Dynamical And-Or Graph Learning for Object Shape Modeling and Detection

Xiaolong Wang
Sun Yat-Sen University
Guangzhou, P.R. China 510006
dragonwxl123@gmail.com

Liang Lin*
Sun Yat-Sen University
Guangzhou, P.R. China 510006
linliang@ieee.org

Abstract

This paper studies a novel discriminative part-based model to represent and recognize object shapes with an "And-Or graph". We define this model consisting of three layers: the leaf-nodes with collaborative edges for localizing local parts, the or-nodes specifying the switch of leaf-nodes, and the root-node encoding the global verification. A discriminative learning algorithm, extended from the CCCP [23], is proposed to train the model in a dynamical manner: the model structure (e.g., the configuration of the leaf-nodes associated with the or-nodes) is automatically determined with optimizing the multi-layer parameters during the iteration. The advantages of our method are two-fold. (i) The And-Or graph model enables us to handle well large intra-class variance and background clutters for object shape detection from images. (ii) The proposed learning algorithm is able to obtain the And-Or graph representation without requiring elaborate supervision and initialization. We validate the proposed method on several challenging databases (e.g., INRIA-Horse, ETHZ-Shape, and UIUC-People), and it outperforms the state-of-the-arts approaches.

1 Introduction

Part-based and hierarchical representations have been widely studied in computer vision, and lead to some elegant frameworks for complex object detection and recognition. However, most of the methods address only the hierarchical decomposition by tree-structure models [5, 25], and oversimplify the reconfigurability (i.e. structural switch) in hierarchy, which is the key to handle the large intra-class variance in object detection. In addition, the interactions of parts are often omitted in learning and detection. And-Or graph models are recently explored in [26, 27] to hierarchically model object categories via "and-nodes" and "or-nodes" that represent, respectively, compositions of parts and structural variation of parts. Their main limitation is that the learning process is strongly supervised and the model structure needs to be manually annotated.

The key contribution of this work is a novel And-Or graph model, whose parameters and structure can be jointly learned in a weakly supervised manner. We achieve the superior performance on the task of detecting and localizing shapes from cluttered backgrounds, compared to the state-of-the-art approaches. As Fig. 3(a) illustrates, the proposed And-Or graph model consists of three layers described as follows.

The **leaf-nodes** in the bottom layer represent a batch of local classifiers of contour fragments. We provide a partial matching scheme that can recognize the accurate part of the contour, to deal with

*Corresponding author is Liang Lin. This work was supported by National Natural Science Foundation of China (no. 61173082), Fundamental Research Funds for the Central Universities (no. 2010620003162041), and the Guangdong Natural Science Foundation (no.S2011010001378).This work was also partially funded by SYSU-Sugon high performance computing typical application project.

the problem that the true contours of objects are often connected to background clutters due to unreliable edge extraction.

The **or-nodes** in the middle layer are "switch" variables specifying the activation of their children leaf-nodes. We utilize the or-nodes accounting for alternate ways of composition, rather than just defining multi-layer compositional detectors, which is shown to better handle the intra-class variance and inconsistency caused by unreliable edge detection. Each or-node is used to select one contour from the candidates detected via the associated leaf-nodes in the bottom layer. Moreover, during detection, location displacement is allowed for each or-node to tackle the part deformation.

The **root-node** (i.e. the and-node) in the top layer is a global classifier capturing the holistic deformation of the object. The contours selected via the or-nodes are further verified as a whole, in order to make the detection robust against the background clutters.

The **collaborative edges** between leaf-nodes are defined by the probabilistic co-occurrence of local classifiers, which relax the conditional independence assumption commonly used in previous tree structure models. Concretely, our model allows nearby contours to interact with each other.

The key problem of training our And-Or graph model is automatic structure determination. We propose a novel learning algorithm, namely dynamic CCCP , extended from the concave-convex procedure (CCCP) [23, 22] by embedding the structural reconfiguration. It iterates to dynamically determine the production of leaf-nodes associated with the or-nodes, which is often simplified by manually fixing in previous methods [25, 16]. The other structure attributes (e.g., the layout of or-nodes and the activation of leaf-nodes) are implicitly inferred with the latent variables.

2 Related Work

Remarkable progress has been made in shape-based object detection [6, 10, 9, 11, 19]. By employing some shape descriptors and matching schemes, many works represent and recognize object shapes as a loose collection of local contours. For example, Ferrari et al. [6] used a codebook of PAS (pairwise adjacent segments) to localize object of interest; Maji et al. [11] proposed a maximum margin hough voting for hypothesis regions combining with intersection kernel SVM(IKSVM) for verification; Yang and Latecki [19] constructed shape models in a fully connected graph form with partially-supervised learning, and detected objects via a Particle Filters (PF) framework.

