Discriminatively Trained And-Or Graph Models for Object Shape Detection

Liang Lin, Xiaolong Wang, Wei Yang, and Jian-Huang Lai

Abstract—In this paper, we investigate a novel reconfigurable part-based model, namely And-Or graph model, to recognize object shapes in images. Our proposed model consists of four layers: leaf-nodes at the bottom are local classifiers for detecting contour fragments; or-nodes above the leaf-nodes function as the switches to activate their child leaf-nodes, making the model reconfigurable during inference; and-nodes in a higher layer capture holistic shape deformations; one root-node on the top, which is also an or-node, activates one of its child and-nodes to deal with large global variations (e.g. different poses and views). We propose a novel structural optimization algorithm to discriminatively train the And-Or model from weakly annotated data. This algorithm iteratively determines the model structures (e.g. the nodes and their layouts) along with the parameter learning. On several challenging datasets, our model demonstrates the effectiveness to perform robust shape-based object detection against background clutter and outperforms the other state-of-the-art approaches. We also release a new shape database with annotations, which includes more than 1500 challenging shape instances, for recognition and detection.

Index Terms—Object Detection, Grammar Model, And-Or Graph, Structural Optimization.

✦

1 INTRODUCTION

As psychophysics experiments suggested, humans can successfully identify objects in images using contour fragments alone [38]. In computer vision, recognizing object shapes from salient contours is an active research area. Several methods [32], [21], [13], [10] have demonstrated that the contours (silhouettes) are robust against variations of illumination, color, and texture. However, there are two long-standing difficulties in the current research.

- Unreliable edge map extraction and contour tracing. Some key contours can be missing or connected to their background, making it difficult for accurately localizing shapes against surrounding clutter.
- Large variations within an object category, e.g. different object poses, views, occlusions, and deformations. Without using appearance or texture information, this challenge might be more serious, as shape contours are somewhat ambiguous and less discriminative.

Some recently proposed approaches addressed the two issues by learning hierarchical and compositional models, and achieved substantial progresses [28], [32],

- The authors are with Sun Yat-sen University, Guangzhou 510006, P. R. China, and L. Lin and J. H. Lai are with the SYSU-CMU Shunde International Joint Research Institute, Shunde, China. E-mail: linliang@ieee.org.

[14]. These models represent an object shape in terms of the parts (i.e. local contours) and the inter-part relations. However, their model structures (e.g. the number of parts and the ways of composition) are often fixed, consequently limiting the performances on complex scenarios.

In this work, we develop a novel reconfigurable part-based model in the form of an And-Or graph representation, which is discriminatively trained from weakly annotated training data (i.e. without annotating the object parts). Our model achieves superior performances on the task of detecting and localizing shapes from cluttered background, compared with other state-of-the-art methods. Figure 1 shows an example of our And-Or graph model. The key component of our model is the "switch variable", referred to the or-node, which incorporates the compositional alternatives and makes the model reconfigurable. Specifically, the or-node specifies the way of compositions by activating the child nodes, to deal with the above-mentioned challenges in shape detection. Our And-Or graph model consists of four layers described as follows.

The **leaf-nodes** at the bottom represent a batch of local classifiers that detect the salient contour fragments of objects. Each leaf-node is defined within a divided block, denoted by the red box in the bottom of Figure 1. Given the edge map extracted from an image, a leaf-node takes the contours fallen into its block as the inputs. Once a long contour exceeds the block, it is automatically truncated. This is actually a partial matching scheme to handle the unreliable bottom-up edge tracing, i.e. to avoid object contours connecting to the background. Moreover, to capture the discriminability of contours, we design a new contour feature that combines the triangle-based descriptor [20] and the Shape Context
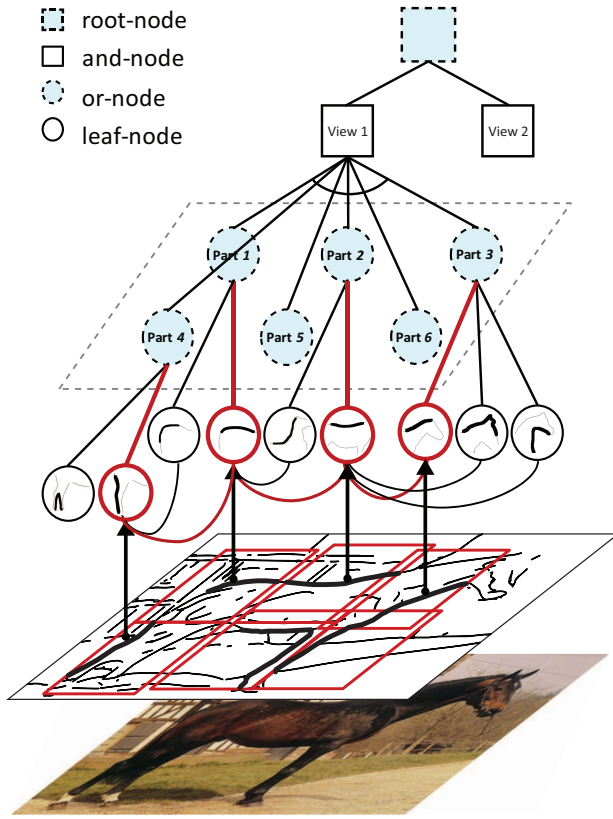
Fig. 1. An example of our And-Or graph model. It comprises four layers from bottom to top: the leaf-nodes (denoted by the solid circles) at the bottom for localizing local contour fragments, the or-nodes (denoted by the dashed blue circles) over the bottom specifying the activations of their child leaf-nodes, the and-nodes (denoted by the solid squares) encoding the holistic (view-based) variances, and the root-node (denoted by the dashed blue squares) on the top to switch the selection of its child and-nodes. The horizontal links incorporate contextual interactions among parts. Note that the leaf-nodes inherit the links that are defined between the layer of or-nodes. The nodes and links in red indicate the activation of leaf-nodes during the detection.

descriptor [3].

The **or-nodes** defined as the switch variables that specify the activation of their child leaf-nodes, denoted by the dashed blue circles in Figure 1. During detection, each or-node activates one of its child lead-nodes and also selects the contour fragment detected by the activated leaf-node. The or-nodes thus represent the parts of an object shape, while the leaf-nodes capture all of the local variabilities. As Figure 1 illustrates, our model can capture not only the local variations (e.g. part 2 of the example), but also the inconsistency caused by missing or broken edges (e.g. part 3 of the example).

The **collaborative edges** in our model impose the contextual information among shape contours, denoted by the horizontal links between the leaf-nodes in Figure 1. Some of the existing compositional shape models ignore the contextual relations among contours, or simplify the relations by calculating the co-occurrence frequencies of neighbor contours [16]. In contrast, we utilize informative spatial layout features to define the edges, motivated by the methods for contextualized object detection [5], [40].

The **and-nodes** aggregate the local shape contours that have been selected via the or-nodes. Each and-node is defined as a potential function that captures the holistic shape deformations and distortions. Once the contour fragments are localized, The and-nodes further verify them as a whole to improve the discriminability of our model.

The **root-node** at the top functions as a switch to choose its child and-nodes, accounting for the large global variations (e.g. different views of shapes). It is defined exactly in the same way as the or-nodes. For example, two horses may appear diversely under different views, so that our model can adaptively activate different and-nodes for detecting them.

From the bottom to the top, our model is hierarchically constructed into an "And-Or-And-Or" structure. Note that the leaf-nodes in our model can also be viewed as the and-nodes, as they are defined in the same way. This structure is very expressive and general to model object variations. The "And" symbol indicates the combination of sub-parts while the "Or" symbol indicates the switch between possible configurations. We introduce the latent variables to make our model reconfigurable. In particular, the latent variables include the activation states of the or-nodes and the root-nodes, and the locations of contour fragments. The leaf-nodes and the and-nodes are defined as classification functions whose coefficients are treated as the observable model parameters. With the latent variables, the graph nodes and edges are explicitly mapped with the discriminative classification function of our model. Figure 4 provides an intuitive illustration of our And-Or graph model, which will be discussed later on. We regard our model as a general extension of the pictorial and deformable part-based models [9], [1], [7], as it incorporates not only the hierarchical decompositions, but also the explicit structural alternatives.

The training of the And-Or graph model is another innovation of this work. The challenges lie in two aspects. First, multiple parameters in different layers need to be optimized along with the latent variables, and the objective function for optimization is non-convex, which cannot be solved directly with the traditional methods such as the support vector machines (SVMs). Second, it is non-trivial to automatically discover the model structures in the model learning, as the training examples are not annotated into object parts. In the literature, learning And-Or graph models (or other reconfigurable models) usually relies on elaborative annotations or initializations [45], [46], [15]. To cope with these two problems, we propose a novel learning method, called Dynamical Structural Optimization (DSO), which is inspired by the recently proposed optimization methods [7], [43], [42].

This algorithm iteratively optimizes the model structures together with the multi-layer parameter learning, which includes three main steps. (i) Apply current model on the training examples while estimating the latent variables for each example. (ii) Discover new model structures. As the model structures are mapped with the discriminative function of our model (see Figure 4), refactoring (rearranging) the feature vectors of training examples can lead to new structures. In brief, we perform clustering on the sub-feature-vectors corresponding to different nodes, and generate new structures according to the clustering results. For example, at one part of the shape, if the corresponding sub-feature-vectors are clustered into three groups, then we create three leaf-nodes accordingly to detect the local contours. (iii) Learn the model parameters with the newly generated structures.

Shape detection using the And-Or graph model is realized by searching over a image pyramid. We first accomplish two testing steps to generate several hypotheses of detection, and each hypothesis represents a configuration comprising detected contour fragments. (i) Local testing uses all leaf-nodes to detector contour fragments within the edge map. (ii) Binding testing imposes the collaborative edges among the contour fragments to further weigh the hypotheses. Afterwards, the and-nodes re-score each hypothesis by measuring the contour fragments as a whole. The root-node decides the final detection by selecting the most possible hypothesis.

