Learning AND-OR Templates for Object Recognition and Detection

Zhangzhang Si and Song-Chun Zhu

Abstract—This paper presents a framework for unsupervised learning of a hierarchical reconfigurable image template — the AND-OR Template (AOT) for visual objects. The AOT includes: (1) hierarchical composition as "AND" nodes, (2) deformation and articulation of parts as *geometric* "OR" nodes, and (3) multiple ways of composition as *structural* "OR" nodes. The terminal nodes are hybrid image templates (HIT) [17] that are fully generative to the pixels. We show that both the structures and parameters of the AOT model can be learned in an unsupervised way from images using an information projection principle. The learning algorithm consists of two steps: i) a recursive block pursuit procedure to learn the hierarchical dictionary of primitives, parts and objects, and ii) a graph compression procedure to minimize model structure for better generalizability. We investigate the factors that influence how well the learning algorithm can identify the underlying AOT. And we propose a number of ways to evaluate the performance of the learned AOTs through both synthesized examples and real world images. Our model advances the state-of-the-art for object detection by improving the accuracy of template matching.

1 Introduction

1.1 Motivation and Objective

Visual objects are fundamentally compositional and exhibit rich structural variations: cats may have sharp or round ears; desks may have long or short legs. Therefore object templates must be reconfigurable to account for structural variabilities. In the literature, deformable templates [3], [7], [6], [20], [5], [22], [21] and compositional hierarchy [11], [12], [9], [23], [19], [16], [8], [2] are widely used in object modeling with shared parts among categories. Furthermore, generative image grammar and AND-OR graphs [24] are proposed to facilitate robust statistical modeling of images. In this article, we propose a framework for learning AND-OR templates (AOT) which combine compositionality represented as AND nodes, and reconfigurability represented as OR nodes.

We address the following issues associated with unsupervised learning of the AOT:

- Identifying a hierarchical dictionary of visual parts and objects;
- Deep mixing of AND, OR nodes, which implies a rich mixture model with combinatorially many configurations of templates;
- Localization accuracy of objects, parts and sub-parts;
- Efficient inference for template matching and object detection in images.

Among these issues a most important challenge with unsupervised learning is the pervasive ambiguity in identifying which elements should be grouped as a visual part. For example, it is hard to determine where to segregate the animal face or horse body into constituent parts.

The proposed AOT model aims at extending our previous work – active basis template (ABT) [20] and hybrid image templates (HIT) [17]. An ABT consists of a small number of Gabor basis elements (sketches) at select locations and orientations, where these basis elements are allowed to perturb their locations and orientations locally. Si and Zhu [17] extended the ABT to a HIT by including local texture patterns (gradient orientation histograms), flatness and color. Both ABT and HIT can be learned automatically from roughly aligned training images. The main limitation of ABT and HIT is the lack of reconfigurability. They can only deal with small deformation, and fall short when there are large articulations or structural changes. It would not be satisfactory by simply enumerating all possible configurations by a mixture model of HITs. These HITs do not share training positives, and require more training images. As a related example, in Felzenszwalb et. al. [5] the deformable parts model is a mixture of object templates which do not share training data. Instead of learning HITs directly for object templates, we propose to use the HITs as reusable building blocks — the terminal nodes of AOT, for smaller visual parts that are relatively rigid.

1.2 Overview of proposed methodology

An AOT is a stochastic reconfigurable template that generates a set of valid configurations or object templates. The AND nodes represent compositions of parts, while the OR nodes account for articulation and structural variation of parts. As an example, Figure 1 shows a learned AOT from 320 animal face images without manual labeling. The solid circles denote AND nodes and the hollow ones denote OR nodes. The branching probabilities are also shown at each OR node. The rectangles

- *The authors are with the Department of Statistics, University of California, Los Angeles. Email: {zzsi, sczhu}@stat.ucla.edu*

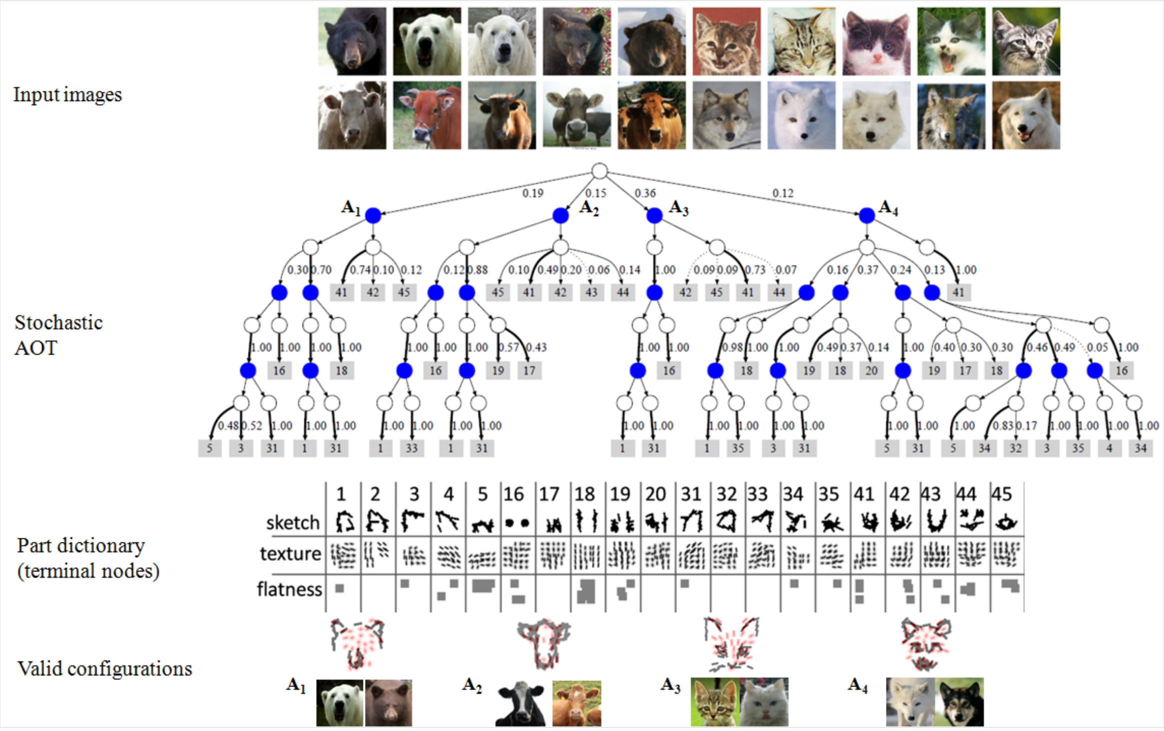


Fig. 1. An AND-OR Template (AOT) learned from 320 animal face images of four categories, with no manual labeling. Shaded circles denote AND nodes, which are combinations of terminal nodes or children OR nodes. Empty circles means structural OR nodes. Shaded rectangles are terminal nodes, which are hybrid image templates (HITs) for part appearances. Each terminal node is also associated with a geometric OR node which accounts for its deformation and articulation. For clarity we removed the geometric OR nodes and OR branches with probability less than 0.1.

denote terminal nodes, where the numbers index the animal facial parts (e.g. eyes, ears) shown as HITs in the third row of the figure. The hierarchical AOT generates a large number of valid configurations.

The learning algorithm for AOT consists of two steps:

i) Block pursuit for hierarchical dictionaries of reusable parts. Firstly we set up a data matrix (e.g. Figure 4) using feature responses of positive training examples. Each row in the data matrix corresponds to one training example; and each column corresponds to one feature (e.g. primitive, texture and color). The data matrix is usually flat, i.e. with a small number of rows (e.g. 100) and a large number of columns (e.g. 10^6). The entries in the data matrix, i.e. the feature responses, are normalized to real numbers between 0 and 1. A response of 1 indicates that a certain feature or part is present in the image, while 0 indicates absence of such feature. Each visual part corresponds to a rectangular *block* in the data matrix. The block is specified by a set of common features (columns) shared by a set of examples (rows). The shared features form a *template* of the block members. The summation over entries in the block measures how significant and how frequent it is. We find that pursuing large blocks with a lot of 1's directly links to the information projection principle [25], [20] and maximum likelihood estimation. Once we pursue the blocks, i.e. the parts, we augment the data matrix with new columns measuring the responses of the parts. This procedure is done recursively until we reach the scale of whole objects. It is worth to note that the pervasive ambiguity of parts (i.e. where to segment objects into parts) can be reduced or eliminated by jointly pursuing for both objects and parts. We show this in the one dimensional text example in Section 5.1.

ii) Graph compression on the AOT. After block pursuit, we encode the training images by a set of configurations of parts (i.e. which parts appear and where they appear). It is a flat and large AOT whose root node has many branches. This flat AOT simply memorizes data, so it suffers from large model complexity and thus poor generalizability. We propose a graph compression procedure to produce a compact AOT. There are two operators in graph compression:

- Sharing. This operator restructures the AOT by sharing parts (e.g. $(A \cap B) \cup (A \cap C) \Rightarrow A \cap (B \cup C)$).
- Merging. This operator merges OR nodes with similar branching probabilities, and re-estimate the merged probabilities.

By applying the two operators, we reduce the model complexity, which is measured by the number of nodes in the AOT. This is accompanied by a slight loss in likelihood. We use a parameter to control the trade-off between data likelihood and model complexity. This is closely related to Bayesian model selection methods like BIC [15].

1.3 Related work

The learning of hierarchical visual dictionaries has been explored in recent literature, where meaningful visual parts are learned from various image features, such as image primitives [9], segmented image regions [19], interest points [18], [23] and histogram of gradients [5], [14]. Different learning algorithms have been used: discriminative criterion[5], [14], data mining heuristics [9], [19], maximum likelihood learning [18] where a hierarchical latent Dirichlet process is assumed, deep neural networks [11] and Grammar-Markov models [23]. These methods have demonstrated the usefulness of the learned structures mainly through good classification performance.

To analyze the merits and limitations of the above methods with respect to the proposed model, we characterize them by five aspects. Table 1 is a side-by-side comparison.

TABLE 1
A comparison with previous work.

	[9], [19], [14]	[18]	[23]	[5]	[20]	ours
compositional hierarchy	√	√	√	√	X	√
unsupervised learning	√	√	√	X	√	√
deep mixing of AND/OR	X	X	X	X	X	√
fully generative	X	X	X	X	√	√
probabilistic model	X	√	√	X	√	√

i) Compositional hierarchy.
ii) Unsupervised learning.
iii) Deep mixing of AND/OR nodes.
iv) Fully generative.
v) Probabilistic model.

In [9], [19], [18], [23], [14], a compositional hierarchy is learned by unsupervised learning. However, the OR nodes for structural variations are largely omitted or oversimplified. And there is no deep mixing of AND/OR nodes, *i.e.* OR nodes across all levels of compositional hierarchy. Another limitation of the above methods is that they do not build on a fully generative model to the level of pixels. For example, in [18] a local bag-of-words model summarizes local geometry and appearance as a pooled histogram, but detailed object deformation is discarded during computation of histograms. As a result, it is difficult to visualize the learned structures to ensure they are semantically meaningful. Many state-of-the-art object detection systems use a part-based latent SVMs model [5] with multi-scale HoG [4] features. They are capable of modeling certain amount of object articulation, but the localization of object boundaries is imprecise because of the local histogram pooling in computing the HoG feature. Another limitation of such models is that their performance relies on training on large amount of negative training examples. The AOT is aimed at addressing these issues.

1.4 Contribution

The contributions of our paper include:

- We propose a fully generative and reconfigurable AOT for object modeling. The AOT includes both structural variabilities (e.g. different co-appearances of parts) and geometric variabilities (articulation of parts). They are modeled by two disjoint sets of OR nodes to enable efficient inference on testing images.
- We present an unsupervised learning algorithm for AND nodes, OR nodes and terminal nodes of the AOT under a principled framework of information projection.
- We study a key issue in unsupervised learning: the identifiability of parts and thus the AOT. We design comprehensive experiments to evaluate several factors that influence the identifiability of AOT, i.e. how well the learning algorithm can identify the underlying AOT.
- In experiments, we show the proposed AOT advances the state-of-the-art in object detection by improving the accuracy of template matching on challenging public benchmarks.

The rest of the paper is organized as follows. Section 2 introduces the representation of the terminal nodes and the graphical structure of the AOT model. Section 3 explains the unsupervised learning algorithm of AOT. Section 4 explains the inference algorithm of AOT for object detection. In Section 5, we show that the AOT can be reliably identified from both synthesized data and real-world images, and it performs on par with state-of-the-art object detection systems on public benchmarks.

2 REPRESENTATION

2.1 Hybrid image templates (HIT) as terminal nodes

In our representation, the terminal nodes of the AOT are HITs (hybrid image templates) [17], which we explain in the following.

Image alphabet. An image $\mathbf{I}$ is divided into many small image patches $\{\mathbf{I}_\Lambda\}$, where $\{\Lambda\}$ are domains of local regions (*e.g.* $11^2 \sim 19^2$ pixels). The space of small image patches are quantized into four categories: sketch, texture, flatness and color.