Recently, the tree structure latent models [25, 5] have provided significant improvements on object detection. Based on these methods, Srinivasan et al. [16] trained the descriptive contour-based detector by using the latent-SVM learning; Song et al. [15] integrated the context information with the learning, namely Context-SVM. Schnitzspan et al. [14] further combined the latent discriminative learning with conditional random fields using multiple features.

Knowledge representation with And-Or graph was first introduced for modeling visual patterns by Zhu and Mumford [27]. Its general idea, i.e. using configurable graph structures with And, Or nodes, has been applied in object and scene parsing [26, 18, 24] and action classification [20].

3 And-Or Graph Representation for Object Shape

The And-Or Graph model is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents three types of nodes and $\mathcal{E}$ the graph edges. As Fig. 3(a) illustrates, the square on the top is the root-node representing the complete object instances. The dashed circles derived from the root are z or-nodes arranged in a layout of $b_1 \times b_2$ blocks, representing the object parts. Each or-node comprises an unfixed number of leaf-nodes (denoted by the solid circles on the bottom); the leaf-nodes are allowed to be dynamically created and removed during the learning. For simplicity, we set the maximum number m of leaf-nodes affiliated to one or-node, and the parameters of non-existing leaf-nodes to zero. Then the maximum number of all nodes in the model is $1 + n = 1 + z + z \times m$. We use $i = 0$ indexing the root node, $i = 1, ..., z$ the or-nodes and $j = z + 1, ..., n$ the leaf-nodes. We also define that $j \in ch(i)$ indexes the child nodes of node i. The horizontal graph edges (i.e., collaborative edges) are defined between the leaf-nodes that are associated with different or-nodes, in order to encode the compatibility of object parts. The definitions of $\mathcal{G}$ are presented as follows.

Leaf-node: Each leaf-node $L_j, j = z + 1, ..., n$ is a local classifier of contours, whose placement is decided by its parent or-node (the localized block). Suppose a contour fragment c on the edge map X is captured by the block located at $p_i = (p_i^x, p_i^y)$, as the input of classifier. We denote $\phi^l(p_i, c)$ as

the feature vector using the Shape Context descriptor [3]. For any classifier, only the part of c fallen into the block will be taken into account, and we set $\phi^l(p_i, c) = 0$ if c is entirely out. The response of classifier L_j at location p_i of the edge map X is defined as:

$$\mathcal{R}_{L_j}(X, p_i) = \max_{c \in X} \omega_j^l \cdot \phi^l(p_i, c), \tag{1}$$

where ω_j^l is a parameter vector, which is set to zero if the corresponding leaf-node L_j is nonexistent. Then we can detect the contour from edge map X via the classifier, $c_j = argmax_{c \in X} \omega_j^l \cdot \phi^l(p_i, c)$.

Or-node: Each or-node $U_i, i = 1, ..., z$ is proposed to specify a proper contour from a set of candidates detected via its children leaf-nodes. Note that we can also consider the or-node activating one leaf-node. The or-nodes are allowed to perturb slightly with respect to the root. For each or-node U_i, we define the deformation feature as $\phi^s(p_0, p_i) = (dx, dy, dx^2, dy^2)$, where (dx, dy) is the displacement of the or-node position p_i to the expected position p_0 determined by the root-node. Then the cost of locating U_i at p_i is:

$$Cost_i(p_0, p_i) = -\omega_i^s \cdot \phi^s(p_0, p_i), \tag{2}$$

where ω_i^s is a 4-dimensional parameter vector corresponding to $\phi^s(p_0, p_i)$. In our method, each or-node contains at most m leaf-nodes, among which one is to be activated during inference. For each leaf-node L_j associated with U_i, we introduce an indicator variable $v_j \in \{0, 1\}$ representing whether it is activated or not. Then we derive the auxiliary "switch" vector for U_i, $\mathbf{v}_i = (v_{j_1}, v_{j_2}, ..., v_{j_m})$, where $||\mathbf{v}_i|| = 1$. Thus, the response of the or-node U_i is defined as,

$$\mathcal{R}_{U_i}(X, p_0, p_i, \mathbf{v}_i) = \sum_{j \in ch(i)} \mathcal{R}_{L_j}(X, p_i) \cdot v_j + Cost_i(p_0, p_i). \tag{3}$$

Collaborative Edge: For any pair of leaf-nodes $(L_j, L_{j'})$ respectively associated with two different or-nodes, we define the collaborative edge between them according to their contextual co-occurrence. That is, how likely it is that the object contains contours detected via the two leaf-nodes. The response of the pairwise potentials is parameterized as,

$$\mathcal{R}_E(V) = \sum_{j=z+1}^{n} \sum_{j' \in neigh(j)} \omega_{(j,j')}^e \cdot v_j \cdot v_{j'}, \tag{4}$$

where $neigh(j)$ is defined as the neighbor leaf-nodes from the other or-node adjacent (in spatial direction) to L_j, and V is a joint vector for each $\mathbf{v}_i$: $V = (\mathbf{v}_1, ..., \mathbf{v}_z) = (v_{z+1}, ..., v_n)$. $\omega_{(j,j')}^e$ indicates the compatibility between L_j and $L_{j'}$.

