Learning a Hierarchical Deformable Template for Rapid Deformable Object Parsing

Long (Leo) Zhu[1], Yuanhao Chen[2], Alan Yuille[1,3,4]

[1]Department of Statistics, [3]Psychology and [4]Computer Science
University of California, Los Angeles, CA 90095
{lzhu,yuille}@stat.ucla.edu

[2]University of Science and Technology of China, Hefei, Anhui 230026 P.R.China
yhchen4@ustc.edu

Abstract

In this paper, we address the tasks of detecting, segmenting, parsing, and matching deformable objects. We use a novel probabilistic object model that we call a hierarchical deformable template (HDT). The HDT represents the object by state variables defined over a hierarchy (with typically 5 levels). The hierarchy is built recursively by composing elementary structures to form more complex structures. A probability distribution – a parameterized exponential model – is defined over the hierarchy to quantify the variability in shape and appearance of the object at multiple scales. To perform inference – to estimate the most probable states of the hierarchy for an input image – we use a bottom-up algorithm called *compositional inference*. This algorithm is an approximate version of dynamic programming where approximations are made (e.g., pruning) to ensure that the algorithm is fast while maintaining high performance. We adapt the *structure-perceptron* algorithm to estimate the parameters of the HDT in a discriminative manner (simultaneously estimating the appearance and shape parameters). More precisely, we specify an exponential distribution for the HDT using a dictionary of potentials which capture the appearance and shape cues. This dictionary can be large and so does not require hand-crafting the potentials. Instead structure-perceptron assigns weights to the potentials so that less important potentials receive small weights (this is like a "soft" form of feature selection). Finally, we provide experimental evaluation of HDTs on different visual tasks including detection, segmentation, matching (alignment) and parsing. We show that HDTs achieve state of the art performance for these different tasks when evaluated on datasets with groundtruth (and when compared to alternative algorithms which are typically specialized to each task).

Index Terms

Hierarchy, Shape Representation, Object Parsing, Segmentation, Shape Matching, Structured Learning.

I. Introduction

Computer vision methods are currently unable to reliably detect, segment, and parse deformable objects in cluttered images. (By parsing, we mean the ability to identify parts, subparts and subsubparts of the object – which is useful for applications such as matching/alignment). Although there have been some partial successes – see [1], [2], [3], [4], [5], [6] and others reviewed in section (II) – none have reached the performance levels and computational speed obtained for detecting faces by using techniques such as AdaBoost [7], [8]. In our opinion, the main disadvantages of current approaches is that they are based on limited representations of deformable objects which only use a small part of the appearance and geometric information that is available. For example, current techniques may rely only on a sparse set of image cues (e.g. SIFT features or edgelets) and limited "flat" representations of the spatial interactions between different parts of the object, see figure (1). As theoretical studies have shown [9], the performance of models degrade if the models fail to represent (and hence exploit) all available information. But improved representation of deformable objects is only useful when it is accompanied by efficient techniques for performing inference and learning and, in practice, the representations used in computer vision are closed tied to the inference and learning algorithms that are available (e.g., the representations used in [1], [2] were chosen because they were suitable for inference by dynamic programming or belief propagation – with pruning). Hence we argue that progress in this area requires us to simultaneously develop more powerful representations together with efficient inference and learning algorithms.

In this paper, we propose a new class of object models – *hierarchical deformable templates (HDT)*. The HDT is specified by a hierarchical graph where nodes at different levels of the hierarchy represent components of the object at different scales, with the lowest level (leaf) nodes corresponding to the object boundary – see figure (1, 2). Formally we define state variables at every node of the graph and the state of the HDT is specified by the state of all the nodes. The state of a node is the position, orientation, and scale of the corresponding component of the object. The clique structure of the HDT (i.e., the sets of nodes that are directly connected to each other) model the spatial relations between different components of the object. The HDT has a rich representation of the object which enables it to capture appearance and spatial information at a range of different scales (e.g., figure (2) shows that nodes at different levels use

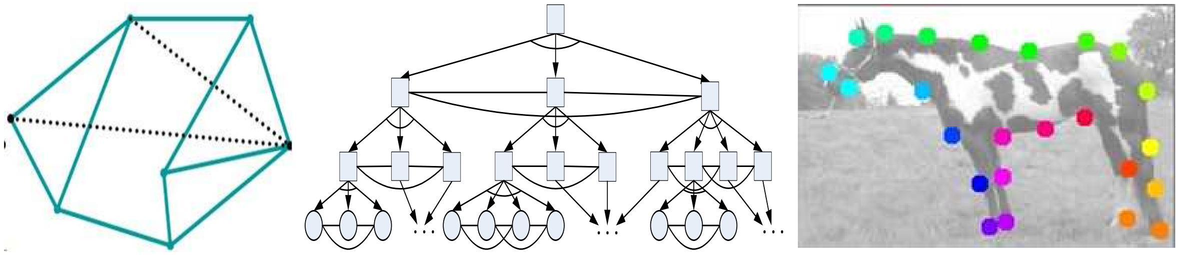

Fig. 1. Two alternative representations for deformable objects. Left panel: standard models use “flat” Markov Random Field (MRF) models relying on sparse cues and with limited spatial interactions [1], [2], [5], [6]. Middle panel: an HDT has a hierarchical representation with a large variety of different images cues and spatial interactions at a range of scales. We can, in theory, obtain a flat model from an HDT by integrating out all the state variables except those at the leaf nodes. This would give a flat model with extremely complicated connections (all nodes would be connected to each other) which, in this form, would be intractable for learning and inference. Right panel: The points along the object boundary correspond to the nodes in the “flat” MRF models or the leaf nodes of the HDT.

different appearance cues). Moreover, the richness of this representation implies that an HDT can be applied to a large range of tasks (e.g., segmentation requires estimating the states of the leaf nodes, detection requires estimating the root node, matching/alignment is performed by estimating and matching all the nodes). The HDT can be thought of as a hybrid discriminative-generative model (e.g., see [10] which does not have a hierarchy). The probability distribution specifying it has terms that can be interpreted as a generative prior for the configurations of the state variables but the appearance terms that relate these variables to the images are of discriminative form.

To ensure practicality of HDT, as discussed above, it is necessary to specify efficient inference and learning algorithms. We perform inference on an HDT – i.e., estimate the most probable states of the HDT for an input image – by *compositional inference* [11], [12], which we show is both rapid and effective. We perform partially-supervised learning of the parameters of HDT (in a discriminative manner) by extending the recent *structure perceptron learning algorithm* [13]. The graph structure of the HDT is learnt in an unsupervised manner by *one-example* learning using a clustering algorithm.

Compositional inference is a bottom-up approximate inference algorithm. The structure of HDTs means that it is possible to perform exact inference and estimate the best state by dynamic programming (DP) in polynomial time in the number of nodes of the hierarchy and the size of the state space. Unfortunately the state space size is very large since object components can occur anywhere in the image (and the state space also includes orientation and scale). Hence *compositional inference* is an approximate version of DP where we represent the state of each node by a set of proposals together with their energies (the terminology is suggested by the MCMC literature). Proposals for the states of a parent node are constructed by composing the proposals for its child nodes. These proposals are pruned by a *threshold* – to remove configurations whose energy is too large (e.g., when the proposals of the child nodes poorly satisfy the spatial relations required by the parent node) – and by *surround suppression* which selects the locally maximal proposal within a fixed window. The key idea is to keep the number of proposals small enough to be tractable but rich enough to yield good performance. Compositional inference was inspired by a compositional algorithm [11] which was applied to a simple model and was only tested on a small number of images. The current version was first reported in [12] where it was applied to AND/OR graphs. Since the HDT only outputs a sparse set of points on the object boundary (the states of the leaf nodes) we obtain a complete contour by using grab-cut [14] initialized by the connecting the estimate states of the leaf nodes.

We learn the graph structure of the HDT by one-example learning using a clustering algorithm and make initial estimates of the parameters of the HDT. Then we make full parameters of the HDT by adapting the structure-perceptron algorithm [13] (note that structure perceptron does not learn the graph structure). This enables us to learn all the parameters globally in a consistent manner (i.e., we learn the parameters at all levels of the hierarchy simultaneously). As we will show, structure-perceptron enables us to select different shape and appearance features from a dictionary and to determine ways to optimally weight them (similar to the selection and weighting strategy used in AdaBoost [7], [8]). Structure-perceptron learning is a discriminative approach that is computationally simpler than standard methods such as maximum likelihood estimation (since it avoids the need to evaluation the normalization constant of the distribution of the HDT). An additional advantage to discriminative learning is that it focusses attention on estimating those parameters of the model which are most relevant to task performance. We first reported on the success of structure-perceptron learning in [15].

We demonstrate the success and versatility of HDT's by applying them to a range of visual tasks. We show that they are very effective in terms of performance and speed (roughly 20 seconds for a typical 300×200 image – the speed increases approximately linearly in the size of the image) when evaluated on large datasets which include horses [16] and cows [17]. In particular, to illustrate versatility, we demonstrate state-of-the-art results for different tasks of object segmentation (evaluated on the Weizmann horse dataset [16]) and matching/alignment (evaluated on the face dataset – [18], [19]). The results on the alignment task on the face dataset are particularly interesting because we are comparing to results obtained by methods such as Active Appearance Models [20] which are specialized for faces and which have been developed over a period of many years (while we spent one week in total to run this application including the time to obtain the dataset). Overall, we demonstrate that HDTs can perform a large range of visual tasks (due to its hierarchical representation) while other computer vision methods typically restrict themselves to single tasks.

We perform diagnostic analysis to quantify how different components of the HDT contribute to overall performance and to the computational cost (e.g., speed). In particular, we compare how different levels of the hierarchy contribute to the overall performance. This type of diagnostic analysis, in particular the trade-offs between performance and computation, is necessary for developing principles for the optimal design of complex computer vision models like HDTs.

II. Background

There is a vast literature on techniques for the separate tasks of object detection, segmentation, and parsing/aligning. But these tasks have typically been studied separately and not addressed by a single model as we do in this paper. We give a brief review of the work that is the most relevant to our approach. The techniques used are generally fairly different although there are some similarities which we will discuss.