The remainder of this paper is organized as follows. Section 2 provides a brief review of related work. Then we present the model representations in Section 3 and follow with a description of the inference procedure in Section 4. Section 5 focuses on discussing the learning algorithm. The experimental results and comparisons are exhibited in Section 6. Section 7 concludes this paper.

2 RELATED WORK

In this section, we review the extant techniques for shape (or contour) matching and shape model learning.

Many methods treat shape detection as a task of matching contours with certain distance measures, and they mostly utilized hand-drawn reference templates [3], [8], [47], [2], [20], [25], [21]. To handle diverse shape deformations and distortions, a number of robust shape (or contour) descriptors have been extensively discussed, such as Shape Context [3], Geodesic-Intensity Histogram [18], Contour Flexibility [39], and Local Angle [25], [20]. Based on these shape features, several effective matching schemes [14], [34], [2] have been proposed to deal with the various challenges. For example, the inner-distance matching algorithm [18] was presented to handle the articulated shape deformations. Tu et al. [34] presented an efficient data-driven EM algorithm to iteratively optimize shape alignment and matching correspondences. Felzenszwalb et al. [8] proposed to hierarchically match shapes using the dynamic programming algorithm, demonstrating good potential in capturing

large shape deformations. An MCMC-based sampling algorithm was discussed in [14] to solve multi-layer shape matching. To overcome the problems caused by incomplete or noisy contours, Zhu et al. [47] presented a many-to-many contour matching algorithm using a voting scheme. Riemenschneider et al. [25] solved the partial shape matching by identifying matches from fragments of arbitrary length to the reference contours.

An alternative to shape detection is addressed by learning shape models for a given category of shape instances. These methods represent shapes as a loose collection of local contour fragments or an ensemble of pairwise constraints [28], [16], [32]. They usually involve the construction of a codebook of contour fragments (e.g. Groups of Adjacent Contours (GAS) [10]) and train the shape models by supervised leaning. For example, the boosting methods were employed to train the discriminative classifiers with contour-based features [28], [24]. Maji et al. [22] incorporated the Hough transform into a discriminative learning framework, in which the contour words and their spatial layout were optimized jointly. Kokkinos and Yuille [13] suggested hierarchically parsing shapes with the bottom-up and top-down computations, and adopted the multiple instance learning algorithm for model training. Another type of shape template is the active basis model proposed by Wu et al. [35], which was trained with a shared sketch algorithm.

Very recently, major progress has been made in appearance-based object recognition using the latent structure models [44], [7], [36], in which the latent variables effectively enrich the representations. These methods owe their success to their ability to cope with deformations, occlusions, and variations. Based on these methods, Srinivasan et al. [32] trained the descriptive contour-based detector by using the latent-SVM algorithm, Song et al. [31] integrated the context information with the SVM-based learning, and Schnitzspan et al. [27] further combined the latent discriminative learning with conditional random fields using multi-types of shape features.

The And-Or graph was originally explored by Zhu and Mumford [46] for modeling complex visual patterns. Its key idea, using And/Or nodes to account for structure reconfigurations and variabilities in hierarchical composition, has been extensively applied in several vision tasks such as object and scene parsing [45], [37], [15] and event analysis [29]. However, these approaches often require elaborate annotations or manual initializations. Si and Zhu [30] recently presented a framework for unsupervised learning of the And-Or image template, and demonstrated very promising results on modeling complex object categories. Our approach is partially motivated by these works, and we target on an alternative way to discriminatively train the And-Or graph model with the non-convex optimization. Our preliminary attempts along this path have been discussed in [36], [17].

3 REPRESENTATIONS

In this section, we define all the components of our And-Or graph models, including the shape features and the potential functions for graph nodes and edges.

3.1 Contour Descriptor

First, we introduce our contour descriptor for characterizing local contour fragments. As Figure 2 illustrates, this feature combines the triangle-based descriptor [20] and the Shape Context [3], capturing local contour deformations with the surrounding contexts. For any contour fragment we extract a sequence of sample points Ω, and for each point in Ω, its triangle-based descriptor and Shape Context descriptor are both computed and concatenated into a vector. Then we pool the vectors of all the sample points into a histogram.
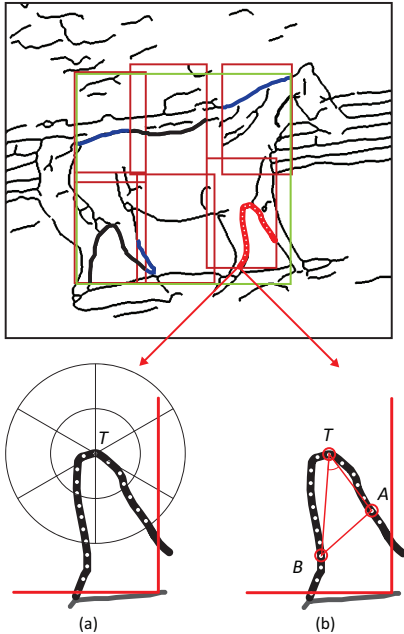

Fig. 2. Illustration of the proposed contour descriptor. This feature combines the Shape Context descriptor in (a) and the triangle-based descriptor in (b) to characterize a local contour fragment.

Given a point $T \in \Omega$ for a contour, we collect triangles that are formed by T and any other two A, B in Ω. Note that each triangle is constructed by three different points. As Figure 2(b) illustrates, the triangle-based descriptor for T is a 3-D histogram, denoted by $\mathbb{H}^t(T)$, which contains the angle values (e.g. $\angle BTA$) and the two distances TA and TB in each dimension, respectively. We use the clockwise orientation to determine the triangle $\angle BTA$, and the distances TB and TA are normalized by the average distance between the points in Ω. The Shape Context descriptor, denoted by $\mathbb{H}^b(T)$, is constructed by T and all other points in Ω.

In our implementation, the number of sample points for each contour fragment is fixed at 20 , and the distances between adjacent points in Ω are equal. For each point T, $(20-1)*(20-2)/2 = 171$ triangles are thus collected. We define the 3-D histogram $\mathbb{H}^t(T)$ including 2 bins for TA, 2 bins for TB, and 6 bins for angle $\angle BTA$ ranging from 0 to π. We transform $\mathbb{H}^t(T)$ into a $2 \times 2 \times 6 = 24$-bin 1-D feature vector. For the Shape Context descriptor $\mathbb{H}^b(T)$, we use 2 bins for lengths and 6 bins for polar angles ranging from 0 to 2π, then its length is $2 \times 6 = 12$. By concatenating these two descriptors, we obtain the feature vector of T including $(24 + 12) = 36$ bins. Thus, the contour fragment is represented by a feature vector of $36 * 20 = 702$ bins.

3.2 And-Or Graph Model

Our model is defined in the form of an And-Or graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ includes four levels of nodes and $\mathcal{E}$ includes the graph edges. The root-node is indexed as 0, indicating the switch among different shape views,(or other different global variations, by analogy). The and-nodes are indexed by $r = 1, ..., m$, with each representing one global classifier. For each and-node, there are a number of z or-nodes arranged in a layout of $b_1 \times b_2$ blocks to represent several object parts, and we index all of the or-nodes as $j = m + 1, ..., (z + 1) * m$. The leaf-nodes in the fourth layer are indexed by $i = (z + 1) * m + 1, ..., (z + 1) * m + 1 + n$, where n is the number of leaf-nodes. For notation simplicity, we define $m' = (z + 1) * m + 1, n' = (z + 1) * m + 1 + n$, and $i \in ch(j)$ indicating a child node of node j. The details of the model $\mathcal{G}$ are described as follows.

Leaf-node: Each leaf-node L_i is a local classifier for detecting partial shape contours. We denote the location of leaf-node L_i as p_i, which is determined by its parent or-node. Given the extracted edge map X, we treat contour fragments within the observed block as the inputs of L_i. For a contour c, we denote $\phi^l(p_i, c)$ as its feature vector using the proposed contour descriptor, and only the part of c that has fallen into the block will be considered. Note that we can prune some very short contours as noises in practice. The response of classifier L_i located at p_i is defined as:

$$\mathcal{R}_i^l(X, p_i) = \max_{c \in X} \omega_i^l \cdot \phi^l(p_i, c), \qquad (1)$$

where ω_i^l is a parameter vector that is set to zero if the corresponding leaf-node L_j is nonexistent. We can thus localize the contour representing the shape part by $c_i = argmax_{c \in X} \omega_i^l \cdot \phi^l(p_i, c)$. This partial detecting scheme enables to partition true object contours from cluttered background.

Or-node: The or-node $U_j, j = m + 1, ..., (z + 1) * m$ specifies one of its child leaf-nodes, and also the contour detected by the leaf-node. Every or-node is allowed to slightly perturb their locations with respect to the root in order to capture the inter-part deformations.

For each or-node U_j, we define the deformation feature, $\phi^s(p_0, p_j) = (dx, dy, dx^2, dy^2)$, where (dx, dy) encodes the displacement of the or-node position p_j to the

expected position p_0 determined by the root-node. The cost of locating U_j at p_j is:

$$D_j(p_0, p_j) = \omega_j^s \cdot \phi^s(p_0, p_j), \qquad (2)$$

where ω_j^s is a 4-dimensional parameter vector corresponding to $\phi^s(p_0, p_j)$.

For each leaf-node L_i associated with U_j, we introduce an indicator variable $v_i \in \{0,1\}$ representing whether it is activated by U_j or not. We also define an auxiliary vector for U_j, $\mathbf{v}_j = \{v_i\}_{i \in ch(j)}$, where $||\mathbf{v}_j|| = 1$ or 0. Note that $||\mathbf{v}_j|| = 1$ only when one of the leaf-nodes under U_j is activated. In this way, the or-node can adaptively activate the different leaf-nodes to capture the diverse local shape variance. It is worth mentioning that the cost of locating the or-node is independent of the selected leaf-nodes because we assume the leaf-nodes belong to the same part (i.e. or-node) act a nearby location.