Root-node: The root-node represents a global classifier to verify the ensemble of contour fragments $C^r = \{c_1, ..., c_z\}$ proposed by the or-nodes. The response of the root-node is parameterized as,

$$\mathcal{R}_T(C^r) = \omega^r \cdot \phi^r(C^r), \tag{5}$$

where $\phi^r(C^r)$ is the feature vector of C^r and ω^r the corresponding parameter vector.

Therefore, the overall response of the And-Or graph is:

$$\mathcal{R}_G(X, P, V) = \sum_{i=1}^{a} \mathcal{R}_{U_i}(X, p_0, p_i, \mathbf{v}_i) + \mathcal{R}_E(V) + \mathcal{R}_T(C^r)$$

$$= \sum_{i=1}^{z} [\sum_{j \in ch(i)} \omega_j^l \cdot \phi^l(p_i, c_j) \cdot v_j - \omega_i^s \cdot \phi^s(p_0, p_i)] + \sum_{j=z+1}^{n} \sum_{j' \in neigh(j)} \omega_{(j,j')}^e \cdot v_j \cdot v_{j'} + \omega^r \cdot \phi^r(C^r), \tag{6}$$

where $P = (p_0, p_1, ..., p_z)$ is a vector of the positions of or-nodes. For better understanding, we refer $H = (P, V)$ as the latent variables during inference, where P implies the deformation of parts represented by the or-nodes and V implies the discrete distribution of leaf-nodes (i.e., which leaf-nodes are activated for detection). The Eq.(6) can be further simplified as :

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

where ω includes the complete parameters of And-Or graph, and $\phi(X, H)$ is the feature vector,

$$\omega = (\omega_{z+1}^l, ..., \omega_n^l, -\omega_1^s, ..., -\omega_z^s, \omega_{(z+1,z+1+m)}^e, ..., \omega_{(n-m,n)}^e, \omega^r). \tag{8}$$

$$\phi(X, H) = (\phi^l(p_1, c_{z+1}) \cdot v_{z+1}, \cdots, \phi^l(p_z, c_n) \cdot v_n, \\ \phi^s(p_0, p_1), \cdots, \phi^s(p_0, p_z), v_{z+1} \cdot v_{z+1+m}, ..., v_{n-m} \cdot v_n, \phi^r(C^r)). \tag{9}$$

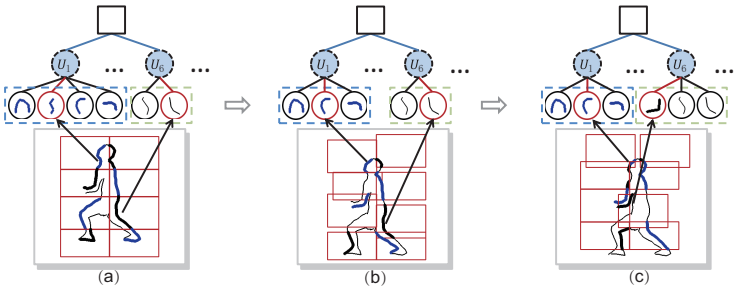

Figure 1: Illustration of dynamical structure learning. Parts of the model, two or-nodes (U_1, U_6), are visualized in three intermediate steps. (a) The initial structure, i.e., the regular layout of an object. Two new structures are dynamically generated during iteration. (b) A leaf-node associated with U_1 is removed. (c) A new leaf-node is created and assigned to U_6.

4 Inference

The inference task is to localize the optimal contour fragments within the detection window, which is slidden at all scales and positions of the edge map X. Assuming the root-node is located at p_0, the object shape is localized by maximizing $\mathcal{R}_G(X, H)$ defined in (6):

$$S(p_0, X) = \max_H \mathcal{R}_G(X, H). \tag{10}$$

The inference procedure integrates the bottom-up testing and top-down verification:

