. At higher levels, it enables us to use coarser spatial representations for parts and is similar to techniques used in biologically inspired models – e.g., [38], [35] – which match the experimental neuroscience findings that receptive fields in the visual cortex become more broadly tuned spatially as they become more specific to stimulus type (e.g, neurons which respond to faces are relatively insensitive to the pose of the face).

Thirdly, we make the pruning robust by the *two out of three rule*. Suppose a node has only three children and consider a situation where two child nodes have proposals which are above threshold but there is no corresponding proposal from the third child node. In this case, the algorithm creates a proposal for the third node by finding the state which best fits the horizontal prior, pays a penalty, and proceeds as above. This procedure can be generalized in the obvious way to cases where the parent node has more than two child nodes, see [15]. The two out of three rule enables the algorithm to deal with situations when there is partial occlusion, failure of the filters to detect a subpart of the object, or an error made by pruning.

The parameters of these pruning mechanisms were determined experimentally using the training dataset with the goal of maximizing the speed of the algorithm while keeping the performance rates high, see [15].

The algorithm has an intuitive interpretation. During the bottom-up pass – equation (7) – the child nodes find state configurations which have low energy values and form compositions of them to make proposals for the states of their parent nodes. The parents check the consistency between child states, imposed by the prior potentials, and incorporate additional image cues using the data potentials. Inconsistent proposals are rejected, typically because they violate the spatial relationships. As we proceed up the levels of the graph, the algorithm enforces increasingly more global constraints on the spatial configuration of the object.

The Learning algorithm:

The structure perceptron algorithm is used to learn the parameters $\boldsymbol{\lambda}$. Initialization is performed by the same hierarchical clustering that was used to learn the graph structure, see [15].

The form of the prior potentials – the deterministic function $w_\mu = f(\mathbf{w}_{ch(\mu)})$– means that we know the states of all the graph nodes provided we know the states of the leaf nodes. Hence we can infer the groundtruth for all graph nodes even if the groundtruth is only specified for the leaf nodes (as it is for all benchmarked datasets).

Postprocessing:

The output of the RCM is the states of the graph nodes. The leaf nodes give a sparse representation of the boundary in terms of a finite set of points. To obtain a complete boundary we link the leaf node points using a standard linking algorithm [15].

Datasets and Results:

This RCM was evaluated for object segmentation and parsing on the Weizmann horse dataset and the cow dataset. Typical results are shown in figure (6). Quantitative comparisons on the Weizmann dataset

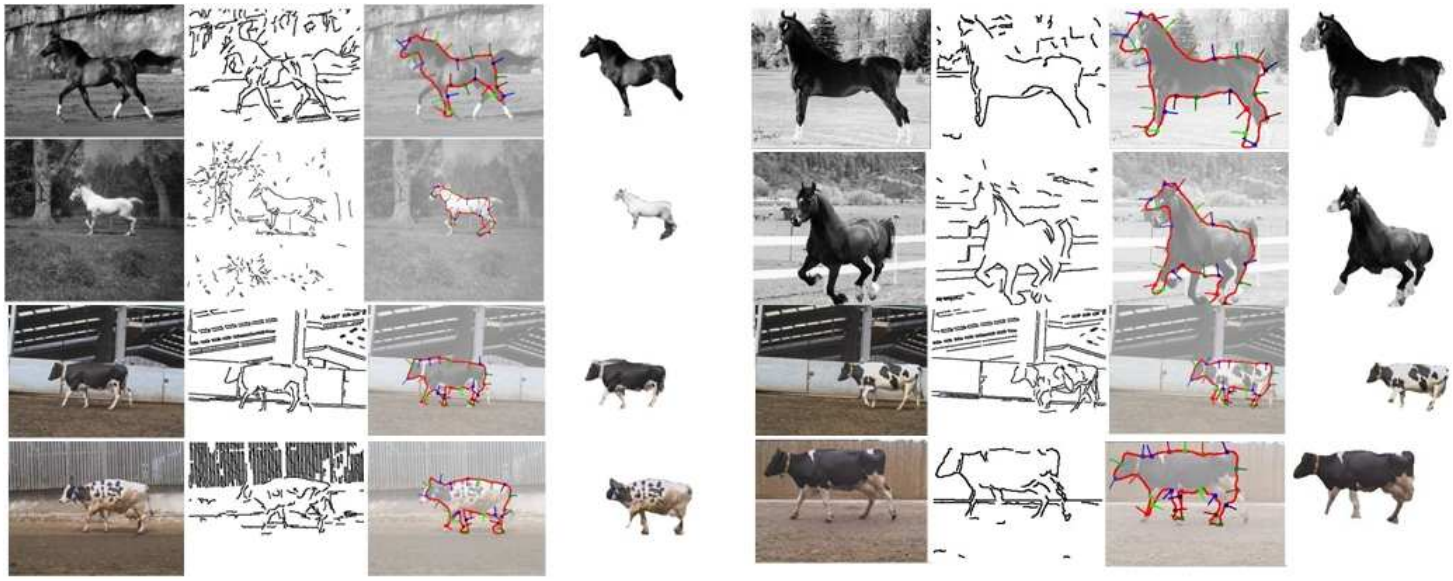

Fig. 6. Segmentation and parsing results on horses and cows. The four columns show the raw images, the edge maps, the parsed results, and the segmentation results (respectively). Example A.

Methods	RCMs	[80]	[81]	[66]	[82]	[83]	[67].
Testing	228	172	328	328	N/A	5	N?A
Seg. Acc.	94.7%	91.0% 93.0%	93.1%	94.2%	95.0%	96.0%	

TABLE I
COMPARISONS OF SEGMENTATION PERFORMANCE ON THE WEIZMANN HORSE DATASET. EXAMPLE A.

are given in table (I). See [15] for more detailed results, convergence analysis, generalization analysis, and feature selection.

This RCM was also evaluated for object matching on a standard face dataset [23]. Examples are given in figure (7). We obtained an average pixel error of 6.0 pixels on this dataset (using the same parameter settings as for horses and cows) compared to the best result of 5.7 pixels [23]. The algorithm ran in about 20 seconds for a typical 300×200 size image (timed in 2008).

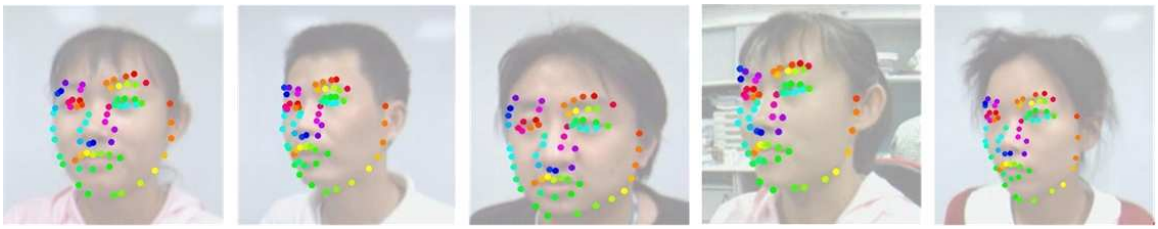

Fig. 7. Multi-view Face Alignment on examples from a face dataset [23]. Example A.

B. The Hierarchical Configural Deformable Template: AND/OR Graphs

This RCM is a deformable model which is capable of dealing with different poses and configurations of objects, see figure (8) and [11]. This requires much greater flexibility than the RCMs in the previous example. Here we supplement the graph structure by introducing OR nodes and switch variables [5]. This allows the graph to change topology and hence enables it to deal with great variations in the appearance and geometry of the object. Essentially the AND/OR is a very efficient way – in terms of representation and computation – to implement mixture models as shown in figure (8).