There has been a range of attempts to model deformable objects in order to detect, register, and recognize them. Many of them can be formulated as maximum a posteriori inference of the position, or pose, states z of the object parts in terms of the data $\mathbf{I}$ (i.e., an input image). Formally, they seek to estimate

$$z^* = \arg\max_z p(z|\mathbf{I}) = \arg\max_z p(\mathbf{I}|z)p(z), \tag{1}$$

where $p(\mathbf{I}|z)p(z) = p(\mathbf{I}, z)$ is of form:

$$p(\mathbf{I}, z) = \frac{1}{Z}\exp\{\sum_i \alpha_i f(\mathbf{I}(x_i), z_i) + \sum_{i,j} \beta_{ij} g(z_i, z_j)\}. \tag{2}$$

where Z is the normalization constant, x_i is image position. The unary potentials $f(\mathbf{I}(x_i), z_i)$ model how well the individual features match to the positions in the image. The binary potentials $g(z_i, z_j)$ impose (probabilistic) constraints about the spatial relationships between feature points. Typically, z is defined on a flat MRF model, see figure (1), and the number of its nodes is small.

Coughlan et al. [1] provided one of the first models of this type, using a sparse representation of the boundary of a hand, and showed that dynamic programming (DP) could be used to detect the object without needing initialization (but using pruning to speed up the DP). This type of work was extended by Felzenswalb [5] and by Coughlan who used a pruned version of belief propagation (BP) [2]. The main limitation of this class of model is that they typically involve local pairwise interactions between points/features (see the second term in equation (2)). This restriction is mainly due to computational reasons (i.e. the types of inference and learning algorithms available) and not for modeling reasons. There are no known algorithms for performing inference/learning for densely connected flat models – for example, the performance of BP is known to degrade for representations with many closed loops.

Other classes of models are more suitable for matching than detection [3], [2], [21]. Some of these models [2], [21] do use longer range spatial interactions, as encoded by shape context and other features, and global transformations. But these models are typically only rigorously evaluated on matching tasks (i.e., in situations where the detection is trivial). They all need good initialization for position, orientation, and scale if they are required to detect objects in images with background clutter.

Recent work has introduced hierarchical models to represent the structure of objects more accurately (and enable shape regularities at multiple scales). Shape-trees were presented [6] to model shape deformations at more than one level. Other work [22] [23] uses image features extracted at different scales but does not formulate them within a hierarchy. Alternative approaches [24] use hierarchies but of very different types. The hierarchical representations most similar to HDTs are the AND/OR graph representations [25], [26], [12]. But there are several differences: (i) these AND/OR model have appearance (imaging) cues at the leaf nodes only, (ii) they are only partially, if at all, learnt from the data, (iii) the image cues and geometrical constraints are mostly hand-specified. Our work has some similarity to the hybrid discriminative/generative models proposed by Tu [10] since HDTs combine discriminative models for the object appearance with generative models for the geometric configuration of the object (but Tu's work does not involve hierarchies).

Object segmentation has usually been formulated as a different task than object detection and has been addressed by different techniques. It aims at finding the boundary of the object and typically assumes that the rough location is known and does not involve recovering the pose (i.e. position, orientation, and scale) of the object. Borenstein and Ullman [16] provided a public horse dataset and studied the problem of deformable object segmentation on this dataset. Torr and his colleagues [27] developed Object-Cut which locates the object via a pictorial model learnt from motion cues and use the min-cut algorithm to segment out the object of interest. Ren et al. [28] addressed the segmentation problem by combining low-, mid- and high-level cues in Conditional Random Field (CRF). Similarly, Levin and Weiss [29] used CRF to segment object but assuming that the position of the object is roughly given. In contrast to supervised learning, Locus [30] explores a unsupervised learning approach to learn a probabilistic object model. Recently, Cour and Shi [31] currently achieve the best performance on this horse dataset. Note that none of these methods report performance on matching/alignment.

III. Hierarchical Deformable Templates (HDT)

This section describes the basic structure of HDTs. Firstly we describe the graphical structure in subsection (III-A). Secondly we specify the state variables and the form of the probability distribution in subsection (III-B). Thirdly, in subsection (III-C), we describe the procedure used to learn the graph structure from one example. The inference and parameter learning algorithms will be described in sections (IV,V) respectively.

A. The Graphical Structure of the HDT

We represent an object by a hierarchical graph defined by parent-child relationships. The top node of the hierarchy represents the pose (position, orientation, and scale) of the center of the object. The leaf nodes represent the poses of points on the object boundary and the intermediate nodes represent the poses of subparts of the object. This is illustrated in figure (2).

Formally, an HDT is a graph $G = (V, E)$ where V is the set of nodes (vertices) and E is the set of edges (i.e., nodes $\mu, \nu \in V$ are connected if $(\mu, \nu) \in E$)). There is a parent-child structure so that $ch(\nu)$ denotes the children of node ν. We require that the graph to be tree-like so that $ch(\nu) \bigcap ch(\mu) = \emptyset$ for all $\nu, \mu \in V$. The edges are defined by the parent-child relationships

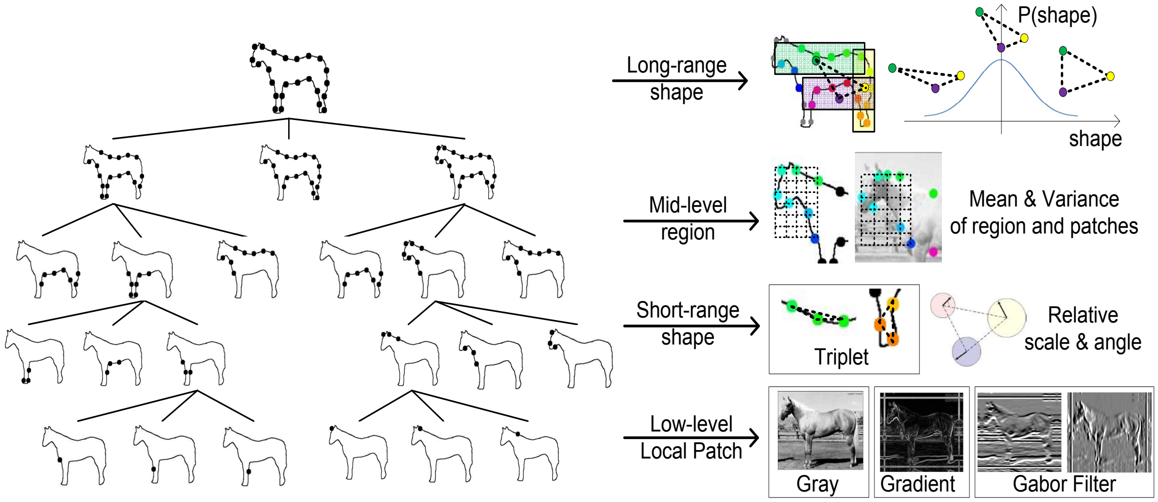

Fig. 2. Left Panel: The hierarchical graph of the HDT is constructed by a hierarchical clustering algorithm (see text for details). Black dots indicate the positions of the leaf nodes in the hierarchy. The arrows indicate how nodes at each level of the hierarchy are linked to their parents nodes at higher levels – i.e., how groups of subparts are composed to form bigger subparts. Right Panel: the appearance and shape deformation are modeled at all levels of the hierarchy and different appearance cues are used at different levels (e.g., mean and variance of features at the mid-levels and edge features at the low-levels).

and by the requirement that all the children of a node have an edge connecting them. Hence $(\nu, \mu) \in E$ provided either $\mu \in ch(\nu)$, $\nu \in ch(\mu)$, or there exist ρ s.t. $\mu, \nu \in ch(\rho)$. We define μ_R to be the root node of the graph. We let V^{LEAF} denote the leaf nodes. For any node ν, we define V_ν to be the subtree formed by the set of descendent node with ν as the root node. (We note that all the nodes of the graph are connected to the image data terms, see subsection (III-B).

B. The state variables and the potential functions

Each node $\mu \in V$ is assigned a state variable z_μ and we denote the state variables for the entire graph by $z = \{z_\mu : \mu \in V\}$. We denote $z_{ch(\mu)} = \{z_\nu : \nu \in ch(\mu)\}$ to be shorthand for the states of the child nodes of μ. The state variable indicates properties of the node and, in particular, the subregion $D(z_\mu)$ of the image domain $D \in R^2$ corresponding to the node. The state variable z_μ of node μ correspond to the position $\vec{x}_\mu$, orientation θ_μ, and scale s_μ of a subpart of the object: hence $z_\mu = (\vec{x}_\mu, \theta_\mu, s_\mu)$ (and $D(z_\mu)$ is calculated from these). For example, the state of the top node for an object model will correspond to the orientation, scale, and center position of the object – while the state of the leaf nodes will correspond to the orientation and position of elements on the boundary of the object. All these state variables are hidden – i.e., not directly observable. Note that the state variables take the same form at all levels of the hierarchy (unlike other standard hierarchical representations [32], [12]) which is important for the inference and learning algorithms that we describe in subsections (IV,V).

We now define probability distributions on the state variables defined on the graph. These distributions are of exponential form defined in terms of potentials $\phi(.)$ which are weighted by parameters α. The specify, for example, the probability distributions for the relative states of the hidden variables and the data terms. There are two types of terms: (i) "prior potentials" defined over the cliques of the graph $\vec{\phi}(z_\mu, z_{ch(\mu)})$, for all $\mu \in V$, which are independent of the image $\mathbf{I}$, (later we decompose the "prior" terms into "vertical" and "horizontal" terms) and (ii) "data potentials" of form $\vec{\phi}_D(\mathbf{I}, D(z_\mu))$, for all $\mu \in V$, which depend on measurements of the image $\mathbf{I}$ within the domain $D(z_\mu)$. The potentials will have coefficients $\vec{\alpha}_\mu, \vec{\alpha}^D_\mu$ respectively for all $\mu \in V$. We use α to denote $\{\vec{\alpha}_\mu, \vec{\alpha}^D_\mu\}$.