Bottom-up testing: For each or-node U_i, its children leaf-nodes (i.e. the local classifiers) are utilized to detect contour fragments within the edge map X. Assume that leaf-node $L_j, j \in ch(i)$ associated with U_i is activated, $v_j = 1$, and the optimal contour fragment c_j is localized by maximizing the response in Eq.(3), where the optimal location $p^*_{i,j}$ is also determined. Then we generate a set of candidates for each or-node, $\{c_j, p^*_{i,j}\}$, each of which is one detected contour fragments via the leaf-nodes. These sets of candidates will be passed to the top-down step where the leaf-node activation $\mathbf{v}_i$ for U_i can be further validated. We calculate the response for the bottom-up step, as,

$$\mathcal{R}_{bot}(V) = \sum_{i=1}^{z} \mathcal{R}_{U_i}(X, p_0, p^*_i, \mathbf{v}_i), \tag{11}$$

where $V = \{\mathbf{v}_i\}$ denotes a hypothesis of leaf-node activation for all or-nodes. In practice, we can further prune the candidate contours by setting a threshold on $\mathcal{R}_{bot}(V)$. Thus, given the $V = \{\mathbf{v}_i\}$, we can select an ensemble of contours $C^r = \{c_1, ..., c_z\}$, each of which is detected by an activated leaf-node, $L_j, v_j = 1$.

Top-down verification: Given the ensemble of contours C^r, we then apply the global classifier at the root-node to verify C^r by Eq. (5), as well as the accumulated pairwise potentials on the collaborative edges defined in Eq.(4).

By incorporating the bottom-up and top-down steps, we obtain the response of And-Or graph model by Eq.(6). The final detection is acquired by selecting the maximum score in Eq.(10).

5 Discriminative Learning for And-Or Graph

We formulate the learning of And-Or graph model as a joint optimization task for model structure and parameters, which can be solved by an iterative method extended from the CCCP framework [22]. This algorithm iterates to determine the And-Or graph structure in a dynamical manner: given the inferred latent variables $H = (P, V)$ in each step, the leaf-nodes can be automatically created or removed to generate a new structural configuration. To be specific, a new leaf-node is encouraged to be created as the local detector for contours that cannot be handled by the current model(Fig. 1(c)); a leaf-node is encourage to be removed if it has similar discriminative ability as other ones(Fig. 1(b)). We thus call this procedure dynamical CCCP (dCCCP).

5.1 Optimization Formulation

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

$$\phi(X, y, H) = \begin{cases} \phi(X, H) & \text{if } y = +1 \\ 0 & \text{if } y = -1 \end{cases}, \tag{12}$$

where H is the latent variables. Thus, Eq.(10) can be rewritten as a discriminative function,

$$S_\omega(X) = argmax_{y,H}(\omega \cdot \phi(X, y, H)). \tag{13}$$

The optimization of this function can be solved by using structural SVM with latent variables,

$$\min_\omega \frac{1}{2}\|\omega\|^2 + D\sum_{k=1}^{N}[\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y, H)) - \max_H(\omega \cdot \phi(X_k, y_k, H))], \tag{14}$$

where D is a penalty parameter(set as 0.005 empirically), and $\mathcal{L}(y_k, y, H)$ is the loss function. We define that $\mathcal{L}(y_k, y, H) = 0$ if $y_k = y$, "1" if $y_k \neq y$ in our method.

The optimization target in Equation(14) is non-convex. The CCCP framework [23] was recently utilized in [22, 25] to provide a local optimum solution by iteratively solving the latent variables H and the model parameter ω. However, the CCCP does not address the or-nodes in hierarchy, i.e., assuming the configuration of structure is fixed. In the following, we propose the dCCCP by embedding a structural reconfiguration step.

5.2 Optimization with dynamic CCCP

Following the original CCCP framework, we convert the function in Eq. (14) into a convex and concave form as,

$$\min_\omega[\frac{1}{2}\|\omega\|^2 + D\sum_{k=1}^{N}\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y, H))] - [D\sum_{k=1}^{N}\max_H(\omega \cdot \phi(X_k, y_k, H))] \tag{15}$$

$$= \min_\omega[f(\omega) - g(\omega)], \tag{16}$$

where $f(\omega)$ represents the first two terms, and $g(\omega)$ represents the last term in (15).

The original CCCP includes two iterative steps: (I) fixing the model parameters, estimate the latent variables H^* for each positive samples; (II) compute the model parameters by the traditional structural SVM method. In our method, besides the inferred H^*, we need to further determine the graph configuration, i.e. the production of leaf-nodes associated with or-nodes, to obtain the complete structure. Thus, we insert one step between two original ones to perform the structure reconfiguration. The three iterative steps are presented as follows.

(I) For optimization, we first find a hyperplane q_t to upper bound the concave part $-g(\omega)$ in Eq.(16),

$$-g(\omega) \leq -g(\omega_t) + (\omega - \omega_t) \cdot q_t, \forall\omega. \tag{17}$$

where ω_t includes the model parameters obtained in the previous iteration. We construct q_t by calculating the optimal latent variables $H_k^* = argmax_H(\omega_t \cdot \phi(X_k, y_k, H))$. Since $\phi(X_k, y_k, H) = 0$ when $y_k = -1$, we only take the positive training samples into account during computation. Then the hyperplane is constructed as $q_t = -D\sum_{k=1}^{N}\phi(X_k, y_k, H_k^*)$.