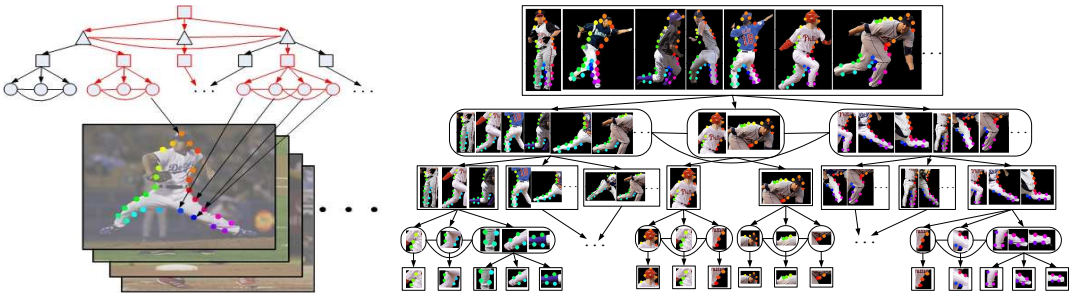

Fig. 8. Including OR nodes, and OR-potentials, enables the topology of the graph to change in response to the input data. This enable the RCM to alter the graph topology to deal with different poses of the object. Each OR node is required to select one of its child nodes by a switch variable. Example B.

Fig. 9. The parse results (odd columns) and segmentation results (even columns). We use colored dots to show the positions of the subparts (i.e. states of the graph nodes) for the parse results. The same color is used in all images to indicate the same subparts. Example B.

The Graph structure: The graph structure is the same as the previous example but we now divide the nodes into two classes: (i) AND nodes $\mathcal{V}^{AND}$, which have AND-potentials identical to those described in the previous example, and (ii) OR nodes $\mathcal{V}^{OR}$ which use OR-potentials where a parent node μ is required to select one of its children by a switch variable z_μ [5]. The graph structure was hand specified, see [5]. The OR nodes enable us to have many different graph topologies (determined by the states of the switch variables) which correspond to different poses of the object. Intuitively, the object is composed of parts and the switch variables determine which parts are chosen. Different object parts will be visible from different poses. In turn, these parts can be expressed as compositions of sub-parts and the choice of subparts again will depend on the object pose. This is illustrated in figure (8) where the graph has alternating levels of AND and OR nodes. Note that each specification of the switch variables corresponds to a hierarchical deformable model as described in the section (V-A). Hence we can think of a probabilistic model defined over an AND/OR graph as a mixture of distributions, where the number of mixture components corresponds to the number of different topologies.

This model can also be described as a hierarchical dictionary. The top level dictionary $\mathcal{T}^L$ contains a single RCM. This is related to the RCMs in the level-$(L-1)$ dictionary $\mathcal{T}^{L-1}$ by an AND-potential. These RCMs are related to RCMs in the level-$(L-2)$ dictionary $\mathcal{T}^{L-2}$ by OR-potentials. This pattern repeats with alternative levels connected by AND-potentials and OR-potentials finishing with AND-potentials relating the RCMs in the level-1 dictionary $\mathcal{T}^1$ to the level-0 RCMs $\mathcal{T}^0$. We constrain that each child RCM has only a single parent RCM, which ensures that the graph has no closed loops and guarantees that the compositional inference algorithm will converge to the optimal solution (subject to pruning).

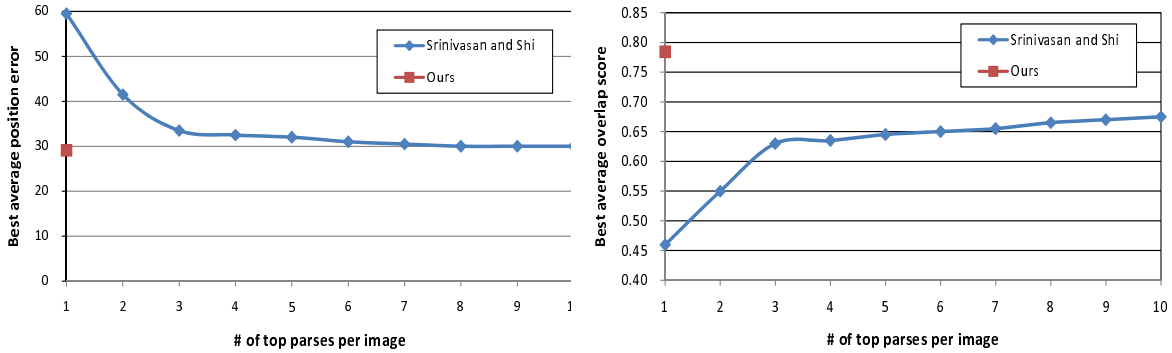

Fig. 10. We compare the results of HCDTs with those of Srinivasan and Shi [84]. The performance for parsing (position error) and segmentation (overlap score) are shown in the top and bottom figures respectively. Note that [84] select the best one (manually) of the top parses. Example B.

The State variables: The AND nodes have the same state variables as the previous RCM – i.e. $w_\mu = (\mathbf{x}_\mu, \theta_\mu, s_\mu)$, where $\mathbf{x}_\mu, \theta_\mu, s_\mu$ represent the position, orientation, and size of the subpart. The OR nodes have an additional 'switch' variable z_μ which indicates which child node is selected (if the OR node itself is not selected, then z_μ is unspecified).

The Potentials:

The data potentials $\boldsymbol{\phi}(w_\mu, \mathbf{I})$ are defined at the leaf nodes only and are identical to those used in the previous RCM. We use AND-potentials $\boldsymbol{\lambda}_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ for the AND nodes which are the same as for the previous model as specified by equation (21) (all AND nodes have the same potential).

We define OR-potentials for the OR nodes. These potential terms bias the selection of the child nodes. The switch variables z_μ for an OR node μ takes values $z_\mu \in ch(\mu)$, and has a potential term:

$$\boldsymbol{\lambda}^P_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, z_\mu \mathbf{w}_{ch(\mu)}) = - \sum_{\nu \in ch(\mu)} \lambda_{\mu\nu} \delta_{z_\mu,\nu} \delta(w_\mu - w_\nu) \tag{22}$$

The Inference Algorithm and Pruning:

The inference algorithm is a natural extension of the compositional inference algorithm defined above in section (V-A) including the three pruning mechanisms.

The only difference is that we modify equation (7) at OR nodes so that we select the best child. This replaces $\min_{\{b_i\}}\{\boldsymbol{\lambda}^P_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, \{w_{\mu_i,b_i}\}) + \sum_{j=1}^{|ch(\mu)|} E^{\min}_{\mu_j}(w_{\mu_j,b_j}, \mathbf{I})\}$ by $\min_{t_\mu}\{\boldsymbol{\lambda}^P_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}_{\mu,ch(\mu)}(w_\mu, t_\mu \mathbf{w}_{ch(\mu)}) + \min_b E^{\min}_{t_\mu}(w_{t_\mu,b}, \mathbf{I})\}$.

The Learning Algorithm:

The graph structure was specified manually (which takes 3 minutes per image). The parameters $\boldsymbol{\lambda}$ are learnt by structure max-margin. The loss function specifies a penalty for each node who pose w_μ differs from the groundtruth position by more than a threshold, see [11]. We also attempted to learn using the structure perceptron algorithm (i.e. without a loss function) but obtained worse results.

Datasets and Results:

The method was evaluated for parsing on the baseball dataset, see figure (10). Some examples are shown in figure (9). Our results were significantly better than previous methods which is probably due to the greater representational power of the AND/OR graph and differences in the learning algorithms. The algorithm ran at 24 seconds for 300×200 image (tested in 2008, time evaluated on the Weizmann dataset).