These probability distributions is specified as a *discriminative model* which directly models the posterior distribution $P(z|\mathbf{I})$:

$$P(z|\mathbf{I}) = \frac{1}{Z(\alpha, \mathbf{I})} \exp\{-\sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)}) - \sum_{\mu \in V} \vec{\alpha}^D_\mu \cdot \vec{\phi}^D_\mu(\mathbf{I}, D(z_\mu))\}. \quad (3)$$

It is important to realize that this discriminative model includes an *explicit* prior on the state z given by the $\sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)})$ term in the exponent in equation (3). This is obtained by applying Bayes rule $P(z|\mathbf{I}) = P(\mathbf{I}|z)P(z)/P(\mathbf{I})$ and identifying the components of $P(z|\mathbf{I})$ which depend on $\mathbf{I}$ as $P(\mathbf{I}|z)/P(\mathbf{I})$ and those which depend only on z as $P(z)$ (up to an unknown normalization constant). Hence an HDT has a prior distribution on the hidden states, specifying a distribution on the relative geometry of the subparts, together with a discriminative model for how the subparts interact with the image (specified by the terms parameterized by the α^D). We use discriminative terms for how the HDT interacts with the image for two main reasons: (i) it is far easier to learn discriminative models for intensities rather than generative ones (e.g. we can use AdaBoost to discriminate between the interiors and backgrounds of cows and horses, but there are no generative models that can realistically synthesize the intensity properties of cows and horses [33], (ii) it is easier to learn discriminative models than generative ones (because of the difficulties of dealing with the normalization factors).

We now describe the terms in more detail. The *data terms* $\phi^D(\mathbf{I}, z)$ contain the appearance terms which indicate how the HDT interacts with the image. The *prior terms* $\phi(z_\mu, z_{ch(\mu)})$ are decomposed into *vertical terms* indicating how the state of the parent node relates to its children and *horizontal terms* defined on triplets of child nodes.

The *data terms* $\phi_\mu^D(\mathbf{I}, D(z_\mu))$ are defined in terms of a dictionary of potentials computed from image features. More precisely, the potentials are of form $\phi^D(\mathbf{I}, D(z_\mu)) = \log \frac{P(F(\mathbf{I}, D(z_\mu))|object)}{P(F(\mathbf{I}, D(z_\mu))|background)}$ where $F(\mathbf{I}, D(z_\mu))$ is the feature response from the region in the image $\mathbf{I}$ specified by $D(z_\mu)$. The distributions $P(F(\mathbf{I}, D(z_\mu))|object)$ and $P(F(\mathbf{I}, D(z_\mu))|background)$ are either histogram distributions, or uni-variate Gaussians, measured when z_μ is in the correct location (object) or on the background, see subsection (V) for more details. We use different features dictionaries at different levels of the hierarchy, see figure (2). For leaf nodes, $\mu \in V^{LEAF}$ the $\phi_\mu^D(\mathbf{I}, D(z_\mu))$ are specified by a dictionary of local image features $F(\mathbf{I}, D(z_\mu))$ computed by different operators – there are 27 features in total including the intensity, the intensity gradient, Canny edge detectors, Difference of Offset Gaussian (DOOG) at different scales (13*13 and 22*22) and orientations $(0, \frac{1}{6}\pi, \frac{2}{6}\pi, ...)$, and so on (see bottom row of figure 2). For non-leaf nodes, $\mu \in V/V^{LEAF}$, the $\phi_\mu^D(\mathbf{I}, D(z_\mu))$ are specified by a dictionary of regional features (e.g. mean, variance, histogram of image features) defined over the sub-regions $D(z_\mu)$ specified by the node state z_ν, see the second row of the right panel of figure (2).

The *prior terms* $\phi(z_\mu, z_{ch(\mu)})$ are decomposed into *horizontal terms* and *vertical terms*. The horizontal terms are defined for each triplet of child nodes of μ, see figure (3). I.e., for each triplet (ν, ρ, τ) such that $\nu, \rho, \tau \in ch(\mu)$ we specify the *invariant shape vector* [32] $l(z_\mu, z_\rho, z_\tau)$ and define $\phi^H(z_\nu, z_\rho, z_\tau)$ to be the Gaussian potential (i.e., the first and second order statistics). Recall that the ITV [32] depends only on functions of z_ν, z_ρ, z_τ, such as the internal angles, which are invariant to the translation, rotation, and scaling of the triple. This ensures that the potential is also invariant to these transformations. The parameters of the Gaussian are learnt from training data as described in section (V). The *vertical terms* $\phi^V(z_\mu, z_{ch(\mu)})$ are used to hold the structure together by relating the state of the parent nodes to the state of their children. The state of the parent node is determined precisely by the states of the child nodes. This is defined by $\phi^V(z_\mu, z_{ch(\mu)}) = h(z_\mu, z_{ch(\mu)})$, where $ch(\mu)$ is the set of child nodes of node μ, $h(.,.) = 0$ if the average orientations and positions of the child nodes are equal to the orientation and position of the parent node. If they are not consistent, then $h(.,.) = \kappa$, where κ is a large positive number.

In summary, the HDT models the appearance and the shape (geometry) at multiple levels.

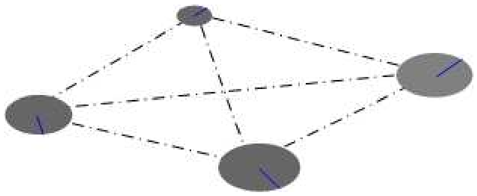

Fig. 3. Representation based on oriented triplet. This gives the cliques for the four children of a node. In this example, four triplets are computed. Each circle corresponds to one node in the hierarchy which has a descriptor (indicated by blue line) of position, orientation and scale. The potentials of the cliques are Gaussians defined over features extracted from triple nodes, such as internal angles of the triangle and relative angles between the feature orientation and the orientations of the three edges of the triangle. These exacted features are invariant to scale and rotation.

Low-levels of the hierarchy model short range shape constraints and local appearance cues, see the third and fourth rows of figure (2). At higher levels – the top and second rows of figure (2) – longer range shape constraints and large scale appearance cues are used.

C. Constructing the Hierarchy by One-example Learning

In this paper, we learn the hierarchical graph from a single example of the object. We call this “one-example learning”. The input is the set $\{(\vec{x}, \theta)\}$ of points on the object boundary curve together with their orientation (i.e. the normal vector to the curve). On this set we specify 24 points corresponding to leaf nodes of the HDT spaced evenly along the boundary (this specification will be used to determine ground-truth during for structure-perceptron learning). The output is the graph structure with initial values of the parameters α (with many set to zero) which gives a *default HDT* that can be used to initialize the structure perceptron learning.

We automatically construct the hierarchical graph by a hierarchical aggregation algorithm which is partially inspired by the “Segmentation by Weighted Aggregation (SWA) ” algorithm [34]. The input is a weighted graph $G = \{V, E, W\}$ where V is the vertex set (the 24 points on the boundary), E is the edge set (the edges are defined by the neighboring points on the contour), and W specifies the weights: $w_{i,j} = \exp\{-\beta_1 dist(\vec{x}_i, \vec{x}_j) + \beta_2 edge(\vec{x}_i, \vec{x}_j)\}$ where $\vec{x}_i$ is the position of point i, $dist(.,.)$ is the distance function and $edge(.,.)$ is an indicator function to measure if point i, j are neighbors or not. β_1 and β_2 are set to be 0.5 and 1 respectively (in all experiments). The algorithm proceeds by iterating through the vertex points and assigning them to clusters based on affinity, so that a new cluster is created if the affinity of the vertex to the current clusters is above threshold. Affinities are computed between the clusters and the procedure repeats using the clusters as the new vertices. See [34] for details.

The output is a hierarchical graph structure (the state variables of the example are thrown out), i.e. a set of nodes and there vertical and horizontal connections. Observe that the hierarchical graph gives a natural parsing of the exemplar – see figure (2).

After one-example learning, we specify a default HDT which will be used to initialize the parameter learning in section (V). We set $\alpha = 0$ for all data terms except those at the leaf nodes. At the leaf nodes we set $\alpha = 1$ for the data terms corresponding to the intensity gradients (we learn the distributions $P(.|object)$ and $P(.|background)$ for the intensity gradient from the responses on and off the object boundary). We set $\alpha = 1$ for the vertical and horizontal terms. For the horizontal terms we set $\phi^H(y)$ to be the Gaussian distribution of the invariant triplet vector $g(\vec{l}(z_\mu, z_\rho, z_\tau))$ where the mean is measured from the example and the covariance is set by hand (to 0.12 times the identity matrix for all levels). This is just the covariance for the default HDT used for initialization. The covariance will be learnt by the HDT by structure-perceptron.

IV. Inference: Parsing the Model

We now describe an inference algorithm suited to the hierarchical structure of the HDT. Its goal is to obtain the best state z^* by estimating $z^* = \arg\max P(z|\mathbf{I})$ which can be re-expressed in terms of minimizing an energy function:

$$z^* = \arg\min\{\sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)}) + \sum_{\mu \in V} \vec{\alpha}^D_\mu \cdot \vec{\phi}^D_\mu(\mathbf{I}, D(z_\mu)).\} \tag{4}$$

To perform inference, we observe that the hierarchical structure of the HDT, and the lack of shared parents (i.e., the independence of different parts of the tree), means that we can express the energy function recursively and hence find the optimum z using dynamic programming in polynomial time in the size of the graph G and the state space of the $\{z_\mu\}$. But the state space of the $\{z_\mu\}$ is very large since every component of the object can occur in any position of the image, at any orientation, and any scale. Hence, as in other applications of DP or BP to vision [1], [2] we need to perform pruning to reduce the set of possible states.

The algorithm is called *compositional inference* [11][12], see table (5). It is a pruned form of dynamic programming (DP) that exploit the independence structure of the graph model. It is run bottom-up starting by estimating possible states for the leaf nodes and proceeding to estimate possible states for the nodes higher up the tree (a top-down stage is sometimes run as a variant). Although DP guaranteed to be polynomial in the relevant quantities (number of layers and graph nodes, size of state space of z) full DP is too slow because of the large size of the state space (i.e. range of values that z_μ can take for each node μ). The pruning reduces the allowable states of a node μ to a set of *proposals* (borrowing terminology from the MCMC literature) together with their energies. These proposals are selected by two mechanisms: (i) *energy pruning* - to remove proposals corresponding to large energy, and (ii) *surround suppression* - to suppress proposals that occur within a surrounding window (similar to non-maximal suppression).

The intuition for compositional inference is that it starts by detecting possible configurations of the elementary (low-level) components of the HDT and combines them to produce possible configurations of high-level components, see figure (4).