Thus, the response of the or-node U_j is defined as,

$$\mathcal{R}_j^u(X, p_0, p_j, \mathbf{v}_j) = \sum_{i \in ch(j)} \mathcal{R}_i^l(X, p_j) \cdot v_i - D_j(p_0, p_j). \quad (3)$$
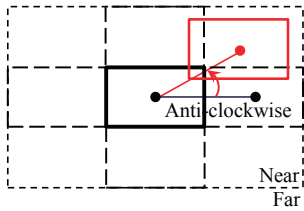

Fig. 3. The spatial contextual features defined for the collaborative edges.

And-node: The and-node, A_r, performs a global verification for the whole shape. For each and-node, we have a set of contour fragments, $C_r = \{c_1, c_2, ..., c_z\}$, which are determined by its child or-nodes. Then we adopt the Shape Context descriptor [3] to describe these contours as a whole, $\phi^a(C_r)$. Thus, we define the and-node's response as,

$$\mathcal{R}_r^a(C_r) = \omega^a \cdot \phi^a(C_r), \qquad (4)$$

where ω^a is the corresponding parameter vector.

Collaborative Edge: We impose contextual interactions among shape parts based on the collaborative edges. Given any two different or-nodes associated with the same and-node, we link an edge between them and their child leaf-nodes inherit the edge. We define the collaborative edges using the spatial contextual features, as Figure 3 illustrates.

Suppose one edge connects two leaf-nodes $(L_i, L_{i'})$ are located at p_i and $p_{i'}$ respectively. We collect a 4-bin feature $\psi(p_i, p_{i'})$ for the two leaf-nodes according to their spatial layout. Each bin of $\psi(p_i, p_{i'})$ represents one of the four relations of $(L_i, L_{i'})$: *clockwise, anti-clockwise, near,* and *far*. In Figure 3, the bold rectangle in the center indicates the location of L_i, which is connected to the red bold rectangle indicating the location of $L_{i'}$. The dashed

line represents the initial layout of the two leaf-nodes, and the red solid line is the adjusted actual layout in detection. Specifically, we define the relations as

- *Near and Far*: If $L_{i'}$ falls into the outer dashed rectangle, it is *near* to L_i, i.e. the bin of *near* is activated (i.e. being set as 1); otherwise it is *far* from L_i.
- *Clockwise and Anti-clockwise*: One of the two relations is activated (i.e. being set as 1) according to the angle between the dashed line and the solid red line.

These relations intuitively encode the spatial contexts of two leaf-nodes $(L_i, L_{i'})$. Let $\{\mathbf{v}_j\}$ represent the activation variables of the leaf-nodes, and we denote P as a vector of the positions of all or-nodes U_j. P also specifies the locations $\{p_i\}$ of the activated leaf-nodes. The response of the collaborative edge is then parametrized as

$$\mathcal{R}_r^e(P, \{\mathbf{v}_j\}) = \sum_{j \in ch(r)} \sum_{i \in ch(j)} \sum_{i' \in \partial(i)} \omega_{(i,i')}^e \cdot \psi(p_i, p_{i'}) \cdot v_i \cdot v_{i'}, \qquad (5)$$

where $\partial(i)$ represents the set of neighbor leaf-nodes of L_i, and each neighbor has a different parent node with L_i. $\omega_{(i,i')}^e$ is the corresponding weight. v_i and $v_{i'}$ are the activation indicators for L_i and $L_{i'}$, respectively, as the edges are imposed only for the activated leaf-nodes.

Root-node: The root-node on the top alternatively activates one of its child and-nodes, whose definition is similar with that of the or-node. Also, we use a variable $v_r \in \{0,1\}$ to specify the activation of each and-node A_r, and the indicator vector for the root-node is $\mathbf{v}_0 = \{v_r\}_{r=1}^m$ and $||\mathbf{v}_0|| = 1$, i.e. only one child is selected.

Let P imply the part-based deformation with or-nodes, and $V = (\mathbf{v}_0, \{\mathbf{v}_j\})$ imply the selection of and-nodes and leaf-nodes, the overall response of our model is then defined as:

$$\mathcal{R}_G(X, P, V) =$$
$$\sum_{r=1}^m v_r \cdot (\sum_{j \in ch(r)} \mathcal{R}_j^u(X, p_0, p_j, \mathbf{v}_j) + \mathcal{R}_r^e(P, \{\mathbf{v}_j\}) + \mathcal{R}_r^a(C_r)). \qquad (6)$$

In this model, $H = (P, V)$ are the latent variables that will be adaptively estimated in testing. For notation simplicity, our model in Equation (6) can be re-written as :

$$\mathcal{R}_G(X, H) = \omega \cdot \phi(X, H), \qquad (7)$$

where $\phi(X, H)$ represents the concatenated feature vector for all nodes and edges in the model, and ω includes all of the parameters corresponding to $\phi(X, H)$. Figure 4 illustrates our And-Or graph model mapped with the discriminative function.

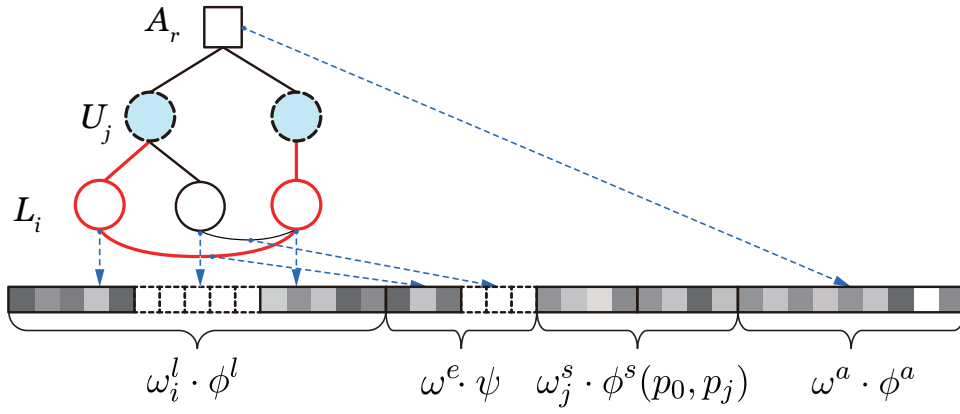We summarize the symbols used in our model in Table 1.

Fig. 4. Mapping the latent And-Or graph with the discriminative function defined in Equation 6. Different layers of nodes in our model are associated with certain bins in the feature vector $\phi(X, H)$ (at the bottom). The activated leaf-nodes are highlighted in red, and the feature bins are set to zeros for the other inactivated nodes. The embedded latent variables $H = (P, V)$ make our model reconfigurable during detection.

Symbol	Meaning
$\{A_r\}_{r=1}^m$	The and-nodes.
$\{U_j\}_{j=m+1}^{(z+1)\cdot m}$	The or-nodes.
$\{L_i\}_{i=m'}^{n'}$	The leaf-nodes.
X	The edge map of an image.
$P = \{p_0, p_j, p_i\}$	The locations of the root-node p_0, or-nodes p_j, and leaf-nodes p_i.
$\mathcal{R}_i^l(X, p_i)$	The response of the classifier associated with leaf-node L_i located at p_i.
$\mathcal{R}_j^u(X, p_0, p_j, \mathbf{v}_j)$	The response of the or-node. $\mathbf{v}_j$ indicates the selection of its child leaf-nodes.
$\mathcal{R}_r^a(C_r)$	The response of the and-nodes, which provides a global verification for the shape C_r.
$\mathcal{R}_r^e(P, \{\mathbf{v}_j\})$	The response of the collaborative edges. $\{\mathbf{v}_j\}$ indicates the selection of the leaf-nodes.
$\mathcal{R}^G(X, P, V)$	The response of the whole model, where P and V represent the latent variables.
$H = (P, V)$	All latent variables (including positions P and activation variables V) of our model.

TABLE 1
Notation summary of this work.

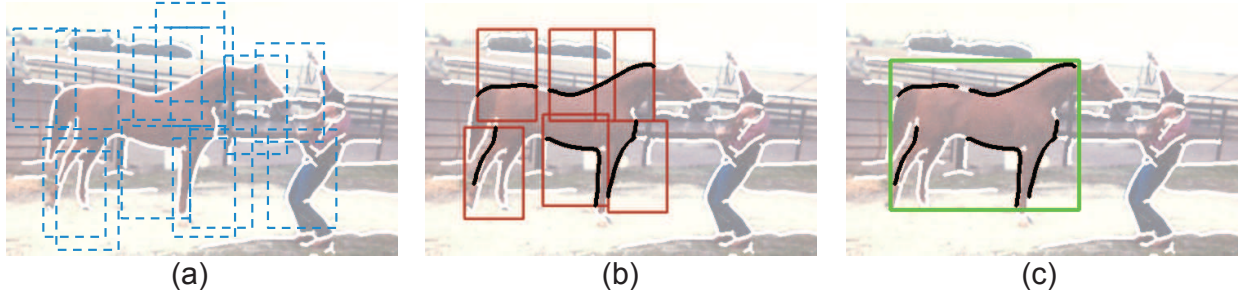(a)   (b)   (c)

Fig. 5. Illustration of the inference procedure. (a) shows local testing for detecting contour fragments within the edge map; the blue dashed boxes represent perturbed blocks associated with the leaf-nodes. (b) shows a hypothesis of detection including candidates (indicated by the red boxes) proposed by all or-nodes, in which the collaborative edges are imposed. (c) shows the global verification, in which the ensemble of contours are measured as a whole.