Definition 1 The *image alphabet*, denoted as $\Delta^{(1)}$, is the set of feature prototypes that are typical and appear frequently in small image patches. They include sketch, texture, flatness and color:

$$\Delta^{(1)} = \{B_j\} \cup \{\mathbf{h}_j^{\mathrm{txt}}\} \cup \{\mathbf{h}_j^{\mathrm{flt}}\} \cup \{\mathbf{h}_j^{\mathrm{clr}}\}$$

$\{B_j\}$ are image primitives or sketches (e.g. Gabor wavelets), which often appear near strong edges and object boundaries. $\{\mathbf{h}_j^{\mathrm{txt}}\}$ are histograms of gradient

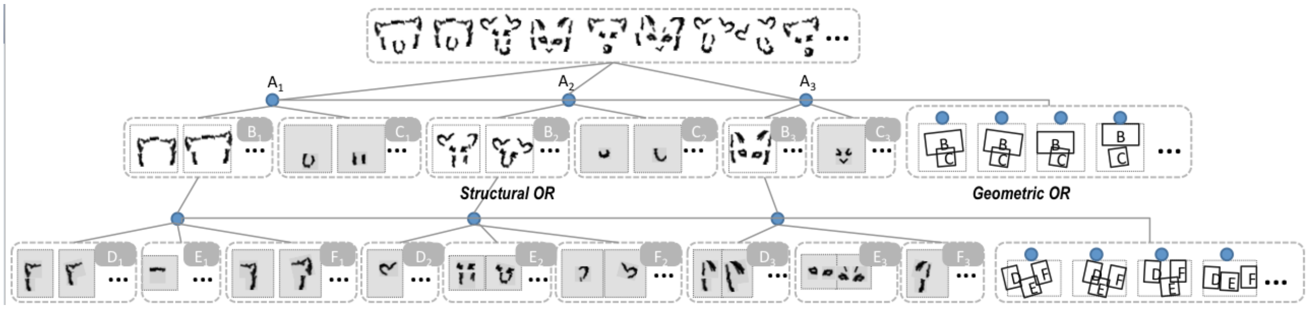


Fig. 2. A more detailed illustration of the animal AOT shown in Figure 1. For OR nodes, we illustrate typical templates with structural variations and geometric transforms. The AND nodes are denoted by solid blue circles. The terminal nodes are individual parts (e.g. ear, mouth) which are represented by automatically learned hybrid image templates.

orientations (e.g. HoG) which are suitable to represent more complex textures inside objects (e.g. fur and hair). $\{\mathbf{h}_j^{\text{flt}}\}$ are flatness features which often appear in empty image patches like cloudless sky. $\{\mathbf{h}_j^{\text{clr}}\}$ are histograms of color, which are most capable of describing objects with distinctive colors (e.g. tomatoes).

HIT model. A HIT is a fully generative probabilistic image model, which consists of a small number of atomic feature prototypes at selected locations and orientations. See Figure 3 for two HITs learned from tomato and pear images. A HIT is specified by a list:

$$\begin{aligned}\text{HIT} = \{&(B_1, x_1, y_1, o_1), (\mathbf{h}_2, x_2, y_2),\\ &(\mathbf{h}_3, x_3, y_3), (B_4, x_4, y_4, o_4), ...\}\end{aligned}$$

where $B_1, B_4, ...$ are image primitives and $\mathbf{h}_2, \mathbf{h}_3, ...$ are histogram descriptors for texture, flatness and color. $\{(x_j, y_j)\}$ denote the selected locations and $\{o_j\}$ are the selected orientations. Though local deformation is allowed for its individual elements, the HIT itself does not deal with large articulation or structural variation. The proposed AOT is aimed at addressing this issue, and the HITs serve as terminal nodes of AOT. Each AOT is an object template consist of a few HITs as parts.

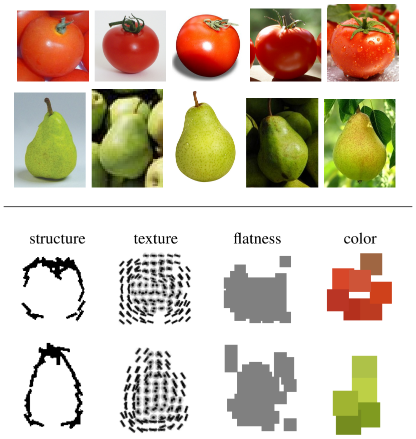


Fig. 3. HITs for tomato and pear [17]. Best viewed in color.

Feature responses. Within a window (e.g. 150 by 150 pixels) of visual object, the image is divided into small image patches using a regular grid. For each patch, we measure a one-dimensional response $r(\mathbf{I})$ which indicates how likely each feature prototype in $\Delta^{(1)}$ appears in this patch. r measures the similarity between the image patch and feature prototype, and it is normalized to a value between 0 and 1. Larger value of r means this feature prototype appears with higher probability.

For image primitives $\{B_j\}$, we compute feature response r as the Euclidean distance between the image patch and the primitive. For texture, color and flatness, we compute locally pooled histograms in the image patch. We refer to [17] for parameters of atomic features and how to compute the one dimensional response $r(\mathbf{I})$.

Let D be the total number of feature responses (i.e. number of patches times the dictionary size $|\Delta^{(1)}|$) on one image. D is often on the order of 10^6 or more. The feature responses of image $\mathbf{I}$ is organized in a vector:

$$\mathbf{R}(\mathbf{I}) = (r_1(\mathbf{I}), ..., r_D(\mathbf{I})).$$

From the D candidate features we select a small subset to compose a HIT. Let $\{j_1, ...j_T\} \subset \{1, ..., D\}$ be indexes of the selected features. T is the number of selected atomic features, and it is usually on the order of 10. To simplify notation, we will use r_t to denote r_{j_t} when there is no ambiguity.

Probability model. Let $\mathcal{X}_+ = \{\mathbf{I}_1, ..., \mathbf{I}_N\}$ be positive example images (e.g. animal faces) governed by the underlying target distribution $f(\mathbf{I})$. Let $\mathcal{X}_-$ be a large set of generic natural images governed by the reference distribution $q(\mathbf{I})$. Our objective of learning is to pursue a model $p(\mathbf{I})$ to approximate $f(\mathbf{I})$ in a series of steps:

$$q(\mathbf{I}) = p_0(\mathbf{I}) \to p_1(\mathbf{I}) \to \cdots p_T(\mathbf{I}) = p(\mathbf{I}) \approx f(\mathbf{I})$$

starting from q.

The model p after T iterations contains T selected features $\{r_t : t = 1, ..., T\}$. If the selected feature responses capture all information about image $\mathbf{I}$, it can be shown by variable transformation [20] that:

$$\frac{p(\mathbf{I})}{q(\mathbf{I})} = \frac{p(r_1, ..., r_T)}{q(r_1, ..., r_T)}. \qquad (1)$$

So p can be constructed by reweighting q with the marginal likelihood ratio on selected features.

Under the maximum entropy principle, $p(\mathbf{I})$ can be expressed in the following log-linear form:

$$p(\mathbf{I}) = q(\mathbf{I}) \prod_{t=1}^{T} \left[\frac{1}{z_t} \exp\left\{\beta_t r_t(\mathbf{I})\right\} \right]. \tag{2}$$

where β_t is the parameter for the t-th selected feature r_t and z_t ($z_t > 0$) is the individual normalization constant determined by β_t:

$$z_t = \sum_{r_t} q(r_t) \exp\{\beta_t r_t\}. \tag{3}$$

It is an expectation over the reference distribution $q(r_t)$. In practice, as a pre-processing step prior to model learning, we estimate $q(r_t)$ in the form of a histogram using a large number of random training examples. And we can compute z_t using Monte Carlo estimation.

For simplicity, we will use an aggregated normalizing constant $Z = \prod_{t=1}^{T} z_t$ when there is no ambiguity.

By the information projection principle [13], [25], [20], we adopt a step-wise procedure to for feature selection. In particular, the t-th feature r_t is selected and model p_t is updated by:

$$\begin{aligned} & p_t = \arg\max \quad \mathcal{K}(p_t | p_{t-1}) \\ s.t. \qquad & E_{p_t}[r_t] = \frac{1}{N} \sum_{i=1}^{N} r_t(\mathbf{I}_i) \end{aligned} \tag{4}$$

where $\mathcal{K}$ denotes the Kullback-Leibler divergence, and by maximizing it over all candidate features, we select a most informative feature r_t to augment p_{t-1} towards p_t. The constraint equation in Eq. (4) ensures that the updated model is consistent with the observed training examples on marginal statistics. The optimal β_t can be found by a simple line search or gradient descent to satisfy the constraint in Eq. (4).

It is convenient to rewrite Eq. (2) using a long and sparse vector $\boldsymbol{\beta}$ of length D, with only few non-zero entries at the indexes $\{j_1, ..., j_T\}$ corresponding to selected features:

$$p(\mathbf{I}) = q(\mathbf{I}) \prod_{j=1}^{D} \left[\frac{1}{z_j} \exp\{\boldsymbol{\beta}_j r_j(\mathbf{I})\} \right] \tag{5}$$

And the logarithm of normalizing constants $(\log z_1, ..., \log z_D)$ is also sparse, with $\log z_j = 0$ (i.e. $z_j = 1$) whenever $\boldsymbol{\beta}_j = 0$. Since $\boldsymbol{\beta}$ encodes both indexes and multiplicative weights of selected features, we may simply consider $\boldsymbol{\beta}$ itself as an HIT.

Definition 2 The *terminal nodes* of AOT, denoted as $\Delta^{(2)}$, is the set of HITs,

$$\Delta^{(2)} = \{\mathrm{HIT}_k : k = 1, ..., K\},$$

which are automatically learned from images. Each entry in $\Delta^{(2)}$ is a part template composed of elements in $\Delta^{(1)}$ at selected locations and orientations. K can also be learned, which will be explained in Section 3. See the third row of Figure 1 for an example of $\Delta^{(2)}$.

In [17] it is shown that the HIT performs well on object categorization. With fewer features and parameters, it achieves on par or better accuracy compared with state-of-the-art methods like HoG + SVM [4] on public benchmarks, especially when there are a small number of training examples.

2.2 AOT: reconfigurable object templates

An AOT consists of a number of configurations of parts, which include

i) structural variabilities (i.e. what parts appear);
ii) geometric variabilities (i.e. where they appear).

The AOT embodies a stochastic context free grammar to regulate the structural and geometric variabilities. It can efficiently capture high-order interaction of parts and compositionality.

Figure 2 illustrates an AOT for animal faces. The terminal nodes are shown as shaded rectangles. AND nodes are denoted as blue solid circles. OR nodes (for both geometric and structural variabilities) are drawn with dashed boxes together with typical configurations. The root node is an OR node with all the valid configurations of animal faces. It is branched into several sets of valid structural configurations as well as geometric configurations (represented as AND nodes) of two subparts: upper face (B) and mouth (C). As we move down the AOT, the upper face is in turn decomposed into left ear (D), right ear (E) and forehead (F). The structural and geometric configurations are not observed in training images, and thus are modeled by two separate sets of latent random variables:

Definition 3 The *structural configuration* $\mathbf{b}$ of AOT is a binary activation vector of length K ($K = |\Delta^{(2)}|$), indicating which parts in $\Delta^{(2)}$ are activated. $\mathbf{b}_k = 1$ means HIT_k is activated and appears in the image.

Definition 4 The *geometric configuration* $\boldsymbol{\tau}$ of AOT is a list of transforms (translation, rotation and scaling) applied to the parts in $\Delta^{(2)}$.

The AOT (e.g. in Figure 2) defines a set of valid configurations for $\mathbf{b}$ and $\boldsymbol{\tau}$, and puts a probability distribution $p(\boldsymbol{\tau}, \mathbf{b}; \mathrm{AOT})$ on this set.