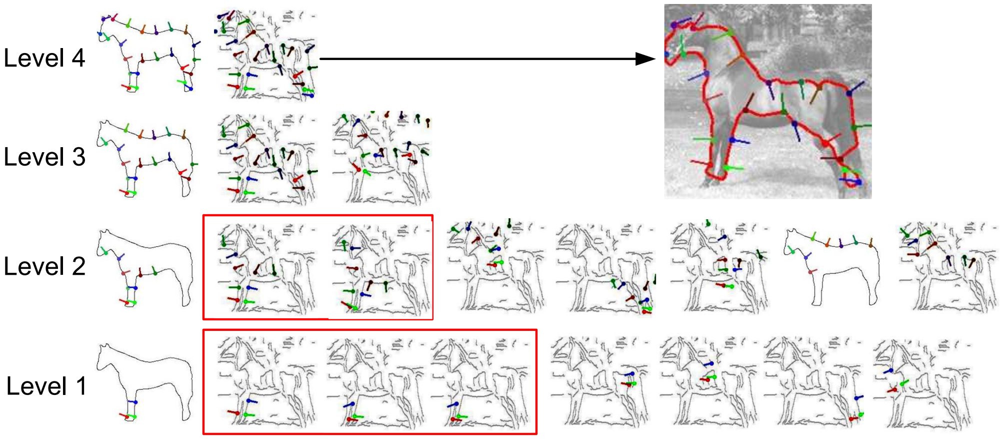

Fig. 4. This snapshot illustrates the compositional inference algorithm. The algorithm proceeds by combining proposals for the states of child nodes to form composite proposals for the states of parent nodes. These composite proposals are pruned to remove those whose energies are too large and then surround suppression is applied (within a window in space, orientation, and scale) so that only locally maximal proposals are accepted.

The pruning threshold, and the window for surround suppression, must be chosen to that there are very few false negatives (i.e. the object is always detected as, at worst, a small variant of one of the proposals). In practice, the window is $(5, 5, 0.2, \pi/6)$ (i.e., 5 pixels in the x and y directions, up to a factor of 0.2 in scale, and $\pi/6$ in orientation – same for all experiments). But rapid inference is achieved by keeping the number of proposals small (avoiding the danger of

```
Input: $\{p^1_{\nu^1}\}$. Output:$\{p^L_{\nu^L}\}$
  • Bottom-Up($p^1$)
    Loop : $l = 1$ to $L$, for each node $\nu$ at level $l$
      1) Composition: $\{p^l_{\nu,b}\} = \oplus_{\rho \in ch(\nu), a=1,\dots,m^{l-1}_\rho} p^{l-1}_{\rho,a}$
      2) Pruning: $\{p^l_{\nu,a}\} = \{p^l_{\nu,a} | \phi_\nu(\mathbf{I}, z^l_{\nu,a}) \cdot \alpha > T_l\}$
      3) Local Maximum: $\{(p^l_{\nu,a}\} = LocalMaximum(\{p^l_{\nu,a}\}, \epsilon_W)$ where $\epsilon_W$ is the size
         of the window $W^l_\nu$ defined in space, orientation, and scale.
```

Fig. 5. The inference algorithm. $\oplus$ denotes the operation of combining proposals from child nodes to make proposals for parent nodes.

combinatorial explosion due to composition). We performed experiments to balance the trade-off between performance and computational speed. Our experiments show that the algorithm has linear scaling with image size, as shown in section (VI), and we empirically quantify the performance of each component of the hierarchy in section (VI-C).

We now specify compositional inference more precisely by first specifying how to recursively compute the energy function – which enables dynamical programming – and then describe the approximations (energy pruning and surround suppression) made in order to speed up the algorithm without decreasing performance.

Recursive Formulation of the Energy The discriminative model, see equation (3), is of Gibbs form and can be specified by an energy function: $E(z|\mathbf{I}) = \sum_{\mu \in V} \vec{\alpha}_\mu \cdot \vec{\phi}(z_\mu, z_{ch(\mu)}) + \sum_{\mu \in V} \vec{\alpha}^D_\mu \cdot \vec{\phi}^D_i(\mathbf{I}, D(z_\mu))$. We exploit the tree-structure and express this energy function recursively by defining an energy function $E_\nu(z_{des(\nu)}|\mathbf{I})$ over the subtree with root node ν in terms of the state variables $z_{des(\nu)}$ of the subtree where $des(\nu)$ stands for the set of descendent nodes of ν – i.e. $z_{des(\nu)} = \{z_\mu : \mu \in V_\nu\}$.

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

Compositional Inference. *Initialization*: at each leaf node $\nu \in V^{LEAF}$ we calculate the states $\{p_{\nu,b}\}$ (b indexes the proposal) such that $E_\nu(p_{\nu,b}|\mathbf{I}) < T$ (*energy pruning* with threshold T) and $E_\nu(p_{\nu,b}|\mathbf{I}) \leq E_\nu(p_\nu|\mathbf{I})$ for all $z_\nu \in W(p_{\nu,b})$ (*surround suppression* where $W(p_{\nu,b})$ is a window centered on $p_{\nu,b}$). (The window $W(p_{\nu,b})$ is $(5, 5, 0.2, \pi/6)$ centered on $p_{\nu,b}$). We refer to the $\{p_{\nu,b}\}$ as proposals for the state z_ν and store them with their energies $E_\nu(p_{\nu,b}|\mathbf{I})$. ***Recursion for parent nodes***: to obtain the proposals for a parent node μ at a higher level of the graph $\mu \in V/V^{LEAF}$ we first access the proposals for all its child nodes $\{p_{\mu_i,b_i}\}$ where $\{\mu_i : i = 1, ..., |ch(\mu)|\}$ denotes the set of child nodes of μ and their energies $\{E_{\mu_i}(p_{\mu_i,b_i}|\mathbf{I}) : i = 1, ..., |ch(\mu)|\}$. Then we compute the states $\{p_{\mu,b}\}$ such that $E_\mu(p_{\mu,b}|\mathbf{I}) \leq E_\mu(z_\mu|\mathbf{I})$ for all $z_\mu \in W(p_{\mu,b})$ where $E_\mu(p_{\mu,b}|\mathbf{I}) = min_{\{b_i\}}\{\sum_{i=1}^{|ch(\mu)|} E_{\mu_i}(z_{des(\mu_i,b_i)}|\mathbf{I}) + \vec{\alpha}_\mu \cdot \vec{\phi}(p_{\mu,b}, \{p_{\mu_i,b_i}\}) + \vec{\alpha}_\mu^D \cdot \vec{\phi}^D(\mathbf{I}, D(p_{\mu,b}))\}$. In our experiments, the thresholds T are set to take values such that the recall in the training data is 95%. In other words, for all object parts corresponding to the nodes in the hierarchy, 95% of training examples are correctly detected by using the thresholds to prune out proposals.

V. Structure-Perceptron Learning

We now describe our parameter learning algorithm. This constructs the HDT probability distribution by selecting and weighting features from the dictionaries. Recall that the graph structure of the HDT has already been learnt from one example by the hierarchical clustering algorithm and a default model has been specified, see subsection (III-C). We now have a training dataset where the boundary is specified. We hand-specify points on the boundaries of the object (24 points for the horses and cows) using a template to ensure consistency (i.e., that the points correspond to similar parts of the object on all training images). This specifies the ground-truth for all the state variables of the HDT because the states of the parent nodes are determined by the states of their child nodes (see section (III-B)). This enables us to learn the distributions $P(F(\mathbf{I}, D(z_\mu))|object)$ and $P(F(\mathbf{I}, D(z_\mu))|background)$ and hence determine the data potentials ϕ^D (recall that the horizontal and vertical potentials are specified by the default HDT). This the remaining task is to estimate the α's described in section (III-B).

A. Background on Perceptron and Structure-Perceptron Learning

Perceptron learning was developed in the 1960's for classification tasks (i.e., for binary-valued output) and its theoretical properties, including convergence and generalization, have been studied [35]. More recently, Collins [13] developed the structure-perceptron algorithm which applies to situations where the output is a structure (e.g. a sequence or a tree of states). He obtained theoretical results for convergence, for both separable and non-separable cases, and for generalization. In addition Collins and his collaborators demonstrated many successful applications of structure-perceptron to natural language processing, including tagging [36] (where the output is sequence/chain), and parsing [37] (where the output is a tree).

Structure-perceptron learning can be extended to learning the parameters of HDTs. The learning proceeds in a discriminative way. By contrast to maximum likelihood learning, which requires calculating the expectation of features, structure-perceptron learning only needs to calculate the energies of the state variables. Hence structure-perceptron learning is more flexible and computationally simpler.

To the best of our knowledge, structure-perceptron learning has never been exploited in computer vision except our previous work [15] (unlike the perceptron which has been applied to binary classification and multi-class classification tasks). Moreover, we are applying structure-perceptron to more complicated models (i.e. HDTs) than those treated by Collins [36] (e.g. Hidden Markov Models for tagging).

B. Details of Structure-Perceptron Learning

The goal of structure-perceptron learning is to learn a mapping from inputs $\mathbf{I} \in \mathcal{I}$ to output structure $z \in \mathcal{Z}$. In our case, $\mathcal{I}$ is a set of images, with $\mathcal{Z}$ being a set of possible parse trees (i.e. configuration of HDT's) which specify the positions, orientations, scales of objects and their subparts in hierarchical form. We use a set of training examples $\{(\mathbf{I}_i, z_i) : i = 1...n\}$ and a dictionary of functions/potentials $\{\phi\}$ which map each $(\mathbf{I}, z) \in \mathcal{I} \times \mathcal{Z}$ to a feature vector $\phi(\mathbf{I}, z) \in R^d$. The task is to estimate a parameter vector $\alpha \in R^d$ for the weights of the features. This can be interpreted as a soft form of feature selection so that unimportant features have small weights. The feature vectors $\phi(\mathbf{I}, z)$ can include arbitrary features of parse trees, as we discussed in section (III-A).

The loss function used in structure-perceptron learning is of form:

$$Loss(\alpha) = \phi(\mathbf{I}, z) \cdot \alpha - \max_{\overline{z}} \phi(\mathbf{I}, \overline{z}) \cdot \alpha, \tag{5}$$

where z is the correct state configuration for input $\mathbf{I}$, and $\overline{z}$ is a dummy variable. (Here $\phi(\mathbf{I}, z)$ denotes all the potentials of the model and α denotes all the parameters).