We also learnt an AND/OR model for horses using the Weizmann dataset. We obtained only slightly better segmentation results than our previous model, see previous section, which we attribute to the limited pose variations in the dataset and the crudeness of the segmentation performance measure (e.g., good performance can be obtained using models which fail to detect the legs of the horse accurately). But the AND/OR graph did perform better on the more challenging parsing task, see [11].

C. Image Labeling. Hierarchical Image Models

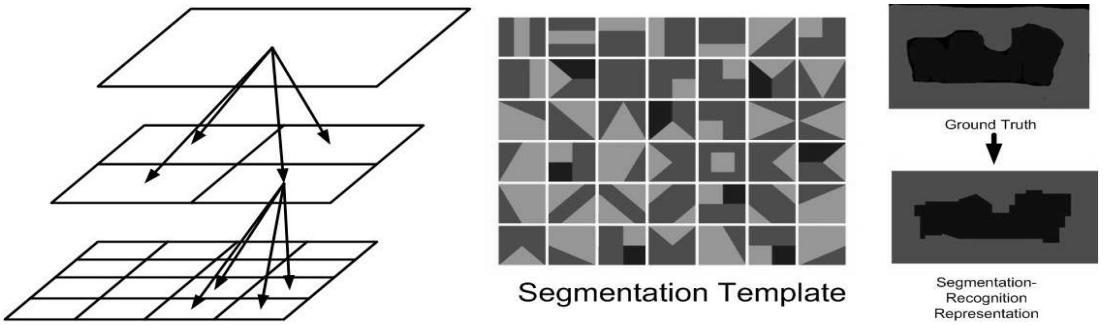

Fig. 11. The left panel shows the structure of a hierarchical image model [13]. The grey circles are the nodes of the hierarchy. The graph structure is a fixed quadtree so that all non-leaf nodes have four children. The center panel shows 30 segmentation templates. The right panel shows an example of the groundtruth and the labeling result. Example C.

We now apply RCMs to the image labeling problem [13]. This is a very different application and requires a different type of state variable, which must represent the labels and segmentation of image regions. We call this a Hierarchical Image Model (HIM). Note that there is a trade-off between the complexity of the graph structure (e.g., the use of OR nodes) and the complexity of the state variables. The last application used a complex graph structure, with variable topology, and a simple state variable. By contrast, this application uses a simple graph structure but a complex state variable.

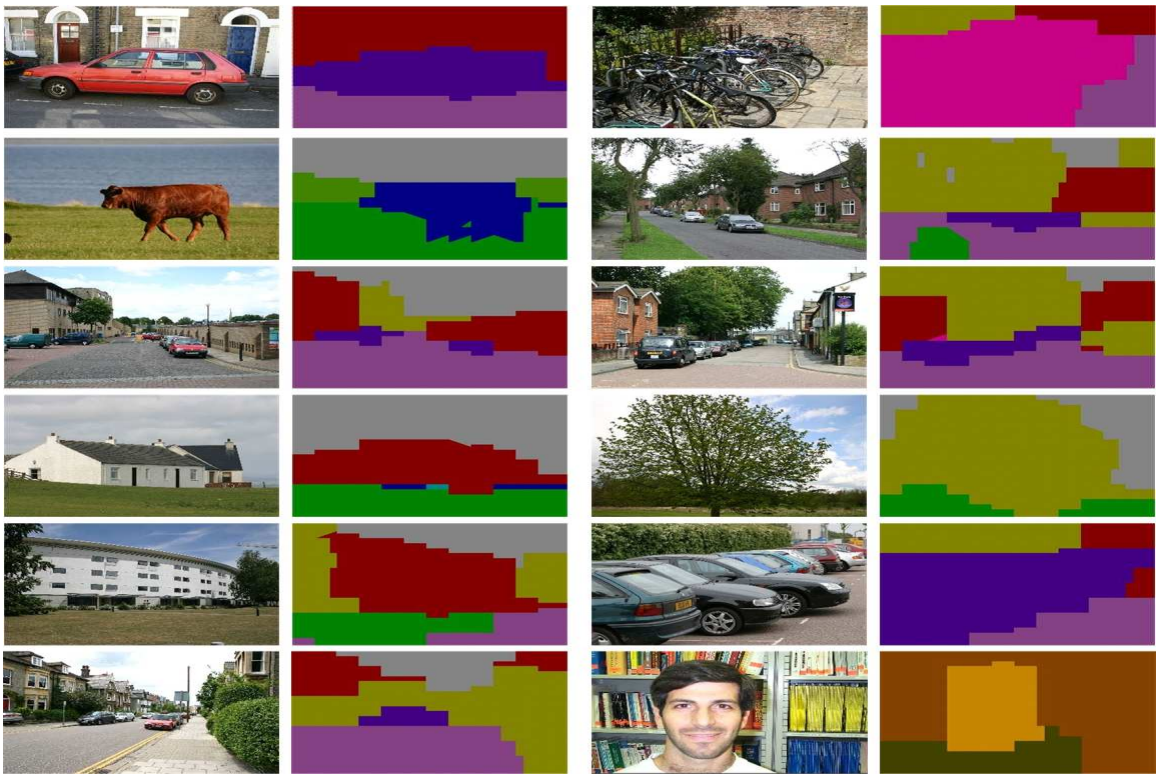

Fig. 12. Images and labels given by the Hierarchical Image Model. Example C.

The Graph structure The graph structure is a fixed quadtree structure where each non-leaf node has four child nodes, see figure (11).

The State variables. For this application, the state variables $w_\mu = (s_\mu, \vec{c}_\mu)$ represent a region of the image, see figure (11) (left panel). The variable $s_\mu \in \mathcal{S}$ indexes a *segmentation-template*, see figure (11) (center panel), which specifies a partition of the region into different sub-regions ($|\mathcal{S}| = 40$ and there

	Textonboost[24]	PLSA-MRF [85]	Auto-context [86]	Classifier only	HIM
Average	57.7	64.0	68	67.2	74.1
Global	72.2	73.5	77.7	75.9	81.2

TABLE II
PERFORMANCE COMPARISONS FOR AVERAGE ACCURACY AND GLOBAL ACCURACY. "CLASSIFIER ONLY" ARE THE RESULTS WHERE THE PIXEL LABELS ARE PREDICTED BY THE CLASSIFIER OBTAINED BY BOOSTING ONLY. EXAMPLE C.

are either one, two or three sub-regions). The variable $\vec{c}_\mu$ specifies labels for all the sub-regions where each label $c \in \mathcal{C}$, where $\mathcal{C}$ is a set of pre-defined labels ($|\mathcal{C}| = 23$ for the Cambridge Microsoft Labeled Dataset – [24]). As above, the states of the high-level nodes give a coarse description of the image while the lower level nodes give higher precision.

The Potentials.

The potential terms $\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}$ impose statistical consistency between the states of the parent and child nodes for the labeling and segmentation variables. They also impose priors on the segmentation templates and on the labeling. In this application there are no horizontal edges linking the child nodes.

There are six potential terms. The first is a data term $\boldsymbol{\lambda}^1_\mu \cdot \boldsymbol{\phi}^1_\mu(w_\mu, \mathbf{I})$ which represents image features of regions (features are color, Gabors, difference of Gaussians, etc,). The second is a data term $\boldsymbol{\lambda}^2_\mu \cdot \boldsymbol{\phi}^2_\mu(w_\mu, \mathbf{I})$ which favors segmentation templates whose pixels within each partition have similar appearances. The third term $\boldsymbol{\lambda}^3_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}^3_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ encourages consistency between the segmentation templates and labeling of parent and child nodes. The fourth term $\boldsymbol{\lambda}^4_{\mu,ch(\mu)} \cdot \boldsymbol{\psi}^4_{\mu,ch(\mu)}(w_\mu, \mathbf{w}_{ch(\mu)})$ captures the co-occurrence of different labels (e.g. a cow is unlikely to be next to an airplane). The fifth term term $\boldsymbol{\lambda}^5_\mu \cdot \boldsymbol{\psi}^5_\mu(w_\mu)$ is a prior over the segmentation templates. The sixth term term $\boldsymbol{\lambda}^6_\mu \cdot \boldsymbol{\psi}^6_\mu(w_\mu)$ models the co-occurrence of the labels and the segmentation templates.