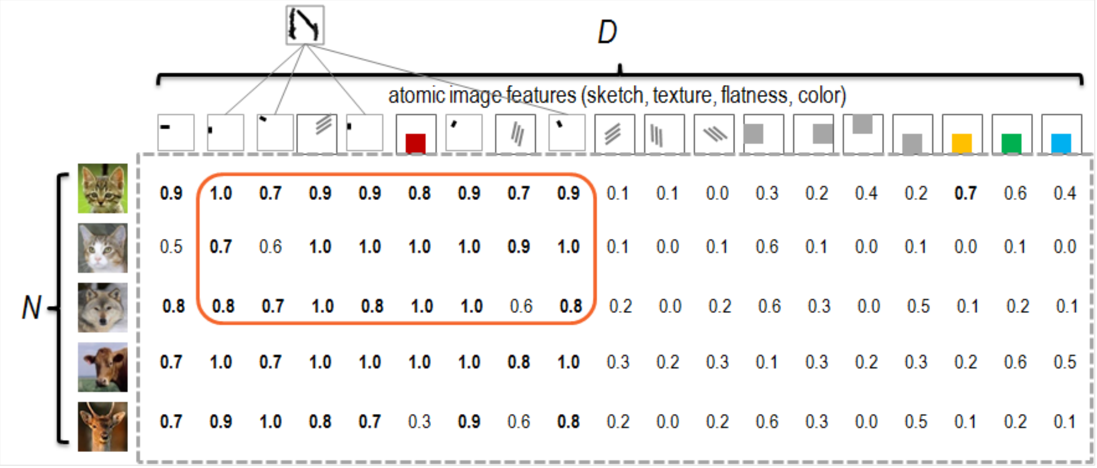
The complete likelihood for an AOT is defined as

$$p(\mathbf{I}, \boldsymbol{\tau}, \mathbf{b} | \mathrm{AOT}, \boldsymbol{\beta}) = p(\boldsymbol{\tau}, \mathbf{b} | \mathrm{AOT}) \cdot p(\mathbf{I} | \boldsymbol{\tau}, \mathbf{b}, \boldsymbol{\beta}), \tag{6}$$

and the image likelihood conditioned on the configuration $(\boldsymbol{\tau}, \mathbf{b})$ is a log-linear form following Eq. (5):

$$\begin{aligned} & p(\mathbf{I} | \boldsymbol{\tau}, \mathbf{b}, \boldsymbol{\beta}) = \\ & \qquad \exp\left\{ \sum_{k=1}^{K} \mathbf{b}_k \left(\sum_{j=1}^{D} \boldsymbol{\beta}_{k,j} r_{\boldsymbol{\tau}(j)}(\mathbf{I}) - \log Z_k \right) \right\} q(\mathbf{I}), \end{aligned} \tag{7}$$

Here we slightly abuse the notation and use $\boldsymbol{\tau}$ as a warping function that maps $\{1, ..., D\}$ to an integer-valued index. This in effects matches a location (and orientation, scale) in the template to a location in the



0.9	1.0	0.7	0.9	0.9	0.8	0.9	0.7	0.9	0.1	0.1	0.0	0.3	0.2	0.4	0.2	0.7	0.6	0.4
0.5	0.7	0.6	1.0	1.0	1.0	1.0	0.9	1.0	0.1	0.0	0.1	0.6	0.1	0.0	0.1	0.0	0.1	0.0
0.8	0.8	0.7	1.0	0.8	1.0	1.0	0.6	0.8	0.2	0.0	0.2	0.6	0.3	0.0	0.5	0.1	0.2	0.1
0.7	1.0	0.7	1.0	1.0	1.0	1.0	0.8	1.0	0.3	0.2	0.3	0.1	0.3	0.2	0.3	0.2	0.6	0.5
0.7	0.9	1.0	0.8	0.7	0.3	0.9	0.6	0.8	0.2	0.0	0.2	0.6	0.3	0.0	0.5	0.1	0.2	0.1

Fig. 4. The data matrix $\mathbf{R}$ measured on images.

image. For out-of-bound situation, $r_{\boldsymbol{\tau}(j)} = 0$ if $\boldsymbol{\tau}(j) < 1$ or $\boldsymbol{\tau}(j) > D$. $\boldsymbol{\beta}$ is a $K \times D$ real-valued matrix denoting a set of K HITs for part templates. $\boldsymbol{\beta}$ is extremely sparse, and there are only around $10 \sim 30$ out of D (D can easily exceed 10^6) non-zero entries in each row. Z_k's are normalizing constants.

We shall call the log-likelihood ratio $\log \frac{p}{q}$ as a *template matching score*, which measures the information gain of explaining the image by the foreground object model instead of the background model:

$$\text{Score}(\mathbf{I}) = \log \frac{p(\mathbf{I}|\boldsymbol{\tau}, \mathbf{b}, \boldsymbol{\beta})}{q(\mathbf{I})} = \sum_{k=1}^{K} \text{Score}(\text{HIT}_k, \mathbf{I}) \tag{8}$$

where

$$\text{Score}(\text{HIT}_k, \mathbf{I}) = \mathbf{b}_k \left(\sum_{j=1}^{D} \boldsymbol{\beta}_{k,j} r_{\boldsymbol{\tau}(j)}(\mathbf{I}) - \log Z_k \right). \tag{9}$$

We assume that the structural configuration is independent from the geometric configuration, and thus

$$p(\boldsymbol{\tau}, \mathbf{b}|\text{AOT}) = p(\boldsymbol{\tau}|\text{AOT}^{\text{geo}}) \cdot p(\mathbf{b}|\text{AOT}^{\text{str}}), \tag{10}$$

where AOT^{geo} is a sub-AOT consisting of only geometric OR nodes (see Figure 2), and AOT^{str} consisting of only structural OR nodes. This facilitates fast inference. In our work, AOT^{geo} is a hierarchical deformation tree similar to the active basis model [20].

3 Learning AOT from Images

The terminal nodes $\Delta^{(2)}$ and the non-terminal nodes in AOT are learned automatically from training images. We first describe how to learn $\Delta^{(2)}$ (the HITs for part templates) using an EM-type block pursuit algorithm. The same algorithm is applied recursively to learn $\Delta^{(3)}$ (compositions of HITs) . Then we introduce a graph compression algorithm to learn a compact set of non-terminal AND/OR nodes.

3.1 Block pursuit on data matrix

Data matrix. The learning is performed on the data matrix $\mathbf{R}$ as shown in Figure 4. Each row of $\mathbf{R}$ is a feature vector for an image in $\mathcal{X}_+$. $\mathbf{R}$ is not necessarily a rectangular matrix, as images of different sizes produce feature vectors of varying lengths. But for simplicity, we assume all positive training images are roughly aligned and have the same size as the object template (this assumption is released in Section 3.2). Therefore $\mathbf{R}$ is a matrix with N (number of positive examples) rows and D (number of all candidate features) columns, and each entry $\mathbf{R}_{ij} = r_j(\mathbf{I}_i)$ is a feature response ($0 \leq \mathbf{R}_{ij} \leq 1$). Larger value of $\mathbf{R}_{ij}$ means feature j appears in image $\mathbf{I}_i$ with higher probability.

On the data matrix, we pursue large blocks $\{\mathcal{B}_k : k = 1, ..., K\}$ with lots of 1's, which correspond to HITs that appear frequently and with high confidence. A block is specified by a set of common features (columns) shared by a set of examples (rows). The significance of block $\mathcal{B}_k$ is measured by the summation over the block:

$$\text{Score}(\mathcal{B}_k) = \sum_{\substack{i \,\in\, \text{rows}(\mathcal{B}_k) \\ j \,\in\, \text{cols}(\mathcal{B}_k)}} (\beta_{k,j}\mathbf{R}_{i,j} - \log z_{k,j}) \tag{11}$$

where rows$(\cdot)$ and cols$(\cdot)$ denote the rows and columns of block $\mathcal{B}_k$. cols$(\mathcal{B}_k)$ corresponds to the selected features in HIT$_k$; and rows$(\mathcal{B}_k)$ are the examples on which HIT$_k$ is activated. $\boldsymbol{\beta}_{k,j}$ is the multiplicative parameter of feature j in HIT$_k$, and $z_{k,j}$ is the individual normalizing constant determined by $\boldsymbol{\beta}_{k,j}$. See Eq. 3 for estimation of $z_{k,j}$.

The score of $\mathcal{B}_k$ is equal with the summation of Eq. (9) over positive examples $\{\mathbf{I}_i : i = 1, ..., N\}$:

$$\text{Score}(\mathcal{B}_k) = \sum_{i=1}^{N} \text{Score}(\text{HIT}_k, \mathbf{I}_i). \tag{12}$$

If we have already identified K blocks $\{\mathcal{B}_k : k = 1, ..., K\}$, then the total score for all the blocks is equal with the summation of Eq. (8) over positive examples:

$$\sum_{k=1}^{K} \text{Score}(\mathcal{B}_k) = \sum_{i=1}^{N} \text{Score}(\mathbf{I}_i) = \sum_{i=1}^{N} \log \frac{p(\mathbf{I}_i)}{q(\mathbf{I}_i)}. \tag{13}$$

So pursuing blocks by maximizing Eq. (13) corresponds to maximum likelihood estimation and the information projection principle.

Information projection. Recall that we pursue a series of models starting from $q(\mathbf{I})$ to approximate the target distribution $f(\mathbf{I})$ governing training positives $\mathcal{X}_+$. This corresponds to maximizing the log-likelihood $\log p(\mathbf{I})$ on $\mathcal{X}_+$. Initially $p = q$, and the data matrix has a log-likelihood $L_0(\mathbf{R})$. After pursuing K blocks, the resulting image log-likelihood is

$$L(\mathbf{R}, \boldsymbol{\beta}, \mathbf{b}) = L_0(\mathbf{R}) + \sum_{k=1}^{K} \text{Score}(\mathcal{B}_k). \tag{14}$$

In the above equation, we have used the vector representation in Eq. (5) and (7). Here we denote the dictionary of HITs in a $K \times D$ real-valued sparse matrix $\boldsymbol{\beta}$. And we denote the structural configurations on all N images in a $K \times N$ binary sparse matrix $\mathbf{b}$. The k-th block can then be denoted as a pair of sparse vectors $\big(\mathbf{b}_{k,:}, \boldsymbol{\beta}_{k,:}\big)$, where the non-zero items in $\boldsymbol{\beta}_{k,:}$ denotes columns of $\mathcal{B}_k$, and the non-zero items in $\mathbf{b}_{k,:}$ denote the rows of $\mathcal{B}_k$.

Block pursuit is a penalized maximum likelihood estimation problem, minimizing a two-term cost function:

$$-L(\mathbf{R}, \boldsymbol{\beta}, \mathbf{b}) + \text{penalty}(\boldsymbol{\beta}), \tag{15}$$

which measures how well the dictionary of templates $\Delta^{(2)}$ (encoded in $\boldsymbol{\beta}$) explain the data matrix through their activations $\mathbf{b}$. The penalty term is an l_0 penalty on $\boldsymbol{\beta}$:

$$\text{penalty}(\boldsymbol{\beta}) = \eta \cdot \sum_{j=1}^{D} \mathbf{1}_{\boldsymbol{\beta} \neq 0} \tag{16}$$

where $\mathbf{1}()$ is an indicator function. η controls the trade-off between the two terms, and we find $\eta = 0.1$ usually leads to good learning results.

In a fully expanded form, the optimization problem in (15) is:

$$\min_{\mathbf{b},\boldsymbol{\beta}} -\frac{1}{N} \left(\sum_{i=1}^{N} \sum_{k} \mathbf{b}_{k,i} \sum_{j=1}^{D} \boldsymbol{\beta}_{k,j} \mathbf{R}_{i,j} - \log Z_k \right) + 0.1 \cdot \sum_{k,j} \mathbf{1}_{\boldsymbol{\beta}_{k,j} \neq 0} \tag{17}$$

$$\text{where } Z_k = \frac{1}{|\mathcal{X}_-|} \sum_{\mathbf{I} \in \mathcal{X}_-} \exp \left\{ \sum_{j=1}^{D} \boldsymbol{\beta}_{k,j} r_j(\mathbf{I}) \right\}$$

We also enforce that the coefficients $\boldsymbol{\beta}_{k,:}$ of HIT$_k$ is confined within a local region of the object window (*e.g.* the top-left subwindow in the 3 by 3 grid in Figure 8) , and for each local region exactly one block is activated for each image example, so that the activated HITs do not overlap.

Due to the highly non-convex l_0 penalty, a naive global optimization algorithm would result in exponential complexity of $O(2^{K \times D})$. Inspired by the matching pursuit algorithm for signal reconstruction under sparse constraints, we propose a shared matching pursuit algorithm generalized from the one used in [20]. This algorithm greedily selects blocks with largest scores, and has a linear complexity $O(KD)$.

