Pattern Recognition 45 (2012) 3648–3660

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Object categorization with sketch representation and generalized samples

Liang Lin [a,b], Xiaobai Liu [b,c], Shaowu Peng [c], Hongyang Chao [a,*], Yongtian Wang [d], Bo Jiang [a]

[a] School of Software, Sun Yat-Sen University, Guangzhou, China
[b] SCTS & CGCL, Huazhong University of Science and Technology, Wuhan, China
[c] Lotus Hill Research Institute, China
[d] Beijing Institute of Technology, Beijing, China

ARTICLE INFO

Article history:
Received 17 February 2011
Received in revised form
30 December 2011
Accepted 26 March 2012
Available online 1 April 2012

Keywords:
Object categorization
And–Or graph
Generalized samples
Cascaded inference

ABSTRACT

In this paper, we present a framework for object categorization via sketch graphs that incorporate shape and structure information. In this framework, we integrate the learnable And–Or graph model, a hierarchical structure that combines the reconfigurability of a stochastic context free grammar (SCFG) with the constraints of a Markov random field (MRF). Considering the computation efficiency, we generalize instances from the And–Or graph models and perform a set of sequential tests for cascaded object categorization, rather than directly inferring with the And–Or graph models. We study 33 categories, each consisting of a small data set of 30 instances, and 30 additional templates with varied appearance are generalized from the learned And–Or graph model. These samples better span the appearance space and form an augmented training set Ω_T of 1980 (60×33) training templates. To perform recognition on a testing image, we use a set of sequential tests to project Ω_T into different representation spaces to narrow the number of candidate matches in Ω_T. We use "graphlets" (structural elements), as our local features and model Ω_T at each stage using histograms of graphlets over categories, histograms of graphlets over object instances, histograms of pairs of graphlets over objects, and shape context. Each test is increasingly computationally expensive, and by the end of the cascade we have a small candidate set remaining to use with our most powerful test, a top-down graph matching algorithm. We apply the proposed approach on the challenging public dataset including 33 object categories, and achieve state-of-the-art performance.

1. Introduction

Object category recognition is plagued by two opposing needs, particularly when the categories have high intra-class variance.

1. One wants *many training instances* to recognize the many appearances an object can have when learning.
2. One wants *few training instances* to match to when performing inference.

This issue is generally resolved by solving one problem or the other – either the task becomes classification [9,23,19,28,14,26], in which case many training instances are used to learn a classifier that labels an entire image, or few candidates are used to learn a single generative model [4,8,10,11,22] that can often only recognize classes of object that are similar in appearance and is computationally intensive to perform inference with.

To solve this problems, we use a compositional model capable of representing object categories together with a cascaded discriminative prune and a top-down graph-matching algorithm in a systematic pipeline.

We solve the problem (1) of needing many data points for training by learning an "And–Or Graph" model [20,17,13] from a few training instances. This model was first presented by Han et al. [13] and Chen et al. [2] to capture the variability of object appearances. The probability model on the And–Or graph are later defined in [18,20] that allows us to learn its parameters in a unified way for each category. We can then draw samples from this model that, though perceptually similar in appearance to the original training data, are novel instances. This gives us better coverage of the appearance space of the object category and augments our training set. The similar idea is discussed for face expression recognition proposed in [11].

We then solve problem (2) by pruning the augmented data set using a cascaded prune to arrive at a small set of candidate categories and objects for a target image. At each stage we project our large training set into a different space and narrow down the candidates that could match our target image. We can then activate a flexible graph-matching algorithm [16] to search the image for the small number of candidates remaining. Similar approaches have

* Corresponding author at: School of Software, Sun Yat-Sen University, Guangzhou, China.
E-mail address: isschhy@mail.sysu.edu.cn (H. Chao).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.03.017

performed top-down matching using just a generative model, such as K-Fans or the constellation model [8,4,10,24]. However, these approaches might not be able to learn large structural variations like the And–Or graph can, and they are computationally expensive to perform inference with [25,21]. Our use of generalized samples helps us represent the object space better and our sequential tests help the efficiency of our top-down matching. A preliminary version of this work was introduced in [17].