The Inference Algorithm and Pruning:

We use the compositional inference algorithm specified by equations (7,8). The state space consists of the set of segmentation templates and the set of labels that can be assigned to them. This is smaller than the size of the state space for the object models, but it is still large. We prune the proposals based on their energies so as to keep a fixed proportion of the proposals (we do not use surround suppression or the two out of three rule).

The Learning Algorithm:

We used the structure perceptron algorithm to learn the parameters. The simple form of the graph structure makes it straightforward to estimate states of the high-level nodes from the groundtruth labels of the image pixels.

Datasets and Results: We tested the RCM on the Microsoft dataset [24]. We show example results in figure (12) and report good performance compared to state of the art methods in table (II). See [13] for more details. the average runtime of the algorithm was 30 seconds for a 320×200 image (tested in 2008).

D. Object Detection on the Pascal Databases

We now apply our approach to the challenging PASCAL datasets see [17]. We use a mixture of graphs with no closed loops. We use complex features based on HOG. For these datasets the groundtruth only specifies whether an object is present or not within an image region and so the state variables of the graph are hidden during learning. Code for the latent hierarchical learning is available by emailing the third author or deform Leo Zhu's website (google Leo Zhu CSAIL – and go to the Pascal application).

The Graph structure

This model consists of two hierarchical graphs. Each graph has three levels, see figure (13)(a,b). There is 1 node at the third level, 9 nodes at the second level, 36 nodes at the first level. Each node at the second level has 4 children. All the edges are from parents to children so there are no closed loops.

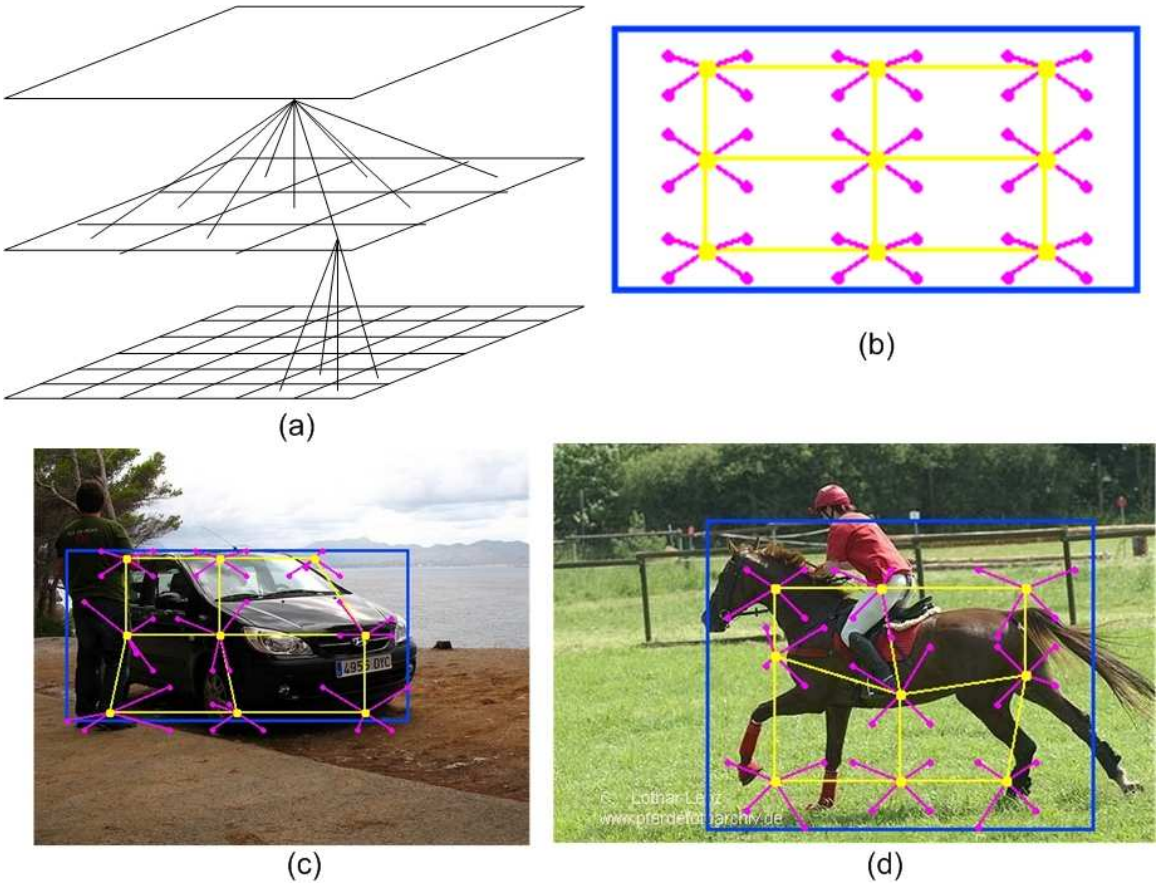

Fig. 13. (a) The graph structure is a three-layer model. The root node has 9 child nodes, each of which has 4 children. (b) Each node corresponds to a rectangular region in the image. The big rectangle corresponds to the root node of the model. The light dots indicate positions of the centers of the rectangles of the 9 nodes at the middle level. The dark dots are the centers of the rectangles of the 36 nodes at the lowest level. (c,d) The positions of the nodes (i.e. their rectangles) are variable and adjust to the structure of the object in the image. Example D.

The State variables

A binary-valued variable $y \in \{-1, +1\}$ indicates whether the object is present or not in a specific image window.

The state variables of the model are $\{V, \mathbf{W}_1, \mathbf{W}_2\}$, where the binary-valued variable V indicates which graphs is selected and $\mathbf{W}_1, \mathbf{W}_2$ indicate the state of each graph. Each graph node corresponds to a rectangular region in the image and represents an object part. The size of these regions is the same for all nodes at each level. Each node has a state variable $w_\mu = p_\mu$ which indicates its position in the image, see figure (13)(c,d). The variables $V, \mathbf{W}_1, \mathbf{W}_2$ are not specified in the training dataset, so we denote them as hidden variables $h = (V, \mathbf{W}_1, \mathbf{W}_2)$.

The Potentials

The data potentials are defined at all graph nodes. The feature functions $\boldsymbol{\phi}(w_\mu, V, \mathbf{I})$ are defined in terms of HOG features in the corresponding image region. For details, see [17].

The prior potentials relate the states of the children and parent nodes – i.e. their relative positions. The feature functions $\boldsymbol{\psi}(w_\mu, \mathbf{w}_{ch(\mu)})$ are quadratic in the difference in positions of the state variables of the child and parent, and hence correspond to Gaussian distributions. We set $\boldsymbol{\Phi}(V, p, y = -1) = 0$ for when the object is not present.

The Inference Algorithm and Pruning

The inference task is to estimate $y^*(\mathbf{I}), h^*(\mathbf{I}) = \arg\max_{y,h} \boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(y, h, \mathbf{I})$.