4 INFERENCE

Given the edge map X extracted from the image, the inference task is to detect the optimal contour fragments within the detection window scanned over an image pyramid. The detection is a search procedure to activate nodes from bottom to top, in which a number of hypotheses are generated and each one specifies a con-

figuration of detected contour fragments. We verify the hypotheses and prune the unlikely ones by maximizing the model response defined in Equation (7).

We conduct the inference algorithm with the following steps. An example illustrating the inference procedure using our model is presented in Figure 5.

Local testing: We use all of the leaf-nodes (i.e. the local contour classifiers) to search for optimal contour frag-

ments within the edge map X. Assume that one or-node U_j, associated with a partitioned block in the detection window, contains a number of leaf-nodes $\{L_i, i \in ch(j)\}$, and that the initial position of U_j is p'_j. Each U_j is allowed to slightly perturb its location. At each location $p'_j + \delta$, we treat all of the contours that have fallen into the block as the inputs to every leaf-node of U_j, as Figure 5(a) illustrates. By maximizing the response in Equation (1), each leaf-node $L_i \in ch(j)$ can find an optimal contour at a certain location. Recall that each or-node can activate only one of the child leaf-nodes. Thus, the possibility of different leaf-node selections can generate a batch of detection hypotheses. In particular, we denote $\hat{H}$ as the latent variables for one hypothesis, and denote $(\hat{\mathbf{v}}_j, \hat{p}_j)$ for a possible activation of U_j, where $\hat{\mathbf{v}}_j$ indicates the leaf-node selection and $\hat{p}_j$ is the location. The cost of $(\hat{\mathbf{v}}_j, \hat{p}_j)$ is then measured by the function $\mathcal{R}^u_j$ defined in Equation (3).

Binding testing: The hypotheses from the local testing are further weighed and filtered by imposing the collaborative edges. In each hypothesis, each or-node proposes one leaf-node, and any two leaf-nodes derived from different or-nodes are connected by an edge. We measure the score by the potential function in Equation (5).

In this way, each detection hypothesis is scored by the two testing steps, as,

$$S^l_r(X, \hat{H}) = \sum_{j \in ch(r)} \mathcal{R}^u_j(X, p_0, \hat{p}_j, \hat{\mathbf{v}}_j) + \mathcal{R}^e_r(\hat{P}, \{\hat{\mathbf{v}}_j\}), \quad (8)$$

where $\hat{P} = \{\hat{p}_j\}$ denotes the locations of all of the or-nodes. In practice, we can prune some of the hypotheses by setting a threshold on the score.

Global verification: In this step, we apply the and-nodes to re-score the hypotheses of detection. For any hypothesis, we obtain an ensemble of contours, $\hat{C}_r = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_z\}$, each of which is proposed by one or-node. We can measure the contours as a whole by $S^g_r(X, \hat{H}) = \mathcal{R}^a_r(\hat{C}_r)$ in Equation (4), as Figure 5(c) illustrates.

Afterwards, the root-node determines the optimal detection by selecting the maximum aggregated score, as

$$H^* = \arg\max_{\hat{H}} \sum_{r=1}^{m} (S_l(X, \hat{H}) + S^g_r(X, \hat{H})) \cdot \hat{\mathbf{v}}_0, \quad (9)$$

where $||\hat{\mathbf{v}}_0|| = 1$ constrains only one of the and-nodes selected by the root-node.

The overall inference procedure appears in Algorithm 1.

5 AND-OR GRAPH LEARNING

We formulate the And-Or graph model learning as a joint optimization task of model structures and parameters. To achieve this goal, we present a novel structure learning algorithm extended from the existing non-convex optimization methods [43], [36]. This algorithm

---

Algorithm 1: Inference with the And-Or graph representation

Input:
  X: the edge map extracted from the test image.
Output:
  H^*: the optimal detection with the maximal detection score $\mathcal{R}_G(X, H^*)$.

Local testing:
  1. Apply leaf-nodes to detect all possible local contour fragments.
  2. Generate a batch of detection hypotheses via the or-nodes.
Binding testing:
  1. Impose the collaborative edges between leaf-nodes in each detection hypothesis.
  2. Score the hypotheses by Equation (8).
  3. Prune unlikely hypotheses by thresholding the score.
Global verification:
  1. For each hypothesis, the local contours are measured as a whole via the and-nodes.
  2. Aggregate all potentials via the root-node in Equation (9).
  3. Merge results by non-maximum suppression over all image positions and scales.

---

optimizes the objective in a dynamical manner: the latent structures $H = (P, V)$ are iteratively determined along with the parameter learning in each step. For example, new leaf-nodes are created or removed to better adapt to the training data by adjusting the latent variables. One instance of our learning procedure is illustrated in Figure 6: from (a) to (b), a leaf-node associated with U_1 is removed and a new leaf-node under U_6 is created in (c).

5.1 Optimization Formulation

Suppose we have a set of positive and negative training samples $(X_1, y_1), \ldots, (X_N, y_N)$, where X is the edge map and $y = \pm 1$ is the label indicating positive and negative samples. We assume that the samples indexed from 1 to K are the positive samples, and that the feature vector for each sample (X, y) is,

$$\phi(X, y, H) = \begin{cases} \phi(X, H) & \text{if } y = +1 \\ 0 & \text{if } y = -1 \end{cases}, \quad (10)$$

where H represents the latent variables and $\phi(X, H)$ the overall feature vector of the And-Or graph model. Then we pose the And-Or graph learning as optimizing model parameters along with the latent structures,

$$\omega = argmax_{y,H}(\omega \cdot \phi(X, y, H)). \quad (11)$$

We further transfer this target into a maximum margin formulation,

$$\min_{\omega} \frac{1}{2}||\omega||^2 + \lambda \sum_{k=1}^{N} [\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y, H)) \\ - \max_{H}(\omega \cdot \phi(X_k, y_k, H))], \quad (12)$$

where λ is a penalty weight (set as 0.005 empirically), and $\mathcal{L}(y_k, y, H)$ is the loss function. In our implementation, we define that $\mathcal{L}(y_k, y, H) = 0$ if $y_k = y$, and 1 otherwise.
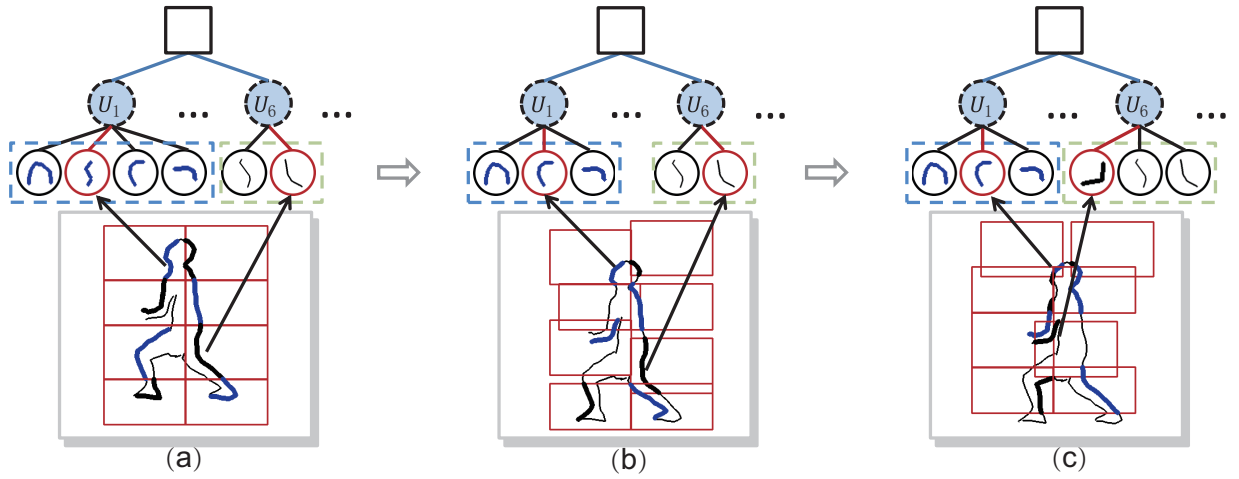
Fig. 6. Illustration of the structure reconfiguration. Parts of the model, two or-nodes (U_1, U_6), are visualized in three intermediate steps. (a) The initial structure, i.e. the regular layout of an object. Two new structures are dynamically generated during the iterations. (b) A leaf-node associated with U_1 is removed. (c) A new leaf-node is created and assigned to U_6.

The target energy in Equation (12) is non-convex making it difficult to be solved analytically. In this work, we propose the Dynamical Structural Optimization (DSO) method to iteratively optimize this objective based on the Concave-Convex Procedure (CCCP) method [43].

5.2 Dynamical Structural Optimization

Following the CCCP method [43], we convert the objective function in Equation (12) into a convex and concave form as,

$$\min_{\omega}[\frac{1}{2}\|\omega\|^2 + \lambda \sum_{k=1}^{N} \max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y, H))]$$

$$- [\lambda \sum_{k=1}^{N} \max_{H}(\omega \cdot \phi(X_k, y_k, H))] \qquad (13)$$

$$= \min_{\omega}[f(\omega) - g(\omega)], \qquad (14)$$

where $f(\omega)$ represents the first two terms, and $g(\omega)$ represents the last term in (13). Assume ω^t is the solution for the t-th iteration. The solution ω^{t+1} for the next iteration can be solved by subjecting it to

$$\nabla f(\omega^{t+1}) = \nabla g(\omega^t). \qquad (15)$$

A geometric explanation of CCCP is presented in Figure 7, where $\nabla g(\omega^t)$ can be regarded as a hyperplane (the red line) at ω^t (the black spot) to upper bound $-g(\omega)$. $\nabla g(\omega^t)$ can be solved analytically once H is fixed. Then, the ω^{t+1} can be estimated accordingly by minimizing $f(\omega^{t+1})$. Please refer to [43] for the theoretical background.