Our data consist of *sketch representations* of objects, in which perceptually important parts have been outlined in a graphical format, with each image composed of nodes and edges. The left and right columns of Fig. 4 show examples of objects in this format. During the inference phase, raw testing images are also converted to a sketch representation, from which we use their local structural elements (called graphlets) as our features. Graphlets are merely combinations of edgelets defined by their topological relations to one another. Fig. 1(b) shows two typical sketch graphs from raw images with graphlet detection. A dictionary of common graphlets is learned for each category for this task, as shown in Fig. 6.

The sequential tests that we implement are a four stage process that prunes the candidates in a coarse to fine manner, both in the number of candidates kept at each stage and the computational complexity of the object representation. Each discriminative test is a simple nearest neighbor ordering of the candidates, but the representation of the candidates changes at each stage. We first model each category as a *category histogram* and use nearest neighbor to select a set of candidate categories that the target may belong to [7]. We then model each remaining object in each remaining category as a sparse vector in a *bag of words* approach. We next augment our vectors to include *graphlet pair relations*, before finally using *shape context* [1] as our distance metric. By the end of this pruning stage, we have a feasible number of candidates left to match, if any.

Our final verification stage uses a graph matching algorithm [16]. This algorithm can match objects in sketch representation, even under large geometric changes in appearance. If the algorithm matches any candidates, it finds the warping of the object that best fits the target. The combination of these two algorithms, along with the set of sequential tests, is what allows us to recognize objects with highly varied appearances.

We show detailed experiments on classifying and recognizing objects from 33 categories, based on the augmented training data. We also show that our system performs better with synthesized data than without generalized samples, proving the importance of augmenting our dataset with samples from the And–Or graph. In addition, we show the classification rate of our approach as the number of training instances increases, compared to the sate-of-the-arts learning-based categorization methods.

All the data used in our experiments are selected from the Lotus Hill dataset [29], including both annotated images for And–Or graph learning and raw testing images with their category labels.

The remainder of this paper is arranged as follows. We briefly review the And–Or graph representation and discuss the principle of synthesizing instances with the learned object models in Section 2. We then follow with a description of the inference schemes with quantitative experiments and comparisons in Section 3. The paper is concluded in Section 4 with a discussion of future work.

2. Augmenting the training set with And–Or graphs

The And–Or graph for representing and recognizing objects was proposed by Lin et al. [17,18] recently, and a learning and sampling algorithm is presented in [20]. It is a compositional model capable of creating novel instances from a small set of training data. By combining a stochastic context free grammar (SCFG) with the constraints of a Markov random field (MRF), it can represent the variability seen in many object categories yet still constrain the appearances of these objects so that they are perceptually equivalent. Intuitively, for each object category, the basic structural components (object parts or sub-parts) and corresponding relationships are essential and finite, like the basic words and grammar rules, and thus can be learned from a few typical instances, as illustrated in Fig. 4(a) and (b). Furthermore, a huge number of object configurations can be synthesized from the learned And–Or graph that are representative of the in-class variability of each object category, as illustrated in Fig. 4(c).

In the following, we first briefly review the And–Or graph model and then describe the principle of sampling new instances from the learned And–Or graph model.