EM-type iterations. The rows of blocks (i.e. activations of parts $\mathbf{b}$) are not observed on training images, and we need to repeatedly infer them using the estimated AOT model. So the block pursuit algorithm as outlined below is an EM-type algorithm that alternate between model parameters $\boldsymbol{\beta}$ and latent variables $\mathbf{b}$:

Algorithm: Block Pursuit for part learning

```
Input: data matrix $\mathbf{R}$, initialization of structural configuration
    $\mathbf{b}^{(0)}$, the number of blocks $K$.
Output: Coefficient matrix $\boldsymbol{\beta}^{(T)}$ and structural configuration
    $\mathbf{b}^{(T)}$ after $T$ iterations.
I1 $t \leftarrow 1$. Compute the correlation matrix $Corr$ between each
    pair $(j, j')$ of features.
I2 For each candidate feature $r_j$, and for a grid of possible
    values $\{\beta^{(m)}\}$ for the coefficient $\beta$, compute the corresponding
    normalizing constant $z_j^m$, and the corresponding expectations
    $\{E\left[r_j; \beta^{(m)}\right]\}$ for all $m$:
        $E\left[r_j; \beta^{(m)}\right] = \frac{1}{|\mathcal{X}_-|} \sum_{\mathbf{I} \in \mathcal{X}_-} r_j(\mathbf{I}) \exp\{\beta^{(m)} r_j(\mathbf{I})\}$
M1 For $k = 1, ..., K$, compute the average response on
    activated positive examples
        $\bar{r}^+_{k,j} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{b}_{k,i} \mathbf{R}_{i,j}, \forall j$
    and find the best $\boldsymbol{\beta}_{k,j}$ and $z_{k,j}$ by $\boldsymbol{\beta}^*_{k,j} \leftarrow \boldsymbol{\beta}^{(\mu)}$, $z^*_{k,j} \leftarrow z_j^{(\mu)}$
    where
        $\mu = \arg\min_m (E\left[r_j; \beta^{(m)}\right] - \bar{r}^+_{k,j})^2.$
    Then compute the gain of feature $j$ for the $k$-th block:
        $\text{gain}_{k,j} = \begin{cases} \boldsymbol{\beta}^*_{k,j} \bar{r}^+_{k,j} - \log z^*_{k,j} & j \in S_{c(t)} \\ 0 & \text{otherwise} \end{cases}$
M2 $\boldsymbol{\beta} \leftarrow \mathbf{0}$. For $k = 1, ..., K$, update $\boldsymbol{\beta}_{k,:}$ iteratively:
        $j^* \leftarrow \arg\max_j \text{gain}_{k,j}$
        $\boldsymbol{\beta}^{(t)}_{k,j} \leftarrow \boldsymbol{\beta}^*_{k,j}$, $z^{(t)}_{k,j} \leftarrow z^*_{k,j}$
        $\text{gain}_{k,j'} \leftarrow 0, \forall j'$ s.t. $Corr(j, j') > \text{Thres} = 0.9$
    until $\max_j \text{gain}_{k,j} < \gamma$, or the maximum allowed number of
    selected features is reached.
E1 $\mathbf{b} \leftarrow \mathbf{0}$. For $i = 1, ..., N$, repeat:
        $k^* \leftarrow \arg\max_k \sum_j \boldsymbol{\beta}^{(t)}_{k,j} \mathbf{R}_{i,j} - \log z^{(t)}_{k,j}$,
        $\mathbf{b}^{(t)}_{k^*,i} \leftarrow 1$,
        set $\mathbf{b}^{(t)}_{k,i} = 0$ if non-zero entries of $\boldsymbol{\beta}_{k,:}$ and $\boldsymbol{\beta}_{k^*,:}$ overlap.
    until all values in $\mathbf{b}$ are assigned.
```

Terminate $t \leftarrow t+1$. Iterate from **M1** through **E1** until **b** converges.

The learned blocks can be ranked by the score (or information gain) in Eq. (11) and the blocks with small scores are discarded.

3.2 Deformable block pursuit

In the previous subsection, we assume the images are aligned so that the the visual parts of different images appear in similar locations. To apply the block pursuit algorithm for non-aligned images or images with articulated objects, we expand the **E1** step in the baseline block pursuit algorithm by inferring both geometric and structural configurations. Now the input is a set of feature vectors $\mathbf{R} = \{\mathbf{R}_{1,:}, ..., \mathbf{R}_{N,:}\}$ with different dimensions, but the coefficient matrix remains to be $K \times D$. Similar to Eq. (15), the objective is:

$$\min_{\boldsymbol{\beta},\mathbf{b},\boldsymbol{\tau}} -L(\mathbf{R}, \boldsymbol{\beta}, \mathbf{b}, \boldsymbol{\tau}) + \text{penalty}(\boldsymbol{\beta}), \tag{18}$$

where

$$L(\mathbf{R}, \boldsymbol{\beta}, \mathbf{b}, \boldsymbol{\tau}) = L_0(\mathbf{R}) + \sum_{i=1}^{N}\sum_{k}\mathbf{b}_{k,i}\sum_{j=1}^{D}\left(\boldsymbol{\beta}_{k,j}\mathbf{R}_{i,\boldsymbol{\tau}_i(j)} - \log z_{k,j}\right). \tag{19}$$

In the E step, we not only solve for the best structural configuration **b** but also infer the best geometric configuration $\boldsymbol{\tau}_i$ on each positive example $\mathbf{R}_{i,:}$ using the inference algorithm in Section 4. $\boldsymbol{\tau}_i$ denotes the localization (i.e. position x, y, rotation o, scale s) for the object bounding box, part bounding boxes as well as primitives on image $\mathbf{I}_i$. Part bounding boxes sit at canonical locations relative to the object center. Each part is subject to a random local perturbation $(\Delta x, \Delta y, \Delta o, \Delta s)$, which is independent from other parts. Similarly, primitives are subject to independent local perturbations around their canonical locations inside their parent part bounding box. The inference algorithm contains one round of bottom-up steps followed by one round of top-down steps. In the bottom-up steps, the local perturbations are accounted in local maximizations (Up-2, Up-4). The optimal localization of object bounding box is first inferred (Up-6); then the optimal localizations of parts are obtained by retrieving the arg-max perturbations of parts (Down-5). The optimal localizations of primitives are found similarly.

3.3 Recursive block pursuit

Now we have pursued K blocks or terminal nodes $\Delta^{(2)} = \{\text{HIT}_k, k = 1, ..., K\}$. We then augment the data matrix by K new columns consisting of responses on the HITs. For clarity, we denote $\mathbf{R}$ as $\mathbf{R}^{(1)}$ to indicate responses on $\Delta^{(1)}$. And we denote $\mathbf{R}^{(2)}$ as the newly computed responses on $\Delta^{(2)}$ for the N images. Each entry of $\mathbf{R}^{(2)}$ is a template matching score:

$$\mathbf{R}^{(2)}_{i,k} = \text{Score}(\text{HIT}_k, \mathbf{I}_i). \tag{20}$$

The block pursuit algorithm or its deformable version can be carried on recursively on $\mathbf{R}^{(2)}$. This leads to a compositional hierarchy. In our experiments we find the simple three level (object-part-primitive) hierarchy works well for detecting articulated objects in cluttered images.

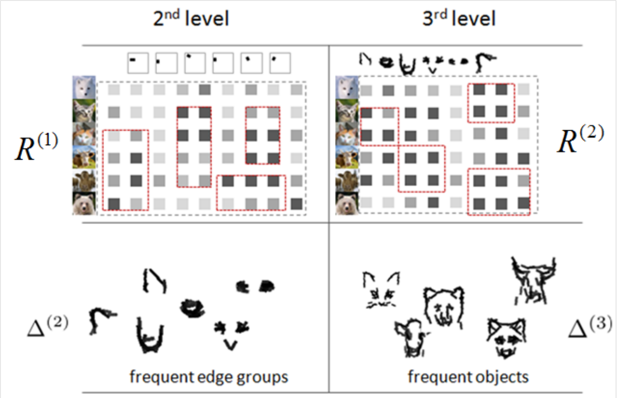


Fig. 5. Block pursuit on images.

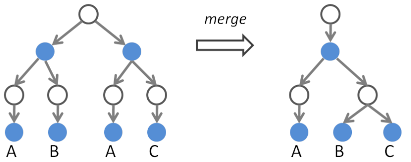


Fig. 6. Graph compression by sharing.

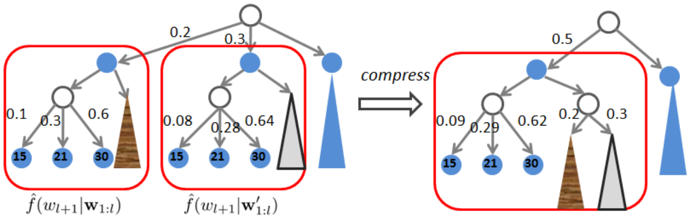


Fig. 7. Graph compression by merging.

3.4 Graph compression

So far we are focused on pursuing blocks to identify meaningful parts from the training images and assume a trivial structural AOT which allows for any activation patterns of parts as long as they don't overlap. This can be problematic as the parts that co-appear in different locations are usually correlated, and certain structural configurations of parts should be prohibited (*e.g.* bear's left ear + cat's right ear). In particular, such a flexible model may easily match a cluttered background patches and cause a large number of false positives. To learn the structural AOT from examples, we extend the optimization problem of block pursuit by adding a prior model $p(\mathbf{b}; \mathrm{AOT}^{\mathrm{str}})$ for structural configuration $\mathbf{b}$ and a hyper prior controlling the complexity of $\mathrm{AOT}^{\mathrm{str}}$. This results in an objective function with four terms:

$$\min_{\boldsymbol{\beta},\mathbf{b},\boldsymbol{\tau},\mathrm{AOT}^{\mathrm{str}}} -L(\mathbf{R},\mathbf{b},\boldsymbol{\beta},\boldsymbol{\tau}) + \mathrm{penalty}(\boldsymbol{\beta},\mathbf{b}) - \log p(\mathbf{b};\mathrm{AOT}^{\mathrm{str}}) + \gamma|\mathrm{AOT}^{\mathrm{str}}| \tag{21}$$

where $|\mathrm{AOT}^{\mathrm{str}}|$ denotes the total number of nodes in the AOT, and the first two terms are exactly the block pursuit cost function and penalty in Eq. (19). To solve (21), we adopt a coordinate descent method which alternate between $(\boldsymbol{\beta},\mathbf{b})$ and $\mathrm{AOT}^{\mathrm{str}}$:

i) Given $\mathrm{AOT}^{\mathrm{str}}$, solve for $(\boldsymbol{\beta},\mathbf{b})$ using the block pursuit algorithm. Here we need to adapt the **E1** step, such that $\mathbf{b}$ is inferred using dynamic programming on $\mathrm{AOT}^{\mathrm{str}}$.

ii) Given $(\boldsymbol{\beta},\mathbf{b})$, solve for $\mathrm{AOT}^{\mathrm{str}}$ by minimizing $-\log p(\mathbf{b};\mathrm{AOT}^{\mathrm{str}}) + \gamma|\mathrm{AOT}^{\mathrm{str}}|$.

Step ii) is a penalized maximum likelihood problem with log-likelihood term $L^{\mathrm{str}} = \log p(\mathbf{b};\mathrm{AOT}^{\mathrm{str}})$. It is solved by a graph compression procedure as follows. Initially we form a giant AOT that has one root OR node branching over the object configurations in $\Delta^{(3)}$. For example:

$$\mathrm{AOT}_0 = (1,6,20,41) \cup (1,6,34,49) \cup (6,18,27,52)...$$

where $\cup$ means OR, and each vector means non-zero entries of $\mathbf{b}$ (i.e. activated parts) in one configuration from $\Delta^{(3)}$.

We call this AOT_0 a *memorization AOT* since it simply memorizes the observed structural configurations. Since the number of configurations is combinatorial, this AOT is huge and tends to overfit. Then we apply an iterative compression procedure that includes two operators:

- Sharing. This operator restructures the AOT by sharing parts. For example, $(1,6,20,41)\cup(1,6,34,49) \Rightarrow (1,6,((20,41)\cup(34,49)))$.
- Merging. This operator merges OR nodes with similar branching probabilities, and re-estimate the merged probabilities.

The two operators are illustrated in Figure 6 and 7. Each of the two operators results in a loss in log-likelihood $\Delta L^{\mathrm{str}} \leq 0$ and a reduction in model complexity (number of nodes) $\Delta|\mathrm{AOT}^{\mathrm{str}}| < 0$. We decide to apply the merging or sharing operator if

$$\Delta L^{\mathrm{str}} - \gamma \cdot \Delta|\mathrm{AOT}^{\mathrm{str}}| > 0 \tag{22}$$

i.e. the reduction in complexity outweighs the loss of log-likelihood.

Re-parameterization of the γ factor. Directly optimizing Eq.(22) requires fine tuning of the γ parameter, and the optimal value of $\gamma \in [0,+\infty)$ is very sensitive to the training data. We adopt a robust re-parameterization of γ using another parameter $\alpha \in [0,1]$. Observing that Eq.22 is essentially testing whether two distributions are the same, we propose to use the χ^2 test with significance level $1-\alpha$ (where $\alpha \in [0,1]$) to approximately implement the decision in Eq.(22). If the branching probabilities of the two OR nodes are : $(a_1,...,a_M)$ and $(b_1,...,b_M)$ with $\sum_i a_i = 1, \sum_i b_i = 1, a_i > 0, b_i > 0, \forall i$, then the χ^2 test statistic is computed as $\chi^2 = \sum_i (a_i - b_i)^2/a_i^2$. We compare this value to $F_{M-1,1-\alpha}$ which can be looked up in the F-table. If $\chi^2 < F_{M-1,1-\alpha}$, then we decide merge these two OR nodes. In the experiments, we use α as the control parameter for model complexity instead of γ.

4 Inference on AOTs

As the AOT is a tree structured graphical model, so a dynamic programming procedure is in place for efficient inference. In particular, the dynamic programming takes the form of recursive SUM and MAX operations, which has been commonly used in hierarchical object models [16], [20], [5] with variations. The inference algorithm is an interleaved procedure between dynamic programming on $\mathrm{AOT}^{\mathrm{geo}}$ and dynamic programming on $\mathrm{AOT}^{\mathrm{str}}$:

Algorithm: Inference by recursive SUM-MAX

Input: Testing image $\mathbf{I}$, AOT, $\boldsymbol{\beta}$ and $\{Z_k\}$.

Output: i) Geometric configuration $\boldsymbol{\tau}$: detected location, rotation and scaling for the whole object, activated parts and activated primitives. ii) Structural configuration $\mathbf{b}$: which parts are activated.

Up-1 Compute atomic feature response $r_j(\mathbf{I}) = \mathrm{SUM1}(x_j,y_j,o_j,s_j)$ for all locations (x_j,y_j), rotations o_j and scalings s_j of the atomic filter (*e.g.* image primitive).