During the training procedure, the model parameters ω^t and latent structures H^t are iteratively updated. To discover the models structures, we add one step called model reconfiguration in each iteration. Recall that the model structures (e.g. graph nodes) are mapped with

the feature vectors, as Figure 4 illustrates. In this step, from the feature vectors of all positive training examples, we first extract the sub-vectors that are corresponding to different nodes (i.e. and-nodes or leaf-nodes), and each node, we perform clustering on these sub-vectors, respectively. Then, according to the clustering results, we rearrange each feature vector by placing the sub-vectors back into the feature vectors (e.g. re-assigning contour fragments to leaf-nodes). Consequently, the new model structures can be generated. Our DSO method iteratively performs with three following steps: (i) estimate the latent variables of training samples; (ii) reconfigure the model structures; (iii) update model parameters for the new structures.
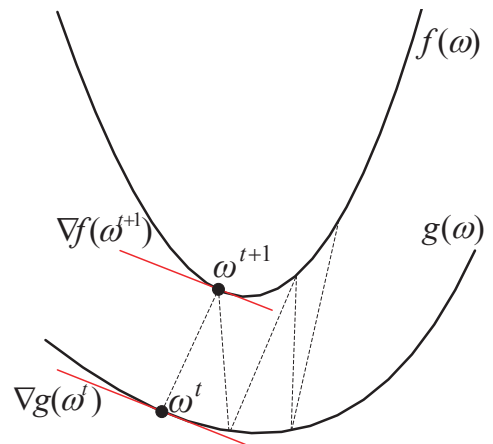

Fig. 7. Geometric illustration of the CCCP procedure. The target energy is decomposed into two functions, $f(\omega)$ and $g(\omega)$. At each step of iteration, a hyperplane (represented by the red line) is calculated as the upper-bound at ω^t for optimizing ω^{t+1}.

(I) The model parameters ω_t in the previous iteration

are fixed. We find a hyperplane q_t to upper bound $-g(\omega)$ in Equation (14),

$$-g(\omega) \leq -g(\omega_t) + (\omega - \omega_t) \cdot q_t, \forall \omega. \qquad (16)$$

The optimal latent variables H_k^* are specified for each positive training example by,

$$H_k^* = argmax_H(\omega_t \cdot \phi(X_k, y_k, H)). \qquad (17)$$

Note that we only take the positive training examples into account as $\phi(X_k, y_k, H) = 0$ when $y_k = -1$. That is, we apply the current model to perform detections on the training samples, and the hyperplane is constructed as

$$q_t = -\lambda \sum_{k=1}^{N} \phi(X_k, y_k, H_k^*). \qquad (18)$$

(II) In the second step, we optimize the model structures based on the estimated latent variables H^*. All graph nodes in our model are mapped with several feature bins (i.e. sub-vectors) of $\phi(X_k, y_k, H_k^*)$ for all of the training samples, as Figure 4 illustrates. Hence, we achieve the model reconfiguration process by rearranging $\phi(X_k, y_k, H_k^*)$. For example, we can remove leaf-node L_j by setting the corresponding bins for L_j into zeros. Specifically, two sub-steps are sequentially performed to generate and-nodes and leaf-nodes, respectively.

(i) Global structure reconfiguration. In the layer of and-nodes, we perform clustering on the feature vectors corresponding to the and-nodes, i.e. the global shape features defined in Equation (4). Note that each vector is a part of $\phi(X_k, y_k, H_k^*)$. The training object shapes detected by the same and-node are initially grouped into one cluster. We then perform clustering on all of the feature vectors by using ISODATA with Euclidean distance. Based on the clustering result, we rearrange the feature vectors mapping with the and-nodes. For example, if one vector is grouped into a new cluster Ω_r, we shall move it into the bins corresponding to And-node A_r, and set its original bins as zeros. In our implementation, we fix the number of and-nodes as m, to simplify the computation.

(ii) Local structure reconfiguration. After the global structure reconfiguration, each and-node is associated with a group of training examples. Suppose the and-node A_r includes a number of or-nodes, and every or-node $U_j, j \in ch(r)$ further derives its child leaf-nodes $L_j, i \in ch(j)$. In this step, we configure the part-level structures rooted by U_j. Note that this step processes each or-node and its leaf-nodes separately.

Each or-node U_j specifies one part of the whole object shape. Given the training examples associating with A_r, we extract the local contour features from $\phi(X_k, y_k, H_k')$, which are corresponding to the shape part of U_j. Then we perform clustering on these vectors, and rearrange these vectors in $\phi(X_k, y_k, H_k')$, similarly as the operation on the and-nodes. In our implementation, the number of leaf-nodes is not fixed, as the local variances of shapes are usually unpredictable. Thus, there are two specific

operators to generate the leaf-nodes according to the clustering.

- One new leaf-node is created if an extra cluster is generated.
- One leaf-node is removed if there are very few samples in the corresponding cluster.

We present a toy example in Figure 8 to illustrate the structure reconfiguration. For the sample X_3, a part of its feature vector $< \phi_5, \ldots, \phi_8 >$ is grouped from one cluster into another while the values of the feature bins are moved from $< \phi_5, \ldots, \phi_8 >$ to $< \phi_1, \ldots, \phi_4 >$.

After the reconfiguration, the latent variables for each training example can be re-calculated, and denoted by H_k^d, in accordance with the arranged feature vectors (refer to Equation (17)). We denote the feature vectors for all examples by $\phi^d(X_k, y_k, H_k^d)$. Then, the hyperplane is transformed accordingly, $q_t^d = -D \sum_{k=1}^{N} \phi^d(X_k, y_k, H_k^d)$.
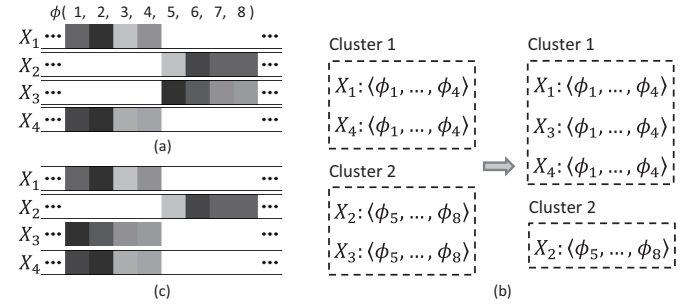

Fig. 8. A toy example for structure reconfiguration. We consider 4 samples, $X_1, \ldots, X_4$, for training the structure of U_i (or A_r). (a) shows the feature vectors ϕ of the samples associated with U_i (or A_r), and the intensity of the feature bin indicates the feature value. (b) illustrates the clustering performed with ϕ'. The vector $\langle \phi_5, \cdots, \phi_8, \rangle$ of X_2 is grouped from cluster 2 to cluster 1. (c) shows the adjusted feature vectors according to the clustering. Note that the model structure reconfiguration is realized by the rearrange of feature vectors, as we discuss in the text. This figure should be viewed in electronic form.

(III) The newly generated model structures can be represented by the feature vectors $\phi^d(X_k, y_k, H_k^d)$, and the model parameters can then be learned by solving Equation (14),

$$\omega_t^d = argmin_\omega[f(\omega) - g(\omega)] \qquad (19)$$

By substituting $-g(\omega)$ with the upper bound hyperplane q_t^d, this optimization task can be transferred as,

$$\min_\omega \frac{1}{2}\|\omega\|^2 + D\sum_{k=1}^{N}[\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y, H))$$
$$- \omega \cdot \phi^d(X_k, y_k, H_k^*)]. \qquad (20)$$

We solve it as a standard structural SVM problem, as,

$$\omega^* = D \sum_{k,y,H} \alpha_{k,y,H}^* \Delta\phi(X_k, y, H), \qquad (21)$$

where $\Delta\phi(X_k, y, H) = \phi^d(X_k, y_k, H_k^*) - \phi(X_k, y, H)$. We calculate α^* by maximizing the dual form in standard SVM, and we apply the cutting plane method [11] to solve it.

With the estimated parameters ω_t^d, the energy $E(\omega_t^d)$ can be calculated for the new model, and we then compare it with the previous energy $E(\omega_t)$ to verify the new model structures. If $E(\omega_t^d) < E(\omega_t)$, we accept the new model structures and have $\omega_{t+1} \leftarrow \omega_t^d$. Otherwise, we keep the model structures as in the previous iteration and optimize the model parameters without the structure reconfiguration, i.e. by using q_t instead: $\omega_{t+1} = argmin_\omega[f(\omega) + \omega \cdot q_t]$.

In this way, we ensure that the optimization objective in Equation (14) continues to decrease in iterations. Thus, the algorithm keeps iterating until the objective converges.

5.3 Initialization

At the beginning of model training, our model can be initialized as follows. For each training example, whose contours have been extracted, we partition it into a regular layout of partitioned blocks, and each block is corresponding to one or-node. The contours that fall into the block are treated as the inputs, and we initially select the one with the largest length if more than one contour are within there. Then, the leaf-nodes are initially generated by clustering the selected contours without any constraints. The and-nodes are initialized by the similar way. We thus obtain the initial feature vectors for all training examples.

Algorithm 2 summarizes the overall algorithm of learning the latent And-Or graph.

6 EXPERIMENTS

To validate the advantage of our model, we present a new shape database, SYSU-Shapes[1], which includes elaborately annotated shape contours. Compared with the existing shape databases, this database includes more realistic challenges in shape detection and localization, e.g. cluttered background, large intraclass variation, and different poses/views, in which part of the instances were originally used for appearance-based object detection. We also validate our model on two other public databases: UIUC-People [33] and INRIA-Horse [12] and show the superior performances over other state-of-the-art methods.