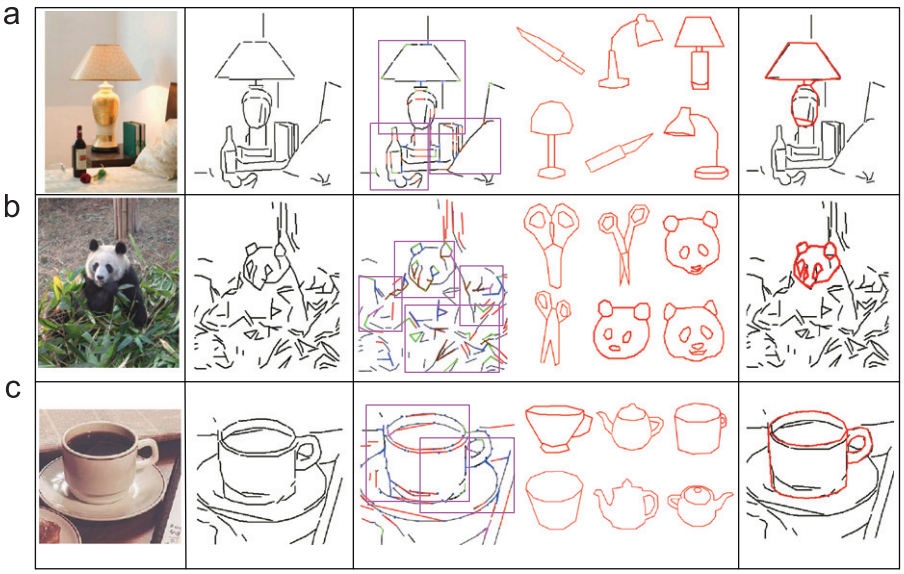

Fig. 1. Illustration of object category recognition. Three typical testing images from 33 categories are shown in (a). The related sketch graphs are computed with graphlet detected in (b). A cascade of discriminative steps prune candidate categories and objects, as well as to localize the objects. Some candidates are shown in (c). A layered graph matching algorithm is finally used for top-down verification, as shown in (d).