(II) In this step, we adjust the model structure by reconfiguring the leaf-nodes. In our model, each leaf-node is mapped to several feature dimensions of the vector $\phi(X, y, H^*)$. Thus, the process of reconfiguration is equivalent to reorganizing the feature vector $\phi(X, y, H^*)$. Accordingly, the hyperplane q_t would change with $\phi(X, y, H^*)$, and would lead to non-convergence of learning. Therefore, we operate on $\phi(X, y, H^*)$ guided by the Principal Component Analysis(PCA). That is, we allow the adjustment only with the non-principal components (dimensions) of $\phi(X, y, H^*)$, in terms of preserving the significant information of $\phi(X, y, H^*)$ [8]. As a result, q_t is assumed to be unaltered. This step of model reconfiguration can be then divided into two sub-steps.

(i) Feature refactoring guided by PCA. Given $\phi(X_k, y_k, H_k^*)$ of all positive samples, we apply PCA on them,

$$\phi(X_k, y_k, H_k^*) \approx u + \sum_{i=1}^{\mathcal{K}}\beta_{k,i}e_i, \tag{18}$$

where $\mathcal{K}$ is the number of the eigenvectors, e_i the eigenvector with its parameter $\beta_{k,i}$. We set $\mathcal{K}$ a large number so that $||\phi(X_k, y_k, H_k^*) - (u + \sum_{i=1}^{\mathcal{K}}\beta_{k,i}e_i)||_2 < \sigma, \forall k$. For the jth bin of the feature

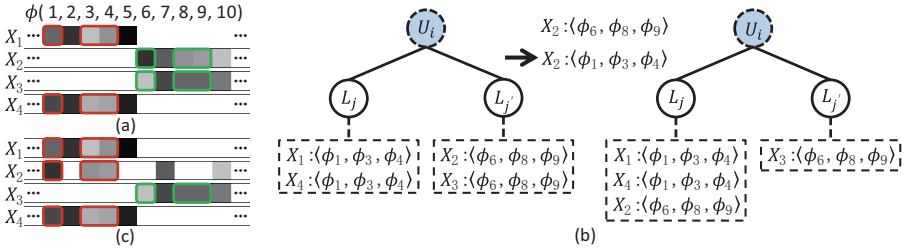

Figure 2: A toy example for structural clustering. We consider 4 samples, $X_1, \ldots, X_4$, for training the structure of U_i. (a) shows the feature vectors ϕ of the samples associated with U_i, and the intensity of the feature bin indicates the feature value. The red and green bounding boxes on the vectors indicate the non-principal features representing the detected contour fragments via two different leaf-nodes. (b) illustrates the clustering performed with ϕ'. The vector $\langle \phi_6, \phi_8, \phi_9 \rangle$ of X_2 is grouped from the right cluster to the left one. (c) shows the adjusted feature vectors according to the clustering. Note that clustering would result in structural reconfiguration, as we discuss in the text. This figure is encouraged to be view in electronic version.

vector, we consider it non-principal only if $e_{i,j} < \delta$ and $u_j < \delta$ for all e_i and u, ($\sigma = 2.0$, $\delta = 0.001$ in experiments).

For each or-node U_i, a set of detected contour fragments, $\{c_i^1, c_i^2, ..., c_i^K\}$, are obtained with the given H_k^* of all positive samples. The feature vectors for these contours that are generated by the leaf-nodes, $\{\phi^l(p_i^1, c_i^1), ..., \phi^l(p_i^K, c_i^K)\}$, are mapped to different parts of the complete feature vector, $\{\phi(X_1, y_1, H_1^*), ..., \phi(X_K, y_K, H_K^*)\}$. More specifically, once we select the jth bin for the all feature vectors ϕ^l, it can be either principal or not in different vectors ϕ. For all feature vector ϕ^l, we select the non-principal bins to form a new vector. We thus refactor the feature vectors of these contours as $\{\phi'(p_i^1, c_i^1), ..., \phi'(p_i^K, c_i^K)\}$.

(ii) Structural reconfiguration by clustering. To trigger the structural reconfiguration, for each or-node U_i, we perform the clustering for detected contour fragments represented by the newly formed feature vectors. We first group the contours detected by the same leaf-node into the same cluster as a temporary partition. Then the re-clustering is performed by applying the ISODATA algorithm and the Euclidean distance. And the close contours are grouped into the same cluster. According to the new partition, we can re-organize the feature vectors, i.e. represent the similar contour with the same bins in the complete feature vector ϕ. Please recall that the vector of one contour is part of ϕ. We present a toy example for illustration in Fig. 2. The selected feature vector (non-principal) $\phi'(p_i^2, c_i^2) = \langle \phi_6, \phi_8, \phi_9 \rangle$ of X_2 is grouped from one cluster to another; by comparing (a) with (c) we can observe that $\langle \phi_6, \phi_8, \phi_9 \rangle$ is moved to $\langle \phi_1, \phi_3, \phi_4 \rangle$.