Up-2 Perform local maximization on SUM1 maps over local translation, rotation and scaling to obtain $\mathrm{MAX1}(x,y,o,s)$ and $\mathrm{ARGMAX1}(x,y,o,s)$, where MAX1 stores local maximum responses and ARGMAX1 stores the local transformation of primitives that result in the local maxima.

Up-3 For each part k ($k = 1,...,K$), compute part score maps $\mathrm{SUM2}_k(x,y,o,s)$ for all locations, orientations and scales by transforming the sparse part template (i.e. HIT template) $\boldsymbol{\beta}_{k,:}$ by translation (x,y), rotation o and scaling s and computing the dot product between $\boldsymbol{\beta}_{k,:}$ and the portion of MAX1 map under the transformed part window:

$$\mathrm{SUM2}_k(x,y,o,s) = \sum_{j=1}^{D} \boldsymbol{\beta}_{k,j}\mathrm{MAX1}(\boldsymbol{\tau}_{x,y,o,s}(x_j,y_j,o_j,s_j)) - \log Z_k,$$

where $\boldsymbol{\tau}_{x,y,o,s}(x_j,y_j,o_j,s_j))$ the destination transformation of the primitive (x_j,y_j,o_j,s_j) after it moves with the template, which itself is transformed by translation (x,y), rotation o and scaling s.

Up-4 Compute MAX2, ARGMAX2 maps for all parts by local maximization on SUM2 maps.

Up-4.5 For each transformation (x,y,o,s) of the object, collect the all the part scores as a K dimensional vector $\mathbf{r}^{(2)} = (r_1^{(2)},...,r_K^{(2)})$. Infer the best structural configuration $\mathbf{b}^*$ using $\mathrm{AOT}^{\mathrm{str}}$ based on $\mathbf{r}^{(2)}$.

Up-5 For each transformation (x,y,o,s) of the object, compute the SUM3 score by applying the object filter specified by $\hat{\mathbf{b}}$:

$$\mathrm{SUM3}(x,y,o,s) = \sum_{k=1}^{K} \mathbf{b}_k^* \cdot \mathbf{r}_k^{(2)},$$

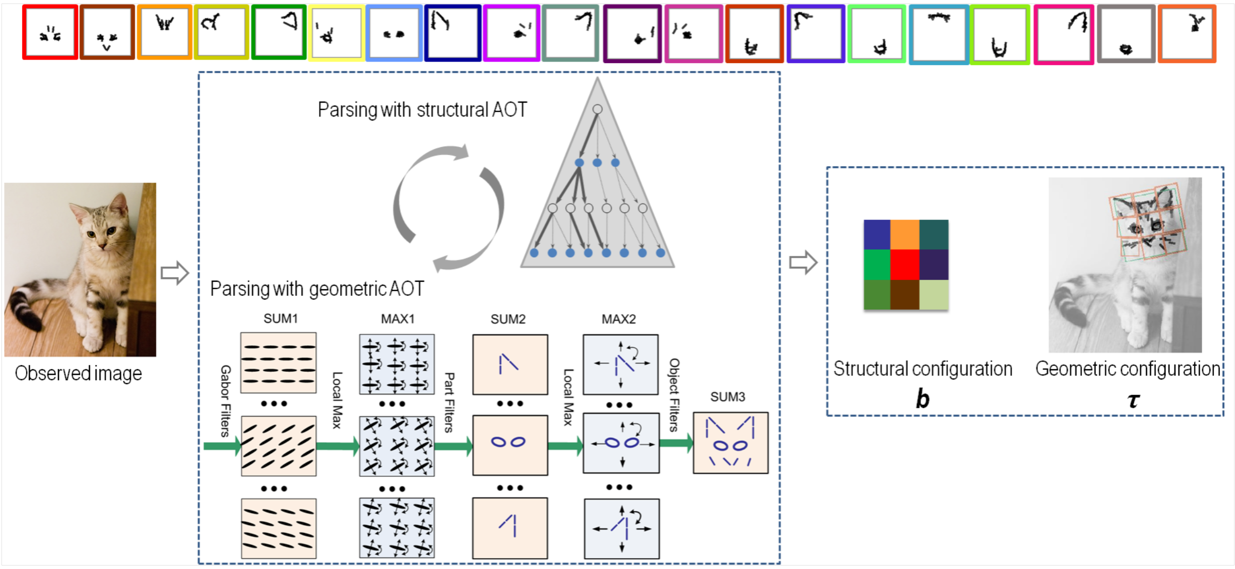


Fig. 8. Inference of AOT on an observed image. **The top row**: the dictionary of parts displayed with different colors in the border. Showing top 20 of them. **The bottom row**: the parsing procedure with the structural AOT and geometric AOT. The structural AOT stores an AND-OR grammar that generates a set of admissible configuration (*i.e.* activation of parts) on an image. In this AOT, we divide the object template into a 3 by 3 grid, and within each of the 9 cells only one part is allowed to be activated. A parse from the structural AOT results in a structural configuration $(k_1, ..., k_9)$ illustrated as a 3 by 3 color code. A parse from the geometric AOT results in the geometric configuration of activated primitives shown as black bars overlayed on the image. We also show the whole geometric parse tree by displaying the transformed object and part bounding boxes.

where $\mathbf{r}^{(2)}$ is the collected vector of MAX2 scores of parts and $\hat{\mathbf{b}}$ is the best structural configuration both computed in **Up-4.5**.

Up-6 Compute the global maximum of SUM3 score maps over all object transformations (x, y, o, s).

Down-6 Retrieve the detected (absolute) transformation $(\hat{x}, \hat{y}, \hat{o}, \hat{s})$ of the object from the maximization in **Up-6**.

Down-5 Compute the temporary transformation $\{\boldsymbol{\tau}_{\hat{x},\hat{y},\hat{o},\hat{s}}(x_k, y_k)\}$ for each activated part k (such that $\mathbf{b}_k > 0$) on **I** by letting the part move with the object, where (x_k, y_k) is the canonical location of the part center relative to the object center.

Down-4 For each activated part $k \in \{k : \mathbf{b}_k > 0\}$, retrieve the detected (absolute) transformation $(\hat{x_k}, \hat{y_k}, \hat{o_k}, \hat{s_k})$ of the k-th part from ARGMAX2 maps computed in **Up-4**.

Down-3 For each activated part k, for each of its activated primitive $j \in \{\boldsymbol{\beta}_{k,j} > 0\}$, compute its temporary transformation $\{\boldsymbol{\tau}_{\hat{x}_k,\hat{y}_k,\hat{o}_k,\hat{s}_k}(x_j, y_j, o_j)\}$ on **I** by letting the primitive move with the part, where (x_j, y_j, o_j) is the canonical location and rotation of the j-th primitive in the part template.

Down-2 For each activated part k, for each of its activated primitive j, retrieve its detected (absolute) transformation $(\hat{x}_j, \hat{y}_j, \hat{o}_j, \hat{s}_j)$ on **I**.

In **Up-4.5** the procedure to infer the best structural configuration $\mathbf{b}$ also takes the form of recursive SUM-MAX. The input is a K dimensional vector $\mathbf{r}^{(2)} = (\mathbf{r}^{(2)}_1, ..., \mathbf{r}^{(2)}_K)$ of all candidate part scores. For notational convenience, we represent the sparse binary vector $\mathbf{b}$ as its non-zero entries $(k_1, k_2, ...) \subset \{1, ..., K\}$. By induction, assume for all the OR nodes in level l we have found the its best parse tree that generates the sub-configuration $(k_1, ..., k_l)$ with exactly l parts being activated. We have also computed the scores of these OR nodes achieved by the best parse trees. Then for an OR node OR^{l+1} in level $l+1$ (the parent level), we find its best parse tree by the following two steps:

SUM Identify this OR node's children nodes, which are AND nodes. For each of the AND nodes AND^{l+1}_i, compute its score by summation of :

$$\mathrm{Score}(\mathrm{AND}^{l+1}_i) = \sum_{\mathrm{OR}^l_j \in \mathrm{Children}(\mathrm{AND}^{l+1}_i)} \mathrm{Score}(\mathrm{OR}^l_j)$$

MAX Find the best branch of this OR node by maximization:

$$i^* = \arg\max_i \mathrm{Score}(\mathrm{AND}^{l+1}_i).$$

Retrieve the best parse tree by concatenating the best sub-parse-trees in $\mathrm{Children}(\mathrm{AND}^{l+1}_{i*})$. From this parse tree, we get the non-zero entries $(k^*_1, ..., k^*_{l+1})$ of the best sub-configuration $\mathbf{b}^*$. We then compute the best score of this OR node by:

$$\mathrm{Score}(\mathrm{OR}^{l+1}) = \sum_{k=1}^{K} \mathbf{b}^*_k \mathbf{r}^{(2)}_k$$

5 Experiment

We evaluate the AOT model in both supervised and unsupervised settings.

In the supervised setting, the ground truth label for training images and object bounding boxes are given. We measure the performance of classification and detection for articulated objects. On challenging object detection tasks with lots of occlusion and clutter, we perform on par with state of art methods for localization of objects, parts and key points, while using a much smaller number of features.

Evaluating the learned AOT in the unsupervised learning is a non-trivial problem, as we often don't have a unique ground-truth, *e.g.* how an object should be divided into parts. We start from the 1D text example, because we have the underlying generating AOT for the training data, which enables us to conduct a thorough numerical evaluation. Then we evaluate the image AOT learned without supervision for both objects and parts, comparing to human subjects.

5.1 The synthesized 1D example

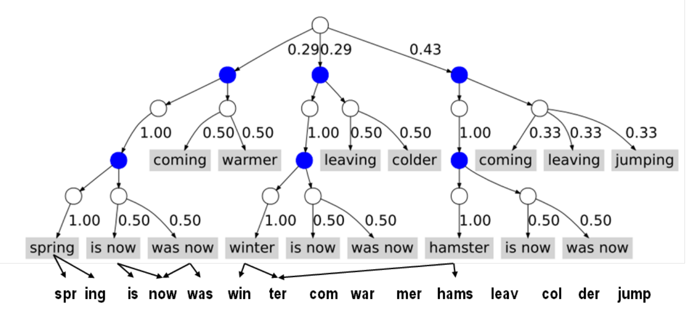


Fig. 9. A stochastic AND-OR template for generating a sentence composed by three parts: subject + linking verb + adjective/present participle, such as "hamster is now jumping". Shaded circles denote AND nodes. Empty circles represent OR nodes. Shaded rectangles are terminal nodes.

To study the identifiability issue, we synthesize a 1D example where we know the underlying AOT as ground truth that generates the training data. Let AOT* be the true AOT. Figure 9 shows AOT* for sentences composed of three parts: subject + linking verb + adjective/present participle (*e.g.* winter is now leaving). The three parts are correlated, such that not all combinations of the three parts are admissible. For example, the combination *spring is now jumping* is not allowed. Each part is in turn composed of a prefix and a postfix. Each part, prefix/postfix and letter can be missing (*i.e.* occluded by random letters) with a small probability (0.01). Finally, random letters of varying lengths are inserted between the three parts of the sentence.

5.1.1 The data and data matrix

Table 2 shows several example strings generated by this underlying AOT. Our goal is to learn an AOT from the sampled example strings, and compare the learned AOT with the underlying one in Figure 9 to study the effectiveness of the learning algorithm in identifying its parts and composite structures.

TABLE 2
String examples.

1. nkfnwkn**spring**yzxyxu**was now**jvzea**warmer**tgprh
2. oqsdq bov**hamster**iwxwo**was now**tdxtzbyc**coming**bjxp
3. lhtuwbcdzfz**hamster**aquo **is now**zgoclu**jumping**mmqrlu
4. jlmzzrsl**winter**vmqdle**is now**napla**leaving**douggkwh

5.1.2 Recursive block pursuit for text data

We first identify frequent substrings of length l ($l = 3$ or 4), such as "ing", "ster", as significant blocks in the data matrix. These blocks are selected into the first level dictionary $\Delta^{(1)}$ (Figure 10).

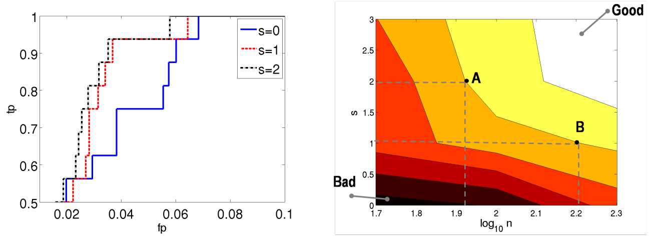


Fig. 12. The effect of the separation parameter s and training sample size n on learned dictionary. **Left:** ROC curves for different separation parameters. **Right:** AUC as a function of separation s and sample size n.