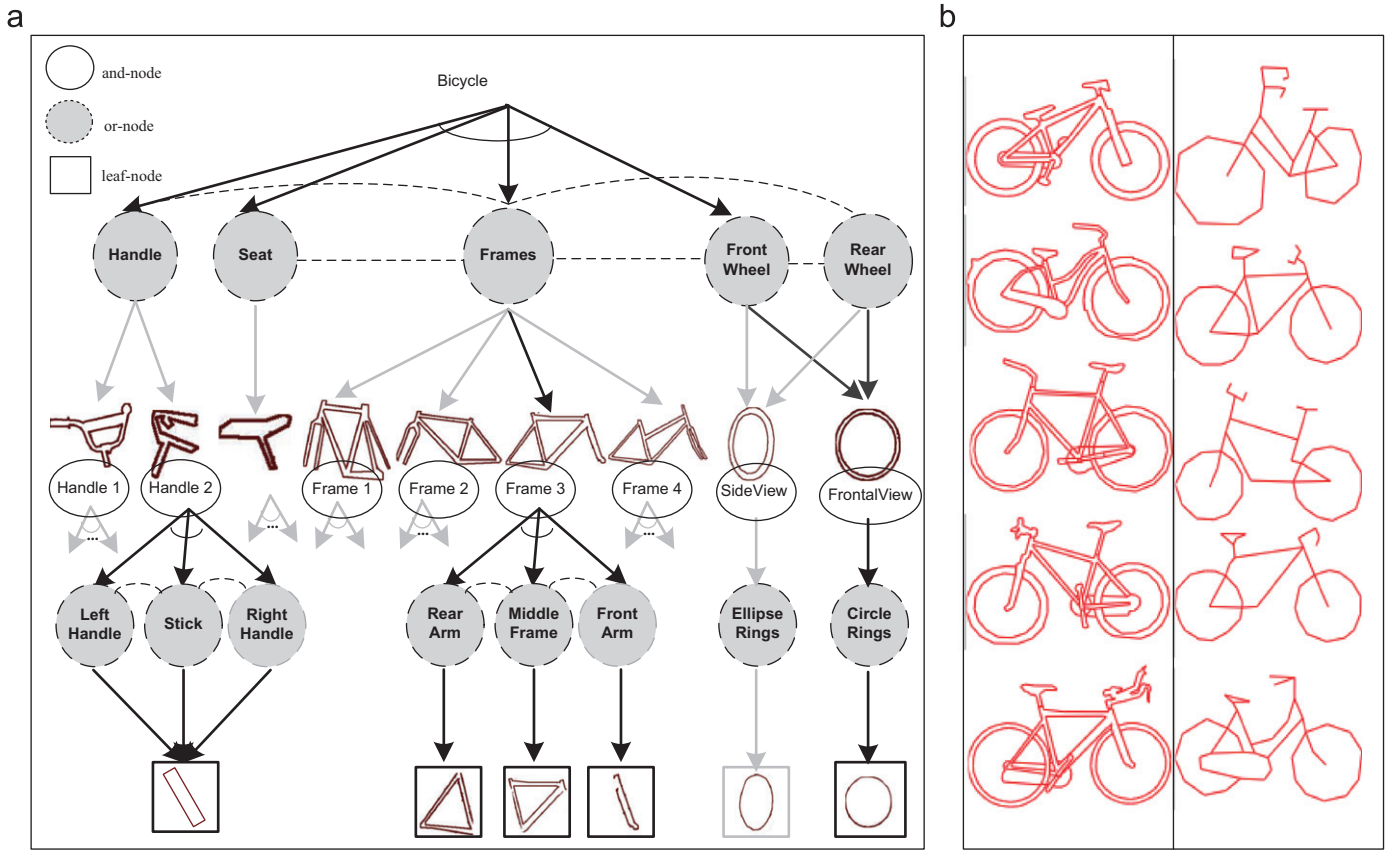

Fig. 2. A specific object category can be represented by an And–Or graph. (a) shows one example for the bicycle category. An Or-node (dashed) is a "switching variable" for possible choices of the components and only one child is assigned for each object instance. An And-node (solid) represents a composition of children with certain relations. The bold arrows form a sub-graph (also called a parse graph) that corresponds to a specific object instance (a bicycle) in the category. The process of object recognition is thus equivalent to assigning values to these Or-nodes to form a "parse graph". This structure of And–Or graph is hand-defined. (b) Several synthesized bicycle instances using the And–Or graph to the left as training data.

2.1. The And–Or graph for object modeling

The And–Or graph can be formalized as the 6-tuple

$$\mathcal{G} = \langle S, V_N, V_T, R, P, \Sigma \rangle,$$

where S is the root node, V_N are non-terminal nodes for objects and parts, V_T are terminal nodes for the atomic description units, R are a set of pairwise relationships, P is the probability model on the graph, and Σ is the set of all valid configurations producible from this graph. We denote $V = V_T \cup V_N$ for all nodes in the graph.

In Fig. 2(a), the root node S is an And node, as bicycle must be expanded into *handle*, *seat*, *frame*, *front wheel*, *rear wheel*, while the handle node is an Or node, as only one appearance of handle should exist for each instance of bicycle. Each Or Node V_i^{OR} has a distribution $p(\omega_i)$ over which of its $\omega = \{1, 2, \ldots, N(\omega_i)\}$ children it will be expanded into. $V_T = \{t_1, t_2, \ldots, t_{T_n}\}$ represents the set of terminal nodes. In our model, the terminal nodes are low-level sketch graphs for object components. They are combined to form more complex non-terminal nodes, as illustrated in Fig. 2(a).

$R = \{r_1, r_2, \ldots, r_{N(R)}\}$ in the formulation represents the set of pairwise relationships defined as functions over pairs of nodes $(v_i, v_j) \in V$, as well as singleton relationships over a single node $v_i \in V$. We define

$$r^a = \phi^a(v_i) r^b = \psi^b(v_i, v_j), \tag{1}$$

where the former denote a singleton relationship function and the latter a pairwise relationship function. These relationships are defined at all levels of the graph. For any graph node v_i, assume its relative position, orientation, and scale are, respectively, denoted as (X_i^x, X_i^y), γ_i, and (σ_i^x, σ_i^y). Then we can specify the relationship functions $\phi(v_i)$ and $\psi(v_i, v_j)$ in Table 1. Clearly, more suitable relationships can be easily incorporated to this list to make the representation richer.