Implementation setting. We extract clutter-free object contours for the positive samples, and the edge maps for the negative samples are extracted using the Pb edge detector [23] with an edge link algorithm. For each contour as the input of the leaf-node, we sample 20 points and compute the contour descriptor for each point. During detection, the edge maps of test images are extracted

1. http://vision.sysu.edu.cn/projects/discriminative-aog/

Algorithm 2: Learning latent And-Or graph model

Input:
  positive and negative training samples,
  $\{X_k, y_k\}^+ \bigcup \{X_{k'}, y_{k'}\}^-, k = 1..K, k' = K+1..N$.
Output:
  The trained And-Or graph model.
Initialization:
  1 Initialize the model structure (the arrangement of nodes).
  2 Initialize the latent variables H and model parameters ω.
repeat
  1 Estimate the latent variables H^* on each positive example (X_k, y_k) with the current model parameters ω_t.
  2 Generate the new graph structures.
    (a) Localize the contour fragments for all examples using the current latent variables H_k^*, and obtain the feature vectors $\phi(X_k, y_k, H_k^*)$.
    (b) In the layer of and-nodes, perform clustering on the global shape features, and rearrange the feature vectors.
    (c) For each or-node U_i, perform clustering on the feature vectors of all its child leaf-nodes.
    (d) Operate on the leaf-nodes to generate a new structure, and the latent variable is updated to H_k^d with the rearranged feature vectors $\phi^d(X_k, y_k, H_k^d)$.
  3 Update the model parameters ω_{t+1}.
    (a) Estimate the parameters ω_t^d with the newly generated structures.
    (b) IF $E(\omega_t^d) < E(\omega_t)$,
        Accept the new model structures, and $\omega_{t+1} \leftarrow \omega_t^d$.
      ELSE
        Calculate ω_{t+1} while keeping the structures in the previous iteration.
until The target function defined in Equation (14) converges.

	SYSU-Shapes	UIUC-People	INRIA-Horses
and-nodes	$m = 3$	$m = 2$	$m = 1$
or-nodes	$z = 6$	$z = 8$	$z = 6$
leaf-nodes	$n \leq 4$	$n \leq 4$	$n \leq 4$

TABLE 2
Numbers of nodes in the and-or graph models for different databases.

---

as for the negative training samples. The objects are searched by sliding windows over 6 different scales and 2 per octave, and detections are reported by non-maximum suppression. We adopt the testing criterion defined in the PASCAL VOC challenge: a detection is counted as correct if its overlap with the groundtruth bounding-box is greater than 50%.

Our model is able to flexibly adapt to the data by setting the numbers of nodes in each layer: m for and-nodes, z for or-nodes, and n for leaf-nodes. Recall that each or-node in our model indicates a part of object shape, so that we can set the number of or-nodes according to the sizes (scales) of the shape categories. The leaf-nodes are produced during the iterative training, and their numbers can be determined automatically. In the experiments, to reduce computational cost, we fix the number for and-nodes and set an upper limit for the number of leaf-nodes. Table 2 summarizes the numbers of nodes on the three databases. In the model training, the initial layout for each sample is a regular partition
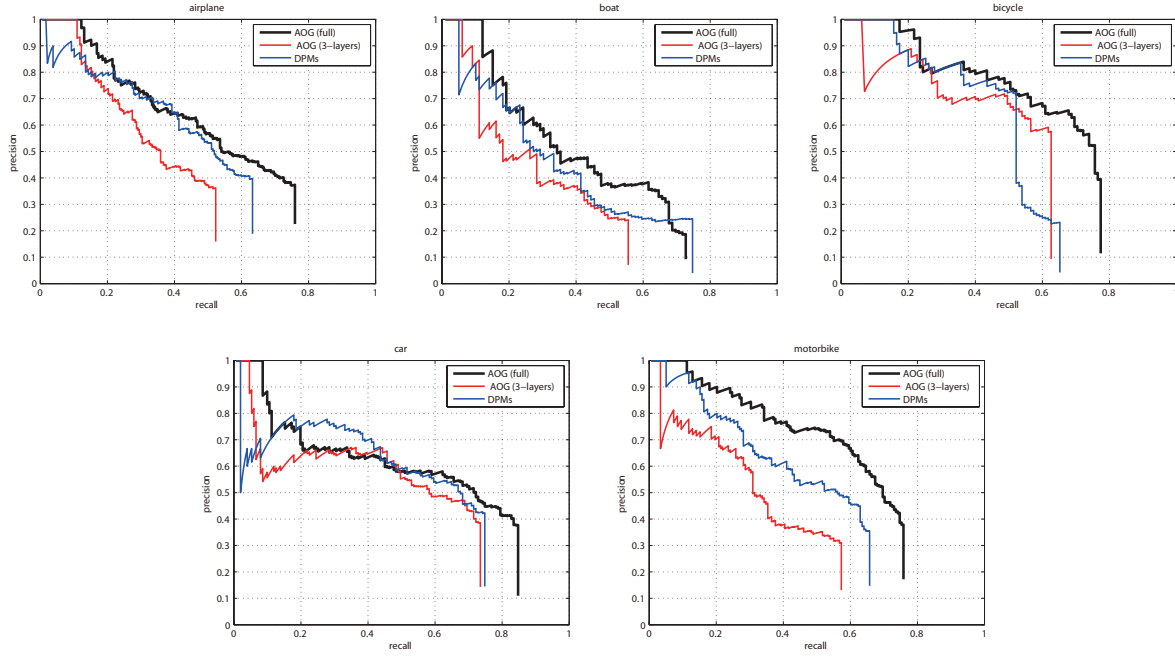
Fig. 9. Precision-Recall (PR) curves on the SYSU-Shape dataset.

Method	Airplane	Bicycle	Boat	Car	Motorbike	MeanAP
AOG (full)	**0.520**	**0.623**	**0.419**	**0.549**	**0.583**	**0.539**
AOG (3-layers)	0.348	0.482	0.288	0.466	0.333	0.383
DPMs	0.437	0.488	0.365	0.509	0.455	0.451

TABLE 3
Detection accuracies on the SYSU-Shape dataset.

(e.g. 2×4 blocks for the UIUC-People dataset and 3×2 for the other two datasets).

If we keep only one and-node (i.e. $m = 1$), our model is simplified into a 3-layer structure that is rooted by the and-node. The training procedure (i.e. Algorithm. 2) for this structure is kept, but we discard the step of generating and-nodes.

We conduct the experiments on a workstation with Core Duo 3.0 GHZ CPU and 16GB memory. On average, it takes $4 \sim 8$ hours to train a shape model, depending on the numbers of training examples, and the time cost for detection on an image is around $1 \sim 2$ minutes.

Experiment I. We first conduct the experiment on the SYSU-Shape database, which is collected from the Internet and other vision databases. There are 5 categories, i.e. airplanes, boats, cars, motorbikes, and bicycles, and each category contains $200 \sim 500$ images. The shape contours are carefully labeled by a professional team using the LabelMe toolkit [26]. It is worth mentioning that each image has at least but not limited to one object of a given category. For each category, half of the images are randomly selected as positive samples and the rest for testing. The images from the other categories are randomly split into two halves as negative samples for training and testing.

For comparison, we apply the well acknowledged deformable part-based models (DPMs) [7] on this database, where we modify the released code by replacing the input feature with our shape descriptor, and keep the other settings. In this implementation, 3 DPMs are merged into a mixture, which accounts for different object views. Moreover, we simplify the model into a 3-layer configuration by setting $m = 1$, and test its performances. Figure 9 shows the Precision-Recall curves for all 5 categories, and the Average Precision values are reported in Table 3. Our complete model achieves the best mean AP and the best APs for all 5 categories, and the results clearly demonstrate the benefit of using the layered And-Or structures. Several representative detection results are exhibited in Figure 10.

Experiment II. The UIUC-People dataset contains 593 images (346 for training, 247 for testing) that are very challenging due to large shape variations caused by different views and human poses. Most of the images contain people playing badminton. The existing methods [37], [4] that are tested on this dataset usually rely on rich appearance-based image features and/or manually labeled prior models. To the best of our knowledge, this work is the first shape-based detector to achieve comparable performances on this dataset. Figure 11(a) shows
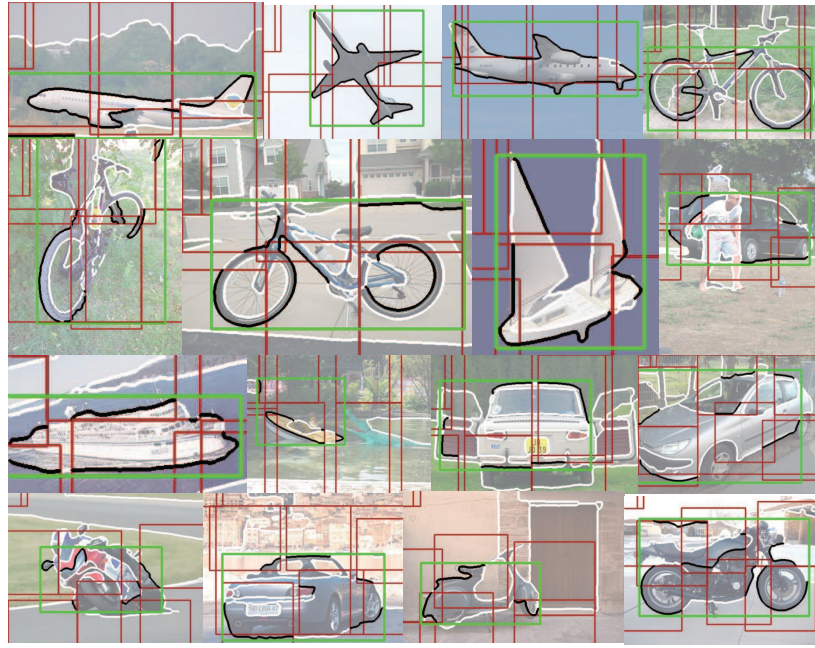
Fig. 10. A few typical object shape detections generated by our approach on the SYSU-Shape dataset. The localized contours are highlighted in black, and the green boxes and red boxes indicate detected shapes and their parts, respectively.
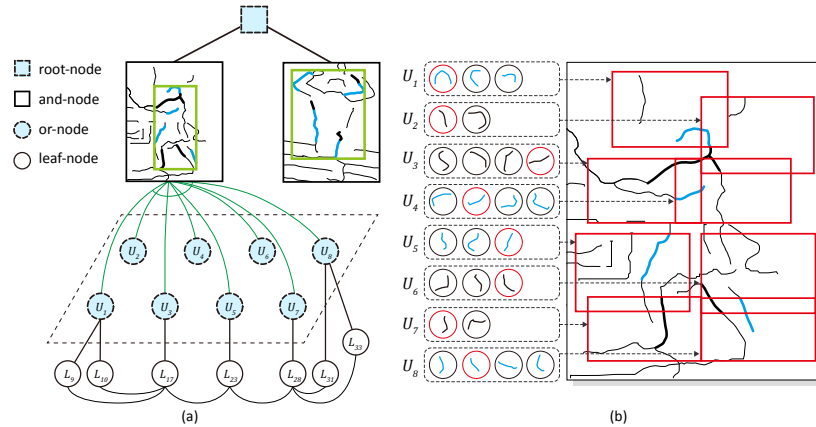