Once $\Delta^{(1)}$ is learned, the strings are re-encoded using the entries in $\Delta^{(1)}$. We then construct a new data matrix by collecting co-occurrences of $\Delta^{(1)}$ entires. As a result, frequent combinations such as "spr"+"ing" are identified as significant blocks and selected into the second level *word* dictionary $\Delta^{(2)}$. An entry in the word level dictionary covers 6 to 8 letters. The word dictionary contains many duplicate or overlapping entries, such as "hamster" and "amster". The nuance entries like "amster" are pruned by a greedy procedure of finding best matching and local inhibition. In the end, only high frequency words remain in the top of $\Delta^{(2)}$ (Figure 10). Notice that compared to $\Delta^{(1)}$, $\Delta^{(2)}$ contains much less ambiguity. Finally the level 3 dictionary (sentences) $\Delta^{(3)} = \{\text{"spring is now coming"}, \cdots\}$ is easily obtained by identifying frequent combinations of words.

5.1.3 Evaluation on the text example

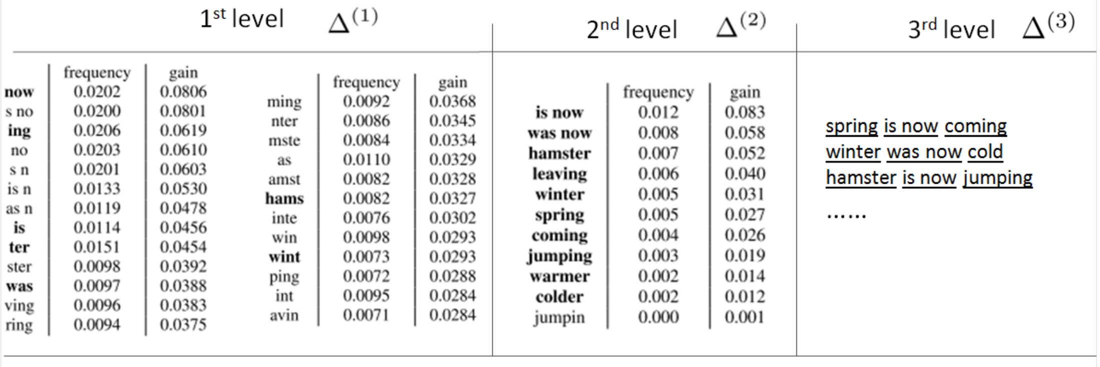
In this experiment, we are interested in studying the following factors that influence model identifiability:

- n: training sample size.
- s: the average length of random letters inserted between two words (see Table 2) in the underlying AOT. When s is small, words are hard to separate, which results in large ambiguity. s implies the articulation and separation of parts in images.
- α: the parameter controlling model complexity.

n and s are parameters of training data, and α is a parameter of the learning algorithm.

We design a comprehensive experiment with about 10^5 combinations of parameters. And the results are summarized in Figure 12 and 13.

Evaluating the learned terminal nodes of AOT. To compare the underlying true terminal nodes Δ_{true} with the



1st level $\Delta^{(1)}$

	frequency	gain
now	0.0202	0.0806
s no	0.0200	0.0801
ing	0.0206	0.0619
no	0.0203	0.0610
s n	0.0201	0.0603
is n	0.0133	0.0530
as n	0.0119	0.0478
is	0.0114	0.0456
ter	0.0151	0.0454
ster	0.0098	0.0392
was	0.0097	0.0388
ving	0.0096	0.0383
ring	0.0094	0.0375

	frequency	gain
ming	0.0092	0.0368
nter	0.0086	0.0345
mste	0.0084	0.0334
as	0.0110	0.0329
amst	0.0082	0.0328
hams	0.0082	0.0327
inte	0.0076	0.0302
win	0.0098	0.0293
wint	0.0073	0.0293
ping	0.0072	0.0288
int	0.0095	0.0284
avin	0.0071	0.0284

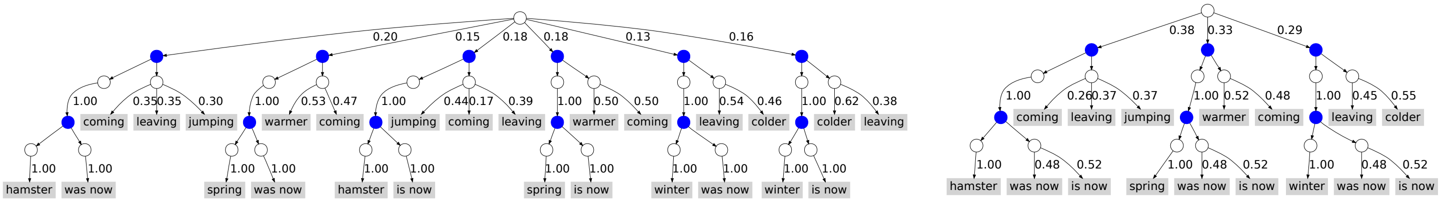
2nd level $\Delta^{(2)}$

	frequency	gain
is now	0.012	0.083
was now	0.008	0.058
hamster	0.007	0.052
leaving	0.006	0.040
winter	0.005	0.031
spring	0.005	0.027
coming	0.004	0.026
jumping	0.003	0.019
warmer	0.002	0.014
colder	0.002	0.012
jumpin	0.000	0.001

3rd level $\Delta^{(3)}$

spring is now coming
winter was now cold
hamster is now jumping
……

Fig. 10. **Left:** The learned dictionary of terminal nodes $\Delta^{(1)}$ for three/four letter groupings (white space is included). We only show the top ones, together with their frequencies and information gains side by side, up to a constant multiple. **Middle:** The learned second level dictionary $\Delta^{(2)}$ for words composed by entries in the children dictionary $\Delta^{(1)}$. **Right:** The learned dictionary for sentences as combinations of $\Delta^{(2)}$ entries.



(a) initial AOT

(b) compressed AOT

Fig. 11. **Left:** the initial AOT naïvely constructed. **Right:** the compressed AOT. Both are obtained from the same 100 training sequences sampled from the underlying AOT in Figure 9. The model complexity parameter α is set to 0.05. The initial AOT has 13 free parameters. The compressed AOT has only 9 free parameters and successfully recovers the structure of true underlying AOT.

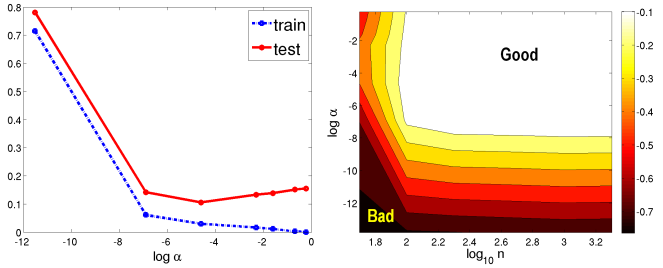


Fig. 13. The effect of the model complexity α and training sample size n on model generalizability. **Left:** Error of learned model (KL divergence) as a function of model complexity α, plotted on training and testing data respectively. **Right:** KL divergence as a function of n and α.

learned terminal nodes Δ. We use the ROC curve and AUC (area under ROC curve) to evaluate the difference between manually labeled ground-truth Δ_{true} and the learned Δ with entries ranked by information gain. Figure 12 (left figure) plots three ROC curves for three different values of s for sample size $n = 100$. After repeating this for different n, we obtain a series of ROC comparisons. To summarize this, Figure 12 (right figure) shows the isolines of AUC as a function of two variables: s the separation parameter, and n the training sample size. Take the two points A, B as an example, when the separation decreases by 1 from A to B, we need about twice ($10^{0.3}$) as many training examples to achieve the same AUC.

We find that the block pursuit algorithm can successfully identify the terminal nodes Δ for $n > 100$ and $s > 2$, up to an accuracy of AUC = 0.99.

Evaluating on the graph topology and branching probabilities of the learned structural AOT. Another important factor is the parameter α which controls model complexity. We set α to different values and compress the sentence-level dictionary $\Delta^{(3)}$ into a series of AOTs with varying complexity. We then compute the distance between the learned AOT and the underlying true AOT* shown in Figure 9. We use the Kullback-Leibler divergence as the distance between AOTs, which is estimated by Monte-Carlo simulation on their samples. We first sample m configurations from AOT*:

$$\{\mathbf{b}_1, ..., \mathbf{b}_m\} \sim p(\mathbf{b}; \text{AOT}^*).$$

Then we compute:

$$\mathcal{K}(\text{AOT}^*|\text{AOT}) \approx \sum_{i=1}^{m} \log \frac{p(\mathbf{b}_i; \text{AOT}^*)}{p(\mathbf{b}_i; \text{AOT})}$$

For each $\mathbf{b}_i$ $(i = 1, ..., m)$ we compute $p(\mathbf{b}; \text{AOT})$ and $p(\mathbf{b}; \text{AOT}^*)$ by the product of branching probabilities along the paths of AOT that generates $\mathbf{b}$.

We perform 10^5 repeated cross validations to compare training error and testing error. Different from the classification scenario, the *training error* for AOT is defined as the KL divergence $\mathcal{K}(\hat{f}_n^{\text{train}}\|\text{AOT})$ between the learned AOT and training data $\hat{f}_n^{\text{train}}$. Here $\hat{f}_n^{\text{train}}$ denotes the empirical distribution on the training data $\{\mathbf{b}_1^{\text{train}}, ..., \mathbf{b}_n^{\text{train}}\}$. The *testing error* for AOT is defined as the KL divergence $\mathcal{K}(\hat{f}_m^{\text{test}}\|\text{AOT})$, where $\hat{f}_m^{\text{test}} = \{\mathbf{b}_1^{\text{test}}, ..., \mathbf{b}_m^{\text{test}}\}$ is another independent sample from AOT*. And $m >> n$. In Figure 13 (left), the horizontal axis is the logarithm of α which is sampled at seven points $(10^{-6}, 10^{-3}, 10^{-2}, 0.1, 0.2, 0.5, 0.8)$, and the vertical axis denotes the model error computed as KL divergence. Recall that large α results in larger more complexity. When the model becomes more complex, the training error always decreases, but the testing error would first decrease but then increase due to overfitting. Figure 13 (right) shows at what sample size and what α values can we successfully recover the generating grammar. The horizontal axis is the logarithm of training sample size $\log_{10} n$, and the vertical axis is $\log \alpha$. The color intensity in this 2D contour map denotes the testing error.

We find that the graph compression algorithm can successfully identify the AOT for $n > 100$ and $10^{-6} < \alpha < 10^{-2}$, up to a tolerance of 0.1 in KL divergence.

5.2 Unsupervised learning of image AOT

Interesting practical issues arise from learning image AOT in the unsupervised setting. Firstly, for the learning algorithm to scale to a large dataset, an online version of the learning algorithm is desirable. Secondly, it is interesting to see how the learned part dictionary grows with the number of training examples. Thirdly, we care about the stability of the learning algorithm with respect to perturbations in the training sequence. We also compare our learning algorithm with state-of-the-art latent SVMs methods for discovering hidden clusters in image data.

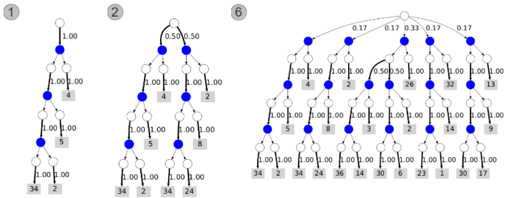


Fig. 14. Online learning of AOT.

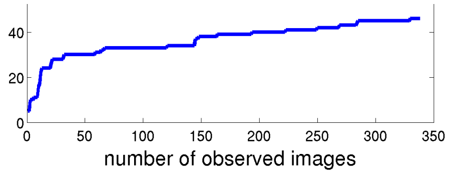


Fig. 15. Sub-linear growth of the learned dictionary.

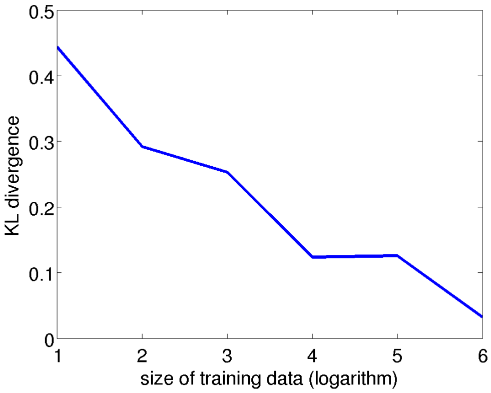


Fig. 16. Robustness of AOT learning evaluated by perturbation in KL divergence.

Online AOT learning. We implemented an online version of both block pursuit for part dictionary and graph compression. When a new training example comes in, we first try to explain it using existing dictionary and AOTs. If the likelihood of the AOT or a part scores a low likelihood on the new image, we then update both the dictionary and the AOT. The online dictionary learning is achieved by an incremental version of EM algorithm. The incremental updating of the stochastic AOT is carried out by inserting the new configuration into the current AOT, creating new OR branches and updating branching probabilities. Figure 14 gives the first, second and sixth updates of the AOT using 320 animal face images.

Dictionary growth. With the online learning algorithm, we are able to observe the size of the dictionary as a function of the number of training examples (Figure 15). The sub-linear growth of the dictionary shows that the learned dictionary of parts is highly re-usable and shared by several animal face categories. The number of nodes in the learned AOT scales linearly with the dictionary size. Thus the complexity of AOT also grows sub-linearly to the number of training examples.