Table 1
Relationship function definitions.

Relation	Type	Function
Aspect ratio	Singleton	σ_i^x / σ_i^y
Position X	Pairwise	$\lvert X_i^x - X_j^x \rvert / \sigma_i^x$
Position Y	Pairwise	$\lvert X_i^y - X_j^y \rvert / \sigma_i^y$
Scale X	Pairwise	σ_i^x / σ_j^x
Scale Y	Pairwise	σ_i^y / σ_j^y
Orientation	Pairwise	$\lvert \gamma_i - \gamma_j \rvert$
Concentricity	Pairwise	$(X_i^x - X_j^x)^2 + (X_i^y - X_j^y)^2$

$P = p(\mathcal{G}, \Theta)$ is the probability model over the graph structure. As the And–Or graph embeds an MRF in a SCFG, it borrows from both of their formulations.

Following the SCFG [3], the structural components of the And–Or graph can be expressed as a parse tree, and its prior model follows the product of all the switch variables ω_i at the Or nodes visited:

$$p(T) = \prod_{i \in V} p_i(\omega_i). \tag{2}$$

Let $p(\omega_i)$ be the probability distribution over the switch variable ω_i at node V_i^{OR}, θ_{ij} be the probability that ω_i takes value j, and n_{ij} the

number of times we observe this production, we can rewrite $p(T)$ as

$$p(T)=\prod_{i\in V^{OR}}\prod_{j=1}^{N(\omega_i)}\theta_{ij}^{n_{ij}}. \quad (3)$$

The MRF is defined as a probability on the configurations of the resulting parts of the parse tree. It can be written in terms of the pairwise energies between parts (i.e. graph nodes) as well the singleton energies. Suppose R_N^1 is the number of singleton constraints, and R_N^2 the number of pairwise constraints, we have

$$p(C)\propto \exp^{-\sum_{i\in V}\sum_{a=1}^{R_N^1}\alpha_i^a(\phi^a(v_i))}\exp^{-\sum_{\langle i,j\rangle\in V}\sum_{b=1}^{R_N^2}\beta_{ij}^b(\psi^b(v_i,v_j))}, \quad (4)$$

where α_i^a and β_{ij}^b denote the co-efficiencies for the corresponding relationship functions.

Therefore, we obtain the final expression of $P=p(\mathcal{G},\Theta)$, as

$$p(\mathcal{G},\Theta)=\frac{1}{Z}\exp\{-E(g)\}, \quad (5)$$

$$E(g)=\log(p(T))+\sum_{i\in V}\sum_{a=1}^{R_N^1}\alpha_i^a(\phi^a(v_i))+\sum_{\langle i,j\rangle\in V}\sum_{b=1}^{R_N^2}\beta_{ij}^b(\psi^b(v_i,v_j)), \quad (6)$$

where Z denotes a normalization constant. $\Theta=(\theta,\alpha,\beta)$ are related parameters of the probability model.

In this work, the meaningful structures of And–Or graphs are hand-defined, and the parameters $\Theta=(\theta,\alpha,\beta)$ can be learned by an efficient minimax entropy method. The algorithm for learning the parameters of the model proceeds in two steps. The detailed And–Or Graph learning process was proposed in [20].

- We first learn the parameter θ for the switch variables of $p(T)$ by MLE, which are just the sample frequencies of the decompositions of each node.
- We then iteratively select significant relationships that help the model best match true statistics for that object class. After each relationship is added, we iteratively update the parameters α and β for all relationships.