Fig. 11. The trained And-Or graph model with the UIUC-People dataset. (a) Visualizes the model of 4 layers. (b) Exhibits leaf-nodes associated with or-nodes, $U_1, \ldots, U_8$. A real detection case with the activated leaf-nodes are highlighted in red.

Method	Accuracy
AOG model	**0.708**
Wang et al. [37]	0.668
Andriluka et al. [1]	0.506
Felz et al. [7]	0.486
Bourdev et al. [4]	0.458

TABLE 4
Comparisons of detection accuracies on the
UIUC-People dataset.

the trained And-Or model (AOG), which includes 2 and-nodes and 8 or-nodes, and each or-node is associated with $2 \sim 4$ leaf-nodes. Since most of the images contain one person, we only consider the detection with the highest score on an image for all of the methods. Table 4 reports the quantitative detection accuracies generated by our method and the competing approaches [37], [1], [4], [7]. The results (except ours) come from [37]. A number of representative detection results are presented in Figure 12, where the localized contours are highlighted in black, and the green boxes and red boxes indicate detected human and parts, respectively. We also present several inaccurate detections indicated by the blue boxes in Figure 12. There are two main reasons for the failure cases: (i) False positives are sometimes created by the background contours segments that appear like the objects-of-interest very much. (ii) The object contours are
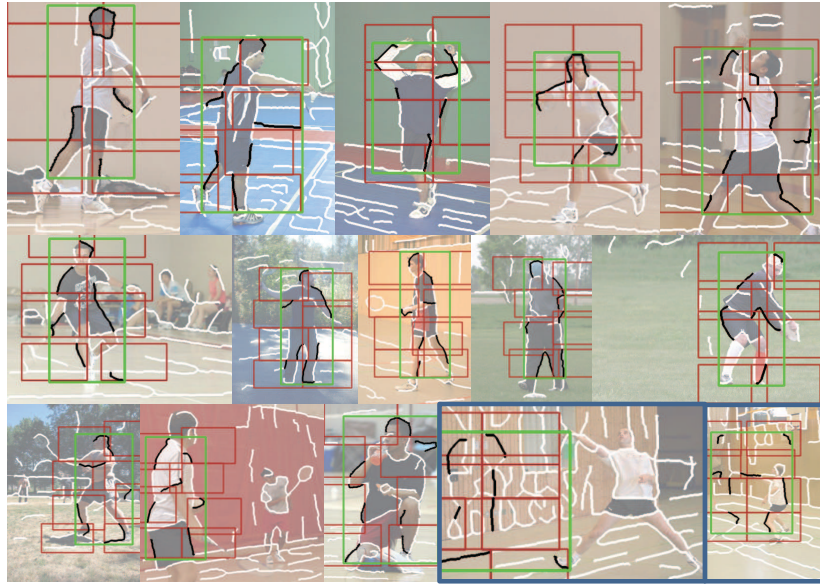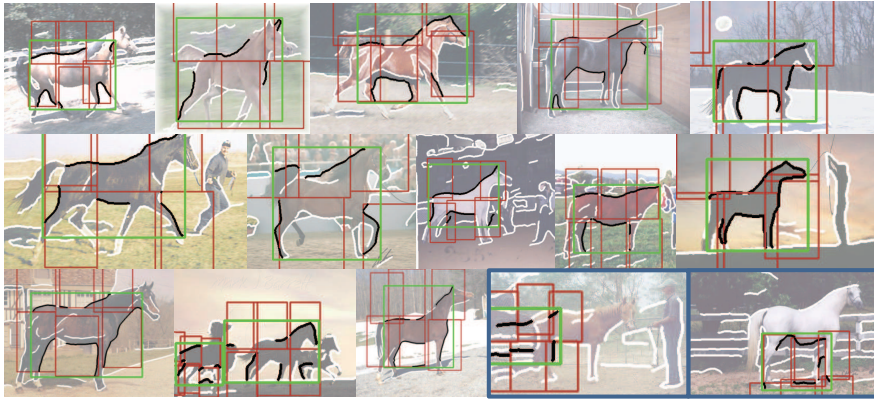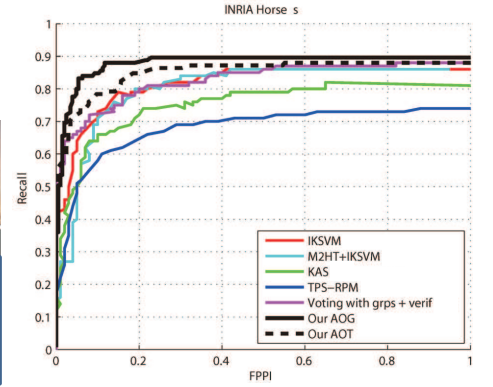
Fig. 12. A few typical object shape detections generated by our method on the UIUC-People database [33]. The localized contours are highlighted in black, and the green boxes and red boxes indicate detected peoples and parts, respectively. Two failure detections are indicated by the blue boxes.

(a)   (b)

Fig. 13. Results on the INRIA-Horse database. (a) shows several detected shapes by our method, where the localized contours are highlighted in black, and two failure detections are indicated by the blue boxes. (b) shows the quantitative results with the recall-FPPI measurement.

insufficiently discriminative for recognition, particularly with unconventional object poses and views.

Experiment III. The INRIA Horse dataset comprise 170 horse images and 170 images without horses. The challenges of this dataset arise from background clutter and large deformations, and some of the images contain more than one horse. Following the common experiment setting, we use 50 positive examples and 80 negative examples for training and the remaining 210 images for testing.

Some typical shape detection results on the INRIA Horse dataset are shown in Figure 13(a). Compared with existing approaches, we use the recall-FPPI (false positive per image) curves for evaluation, as Figure 13(b) reports. It is shown that our approach (denoted as AOG)

substantially outperforms the competing methods. Our model achieves detection rates of 89.6% at 1.0 FPPI; in contrast, the results of competing methods are: 87.3% in [41], 85.27% in [22], 80.77% in [10], and 73.75% in [6].

Empirical analysis. For further evaluation, we present two empirical analysis under different model settings as follows.

(I) We validate the benefit of the contextual collaborative edges. Our model can be further transferred into a tree structure by removing the interactions, which is denoted as "And-Or Tree (AOT)". On the UIUC-People dataset, the detection accuracy of the AOT model is 0.69, which is lower than the complete form of our model, but it is also comparable to the state-of-arts. On the INRIA-Horse dataset, we also present the results yielded by the

AOT model in Figure 13(b). Based on these results, we can observe that the collaborative edges effectively boost the detection against disturbing surrounding clutter and occlusions.
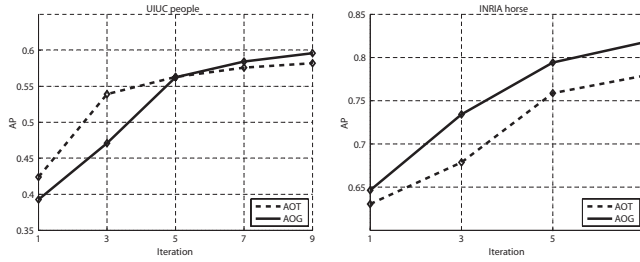

Fig. 14. Model capabilities during the iterative training. We plot the average-precision (AP) with the increasing iterations: the intermediate performances of our models in the iteration steps. We conduct the experiments on the UIUC-People dataset (on the left) and INRIA Horse dataset (on the right). The results of disabling the collaborative edges are also reported.

(II) To analyze the model capacity during the iterative training, we output the intermediate performance measures of our models in the iteration steps.

We execute the experiments on the UIUC-People and the INRIA-Horse databases. The quantitative results represented by average precisions (APs) are visualized in Figure 14. We also report the results generated by the models without collaborative edges, i.e. AOT models. We observe that the discriminative capabilities of our model increase proportinately with the iterations, and converge after a few rounds.

7 CONCLUSION AND FUTURE WORK

In this paper, we have introduced, first, a hierarchical and reconfigurable object shape model in the form of an And-Or graph representation. Second, an efficient inference algorithm for shape detection with the proposed model. Third, a principled learning method that iteratively determine the model structures while optimizing multi-layer parameters. We demonstrated the practical applicability of our approach by effectively detecting and localizing object shapes from cluttered edge maps. Our model effectively captured large shape variations in deformation for different views and poses. Experiments were implemented on several very challenging databases, (e.g. SYSU-Shapes, UIUC-People, and INRIA-Horse), and our model outperformed other current state-of-the-art approaches.