Robustness of AOT learning. Another important question is, how many training examples are needed so that the AOT is reliably learned. We study this question by performing the following cross validation: two independent samples of size n are drawn from the training data of animal faces, from which two AOTs are learned. Both AOTs are then evaluated on a third independent sample of data (testing examples) by KL divergence. The difference in KL divergence is recorded. A smaller difference indicates that the learned AOT is closer to convergence, and thus more identifiable. We do this for 5 repetitions and for different training sizes n. The result is summarized in Figure 16. The AOT is increasingly identifiable as n grows, and when n reaches 100 an AOT can be reliably identified up to a KL divergence of 0.1 (bits). The KL divergence is computed by Monte Carlo

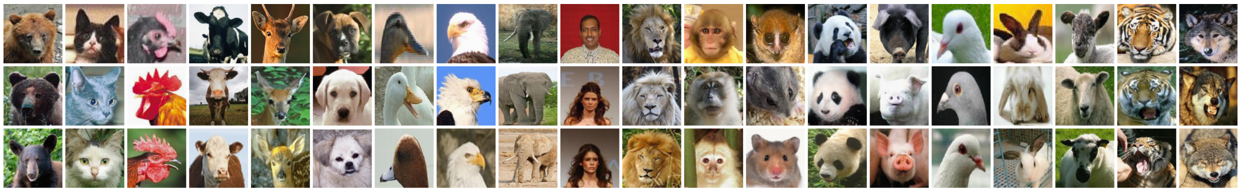


Fig. 17. Training examples from 20 animal face categories.

estimation:

$$\mathcal{K}(\mathrm{AOT}^*|\mathrm{AOT}) \approx \sum_{i=1}^{m} \log \frac{p(\mathbf{b}_i; \mathrm{AOT}^*)}{p(\mathbf{b}_i; \mathrm{AOT})}$$

Evaluating learned AOTs by performance of clustering. For the object dictionary, we evaluate on two examples of unsupervised clustering on i) 142 cat and wolf images, and ii) 100 rabbit images. The rabbit images contain two distinct types with standing ears and relaxed ears. For both examples, the number of clusters is 2. We use the labeling cost by [10] to measure the performance of clustering. This cost reflects the labeling effort needed for a user to label all the images, given the clustering result. The user is assumed to be able to use two buttons: one to assign a single label to all images in a cluster, and one to assign a label to a single image. In Table 3, we compare the label cost of clustering for AOT and latent SVMs. We also show the cost for an optimal clustering (*e.g.* cats in one cluster, wolves in the other) and a random clustering. From this result, we can see that the learned AOT is able to identify meaningful object clusters more reliably. On a larger scale, Figure 1 shows the learned AOT from a mixture of 320 animal face images (bear, cat, wolf and cow) without knowing their categories. The AOT successfully identifies the four hidden clusters.

TABLE 3
Cost of unsupervised clustering.

	AOT	LSVM	optimal	random
cat & wolf	3	2	2	61
rabbit	5	36	2	38

5.3 Animal faces classification

We perform classification experiments on the **LHI-Animal-Faces** dataset[1] [17]. Figure 17 is an overview of the dataset, which consists of about 2200 images for 20 categories of animal faces. Compared to other benchmarks, LHI-Animal-Faces has several desirable properties: (1) the animal face categories are similar to each other due to evolutional relationship and shared parts, and it is a challenging task to discern them; (2) the animal face categories exhibit interesting within-class variation, which includes (i) rotation and flip transforms, e.g. rotated panda faces and left-or-right oriented pigeon heads; (ii) posture variation, e.g. rabbits with standing ears and relaxed ears; and (iii) sub-types, e.g. male and female lions.

TABLE 4
Classification accuracy on animal faces.

HoG+SVM [4]	HIT [17]	LSVM [5]	AOT
70.8%	75.6%	77.6%	79.1%

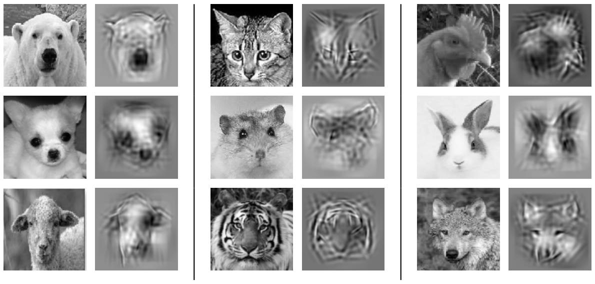


Fig. 18. Reconstructed image generated by AOT templates matched to objects. For each pair of images, the observed image is shown on the left, followed by the reconstructed image is on the right.

In Table 4, we compare the classification accuracy to three related works: (i) HoG feature trained with linear SVM [4], (ii) our prevoius work HIT [17] and (iii) part-based HoG features trained with latent SVM [5]. For all methods, we use 30 positive training images for each category, and we use the rest as testing examples. For AOT, the log-likelihood ratio score in Eq. (8) is used for evluating the classification performance. We use parameter $\eta = 0.1$ as the stopping criterion for feature selection, i.e. stop when the information gain of newly selected feature is below 0.1 (bits). For each category, the number of selected primitive features are around 200, and the number of selected blocks are around 20.

Our AOT model is fully generative to the pixels. In Figure 18 we show several examples of reconstructed image generated from the inferred (or matched) AOT. The reconstructed image captures rich information of the articulated object.

While the AOT shows competitive performance on object categorization against state-of-the-art methods, it requires training to be carried out separately per category. This poses heavy computational burden when scaling up to hundreds or thousands of categories. One way to tackle this problem is to use a dictionary of shared parts, in the hope that the number of distinctive parts grows sub-linearly to the number of object categories. In

[1] http://www.stat.ucla.edu/~zzsi/hit/exp5.html

Figure 15 we show an example of this sub-linear growth of shared dictionary. We are working on using shared dictionary to improve the learning of AOT models.

5.4 Detection of AOT

5.4.1 *Weizmann horse*

We apply our learning algorithm for the Weizmann horse dataset [1] and perform detection of articulated horses. There are 328 horses in the dataset. We train on the first 100 images, and test it on the rest of horse images and large generic images. Figure 19 shows the precision-recall curves for horse detection. Again, we compare with the active basis [20], which is the closest work to ours. The proposed AOT model compares favorably with the active basis which has no compositional hierarchy. In computing the precision-recall measurement, we use 8.9×10^4 negative examples collected from random background images. A detection is counted correct if the bounding box overlaps with ground-truth by 80%. The comparison is more evident when one inspects the detailed template matching result in Figure 20, in which we also show the learned AOT (parts illustrated with different colors) together with the active basis template. For most testing images, the AOT can capture the articulation more accurately, while the active basis model only allows local deformation and does not capture the articulation as well.

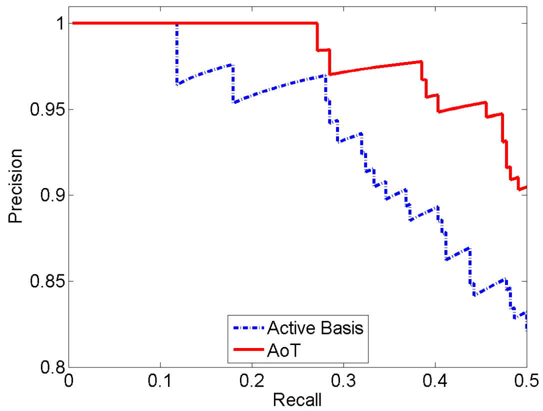


Fig. 19. Precision-recall curves for horse detection on the Weizmann horse dataset.

5.4.2 *More challenging datasets*

We test the detection performance of AOT on three more challenging datasets: egret, deer and side-view PASCAL VOC 2007 bikes, in comparison to the state-of-the-art part-based latent SVMs model (LSVM) [5]. In particular, the VOC bike dataset contains some difficult examples with large occlusion. For all the three categories, we use a small training set of around 20 images. More detailed setting for the training/testing split can be found in Table 5.

Figure 21 and Figure 22 are the learned object templates for the egret and VOC bike using AOT and the

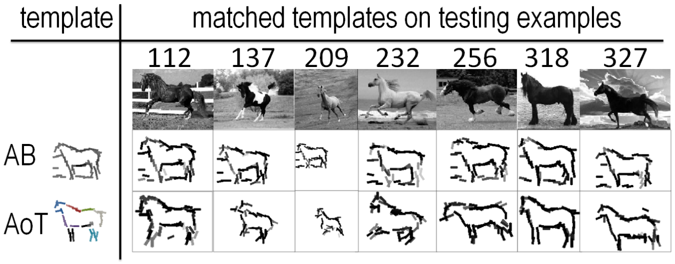
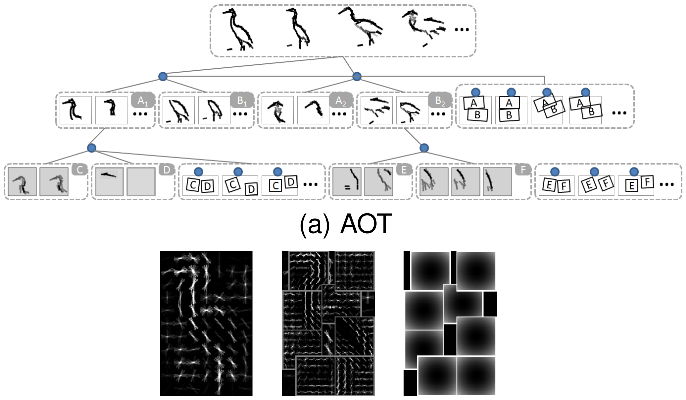


Fig. 20. Learned common templates and their deformed versions registered on testing images of the Weizmann horse dataset. For both AOT and active basis the total number of Gabor elments is 60.

TABLE 5
Training and testing sizes for the detection experiment.

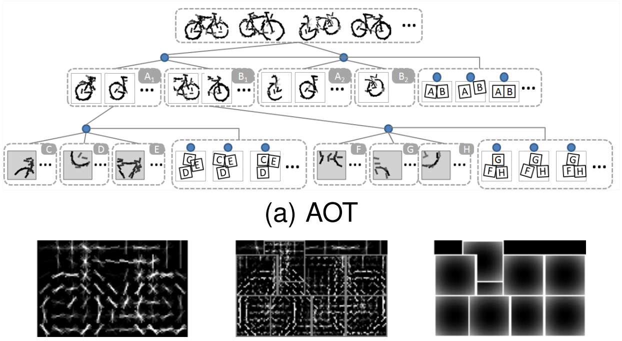
	egret	deer	bike
train	25	15	20
test	67	128	161



(a) AOT

(b) LSVM

Fig. 21. Templates learned for egret. (a) AOT. 4 (deformed) templates from AOT are shown on the top row, and the learned part templates are shown on the bottom row. (b) Part-based model learned by Latent SVM.



(a) AOT

(b) LSVM

Fig. 22. Templates learned for VOC bike. (a) AOT. 4 sampled (deformed) templates from AOT are shown on the top row, and the learned part templates are shown on the bottom row. (b) Part-based model learned by Latent SVM.

part-based latent SVMs. In contrast to the discriminatively trained system [5] which uses a large number of features ($O(10^4)$), the AOT learned for each category only contains less than 200 features in total. It takes less than 10 minutes to learn an AOT with around 20 positive example images, on a machine with Intel i5 processor (3GHz) and 8GB RAM. On the same machine, detection in one image takes half a minute. No negative examples are used in training as the computation of the normalizing constant is put in the pre-computation stage. While for latent SVMs, a large number of negative examples need to be used and going over them in each optimization iteration requires heavy computation. So despite the highly optimized implementation of the training algorithm, it takes much more time (several hours) to train a latent SVMs model.

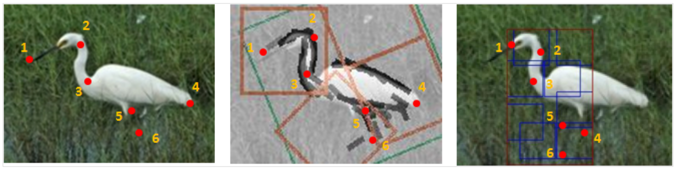
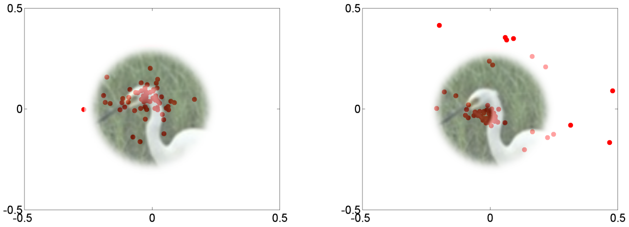


Fig. 23. Measuring the accuracy object localization with key points.



(a) AOT

(b) LSVM

Fig. 24. 2D scatter plots of localization error (normalized by the object size) for the second key point of the egret.

Evaluating the accuracy of localizing objects, parts and keypoints. We evaluate the localization of objects, parts and pixel-level key points as the performance metric for detection. This is done using manually selected key points that are easy to be identified by a human labeler. Figure 23 shows the ground truth key point labelings (in total 6 key points), along sided with the detected key points by AOT and by latent SVMs. The key point labeling is only used in evaluating the detection results, and not used in the training of either models. After the AOT model is learned, we find the nearest edge element to each key point in the training images, and record the most likely location of the key point relative to that edge element. For the latent SVMs, we find the nearest part for each key point and record the most likely location of the key point relative to that part. We then use this information to locate the key points for each testing image.