2.2. Generating new object instances

One of the important features of the And–Or graph model is its ability to learn representations from a small sample set as well as its ability to generalize to a combinatorial number of novel instances. Thus, we can first learn the model for each category, yet still recognize objects in a testing set that were never seen before. The generation of new object instances from the model comprises two steps: (i) sampling for tree structure (i.e. to select object parts) and (ii) sampling for relationships (i.e. to decide the spatial arrangements of parts).

(*I*) We first sample the tree structure $p(T)$ of the And–Or graph. This is achieved by starting at the root of the And–Or graph and decomposing each node into its children nodes recursively. And nodes are decomposed into all of their children, while an Or node V_i^{OR} selects one of its children to decompose into according to the learned switch probability $p(\omega_i)$. Intuitively, this process is very similar to generating a sentence from a SCFG [3] model. This continues until all nodes have decomposed into terminal nodes. The output from this step is a parse graph of the hierarchy of object parts, though no spatial constraints have been imposed yet. For example, we may have selected certain templates for the wheels, frame, seat, and handle bars of a bike, but we have not yet constrained their relative appearances and spatial arrangements.

(*II*) To further arrange the parts spatially, we use *Gibbs* sampling to explore the relationships for each pair of parts (i.e. graph nodes). At each iteration, we update the value of each of the relationships that exists between each pair of nodes at the same level in the parse graph. The parameters for each relationship are inherited from the And–Or graph, as defined in Table 1. The process of relationship sampling is consistent with the relationship learning described in [20], and the relationships are represented as histograms of the relationship functions. Given the relationship functions in Table 1, we determine the range of the histograms using the maximum and minimum values observed over the training examples in the learning process. In practical, these histograms each are divided into 15 bins. Given a learned relationship in the model, we can thus calculate its response for every value of the relationship function.

In our algorithm, for each relationship $r_k\in R$ between a pair of graph nodes $(v_i,v_j)\in V$, we arrange the two nodes so that the response of this relationship is changed. And the responses of other relationships co-related with this arrangement should be taken into account as well. Then we can accordingly obtain the change of energy δ_k for the parse graph by observing the new values of any relationship functions that were altered by the arrangement. This energy is easily calculated by plugging the affected relationship responses into our probability model $p(\mathcal{G},\Theta)$. We record this energy and then make a new arrangement. Therefore, we can collect a vector of energy changes $\{\delta_k\}$ for every possible arrangements.

For example, given a parse graph of a car, we would need to update the relative position between the wheels and the body. This would be two of the relationships (i.e. Position X and Position Y) between this pair of parts that we would update during this iteration. To achieve this, we set the size of the wheels to each possible value that the relative position can take on, and then compute the resulting change in energy of the parse graph. Any relationships between the body and wheels that depend on the relative position would be affected, as well as any relationships between other parts that depend on the position of the wheels.

Once we have a vector of energy changes for a relationship r_k, we exponentiate to get a probability distribution from which we sample the optimal value (i.e. arrangement with respect to the relationship function) by *Gibbs* sampler. We then move on to the next relationship or next pair of graph nodes. Note that the procedure is the same for the singleton relationships and only one single graph node is proceed.

This two-step sampling continues for 150 iterations in our experiment, at which point we output the final parse graph, which contains the hierarchy we selected as well as the spatial arrangements of the parts. In practice, we limit the number of graph levels is less than 3. The overall process is described in the algorithm below.

Algorithm 1. The sketch of generating new object instances.

Input: A learned And–Or graph model for a category $p(\mathcal{G},\Theta)$
Output: A valid object instance (parse graph) G
1. Sampling for tree structure;
(1) Traverse the graph from root and activate children for Or nodes by sampling $p(T)$.
(2) Obtain an initial parse graph without relationship constraints G_0.
2. Sampling for relationships;
(1) Compute the initial energy of G_0 by randomly setting values for all relationships.
Loop for each pair of nodes (v_i,v_j) at all levels;
Loop for each relationship r_k between v_i and v_j;
(i) Arrange v_i and v_j so that obtain the response of r_k.
(ii) For every relationship co-related with v_i and v_j, compute new responses.
(iii) We can accordingly obtain the change of energy δ_k.
(iv) Repeat step (i)–(iii) for each value r_k can take.