For $y = 1$ and for each V, we perform inference by dynamic programming using equations (7,8). We

class	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
us	**.294**	.558	.094	**.143**	.286	**.440**	**.513**	.213	**.200**	**.193**	**.252**	.125	**.504**	.384	**.366**	.151	**.197**	**.251**	**.368**	**.393**
UoCTTI	.290	.546	.006	.134	.262	.394	.464	.161	.163	.165	.245	.050	.436	.378	.350	.088	.173	.216	.340	.390
UCI	.288	**.562**	.032	.142	**.294**	.387	.487	.124	.160	.177	.240	.117	.450	**.394**	.355	**.152**	.161	.201	.342	.354
V07	.262	.409	**.098**	.094	.214	.393	.432	**.240**	.128	.140	.098	**.162**	.335	.375	.221	.120	.175	.147	.334	.289

TABLE III

PERFORMANCE COMPARISONS ON THE 20 PASCAL VOC 2007 CHALLENGE CATEGORIES [19]. (US) REFERS TO OUR 3-LAYER MODEL. (UOCTTI) REPORTS THE RESULTS FROM [68] WITHOUT SPECIAL POST-PROCESSING. (UCI) [87] IS A METHOD USING MULTI-OBJECT RELATIONSHIP. (V07) IS THE BEST RESULT FOR EACH CATEGORY AMONG ALL METHODS SUBMITTED TO THE VOC 2007 CHALLENGE. OUR METHOD OUTPERFORMS THE OTHER METHODS IN 13 CATEGORIES. OUR 3-LAYER MODEL IS BETTER THAN UOCTTI'S 2-LAYER MODEL IN ALL 20 CATEGORIES. THE AVERAGE APS PER CATEGORY ARE 0.296(US), 0.268 (UOCTTI) AND 0.271 (UCI).

use energy pruning and surround suppression as for our first example. We compute the best energy for $y = 1$ and $V = 1, 2$ and compare the energy 0 for $y = -1$. Then we select the configuration with lowest energy. We apply this to images by moving a sliding region and then applying dynamic programming to each image region.

The Learning Algorithm

The training data is a set of image windows which contain either an image of the object or a background image. We apply the latent-SVM algorithm to deal with the hidden variables $h = (V, \mathbf{W})$. The loss function $l(y_i, y) = 0$ if $y = y_i$ and $= 1$ if $y \neq y_i$.

Results

This model worked well on PASCAL challenge datasets, see table (III) [17], and example results are shown in figure (13)(c,d). The model described here was adapted to include active windows yielding improved performance [88]. A later version obtain second position in the Pascal Challenge for object detection at ECCV 2010. The runtime per image was 8 seconds (tested in 2010).

E. Families of RCMs for Multi-Object/Viewpoint Detection

In this section we consider families of RCMs for representing multiple objects and viewpoints. We use the hierarchical dictionary representation introduced earlier in section (IV-D).

The Graph Structure

The graph structure for a family of RCMs is a hierarchical graph where the root node is an OR node whose child nodes correspond to different objects, see figure (5), represented by RCMs, see figure (4). Because these object models share subparts it is more convenient to describe them in terms of hierarchical dictionaries of RCMs. In this case, the dictionaries at neighboring levels are related only by AND-potentials (i.e. all nodes except the root node are AND nodes). We constrain each parent RCM to have exactly three child RCMs, but child RCMs can have multiple parents and so the graph structure will have closed loops. The graph structure was learnt automatically [16] by a variant of the unsupervised learning algorithm described in section (VI).

The State Variables

The state variables $w_\mu = (\mathbf{x}_\mu, \theta_\mu, s_\mu)$ are the positions, poses, and sizes of subparts. Hence the RCMs in each dictionary have the same form as the Hierarchical Deformable Models described in section (V-A), see figure (4).

The Potentials

The prior potentials are all AND-potentials with the same form as those in section (V-A).

There are two types of data potentials. The *first type* associates a rectangular mask $R(\Omega, w_{\mathcal{R}})$ to the root node for each object. These model the body appearance, or *body map*, using regional cues which capture the texture and material properties of objects, see figure (14). These are described by the feature functions $\phi^{body}(w_{\mathcal{R}}; \mathbf{I}) = (1/|R(\Omega, w_{\mathcal{R}})|) \sum_{\mathbf{x} \in R(\Omega, w_{\mathcal{R}})} \log P(b(\mathbf{x})|\mathbf{I})$. These *body potentials* are specific to each object but are independent of viewpoint. The *second type* are defined at the leaf nodes and are intended to model the image features at the boundary of the object, or *boundary map*. They are similar

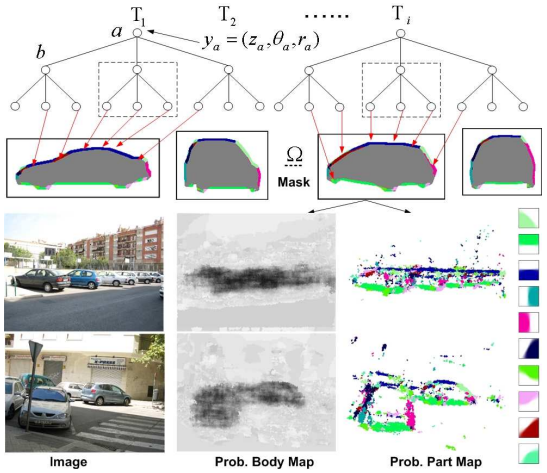

Fig. 14. The top panels show RCMs of cars from different viewpoints. The leaf nodes correspond to the features on the boundary of the cars (indicated by arrows). Each root node has a mask Ω which helps capture the texture and appearance of the object (assumed to be the same for each viewpoint). The bottom panels show an image (left), the *body map* (center) which indicates the response of texture and appearance cues, and the *boundary map* (right) which shows the response of edge-like cues.

to those used in previous models, such as HIM (at the leaf nodes) in section (V-A) and the Hierarchical Configural Deformable Template in section (V-B).

The Inference Algorithm and Pruning:

As discussed in section (IV-D), the compositional inference algorithm is applied directly to the dictionaries. We use the same pruning techniques as in section (V-A).

For this application, each child RCM can have multiple parent RCMs and so there will be closed loops. Hence the compositional inference algorithm is not guaranteed to converge to the optimal solution (subject to pruning). Instead we treat it as a message parsing algorithm [1] which are known to perform well on loopy graphs.

We stress that the inference algorithm outputs the objects/viewpoints which have been detected as well as their parses – there is no need for a further classification stage.

Learning

The family of RCMs is learnt by a hierarchical clustering algorithm whose input is a set of silhouettes of the different objects/viewpoints. To encourage part-sharing the labels of the objects/viewpoints are ignored. The algorithm outputs the hierarchical dictionaries, including the parameters of the RCMs. For reasons of space, we do not describe this algorithm in detail but we will sketch it in section (VI).

The object/viewpoint masks are learnt from the bounding boxes (and object boundaries) specified by LabelMe [20]. This is performed by a simple averaging over the boundaries of different training examples (for fixed object and viewpoint).

The body and edge potentials $P(b(\mathbf{x})|\mathbf{I})$ and $P(e(w_\mu)|\mathbf{I})$ are learnt using standard logistic regression techniques [89] with image features as input (see below). The edge potentials are the same for all objects and viewpoints.

The image features include the greyscale intensity, the color (R,G, B channels), the intensity gradient, Canny edges, the response of DOG (difference of Gaussians) filters at different scales and orientations. A total of 55 spatial filters are used to calculate these features (similar to [24])). The scales of these features are small for the edges and larger for the body (which need greater spatial support to capture texture and material properties). Two types of edge features were used depending on the quality of the groundtruth in the applications. For single objects, we use features such as edges and corners – see the probabilistic maps calculated by the body and part potentials in figure (14). For multiple objects applications we had poorer quality groundtruth and used simpler edge-based potentials similar to [14] based on the log-likelihood ratio of filter responses on and off edges.