The basic structure-perceptron algorithm – *Algorithm I* – is designed to minimize the loss function. Its pseudo-code is given in figure (6). The algorithm proceeds in a simple way (similar to the perceptron algorithm for classification). The HDT is initialized by the default model (i.e., $\alpha = 1$ for the vertical, horizontal, and leaf node intensity terms and $\alpha = 0$ for the other data terms). Then the algorithm loops over the training examples. If the highest scoring parse tree for input $\mathbf{I}$ is not correct, then the parameters α are updated by an additive term. The most difficult step of the method is to find $z^* = \arg\max_z \phi(\mathbf{I}^i, z)\cdot\alpha$. But this can be performed by the inference algorithm described in section (V). Hence the performance and efficiency (empirically polynomial complexity) of the inference algorithm is a necessary pre-condition to using structure-perceptron learning for HDTs.

```
Input: A set of training images with ground truth $(\mathbf{I}^i, z^i)$ for $i = 1..N$. Initialize the parameter vector $\alpha$ by the default model.
Algorithm I:
For $t = 1..T, i = 1..N$
    • Use bottom-up inference to find the best state of the model on the i'th training image with current parameter setting, i.e., $z^* = \arg\max_z \phi(\mathbf{I}^i, z) \cdot \alpha$
    • Update the parameters: $\alpha = \alpha + \phi(\mathbf{I}^i, z^i) - \phi(\mathbf{I}^i, z^*)$
Output: Parameters $\alpha$
```

Fig. 6. Algorithm I: a simple training algorithm of structure-perceptron learning

```
Algorithm II:
For $t = 1..T, i = 1..N$
    • Parse: $z^* = \arg\max_z \phi(\mathbf{I}^i, z) \cdot \alpha$
    • Store: $\alpha^{t,i} = \alpha$
    • Update: $\alpha = \alpha + \phi(\mathbf{I}^i, z^i) - \phi(\mathbf{I}^i, z^*)$
Output: Parameters $\gamma = \sum_{t,i} \alpha^{t,i}/NT$
```

Fig. 7. Algorithm II: a modification of Algorithm I with same training images and initialization.

C. Averaging Parameters

There is a simple refinement to Algorithm I, called *"the averaged parameters"* method (Algorithm II) [13], whose pseudo-code is given in figure (7). The averaged parameters are defined to be $\gamma = \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha^{t,i}/NT$, where NT is the averaging window. It is straightforward to store these averaged parameters and output them. The theoretical analysis in [13] shows that Algorithm II (with averaging) gives better performance and convergence rate than Algorithm I (without averaging). We will empirically compare these two algorithms in section (VI).

D. Soft Feature Selection

Structure-perceptron learning uses a dictionary of features $\{\phi\}$ with parameters $\{\alpha\}$ initialized by the default HDT (after one-example learning). As the algorithm proceeds, it assigns weights to the features so that more important features receive larger weights. This can be thought of as form of "soft" feature selection (by contrast to the "hard" feature selection performed by algorithms like AdaBoost). This ability to perform soft feature selection allows us to specify a large dictionary of possible features and enable the algorithm to select those features which are most effective. This allows us to learn HDTs for different objects *without* needing to specially design features for each object.

This ability to softly select features from a dictionary means that our approach is more flexible than existing conditional models (e.g., CRF [28], [29], [38]) which use multi-level features but with fixed scales (i.e. not adaptive to the configuration of the hidden state). In section (VI-E), we empirically study what features the structure-perceptron algorithm judges to be most important for a specific object like a horse. Section (VI-F) also illustrates the advantage of soft feature selection by applying the same learning algorithm to the different task of face alignment without additional feature design.

VI. Experimental Results

A. Dataset and Evaluation Criterions

Dataset. We evaluate HDT for different tasks on different public datasets. Firstly, we use two standard public datasets, the Weizmann Horse Dataset [16] and cows [17], to perform experimental evaluations for HDTs. See some examples in figure (8). These datasets are designed to evaluate segmentation, so the groundtruth only gives the regions of the object and the background. To supplement this groundtruth, we asked students to manually parse the images by locating the states of leaf nodes of the hierarchy in the images which deterministically specifies the states of the nodes of the remainder of the graph (this is the same procedure used to determine

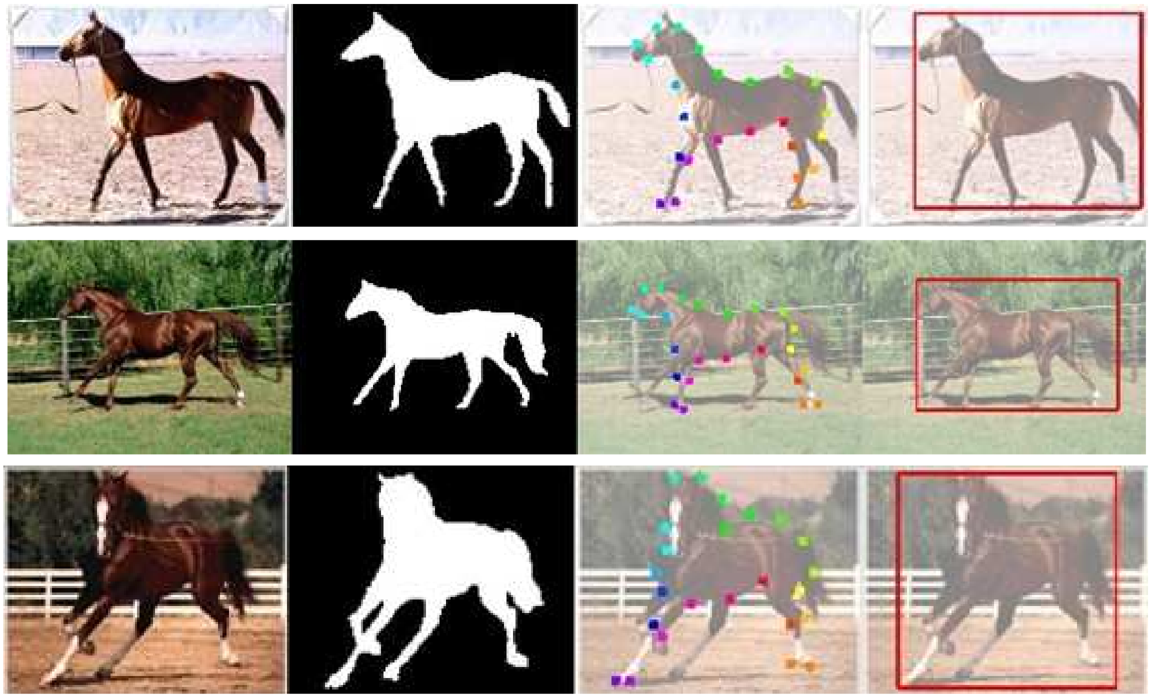

Fig. 8. Examples of the Weizmann horse data set. This figure shows input image, ground truth of segmentation, parsing (position of leaf nodes) and detection, from left to right respectively.

groundtruth for learning, see section (V)). These parse trees are used as ground truth to evaluate the ability of the HDT to parse the horses (i.e. to identify different parts of the horse).

Secondly, to show the generality and versatility of our approach and its ability to deal with different objects without hand-tuning the appearance features, we apply it to the task of face alignment (this requires parsing). We use a public dataset [18] which contains the groundtruth for 65 key points which lie along the boundaries of face components with semantic meaning, i.e, eyes, nose, mouth and cheek. We use part of this dataset for training (200 images) and part for testing (80 images).

The measure for parsing/alignment. For a given image $\mathbf{I}$, we apply the HDT to parse the image and estimate the configuration z. To evaluate the performance of parsing (for horses) and matching/alignment (for faces) we use the **average position error** measured in terms of pixels. This quantifies the average distance between the positions of leaf nodes of the ground truth and those estimated in the parse tree.

The measure for segmentation. The HDT does not directly output a full segmentation of the object. Instead the set of leaf nodes gives a sparse estimate for the segmentation. To enable HDT to give full segmentation we modify it by a strategy inspired by grab-cut [14] and obj-cut [27]. We use a rough estimate of the boundary by sequentially connecting the leaf nodes of the HDT, to initialize a grab-cut algorithm (recall that standard grab-cut [14] requires human initialization, while obj-cut needs motion cues). We use **segmentation accuracy** to quantify the proportion of the correct pixel labels (object or non-object). Although segmentation accuracy is widely used as a measure for segmentation, it has the disadvantage that it depends on the relative size of the object and the background. For example, you can get 80% segmentation accuracy on the weizmann horse dataset by simply labelling every pixel as background. Therefore, to overcome the shortcoming of segmentation accuracy, we also report **precision/recall**, see [28], where $precision = \frac{P \cap TP}{P}$ and $recall = \frac{P \cap TP}{TP}$ (P is the set of pixels which are classifier as object by HDT and TP is the set of object pixels in ground truth). We note that segmentation accuracy is commonly used in the computer vision community, while precision/recall is more standard in machine learning.

The measure for detection. We use **detection rate** to quantify the proportion of successful detections. We rate *detection* to be successful if the area of intersection of the labeled object region (obtained by graph-cut initialized by the HDT) and the true object region is greater than half the area of the union of these regions.

The measure for performance analysis. We judge that an object(or part) is correctly parsed if each subpart (i.e. the location of each node in the hierarchy) is located close (within $k_1 \times l + k_2$ pixels where l is the level with $k_2 = 5$ and $k_1 = 2.5$) to the ground-truth. The thresholds in the distance measure vary proportionally to the height of levels so that the distance is roughly normalized according to the size of object parts. We plot the **precision-recall curve** to study the performance of the components of the whole model.

B. *Experiment I: One-example Learning*

We first report the performance of the HDT with one-example learning. The two exemplars used to obtain the horse and cow hierarchies are shown in figure (9). We use identical parameters for each model (i.e. for the hierarchical aggregation algorithm, for the data terms, and the horizontal and vertical terms, for the proposal thresholds and window sizes).

We illustrate the segmentation and parsing results in figure (10). Observe that the algorithm is

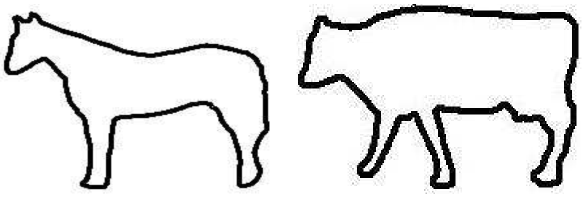

Fig. 9. This shows the exemplars used for the horse (left) and the cow (right).