With the re-organization of feature vectors, we can accordingly reconfigure the leaf-nodes corresponding to the clusters of contours. There are two typical states.

- New leaf-nodes are created once more clusters are generated than previous. Their parameters can be learned based on the feature vectors of contours within the clusters.
- One leaf-node is removed when the feature bins related to it are zero, which implies the contours detected by the leaf-node are grouped to another cluster.

In practice, we constrain the extent of structural reconfiguration, i.e., only few leaf-nodes can be created or removed for each or-node per iteration. After the structural reconfiguration, we denote all the feature vectors $\phi(X_k, y_k, H_k^*)$ are adjusted to $\phi^d(X_k, y_k, H_k^*)$. Then the new hyperplane is generated as $q_t^d = -D \sum_{k=1}^N \phi^d(X_k, y_k, H_k^*)$.

(III) Given the newly generated model structures represented by the feature vectors $\phi^d(X_k, y_k, H_k^*)$, we can learn the model parameters by solving $\omega_{t+1} = argmin_\omega[f(\omega) + \omega \cdot q_t^d]$. By substituting $-g(\omega)$ with the upper bound hyperplane q_t^d, the optimization task in Eq. (15) can be rewritten as,

$$\min_\omega \frac{1}{2}\|\omega\|^2 + D\sum_{k=1}^N [\max_{y,H}(\omega \cdot \phi(X_k, y, H) + \mathcal{L}(y_k, y, H)) - \omega \cdot \phi^d(X_k, y_k, H_k^*)]. \tag{19}$$

This is a standard structural SVM problem, whose solution is presented as,

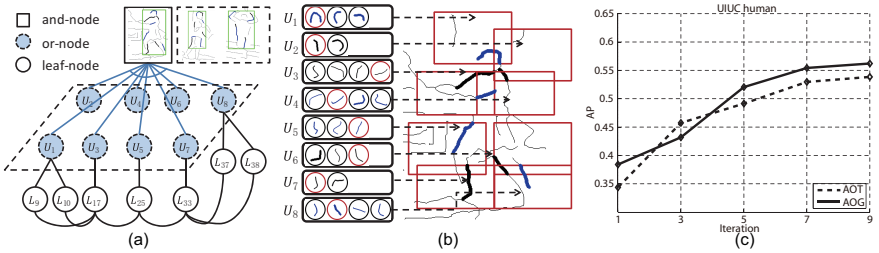

Figure 3: The trained And-Or graph model with the UIUC-People dataset. (a) visualizes the three layer model, where the images on the top imply the verification via the root-node. (b) exhibits the leaf-nodes associated with the or-nodes, $U_1, \ldots, U_8$; a practical detection with the activated leaf-nodes are highlighted by red. (c) shows the average precisions (AP) results generated by the And-Or tree (AOT) model and the And-Or graph (AOG) model.

$$\omega^* = D \sum_{k,y,H} \alpha^*_{k,y,H} \Delta\phi(X_k, y, H), \tag{20}$$

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 function:

$$\max_{\alpha} \sum_{k,y,H} \alpha_{k,y,H} \mathcal{L}(y_k, y, H) - \frac{D}{2} \sum_{k,k'} \sum_{y,H,y',H'} \alpha_{k,y,H} \alpha_{k',y',H'} \Delta\phi(X_k, y, H) \Delta\phi(X_{k'}, y', H'). \tag{21}$$

It is a dual problem in standard SVM, which can be solved by applying the cutting plane method [1] and Sequential Minimal Optimization [13]. Thus, we obtain the updated parameters ω_{t+1}, and continue the 3-step iteration until the function in Eq.(16) converges.

5.3 Initialization

At the beginning of learning, the And-Or graph model can be initialized as follows. For each training sample (whose contours have been extracted), we partition it into a regular layout of several blocks, each of which corresponds to one or-node. The contours fallen into the block are treated as the input for learning. Once there are more than two contours in one block, we select the one with largest length. Then the leaf-nodes are generated by clustering the selected contours without any constraints, and we can thus obtain the initial feature vector ϕ^d for each sample.

6 Experiments

We evaluate our method for object shape detection, using three benchmark datasets: the UIUC-People [17], the ETHZ-Shape [7] and the INRIA-Horse [7].