Results

We evaluated our approach in two ways. Firstly, by the families of RCMs that we learn from the training data. Secondly, by the performance of the family of RCMs on Pascal datasets.

Figure (15) shows the mean shapes of typical RCMs at different levels of the hierarchy. Note that the first few levels appear to capture generic shapes which are typical of those proposed by Gestalt theorists and used in theories of perceptual grouping. The higher levels are more object specific and describe parts which are shared between only a few objects. The amount of part sharing is illustrated in figure (16), which shows that the majority of shared RCMs are those in the lower level dictionaries.

Fig. 15. This figure shows the mean shapes of sub-RCMs at different levels. RCMs learn 5-level dictionary from 120 object templates (27 classes).

The performance of the RCMs were quantified on three datasets: (i) Multi-View Motorbike detection (PASCAL), (ii) The Weizmann horse dataset, and (iii) a LabelMe car dataset. We compared our approach to published results on the first two datasets and trained a HOG-SVM for comparison the LabelMe dataset. The results, see figure (17), show competitive performance to state of the art methods. The runtimes on the first two datasts were typically 1.5 minutes for images of size 1200×900 (tested in 2009).

VI. Unsupervised Structure Learning

Learning the graph structure for an RCM from unsupervised training data is more challenging than learning the parameters. But it is also more exciting since it offers the possibility of avoiding the need for hand-labeled datasets and of obtained deeper understanding of the structure of images and objects. Our unsupervised learning was reported in [14], and was modified to learn the family of RCMs reported in section (IV-D) [16]. The approach has some similarity to the work of [90] which performs hierarchical grouping but which is not formulated in probabilistic terms.

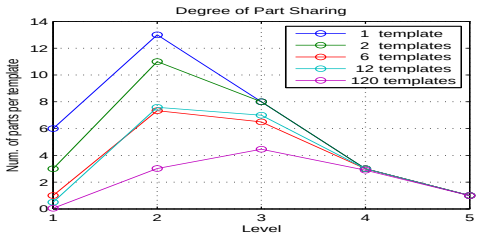

Fig. 16. Each curve plots the proportion of parts to object/viewpoints as a function of the levels of the parts (for five families containing different numbers of object/viewpoints). The overall representation cost can be quantified by the size of area under the curves. Observe that the larger the families then the lower the cost and the more the amount of part sharing.

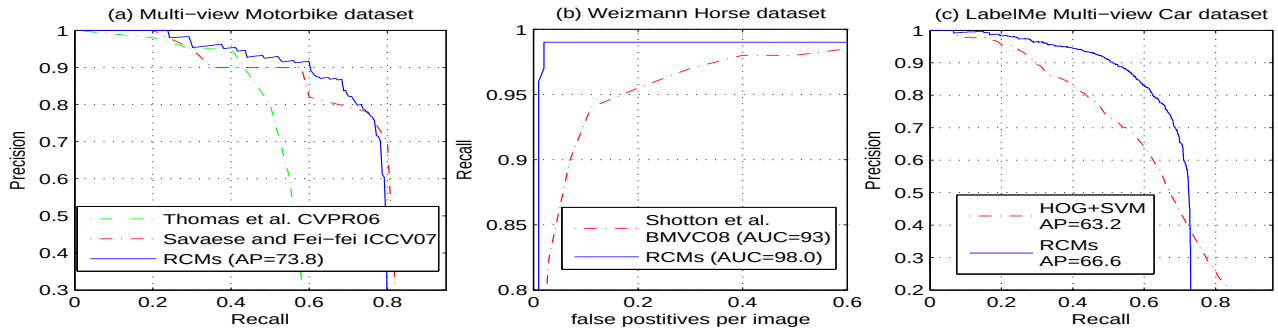

Fig. 17. RCMs (blue curves) show competitive performance compared with state-of-the-art methods on several datasets.

For reasons of space, we will only briefly sketch this method and refer to conference papers [14],[16] for more details.

The input is a set of images containing the same object, but with varying background, and the output is an RCM for the object [14]. The strategy is to learn the RCM by building it up from elementary components and is analogous to the compositional inference algorithm, except that it works in the space of object models. It can be thought of as learning a hierarchical dictionary as described in section (IV-D). This is performed in a *bottom-up pass* followed by a *top-down pass* which combines dictionary elements to obtain the

The bottom-up pass proceeds by defining a zeroth level dictionary $\mathcal{T}^0$ which consists of RCMs with a single node and potentials and parameters which tune them to edges at different orientations (four different orientations in [14], six in [16]). Then we parse the training images to find instances of these level-0 RCMs. Next we perform a series of operations, illustrated in figure (18)(left) and described below, which generate a level-1 dictionary of RCMs. We proceed from level to level until our procedure stops automatically when it fails to output an RCM. The output of the bottom-up pass is a set of dictionaries, whose mean shapes are shown in figure (18)(right). The top-down process performs operations on the dictionaries – e.g., adding subparts – to build a complete model of the horse, see figure (19)(left).

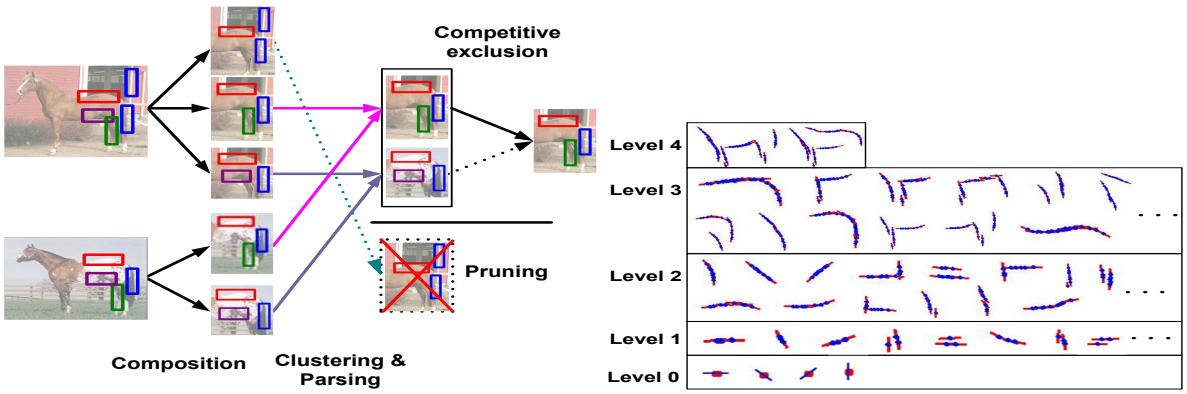

Fig. 18. Left panel: This figure illustrates the bottom-up process of unsupervised learning. The far left panel shows the instances of the RCMs in the level l dictionary. The left panel shows the compositions (step 1). Clustering is performed to generate level $l+1$ RCMs which are used to parse the image (steps 2 and 3). We perform suspicious coincidences and competitive exclusion (steps 4 and 5) as shown in the right panel. This results in the level $l+1$ dictionary, shown in the far right panel. 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.

We evaluated these models on the Weizmann dataset using twelve images for testing. We applied some postprocessing using a method inspired by Grab-Cut [91] to improve the boundaries found by the model, see [12],[15] for details (we use the initial estimate provided by the RCM to initialize a max-flow/min-cut algorithm),

Some results are shown in figure (19)(right). The overall performance of our algorithm was 93.3 % when tested on 316 images (and trained on 12). This is only slightly worse than our supervised method, see section (V), and hence is close to the state of the art. Other examples are shown in [14]. In general, the dictionaries at the first three levels appear very similar for all objects.