Dataset	Size	Detection Rate	Parsing (Average Position Error)	Segmentation Precision/Recall	Speed
Horse	328	86.0	18.7	81.3% /73.4%	3.1s
Cow	111	88.2	15.8	81.5% /74.3%	3.5s

TABLE I

THE PERFORMANCE OF THE DEFAULT HDT PROVIDED BY ONE-EXAMPLE LEARNING.

successful even for large changes in position, orientation and scale – and for object deformations and occlusion. The evaluation results for detection, parsing, and segmentation are shown in table (I). Overall, the performance is very good and the average speed is under 4 seconds for an image of 320×240.

C. Experiment II: Contributions of Object Parts: Complexity and Performance Analysis

We use the default model provided by one-example learning to analyze the effectiveness of different components of the HDT in terms of performance and time-complexity. This is shown in table (II) and figure (11). We anticipate that this analysis of the tradeoffs between speed and performance will yield general principles for optimal design of modeling and inference for computer vision systems particularly those requiring multi-level processing.

Performance Contributions of Multi-level Object Parts. Figure (11) shows how different components of the hierarchy contribute to performance. It is easy to note that smaller object parts have worse performance in terms of precision-recall. More high-level knowledge including both appearance and shape prior makes object parts more distinct from background and thus improves the overall performance. One can see that there is a jump in performance when we move from level 2 to level 3, indicating that the information at level 3 (and below) is sufficient

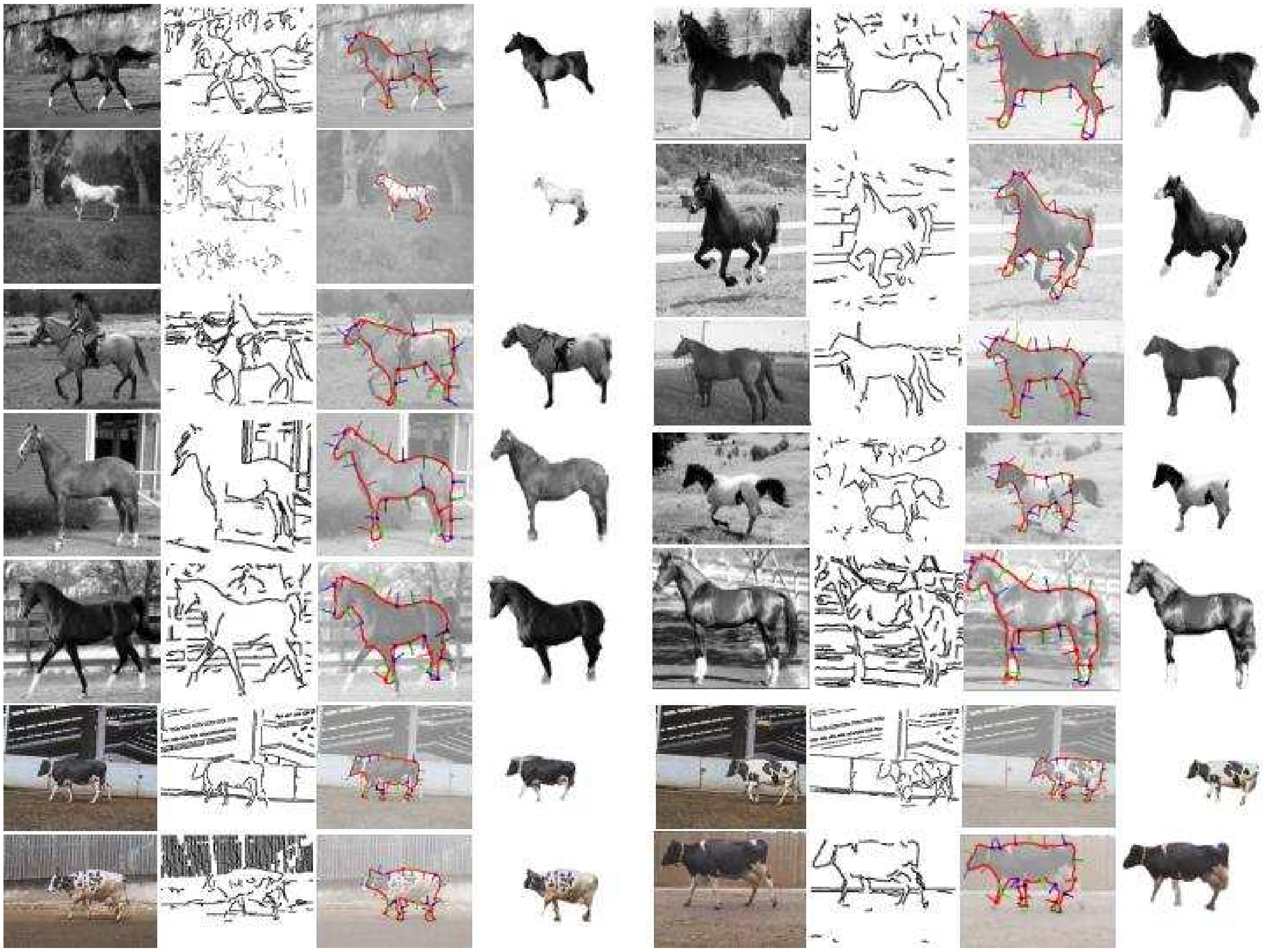

Fig. 10. Segmentation and parsing results on the horse and cows datasets using the default HDT obtained by one-example learning. The first column shows the raw images. The second one show the edge maps. The third one shows the parsed result. The last one shows the segmentation results. The main errors are at the head and legs due to their large variability which may require a model with OR nodes, see [39].

to disambiguate the object from a cluttered background.

Computational Complexity Analysis. Table (II) shows that the number of proposals scales almost linearly with the level in the hierarchy, and the time cost for each level is roughly constant. This demonstrates that the pruning and surround suppression are important factors for making bottom-up processing effective. Overall, this helps understand the effectiveness of the bottom-up processing at different levels.

	Number of Proposals / Node	Time/Node	Time/Img
Level 4	51	0.14s	0.14s
Level 3	77	0.29s	0.88s
Level 2	105	0.21s	1.05s
Level 1	158	0.10s	1.22s
Level 0	225	0.01s	0.18s
Hierarchy (average)	180	0.08s	3.47s

TABLE II

ANALYSIS OF COMPOSITIONAL INFERENCE. LEFT PANEL: THE NUMBERS OF PROPOSALS AT FOR EACH NODES AT DIFFERENT LEVELS (AFTER ENERGY PRUNING AND SURROUND SUPPRESSION). CENTER PANEL: THE TIME COSTS FOR EACH NODES. RIGHT PANEL: THE TIME COSTS FOR THE IMAGE (AVERAGED OVER THE NODES OF ALL THE LEVELS OF THE HIERARCHY).

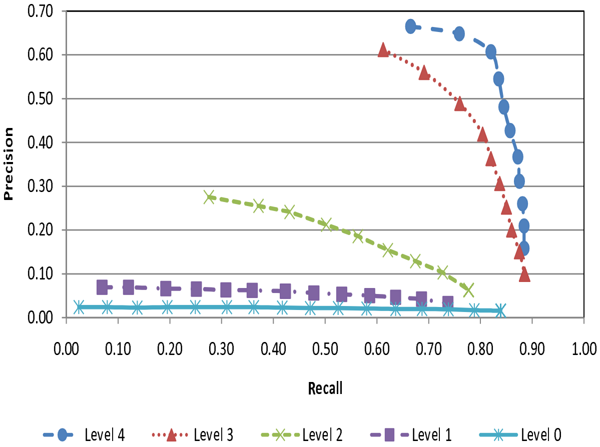

Fig. 11. This figure shows the Precision-Recall curves for different levels. Level 4 is the top level. Level 0 is the bottom level.

D. Experiment III: Evaluations of Structure-Perceptron Learning for Deformable Object Detection, Segmentation and Parsing

In this experiment, we apply structure-perceptron learning to include all image features for the leaf nodes and non-leaf nodes, and estimate the parameters α. The hierarchical structure is obtained by one-example learning. We use the Weizeman horse dataset [16] for evaluation where a total of 328 images are divided into three subsets – 50 for training, 50 for validation,

TABLE III

COMPARISONS OF ONE-EXAMPLE LEARNING AND STRUCTURE-PERCEPTRON LEARNING

Learning Approaches	Training	Validation	Detection	Parsing	Segmentation (Precision/Recall)	Speed
One-example learning	1	–	86.0 %	18.7	81.3% / 73.4%	3.1s
Structure-perceptron learning	50	50	99.1%	16.04	93.6% / 85.3%	23.1s

TABLE IV

COMPARISONS OF SEGMENTATION PERFORMANCE ON WEIZMANN HORSE DATASET

Methods	Testing	Seg. Accu.	Pre./Rec.
Our approach	228	94.7%	93.6% / 85.3%
Ren [28]	172	91.0%	86.2%/75.0%
Borenstein [40]	328	93.0%	
LOCUS [30]	200	93.1%	
Cour [31]	328	94.2%	
Levin [29]	N/A	95.0%	
OBJ CUT [27]	5	96.0%	
Grabcut (bounding box init.)	N/A	83.3%	

and 228 for testing. The parameters learnt from the training set, and with the best performance on validation set, are selected.

Results. The best parse tree is obtained by performing inference algorithm over HDT learnt by structure-percepton learning. Figure 12 shows several parsing and segmentation results. The states of the leaf nodes of parse tree indicate the positions of the points along the boundary which are represented as colored dots. The points of same color in different images correspond to the same semantic part. One can see our model's ability to deal with shape variations, background noise, textured patterns, and changes in viewing angles. The performance of detection and parsing on this dataset is given in Table III. Structure-perceptron learning which include more visual cues outperforms one-example learning in all tasks. The localization rate is around 99%. Our model performs well on the parsing task since the average position error is only 16 pixels (to give context, the radius of the color circle in figure 12 is 5 pixels). Note no other papers report parsing performance on this dataset since most (if not all) methods do not estimate the positions of different parts of the horse (or even represent them). The time of inference for image with typical size 320×240 is 23 seconds.