There are several directions in which we intend to extend this work. The first is to complement our contour-based features with rich appearance information, thereby adapting our model to more general object recognition. The second is to generalize our model in the context of multiclass recognition and investigate part-based structure sharing among classes. For example, the feet of horse and sheep have similar appearances, and thus can be detected by the same local classifier, that is, we can make local classifiers (i.e. the leaf-nodes in our model) shared across categories. Model sharing will keep the model compact while representing multiple categories. Moreover, the inference algorithm will be revised accordingly, to deal with a large number of candidate compositions.

REFERENCES

[1] M. Andriluka, S. Roth, and B. Schiele, Pictorial structures revisited: People detection and articulated pose estimation, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[2] X. Bai, X. Yang, L. J. Latecki, W. Liu, and Z. Tu, Learning Context Sensitive Shape Similarity by Graph Transduction, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(5): 861-874, 2010.

[3] S. Belongie, J. Malik, and J. Puzicha, Shape Matching and Object Recognition using Shape Contexts, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1): 705-522, 2002.

[4] L. Bourdev, S. Maji, T. Brox, and J. Malik, Detecting people using mutually consistent poselet activations, In *Proc. of European Conference on Computer Vision (ECCV)*, 2010.

[5] C. Desai, D. Ramanan, and C. C. Fowlkes, Discriminative Models for Multi-Class Object Layout, *International Journal of Computer Vision*, 2011.

[6] V. Ferrari, F. Jurie, and C. Schmid, From Images to Shape Models for Object Detection, *International Journal of Computer Vision*, 87(3): 284-303, 2010.

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, Object Detection with Discriminatively Trained Part-based Models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9): 1627-1645, 2010.

[8] P. Felzenszwalb, and J. D. Schwartz, Hierarchical Matching of Deformable Shapes, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[9] Felzenszwalb, Pedro F., and Daniel P. Huttenlocher. Pictorial structures for object recognition, *International Journal of Computer Vision*, 61(1)55-79, 2005.

[10] V. Ferrari, L. Fevrier, F. Jerie, and C. Schmid, Groups of Adjacent Contour Segments for Object Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1): 36-51, 2008.

[11] V. Franc and S. Sonnenburg, Optimized Cutting Plane Algorithm for Support Vector Machines, In *Proc. of International Conference on Machine Learning (ICML)*, 2008.

[12] F. Jurie and C. Schmid, Scale-invariant Shape Features for Recognition of Object Categories, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[13] I. Kokkinos, and A. Yuille, Inference and Learning with Hierarchical Shape Models, *International Journal of Computer Vision*, 93: 201-225, 2011.

[14] L. Lin, X. Liu, and S.C. Zhu, Layered Graph Matching with Composite Cluster Sampling, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8): 1426-1442, 2010.

[15] L. Lin, T. Wu, J. Porway, and Z. Xu, A Stochastic Graph Grammar for Compositional Object Representation and Recognition, *Pattern Recognition*, 42(7): 1297-1307, 2009.

[16] L. Lin, S. Peng, J. Porway, S.C. Zhu, and Y. Wang, An Empirical Study of Object Category Recognition: Sequential Testing with Generalized Samples, In *Proc. of International Conference on Computer Vision (ICCV)*, 2007.

[17] L. Lin, X. Wang, W. Yang, and J. Lai, Learning Contour-Fragment-based Shape Model with And-Or Tree Representation, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] H. Ling, and D.W. Jacobs, Shape Classification Using the Inner-Distance, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):286-299, 2007.

[19] P. Luo, L. Lin, and H. Chao, Learning Shape Detector by Quantizing Curve Segments with Multiple Distance Metrics, In *Proc. of European Conference on Computer Vision (ECCV)*, 2010.

[20] C. Lu, L. J. Latecki, N. Adluru, X. Yang, and H. Ling, Shape Guided Contour Grouping with Particle Filters, In *Proc. of International Conference on Computer Vision (ICCV)*, 2009.

[21] T. Ma and L. J. Latecki, From Partial Shape Matching through Local Deformation to Robust Global Shape Similarity for Object Detection, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[22] S. Maji and J. Malik, Object Detection using a Max-Margin Hough Transform, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[23] D. Martin, C. Fowlkes, and J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5): 530-549, 2004.

[24] A. Opelt, A. Pinz, and A. Zisserman, A Boundary-Fragment-Model for Object Detection, *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[25] H. Riemenschneider, M. Donoser, and H. Bischof, Using Partial Edge Contour Matches for Efficient Object Category Localization, In *Proc. of European Conference on Computer Vision (ECCV)*, 2009.

[26] B. C. Russell, A. Torralba, K. P. Murphy, W. T. Freeman, LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, pages 157-173, Volume 77, Numbers 1-3, May, 2008.

[27] P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele, Discriminative structure learning of hierarchical representations for object detection, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[28] J. Shotton, A. Blake, and R. Cipolla, Multi-Scale Categorical Object Recognition Using Contour Fragments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7): 1270-1281, 2008.

[29] Z. Si, M. Pei, Z.Y. Yao, and S.C. Zhu, Unsupervised Learning of Event And-Or Grammar and Semantics from Video, In *Proc. of International Conference on Computer Vision (ICCV)*, 2011.

[30] Z. Si, and S. C. Zhu, Learning And-Or Templates for Object Modeling and Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[31] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, Contextualizing Object Detection and Classification, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[32] P. Srinivasan, Q. Zhu, and J. Shi, Many-to-one Contour Matching for Describing and Discriminating Object Shape, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[33] D. Tran and D. Forsyth, Improved human parsing with a full relational model, In *Proc. of European Conference on Computer Vision (ECCV)*, 2010.

[34] Z. Tu, S. Zheng, and A. Yuille, Shape Matching and Registration by Data-driven EM, *Computer Vision and Image Understanding*, 109(3): 290-304, 2008.

[35] Y. Wu, Z. Si, H. Gong, and S.C. Zhu, Learning Active Basis Model for Object Detection and Recognition, *International Journal of Computer Visison*, 90(2): 198-235, 2010.

[36] X. Wang, and L. Lin, Dynamical And-Or Graph Learning for Object Shape Modeling and Detection, In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2012.

[37] Y. Wang, D. Tran, and Z. Liao, Learning Hierarchical Poselets for Human Parsing, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[38] J. De Winter and J. Wagemans. Contour-based object identification and segmentation: stimuli, norms and data, and software tools, *Behavior Research Methods, Instruments, and Computers*, 36(4): 604-624, 2004.

[39] C. Xu, J. Liu, and X. Tang, 2D Shape Matching by Contour Flexibility, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1): 180-186, 2009.

[40] X. Yang and L. J. Latecki, Weakly Supervised Shape Based Object Detection with Particle Filter, In *Proc. of European Conference on Computer Vision (ECCV)*, 2010.

[41] P. Yarlagadda, A. Monroy and B. Ommer, Voting by Grouping Dependent Parts, In *Proc. of European Conference on Computer Vision (ECCV)*, 2010.

[42] C.-N. J. Yu, and T. Joachims, Learning structural svms with latent variables, In *Proc. of International Conference on Machine Learning (ICML)*, 2009.

[43] A.L. Yuille, and A. Rangarajan, The Concave-Convex Procedure (CCCP). *Neural Computation*, 15(4): 915-936, 2003.

[44] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, Latent Hierarchical Structural Learning for Object Detection, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[45] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. Yuille, Max Margin AND/OR Graph learning for parsing the human body, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[46] S.C. Zhu and D. Mumford, A stochastic grammar of images, *Foundations and Trends in Computer Graphics and Vision*, 2(4): 259-362, 2006.

[47] Q. Zhu, L. Wang, Y. Wu, and J. Shi, Contour Context Selection for Object Detection: A Set-to-Set Contour Matching Approach, In *Proc. of European Conference on Computer Vision (ECCV)*, 2008.

Liang Lin is a full Professor with the School of Advanced Computing, Sun Yat-Sen University (SYSU), China. He received the B.S. and Ph.D. degrees from the Beijing Institute of Technology (BIT), Beijing, China, in 1999 and 2008, respectively. From 2006 to 2007, he was a joint Ph.D. student with the Department of Statistics, University of California, Los Angeles (UCLA). He was a Post-Doctoral Research Fellow with the Center for Vision, Cognition, Learning, and Art of UCLA. His research focuses on new models, algorithms and systems for intelligent processing and understanding of visual data such as images and videos. He has published more than 20 papers in highly ranked academic journals and more than 30 papers in top tier conferences CVPR, ICCV, ECCV, ACM MM and NIPS. He was supported by several promotive programs or funds for his works, such as "Program for New Century Excellent Talents" of Ministry of Education (China) in 2012 and Guangdong Distinguished Young Scholar Fund in 2013. He was a recipient of Best Paper Runners-Up Award in ACM NPAR 2010, Google Faculty Award in 2012, and Best Student Paper Award in IEEE ICME 2014.

Xiaolong Wang received the BS degree from the School of Information, South China Agricultural University in 2011, and the MS degree in Computer Science from Sun Yat-sen University in 2014. His research interest include computer vision and machine learning.
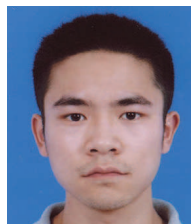
Wei Yang received the BS degree in Software Engineering from Sun Yat-sen University in 2011, and the MS degree in Computer Science from Sun Yat-sen University in 2014. His research interest include computer vision and machine learning.

Jianhuang Lai received his M.Sc. degree in applied mathematics in 1989 and his Ph.D. in mathematics in 1999 from SUN YAT-SEN University, China. He joined Sun Yat-sen University in 1989 as an Assistant Professor, where currently, he is a Professor with the Department of Automation of School of Information Science and Technology and dean of School of Information Science and Technology. His current research interests are in the areas of digital image processing, pattern recognition, multimedia communication, wavelet and its applications.