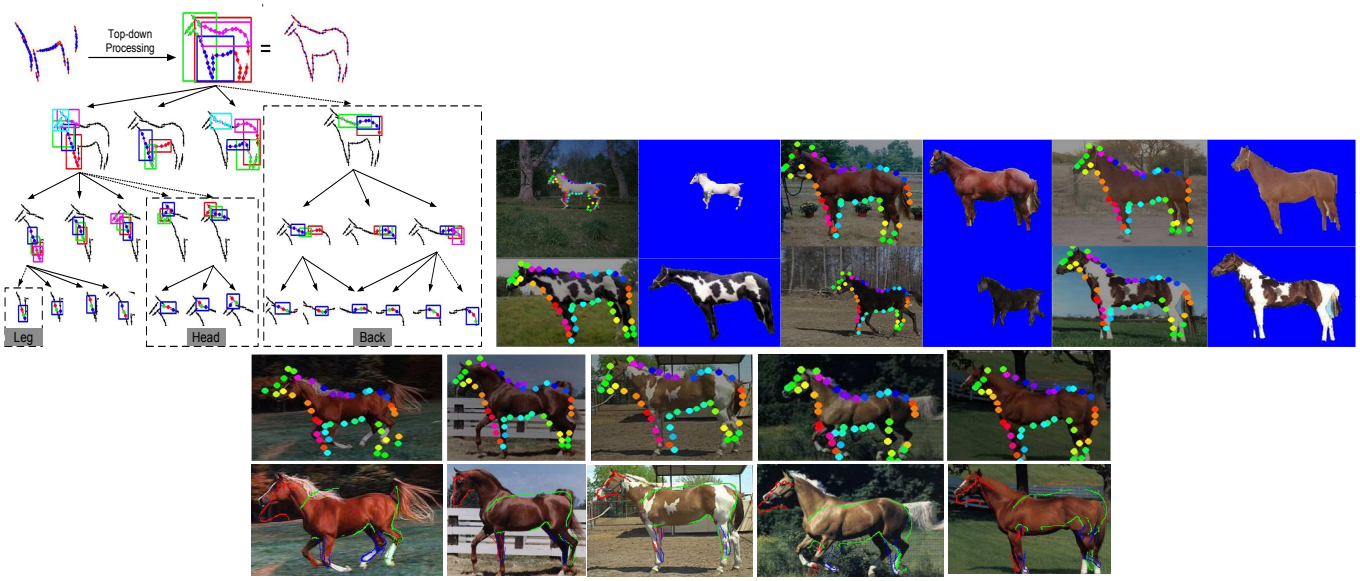

Fig. 19. In the top two rows, Columns 1, 3 and 5 shows the parse results of the RCM followed by the segmentation results. The parse results are illustrated by the colored dots which correspond to the leaf nodes of the object. Note that these are consistent between images. Rows 3 and 4 show the parse results of our method and OBJ CUT (cropped from [67]) on 5 images used in [67] respectively. Note our method obtains more complete boundaries.

VII. Conclusion

This paper has reviewed recent work on Recursive Compositional Models (RCMs). It has given described the basic mathematical and computational framework and shown applications to different visual tasks. In particular, we have shown that RCMS can be successfully applied both to tasks involving objects (i.e. detection and parsing) and to image interpretation asks (i.e. image labeling). In all cases, RCMs obtain results which are at, or close to, the state of the art when evaluated on the most challenging benchmarked datasets.

The key ideas are the use of hierarchical models defined in a recursive way. The hierarchical structure enables us to represent object and image properties at a range of difference scales so that we can take into account statistical regularities at all scales. The executive summary principle ensures that the high level nodes represent coarse, or summary, information about objects and images while the lower level nodes provide the finer scale details. Part sharing helps simply the representation when dealing with multiple objects/viewpoints/poses and greatly speeds up computation.

There are many ways to extend RCMs in the future. The framework proposed here can be applied to other visual tasks – for example, recent work applied the same principles for modeling motion correspondence and provides a model for human perception of motion [75]. The models described here can also be strengthened by adding richer visual cues and more complex graph structures. There are many issues to be addressed – for example, what is the tradeoff between representation with simple graphical topology and complex state variables as against complex graphs (with variable topology) and simple state variables? Also it is interesting to explore the relationship between RCMs and neuroscience since there appear to be intriguing similarities. the structure of the visual cortex.

Acknowledgements

We acknowledge funding from NSF with grants IIS-0917141 and 0613563 and from AFOSR FA9550-08-1-0489. This research was also supported by the WCU (World Class University) program through the

National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (R31-10008). For the work in subsection (V-E) the first author was partially supported by NGA NEGI-1582-04-0004 and MURI Grant N00014-06-1-0734. We thank George Papandreou for giving feedback on a draft of the paper and for drawing some of the figures. We thank our collaborators in our referenced paper for their many contributions.

REFERENCES

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kaufmann, 1988.
[2] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press, 1999.
[3] D. Heckerman, "A tutorial on learning with bayesian networks," pp. 301–354, 1999.
[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2003.
[5] S. Zhu and D. Mumford, "A stochastic grammar of images," vol. 2, no. 4, pp. 259–362, 2006.
[6] U. Grenander, *Pattern Synthesis: Lectures in Pattern Theory 1*. New York, NY, USA: Springer, 1976.
[7] ——, *Pattern Analysis: Lectures in Pattern Theory 2*. New York, NY, USA: Springer, 1978.
[8] J. Tenenbaum and A. Yuille, *IPAM Summer School: The Mathematics of the Mind*. IPAM, UCLA., 2007.
[9] Y. Jin and S. Geman, "Context and hierarchy in a probabilistic image model," in *CVPR (2)*, 2006, pp. 2145–2152.
[10] L. Zhu and A. L. Yuille, "A hierarchical compositional system for rapid object detection," in *NIPS*, 2005.
[11] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. L. Yuille, "Max margin and/or graph learning for parsing the human body," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.
[12] L. Zhu, Y. Chen, X. Ye, and A. L. Yuille, "Learning a hierarchical log-linear model for rapid deformable object parsing," in *CVPR*, 2008.
[13] L. Zhu, Y. Chen, Y. Lin, and A. L. Yuille, "A hierarchical image model for polynomial-time 2d parsing," in *Advances in Neural Information Processing System*, 2008.
[14] L. Zhu, C. Lin, H. Huang, Y. Chen, and A. L. Yuille, "Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion," in *Proceedings of The 10th European Conference on Computer Vision*, 2008.
[15] L. Zhu, Y. Chen, and A. L. Yuille, "Learning a hierarchical deformable template for rapid deformable object parsing," in *Transactions on Pattern Analysis and Machine Intelligence*, 2009.
[16] L. Zhu, Y. Chen, A. Torrabla, W. Freeman, and A. L. Yuille, "Recursive compositional models with re-usable parts for multi-view multi-object detection and parsing," in *CVPR*, 2010.
[17] L. Zhu, Y. Chen, A. L. Yuille, and W. Freeman, "Latent hierarchical structure learning for object detection," in *CVPR*, 2010.
[18] E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," in *ECCV (2)*, 2002, pp. 109–124.
[19] E. M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
[20] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *IJCV*, 2008.
[21] G. Mori, "Guiding model search using segmentation," in *Proceedings of IEEE International Conference on Computer Vision*, 2005, pp. 1417–1423.
[22] D. R. Magee and R. D. Boyle, "Detecting lameness using 're-sampling condensation' and 'multi-stream cyclic hidden markov models'," *IMAGE AND VISION COMPUTING*, vol. 20, p. 2002, 2002.
[23] H. Li, S.-C. Yan, and L.-Z. Peng, "Robust non-frontal face alignment with edge based texture," *J. Comput. Sci. Technol.*, vol. 20, no. 6, pp. 849–854, 2005.
[24] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi, "*TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *ECCV (1)*, 2006, pp. 1–15.
[25] Z. Tu and S. C. Zhu, "Image segmentation by data-driven markov chain monte carlo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 657–673, 2002.
[26] Z. Tu, X. Chen, A. L. Yuille, and S. C. Zhu, "Image parsing: Unifying segmentation, detection, and recognition," in *ICCV*, 2003, pp. 18–25.
[27] H. Chen, Z. Xu, Z. Liu, and S. C. Zhu, "Composite templates for cloth modeling and sketching," in *CVPR (1)*, 2006, pp. 943–950.
[28] Y. Wu, Z. Si, C. Fleming, and S. Zhu, "Deformable template as active basis," in *Proceedings of International Conference of Computer Vision*, 2007.
[29] Y. Amit, D. Geman, and X. Fan, "A coarse-to-fine strategy for multiclass shape detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 12, pp. 1606–1621, 2004.
[30] A. S. Willsky, "Multiresolution Markov models for signal and image processing," *Proc. of the IEEE*, vol. 90, no. 8, pp. 1396–1458, Aug. 2002.
[31] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[32] M. Ranzato, Y.-L. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *NIPS*, 2007.
[33] Y. Bengio, P. Lamblin, P. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 19, 2007.
[34] H. Lee, R. Grosse, R. Ranganath, and A. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *International Conference on Machine Learning (ICML)*, 2009.
[35] M. Riesenhuber and T. Poggio, "Cbf: A new framework for object categorization in cortex," in *Biologically Motivated Computer Vision*, 2000, pp. 1–9.
[36] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *CVPR (2)*, 2005, pp. 994–1000.