Comparisons. In table IV, we compare the segmentation performance of our approach with other successful methods. Note that the object cut method [27] was reported on only 5 images. Levin and Weiss [29] make the strong assumption that the position of the object is given (other methods do not make this assumption) and not report how many images they tested on. Overall, Cour and Shi's method [31] was the best one evaluated on large dataset. But their result is obtained by manually selecting the best among top 10 results (other methods output a single result). By contrast, our approach outputs a single parse only but yields a higher pixel accuracy of 94.7%. We put in results of Grabcut using the groundtruth bounding box as initialization to illustrate the big advantage of using HDT to initialize grabcut. Hence we conclude that our approach outperforms those alternatives which have been evaluated on this dataset. As described above, we prefer the precision/recall criteria [28] because the segmentation accuracy is not very distinguishable (i.e. the baseline starts at 80% accuracy, obtained by simply classifying every image pixel as being background). Our algorithm outperforms the only other method evaluated in this way (i.e. Ren et al.'s [28]). For comparison, we translate Ren et al.'s performance (86.2%/75.0%) into segmentation accuracy of 91% (note that it is impossible to translate segmentation accuracy back into precesion/recall).

E. Experiment IV: Diagnosis of structure-perceptron learning

In this section, we will conduct diagnosis experiments to study the behavior of structure-perceptron learning.

Convergence Analysis. Figure 13 shows the average position error on training set for both Algorithm II (averaged) and Algorithm I (non-averaged). It shows that the averaged algorithm converges much more stablely than non-averaged algorithm.

Generalization Analysis. Figure 14 shows average position error on training, validation and testing set over a number of training iterations. Observe that the behavior on the validation set and the testing set are quite similar. This confirms that the selection of parameters decided by the validation set is reasonable.

Soft Feature Selection. Structure-perceptron effectively performs soft feature selection by

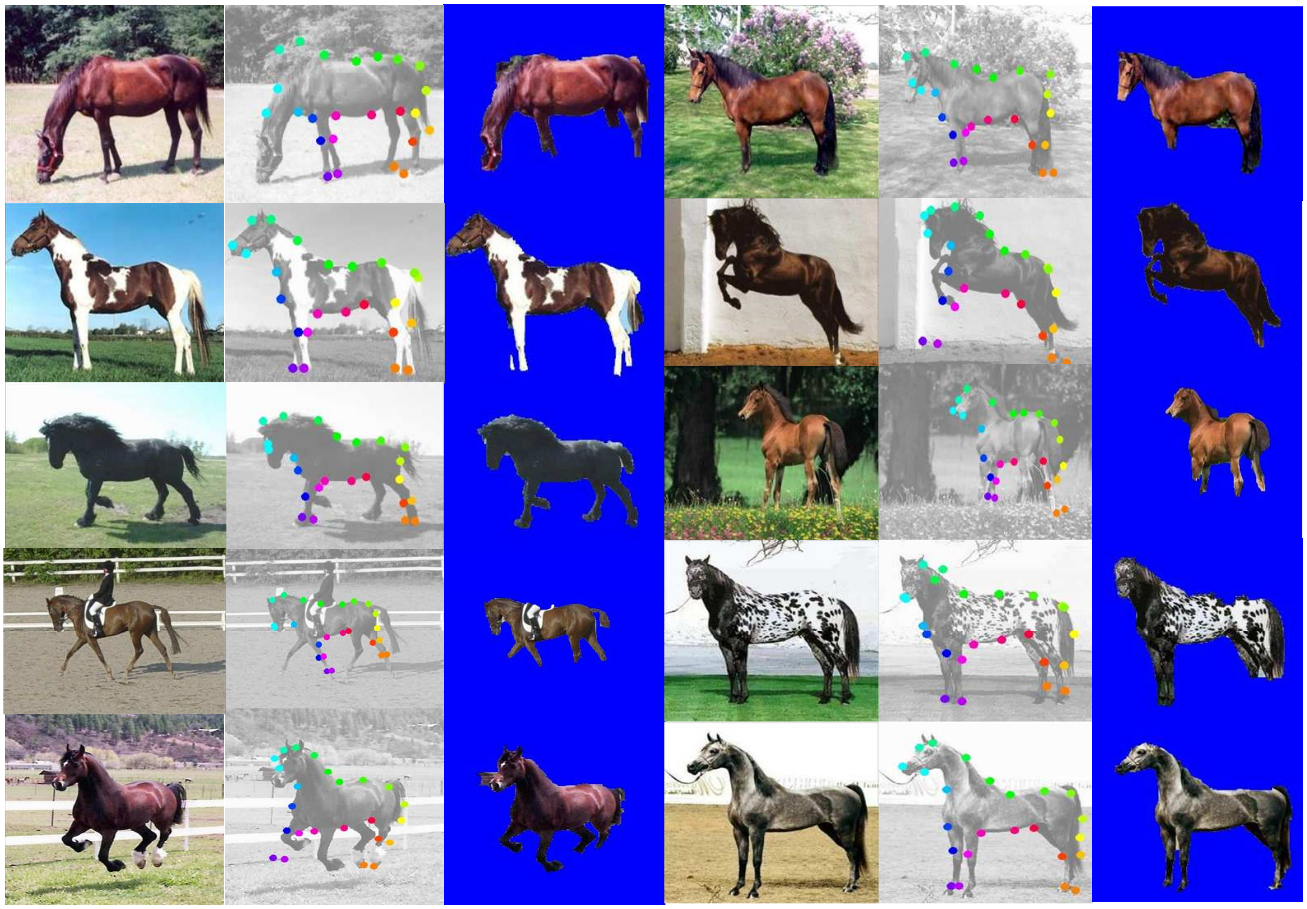

Fig. 12. Examples of Parsing and Segmentation. Column 1 , 2 and 3 show the raw images, parsing and segmentation results respectively. Column 4 to 6 show extra examples. Parsing is illustrated by dotted points which indicate the positions of leaf nodes (object parts). Note that the points in different images with the same color correspond to the same semantical part. As for the HDT default model, the main errors are at the head and legs due to their large variability which may require a model with OR nodes, see [39]

assigning low values of the weights α to many features/potentials, see figure (15). This enables us to specify large dictionaries of features/potentials and allow structure-perceptron to select which ones to use. We illustrate the types of features/potentials that structure-perceptron prefers in figure (16) (we only show the features are shown at the bottom level of the hierarchy for reasons of space). The top 5 features, ranked according to their weights, are listed. The top left, top right and bottom left panels show the top 5 features for all leaf nodes, the node at the back of horse and the node at the neck respectively. Recall that structure-perceptron learning performs soft feature selection by adjusting the weights of the features.

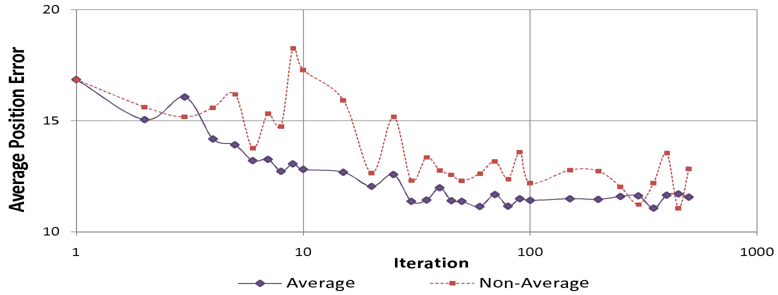

Fig. 13. The average position errors (y-axis) across iterations (x-axis) are compared between Algorithm-II(average) and Algorithm-I (non-average).

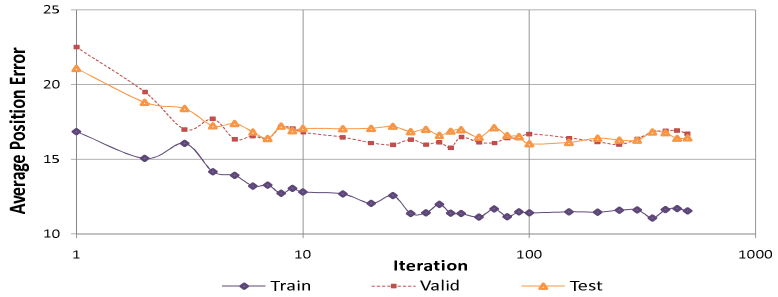

Fig. 14. The average positions errors on training, validation and testing dataset are reported.

F. Experiment V: Multi-view Face Alignment

To demonstrate the versatility of HDTs we applied them to the task of multi-view face alignment. This is a tougher test for the ability of HDTs to parse images because there have been far more studies of face alignment than horse parsing. The input is a set of 64 points marked on the faces. We applied one-example learning followed by structure-perceptron to learn HDTs for faces. We then perform alignment by applying HDTs to each image and using compositional inference to estimate the state variables. Our HDT approach, using identical settings for horse parsing, *achieves an average distance error of* 6.0 *pixels, comparable with the best result* 5.7 *pixels*, obtained by [19]. Their approach is based mostly on Active Appearance Models (AAMs) [20] which were designed specifically to model faces and which are a mature computer vision technique. Figure 17 shows the typical parse results for face alignment. We note that HDTs allow considerable more deformability of objects than do AAMs. Moreover, HDTs required no special training or tuning for this problem (we simply acquired the dataset and trained and tested

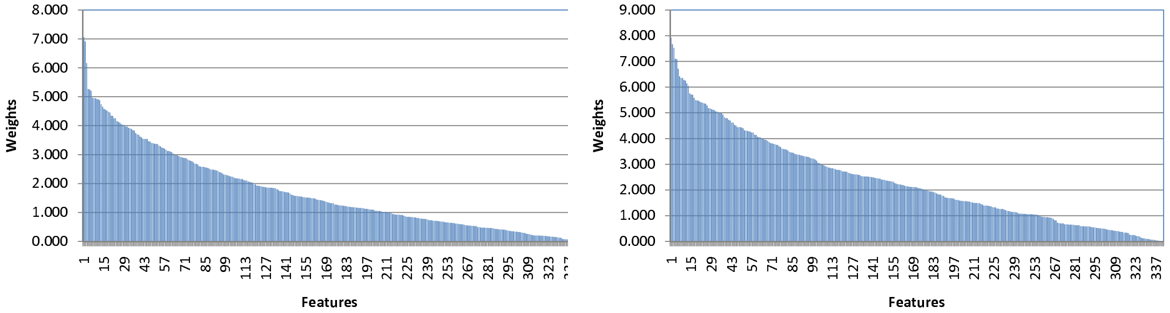

Fig. 15. The weights α of the features for the horse's back (left panel) and the horse's neck (right panel). These experiments use 380 features and show that most are assigned small weights α and hence are effectively not selected.