Figure 24 shows the detections for the second key point (joint of neck and head) of egret using the two methods. Each red dot in the point cloud denotes one detection, and its coordinate means the displacement in horizontal and vertical directions. The displacements are normalized by dividing the size of the object. The AOT can locate the key points much more accurately.

To numerically measure the performance of localization, we use an imprecision-recall curve. A higher curve indicates better performance. In this curve, the horizontal axis is the tolerance for normalized displacement $\sqrt{\Delta x^2 + \Delta y^2}$ by dividing the object size. We restrict the range to be $[0, 1]$ for convenience. The vertical axis is the recall rate (between 0 and 1), *i.e.* the percentage of correctly detected points that fall within the specified displacement tolerance. As we tolerate more displacement, the recall rate increases. We use the area under curve (AUC) to measure the average recall rate.

Figure 25 shows the imprecision-recall curves for 6 key points (tip of beak, joint of head and neck, joint of neck and body, tail, top of standing leg, and bottom of standing leg) of egret. We also show the curves for 4 parts (head, neck, body, and leg) and the whole object. To get the curves for parts and the object, the displacement of the part is computed by averaging the displacements of key points associated with that part; and the displacement of the object is computed by averaging the displacements of all the key points. Our model performs localization consistently better than the latent SVMs, for all the parts and key points and for all the imprecision values (displacement tolerances). Table 6 provides a numerical comparison for egret, deer and VOC bikes, using the area under curve (AUC) measure. The part and key point AUCs are computed by averaging over all parts and key points. We also measured the average precision using PASCAL standard criteria on VOC2007 bikes, with AOT getting an average precision of 60.8% vs. 59.5% for LSVM.

TABLE 6
AUCs for localization of object, parts and key points.

	object		part		keypoint	
	AOT	LSVM	AOT	LSVM	AOT	LSVM
egret	.93	.80	.88	.76	.88	.73
deer	.93	.83	.91	.79	.90	.75
bike	.78	.76	.70	.66	.68	.61

In Figure 26, 27 and 28 we show the detection results on several challenging testing images. From these examples we can see that the AOT can locate the object boundary and inner structures with a higher precision, which leads to more accurate localization overall.

6 Conclusion

We propose a hierarchical reconfigurable object template called the AND-OR Template (AOT) model, which can capture rich structural variations and shape variations of

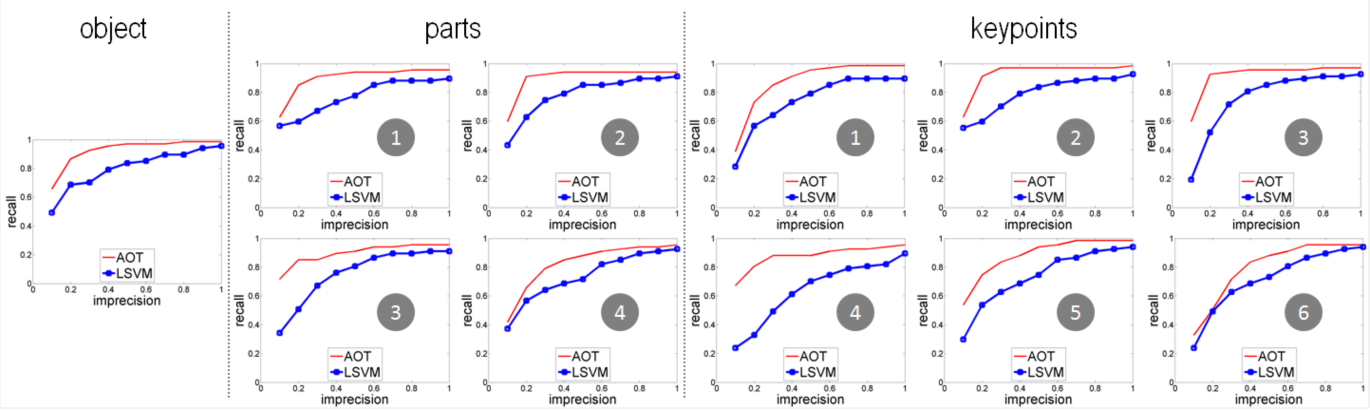


Fig. 25. Egret: the quantitative performance measure for localization of object, parts and key points.

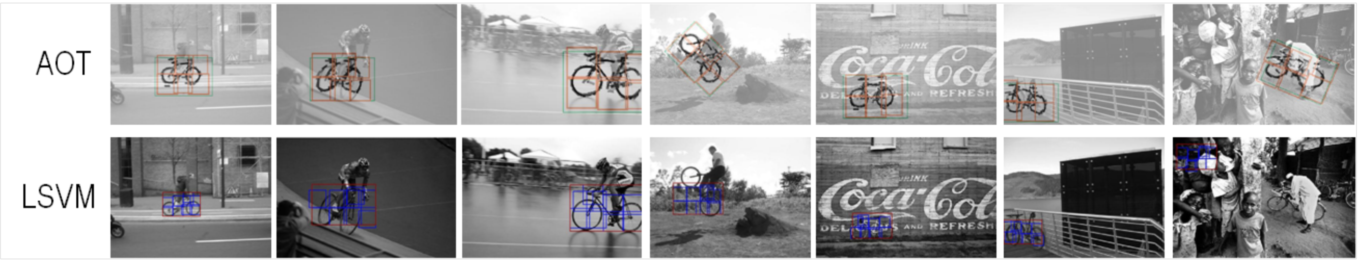


Fig. 26. Detection results on VOC bikes. For AOT, we show the detected bounding boxes for objects and parts, as well as individual sketches. As a comparison, we also show detection results by latent SVMs model [5]. Best viewed in color.

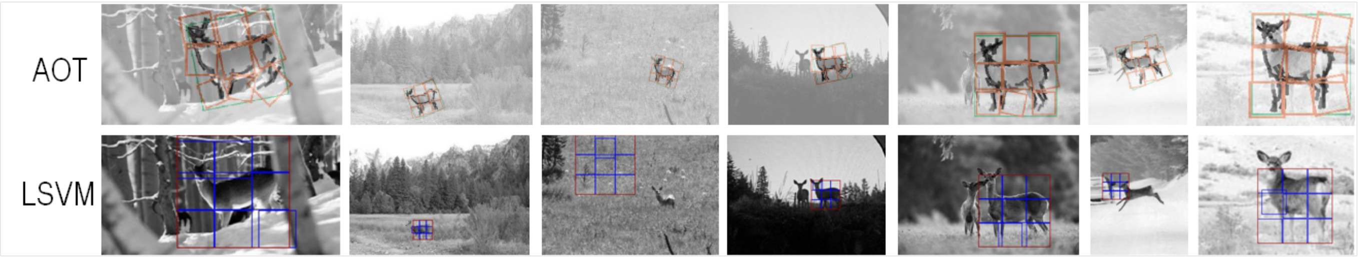


Fig. 27. Detection results on Deer. For AOT, we show the detected bounding boxes for objects and parts, as well as individual sketches. As a comparison, we also show detection results by latent SVMs model [5]. Best viewed in color.

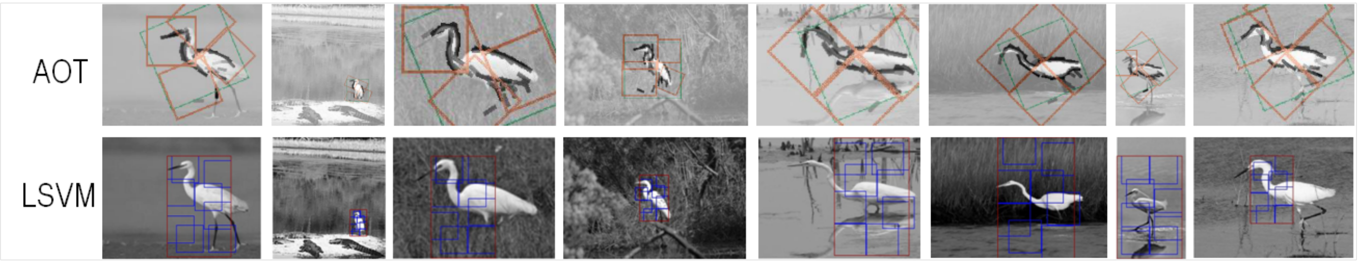


Fig. 28. Detection results on Egrets. For AOT, we show the detected bounding boxes for objects and parts, as well as individual sketches. As a comparison, we also show detection results by latent SVMs model [5]. Best viewed in color.

objects. We propose an unsupervised learning algorithm for learning AOTs from only images and no manual labeling. In our learning algorithm, the pervasive ambiguity of parts is overcome by 1) articulation of parts, 2) alternative compositions, both of which imply the importance of OR nodes. This is a major contribution of our work. We investigate the factors that influence how well the learning algorithm can identify the underlying AOT, and we design a number of ways to evaluate the performance of the proposed learning algorithm through both synthesized examples and real world images. The proposed AOT model achieves good performance on par with state-of-the-art systems in public benchmarks for object detection. Moreover, the AOT model has the advantage of significantly less training time and fewer parameters. In future work, we will allow the parts of visual objects to be non-square regions and search for the optimal part segmentation with flexible shape decompositions.

ACKNOWLEDGEMENT

We thank Dr. Ying Nian Wu for detailed suggestions on both the theory and experiments for this paper. This work is supported by DARPA grant FA 8650-11-1-7149, NSF IIS1018751 and MURI grant ONR N00014-10-1-0933.

REFERENCES

[1] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, pages 109–124, 2002.
[2] L.-B. Chang, Y. Jin, W. Zhang, E. Borenstein, and S. Geman. Context, computation, and optimal roc performance in hierarchical models. *IJCV*, 93(2):117–140, 2011.
[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *TPAMI*, 23(6):681–685, 2001.
[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
[5] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627 – 1645, 2010.
[6] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.
[7] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *IJCV*, 71(3):273–303, 2007.
[8] S. Fidler, M. Boben, and A. Leonardis. A coarse-to-fine taxonomy of constellations for fast multi-class object detection. In *ECCV*, 2010.
[9] S. Fidler and A. Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. In *CVPR*, 2007.
[10] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *ICCV*, 2009.
[11] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.
[12] Y. Jin and S. Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, 2006.
[13] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *TPAMI*, 19(4):380–393, 1997.
[14] P. Schnitzspan, M. Fritz, S. Roth, and B. Schiele. Discriminative structure learning of hierarchical representations for object detection. In *CVPR*, 2009.
[15] G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):464–464, 1978.
[16] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Object recognition with cortex-like mechanisms. *TPAMI*, 29:411–426, 2007.
[17] Z. Si and S.-C. Zhu. Learning hybrid image templates (hit) by information projection. *TPAMI*, 99(PrePrints), 2011.
[18] E. B. Sudderth, A. B. Torralba, W. T. Freeman, and A. S. Willsky. Describing visual scenes using transformed objects and parts. *IJCV*, 77(1-3):291–330, 2008.
[19] S. Todorovic and N. Ahuja. Unsupervised category modeling, recognition, and segmentation in images. *TPAMI*, 30(12):2158–2174, 2008.
[20] Y. N. Wu, Z. Si, H. Gong, and S.-C. Zhu. Learning active basis model for object detection and recognition. *IJCV*, 90(2):198–230, 2010.
[21] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *CVPR*, 2011.
[22] L. Zhu, Y. Chen, A. Torralba, A. Yuille, and W. T. Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010.
[23] L. Zhu, Y. Chen, and A. Yuille. Unsupervised learning of probabilistic grammar-markov models for object categories. *TPAMI*, 2009.
[24] S. C. Zhu and D. Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362, 2006.
[25] S.-C. Zhu, Y. N. Wu, and D. B. Mumford. Minimax entropy principle and its applications to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.

Zhangzhang Si received a BS degree in Computer Science from Tsinghua Univ. in 2006 and a PhD degree in Statistics at UCLA in 2011. He worked as a postdoc in the UCLA Center for Vision, Cognition, Learning and Arts (VCLA), where his research focuses on deformable image models for object recognition. He received a honorary nomination for Marr's Prize in 2007 with Y.N. Wu and S.-C. Zhu. He is currently working at Apple. Inc.

Song-Chun Zhu received a BS degree from Uunversity of Science and technology of China, at Hefei in 1991 and a PhD degree from Harvard Univ. in 1996. He is currently a professor of Statistics and Computer Science at UCLA. His research interests include computer vision, statistical modeling and learning, cognition, and visual arts. He received a number of honors, including the J.K. Aggarwal prize from the Intl Association of Pattern Recognition in 2008 for ”contributions to a unified foundation for visual pattern conceptualization, modeling, learning, and inference”, the David Marr Prize in 2003 with Z. Tu et al. for image parsing , the Marr Prize honorary nominations in 1999 for texture modeling and in 2007 for object modeling with Z. Si and Y.n. Wu. He received the Sloan Fellowship in 2001, a US NSF Career Award in 2001, and an US ONR Young Investigator Award in 2001. He is a Fellow of IEEE.