[37] S. Thorpe and M. Fabre-Thorpe, "Seeking categories in the brain," *Science*, vol. 291(5502), pp. 260–263, 2001.
[38] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.
[39] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen, "Bayesian updating in causal probabilistic networks by local computations," *Computational Statistics Quaterly*, vol. 4, pp. 269–282, 1990.
[40] J. Tenenbaum, T. Griffiths, and C. Kemp, "Theory-based bayesian models of inductive learning and reasoning," *Trends in Cognitive Sciences*, vol. 10(7), pp. 309–318, 2006.
[41] M. Collins, "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms," in *EMNLP*, 2002, pp. 1–8.
[42] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *NIPS*, 2003.
[43] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning, "Max-margin parsing," in *EMNLP*, 2004.
[44] Y. Altun, I. Tsochantaridis, and T. Hofmann, "Hidden markov support vector machines," in *ICML*, 2003, pp. 3–10.
[45] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *ICML*, 2004.
[46] S. Geman and D. Geman, "Stochastic relaxation, gibbs distribution and bayesian restoration of images," 1984.
[47] A. Blake and A. Zisserman, *Visual Reconstruction*. MIT Press, 1987.
[48] D. Geiger, B. Ladendorf, and A. Yuille, "Occlusions and binocular stereo," 1995.
[49] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 359–374, 2001.
[50] D. Geiger and A. Yuille, "A common framework for image segmentation," 1991.
[51] P. Felzenswalb and D. Huttenlocher, "Efficient belief propagation for early vision," 2004.
[52] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.
[53] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu, "Statistical edge detection: Learning and evaluating edge cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 57–74, 2003.
[54] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Computation*, vol. 9, no. 8, pp. 1627–1660, 1997.
[55] S. Kumar and M. Hebert, "Discriminative random fields: A discriminative framework for contextual interaction in classification," in *In ICCV*, 2003, pp. 1150–1157.
[56] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. 22, no. 1, pp. 67–92, 1973.
[57] A. L. Yuille, P. W. Hallinan, and D. S. Cohen, "Feature extraction from faces using deformable templates," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99–111, 1992.
[58] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *Lecture Notes in Computer Science*, vol. 1407, pp. 484–??, 1998. [Online]. Available: citeseer.ist.psu.edu/cootes98active.html
[59] J. M. Coughlan, A. L. Yuille, C. English, and D. Snow, "Efficient deformable template detection and localization without user initialization," *Computer Vision and Image Understanding*, vol. 78, no. 3, pp. 303–319, 2000.
[60] H. Chui and A. Rangarajan, "A new algorithm for non-rigid point matching," in *CVPR*, 2000, pp. 2044–2051.
[61] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, 2002.
[62] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.
[63] R. Fergus, P. Perona, and A. Zisserman, "A sparse object category model for efficient learning and exhaustive recognition," in *CVPR (1)*, 2005, pp. 380–387.
[64] Z. Tu and A. L. Yuille, "Shape matching and recognition - using generative models and informative features," in *ECCV (3)*, 2004, pp. 195–209.
[65] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, "Multiscale conditional random fields for image labeling," in *CVPR (2)*, 2004, pp. 695–702.
[66] J. M. Winn and N. Jojic, "Locus: Learning object classes with unsupervised segmentation," in *ICCV*, 2005, pp. 756–763.
[67] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Obj cut," in *CVPR (1)*, 2005, pp. 18–25.
[68] P. F. Felzenszwalb, R. B. Grishick, D. McAllister, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, 2009.
[69] N. Ahuja and S. Todorovic, "Learning the taxonomy and models of categories present in arbitrary image," in *ICCV*, 2007.
[70] E. Sharon, A. Brandt, and R. Basri, "Fast multiscale image segmentation," in *CVPR*, 2000, pp. 1070–1077.
[71] I. Kokkinos and A. L. Yuille, "Hop: Hierarchical object parsing," in *CVPR*, 2009.
[72] L. Zhu and A. L. Yuille, "A hierarchical compositional system for rapid object detection," in *NIPS*, 2005.
[73] 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.
[74] C. He, "Empirical studies of structured learning for deformable object parsing," *Master's Thesis*, vol. Deaprtment of Statistics, p. UCLA, 2008.
[75] S. Wu, X. He, H. Lu, and A. Yuille, "A unified model of short-rnag and long-range motion perception," in *NIPS*, 2010.
[76] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *International Conference on Machine Learning (ICML)*, 2009.
[77] A. L. Yuille and A. Rangarajan, "The concave-convex procedure (cccp)," in *NIPS*, 2001, pp. 1033–1040.
[78] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *ECCV'04 Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004, pp. 17–32.

[79] J. Coughlan and A. L. Yuille, "Bayesian a* tree search with expected o(n) node expansions for road tracking," *Neural Computation*, vol. 14(8), pp. 1929–58, 2002.
[80] X. Ren, C. Fowlkes, and J. Malik, "Cue integration for figure/ground labeling," in *NIPS*, 2005.
[81] E. Borenstein and J. Malik, "Shape guided object segmentation," in *CVPR (1)*, 2006, pp. 969–976.
[82] T. Cour and J. Shi, "Recognizing objects by piecing together the segmentation puzzle," in *CVPR*, 2007.
[83] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *ECCV (4)*, 2006, pp. 581–594.
[84] P. Srinivasan and J. Shi, "Bottom-up recognition and parsing of the human body," in *CVPR*, 2007.
[85] J. Verbeek and B. Triggs, "Region classification with markov field aspect models," in *CVPR*, 2007.
[86] Z. Tu, "Auto-context and its application to high-level vision tasks," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.
[87] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *Proceedings of the International Conference on Computer Vision*, 2009.
[88] Y. Chen, L. Zhu, and A. L. Yuille, "Active mask hierarchies for object detection," in *Proceedings of The 12th European Conference on Computer Vision*, 2010.
[89] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2000.
[90] S. Fidler and A. Leonardis, "Towards scalable representations of object categories: Learning a hierarchy of parts," in *CVPR*, 2007.
[91] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.