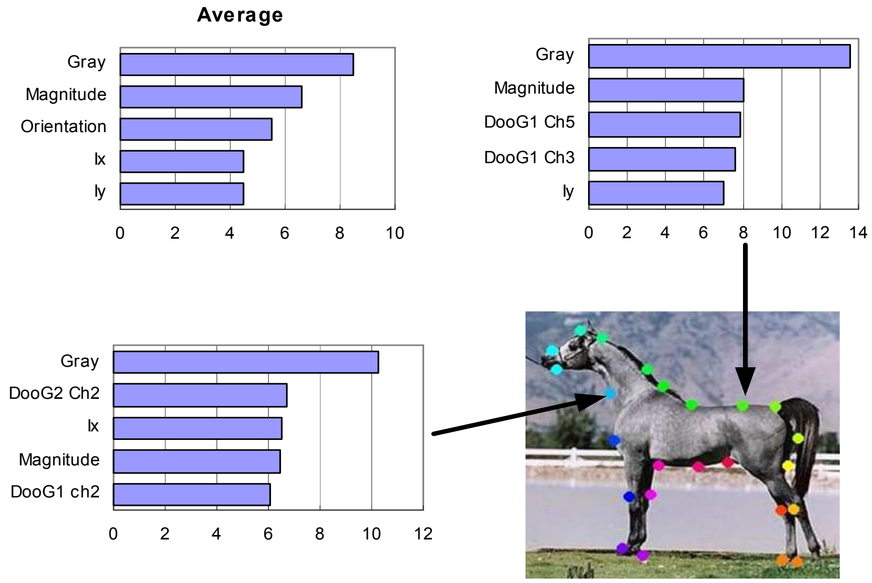

Fig. 16. Weights of Features. The most useful features overall are gray value, magnitude and orientation of gradient, and difference of intensity along horizontal and vertical directions (Ix and Iy). DooG1 Ch5 means Difference of offset Gaussian (DooG) at scale 1 (13*13) and channel (orientation) 5 ($\frac{4}{6}\pi$).

HDTs the next day).

VII. Conclusion

We developed a novel Hierarchical Dynamic Template (HDT) model for representing, detecting, segmenting, and parsing objects. The model is obtained by one-example learning followed by the structure-perceptron algorithm. We detect and parse the HDT by the compositional inference algorithm. Advantages of our approach include the ability to select shape and appearance features

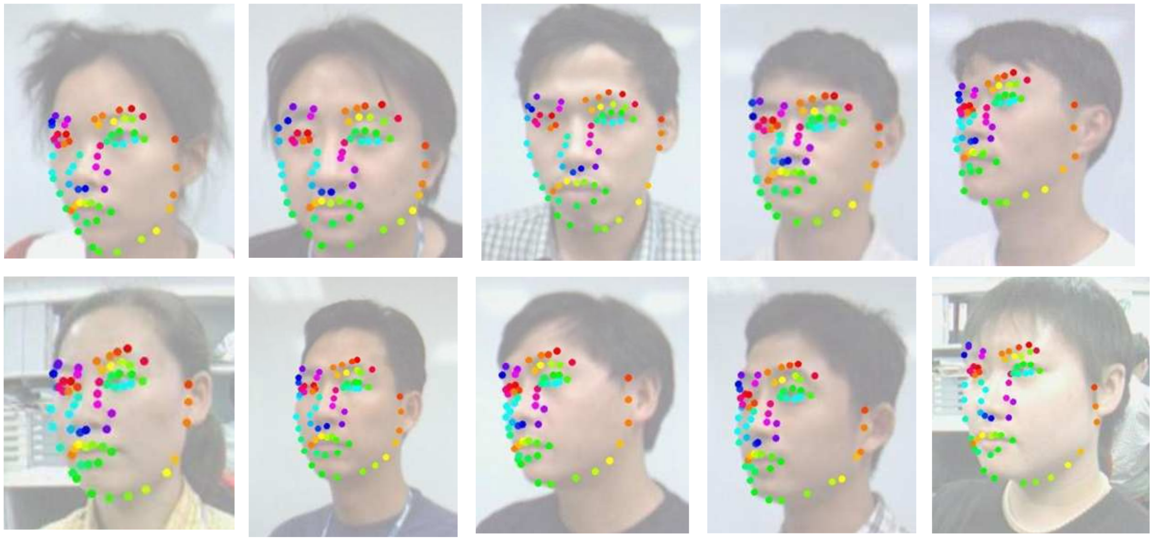

Fig. 17. Multi-view Face Alignment.

at a variety of scales in an automatic manner.

We demonstrated the effectiveness and versatility of our approach by applying it to very different problems, evaluating it on large datasets, and giving comparisons to the state of the art. Firstly, we showed that the HDT outperformed other approaches when evaluated for segmentation on the weizmann horse dataset. It also gave good results for parsing horses (where we supplied the groundtruth), though there are no other parsing results reported for this dataset for comparison. Secondly, we applied HDTs to the completely different task of multi-view face alignment (without any parameter tuning or selection of features) and obtained results very close to the state of the art (within a couple of days). The current limitations of the HDT are due to their lack of OR nodes which decreases their ability to represent objects that vary greatly in appearance and shape, see [39].

We note that certain aspects of HDT's have similarities to the human visual system and, in particular, to biologically inspired vision models. The bottom-up process by its use of surround suppression and its transition from local to global properties is somewhat analogous to Fukushima's neocognitron [41] and more recent embodiments of this principle [42], [43].

Acknowledgements

We gratefully acknowledge support from the National Science Foundation with NSF grant number 0413214 and the W.M. Keck Foundation. We thank YingNian Wu and Zhuowen Tu for helpful discussions.

References

[1] 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.

[2] J. M. Coughlan and S. J. Ferreira, "Finding deformable shapes using loopy belief propagation," in *ECCV (3)*, 2002, pp. 453–468.

[3] H. Chui and A. Rangarajan, "A new algorithm for non-rigid point matching," in *CVPR*, 2000, pp. 2044–2051.

[4] 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.

[5] 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.

[6] P. F. Felzenszwalb and J. D. Schwartz, "Hierarchical matching of deformable shapes," in *CVPR*, 2007.

[7] P. A. Viola and M. J. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *NIPS*, 2001, pp. 1311–1318.

[8] ——, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[9] A. L. Yuille, J. Coughlan, Y. Wu, and S. Zhu, "Order parameters for detecting target curves in images: When does high-level knowledge help?" *International Journal of Computer Vision*, vol. 41(1/2), pp. 9–33, 2001.

[10] Z. Tu, C. Narr, P. Dollar, I. Dinov, P. Thompson, and A. Toga, "Brain anatomical structure segmentation by hybrid discriminative/generative models," *IEEE Tran. on Medical Imaging*, vol. 27(4), pp. 495–508, 2008.

[11] L. Zhu and A. L. Yuille, "A hierarchical compositional system for rapid object detection," in *NIPS*, 2005.

[12] 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.

[13] M. Collins, "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms," in *EMNLP*, 2002, pp. 1–8.

[14] 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.

[15] L. Zhu, Y. Chen, X. Ye, and A. L. Yuille, "Structure-perceptron learning of a hierarchical log-linear model," in *CVPR*, 2008.

[16] E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," in *ECCV (2)*, 2002, pp. 109–124.

[17] 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. [Online]. Available: citeseer.ist.psu.edu/leibe04combined.html

[18] S. Z. Li, H. Zhang, S. Yan, and Q. Cheng, "Multi-view face alignment using direct appearance models," in *FGR*, 2002, pp. 324–329.

[19] 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.

[20] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *ECCV (2)*, 1998, pp. 484–498.

[21] Z. Tu and A. L. Yuille, "Shape matching and recognition - using generative models and informative features," in *ECCV (3)*, 2004, pp. 195–209.

[22] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30(1), pp. 36–51, 2008.

[23] J. Shotton, A. Blake, and R. Cipolla, "Multi-scale categorical object recognition using contour fragments," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 30(7), pp. 1270–1281, 2008.

[24] M. Marszalek and C. Schmid, "Semantic hierarchies for visual object recognition," in *Computer Vision and Pattern Recognition*, 2007, pp. 1–7.

[25] H. Chen, Z. Xu, Z. Liu, and S. C. Zhu, "Composite templates for cloth modeling and sketching," in *CVPR (1)*, 2006, pp. 943–950.

[26] Y. Jin and S. Geman, "Context and hierarchy in a probabilistic image model," in *CVPR (2)*, 2006, pp. 2145–2152.

[27] M. P. Kumar, P. H. S. Torr, and A. Zisserman, "Obj cut," in *CVPR (1)*, 2005, pp. 18–25.

[28] X. Ren, C. Fowlkes, and J. Malik, "Cue integration for figure/ground labeling," in *NIPS*, 2005.

[29] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *ECCV (4)*, 2006, pp. 581–594.

[30] J. M. Winn and N. Jojic, "Locus: Learning object classes with unsupervised segmentation," in *ICCV*, 2005, pp. 756–763.

[31] T. Cour and J. Shi, "Recognizing objects by piecing together the segmentation puzzle," in *CVPR*, 2007.

[32] L. Zhu, Y. Chen, and A. L. Yuille, "Unsupervised learning of a probabilistic grammar for object detection and parsing," in *NIPS*, 2006, pp. 1617–1624.

[33] S.-F. Zheng, Z. Tu, and A. Yuille, "Detecting Object Boundaries Using Low-, Mid-, and High-Level Information," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[34] E. Sharon, A. Brandt, and R. Basri, "Fast multiscale image segmentation," in *CVPR*, 2000, pp. 1070–1077.

[35] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277–296, 1999.

[36] M. Collins and N. Duffy, "New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron," in *ACL*, 2001, pp. 263–270.

[37] M. Collins and B. Roark, "Incremental parsing with the perceptron algorithm," in *ACL*, 2004, p. 111.

[38] 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.

[39] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. L. Yuille, "Max margin and/or graph learning for parsing the human body," in *CVPR*, 2008.

[40] E. Borenstein and J. Malik, "Shape guided object segmentation," in *CVPR (1)*, 2006, pp. 969–976.

[41] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[42] Y. Amit, D. Geman, and X. Fan, "A coarse-to-fine strategy for multiclass shape detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 12, pp. 1606–1621, 2004.

[43] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *CVPR (2)*, 2005, pp. 994–1000.