Implementation setting. We fix the number of or-nodes in the And-Or model as 8 for the UIUC-People dataset, and 6 in other experiments. The initial layout is a regular partition (e.g. 4×2 blocks for the UIUC-People dataset and 2×3 for others). There are at most $m = 4$ leaf-nodes for each or-node. For positive samples, we extract their clutter-free object contours; for negative samples, we compute their edge maps by using the Pb edge detector [12] with an edge link method. The convergence of our learning algorithm take $6 \sim 9$ iterations. During detection, the edge maps of test images are extracted as for negative training samples, within which the object is searched at 6 different scales, 2 per octave. For each contour as the input to the leaf-node, we sample 20 points and compute the Shape Context descriptor for each point; the descriptor is quantized with 6 polar angles and 2 radial bins. We adopt the testing criterion defined in the PASCAL VOC challenge: a detection is counted as correct if the intersection over union with the groundtruth is at least 50%.

Experiment I. The UIUC-People dataset contains 593 images (346 for training, 247 for testing). Most of the images contain one person playing badminton. Fig. 3(b) shows the trained And-Or model(AOG) in that each of the 8 or-nodes associates with $2 \sim 4$ leaf-nodes. To evaluate the benefit from the collaborative edges, we degenerate our model to the And-Or Tree (AOT) by removing the collaborative edges. As Fig. 3(c) illustrates, the average precisions (AP) of detection by applying AOG and AOT are $56.20\%$and 53.84% respectively. Then we compare our model with the state-of-the-art detectors in [18, 2, 4, 5], some of which used manually labeled models. Following the

	Accuracy
Our AOG	**0.680**
Our AOT	0.660
Wang et al. [18]	0.668
Andriluka et al. [2]	0.506
Felz et al. [5]	0.486
Bourdev et al. [4]	0.458

(a)

	Applelogos	Bottles	Giraffes	Mugs	Swans	Average
Our method	**0.910**	0.926	**0.803**	0.885	**0.968**	**0.898**
Ma et al. [10]	0.881	0.920	0.756	0.868	0.959	0.877
Srinivasan et al. [16]	0.845	0.916	0.787	**0.888**	0.922	0.872
Maji et al. [11]	0.869	0.724	0.742	0.806	0.716	0.771
Felz et al. [5]	0.891	**0.950**	0.608	0.721	0.391	0.712
Lu et al. [9]	0.844	0.641	0.617	0.643	0.798	0.709

(b)

Table 1: (a) Comparisons of detection accuracies on the UIUC-People dataset. (b) Comparisons of average precision (AP) on the ETHZ-Shape dataset.

metric mentioned in [18], to calculate the detection accuracy, we only consider the detection with the highest score on an image for all the methods. As Table. 1a reports, our methods outperforms other approaches.

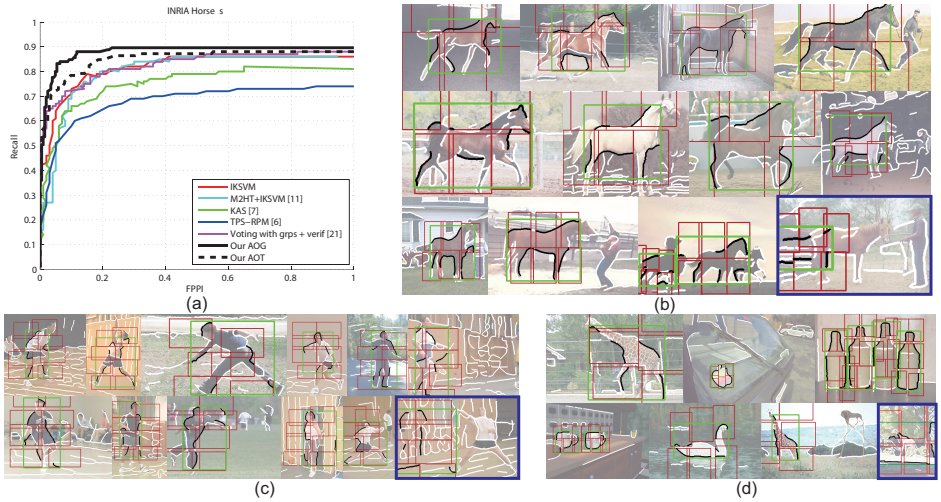

Figure 4: (a)Experimental results with the recall-FPPI measurement on the INRIA-Horse database. (b),(c) and (d) shows a few object shape detections by applying our method on the three datasets, and the false positives are annotated by blue frames.

Experiment II. The INRIA-Horse dataset consists of 170 horse images and 170 images without horses. Among them, 50 positive examples and 80 negative examples are used for training and remaining 210 images for testing. Fig. 4 reports the plots of false positives per image (FPPI) vs. recall. It is shown that our system substantially outperforms the recent methods: the AOG and AOT models achieve detection rates of 89.6% and 88.0% at 1.0 FPPI, respectively; in contrast, the results of competing methods are: 87.3% in [21], 85.27% in [11], 80.77% in [7], and 73.75% in [6].

Experiment III. We test our method with more object categories on the ETHZ-Shape dataset: Applelogos, Bottles, Giraffes, Mugs and Swans. For each category (including $32 \sim 87$ images), half of the images are randomly selected as positive examples, and $70 \sim 90$ negative examples are obtained from the other categories as well as backgrounds. The trained model for each category is tested on the remaining images. Table 1b reports the results evaluated by the mean average precision. Compared with the current methods [11, 16, 5, 9, 10], our model achieves very competitive results.

A few results are visualized in Fig.4(b),(c) and (d) for experiment I, II, and III respectively.

7 Conclusion

This paper proposes a discriminative contour-based object model with the And-Or graph representation. This model can be trained in a dynamical manner that the model structure is automatically determined during iterations as well as the parameters. Our method achieves the state-of-art of object shape detection on challenging datasets.

References

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann, Hidden markov support vector machines, In *ICML*, 2003. 7

[2] M. Andriluka, S. Roth, and B. Schiele, Pictorial structures revisited: People detection and articulated pose estimation, In *CVPR*, 2009. 7, 8

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

[4] L. Bourdev, S. Maji, T. Brox, and J. Malik, Detecting people using mutually consistent poselet activations, In *ECCV*, 2010. 7, 8

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, Object Detection with Discriminatively Trained Part-based Models, *IEEE TPAMI*, 2010. 1, 2, 7, 8

[6] V. Ferrari, F. Jurie, and C. Schmid, From Images to Shape Models for Object Detection, *Int'l J. of Computer Vision*, 2009. 2, 8

[7] V. Ferrari, L. Fevrier, F. Jerie, and C. Schmid, Groups of Adjacent Contour Segments for Object Detection, *IEEE TPAMI*, 30(1): 36-51, 2008. 7, 8

[8] N. Kambhatla and T. K. Leen, Dimension Reduction by Local Principal Component Analysis, *Neural Computation*, 9: 1493-1516, 1997. 5

[9] C. Lu, L. J. Latecki, N. Adluru, X. Yang, and H. Ling, Shape Guided Contour Grouping with Particle Filters, In *ICCV*, 2009. 2, 8

[10] T. Ma and L. J. Latecki, From Partial Shape Matching through Local Deformation to Robust Global Shape Similarity for Object Detection, In *CVPR*, 2011. 2, 8

[11] S. Maji and J. Malik, Object Detection using a Max-Margin Hough Transform, In *CVPR*, 2009. 2, 8

[12] D. R. Martin, C. C. Fowlkes, and J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, *IEEET PAMI*, 26(5): 530-549, 2004. 7

[13] J. C. Platt, Using analytic qp and sparseness to speed training of support vector machines, In *Advances in Neural Information Processing Systems*, pages 557-563, 1998. 7

[14] P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele, Discriminative structure learning of hierarchical representations for object detection, In *CVPR*, 2009. 2

[15] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, Contextualizing Object Detection and Classification, In *CVPR*, 2010. 2

[16] P. Srinivasan, Q. Zhu, and J. Shi, Many-to-one Contour Matching for Describing and Discriminating Object Shape, In *CVPR*, 2010. 2, 8

[17] D. Tran and D. Forsyth, Improved human parsing with a full relational model, In *ECCV*, 2010. 7

[18] Y. Wang, D. Tran, and Z. Liao, Learning Hierarchical Poselets for Human Parsing, In *CVPR*, 2011. 2, 7, 8

[19] X. Yang and L. J. Latecki, Weakly Supervised Shape Based Object Detection with Particle Filter, In *ECCV*, 2010. 2

[20] B. Yao, A. Khosla, and L. Fei-Fei, Classifying Actions and Measuring Action Similarity by Modeling the Mutual Context of Objects and Human Poses, In *ICML*, 2011. 2

[21] P. Yarlagadda, A. Monroy and B. Ommer, Voting by Grouping Dependent Parts, In *ECCV*, 2010. 8

[22] C.-N. J. Yu and T. Joachims, Learning structural svms with latent variables, In *ICML*, 2009. 2, 4, 5

[23] A. Yuille and A. Rangarajan, The concave-convex procedure(cccp), In *NIPS*, pages 1033-1040, 2001. 1, 2, 5

[24] Y.B. Zhao and S.C. Zhu, Image Parsing via Stochastic Scene Grammar, In *NIPS*, 2011. 2

[25] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, Latent Hierarchical Structural Learning for Object Detection, In *CVPR*, 2010. 1, 2, 5

[26] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. Yuille, Max Margin AND/OR Graph Learning for Parsing the Human Body, In *CVPR*, 2008. 1, 2

[27] S.C. Zhu and D. Mumford, A stochastic grammar of images, *Foundations and Trends in Computer Graphics and Vision*, 2(4): 259-362, 2006. 1, 2