(v) Normalize the energy δ_k to get a probability distribution. Sample a new value from it to achieve an updated parse graph $G_{i,j,k}$.

End loop.

End loop.

3. Output the final generated parse graph G.

This sampling procedure thus selects the new relationship values for the pairs of parts proportional to how much they lower the energy at each iteration. As lowering the energy corresponds to increasing the probability of this tree, we are sampling configurations proportional to their probabilities from our learned model.

Once we have a parse graph with all of the leaf nodes appropriately arranged, we must impose an ordering on the terminal nodes to create an appropriately occluded object. This is a logistic issue that is needed to transform our object from overlapping layers of parts into one connected structure. In our experiment we hard-coded these layer values by part type. For example, teapot spouts are always in front of the teapot base. We are currently experimenting with learning an occlusion relationship between pairs so that we can sample this ordering in the future as well. Once the ordering is determined, intersection points between layers are determined and the individual leaf templates are flattened into one final template.

By the end of this process we can produce samples that appear similar to the training data, but vary in the arrangements and configurations of their parts. Fig. 4 shows three And–Or graphs, simplified for the sake of space. On the left of each we see instances from three categories, teapot, clock, and car, and on the right we see high and low resolution samples produced by the And–Or graphs for these categories. We can see that the output images are perceptually similar to the input images, though they may have different part configurations than were observed, thus comprising novel instances of the object.

The And–Or graph's ability to generate novel instances is what makes it particularly powerful for our task. By sampling a large number of instances, we better cover the appearance space of an object category, thus more accurately representing it in our discriminative tests and providing more candidates for our top-down match. Fig. 3 illustrates this concept. The asterisks represent our initial sample set, which does not cover much of the appearance space. Using just the initial set, we would likely not match many of the target images using nearest neighbor. However, with the samples from the And–Or graph included, pictured as gray circles, we can cover a much wider portion of the space, thus increasing the probability that we would find matches for this category. Fig. 4(c) shows examples of these samples, along with the corresponding And–Or graph for that category (Fig. 4(b)). Compared with the traditional methods having generalization ability (e.g., SVM, PCA), the And–Or graph model advances in three aspects. First, due to the meaningful sketch representation, the generated examples are easy to be visualized and validated. Second, our generalization is more effective, i.e. largely generalizing the original samples, because we represent objects with compositional structures rather than simply mapping objects into feature vectors. Last, the learning and generalizing are separated in our approach, which helps us to well supervise the object modeling process.

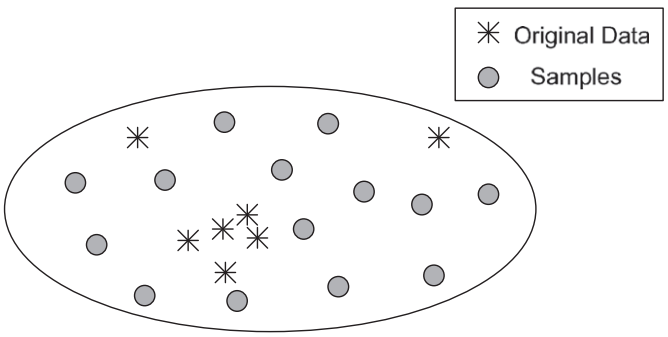

Fig. 3. Instances mapped into the appearance space. The original samples are too sparse to cover the space, but samples from the And–Or graph increase coverage.

To quantitatively illustrate the And–Or graph's ability to synthesize new object configurations, we introduce a metric, namely *coverage*, to show the minimum training size needed to represent a category with our model. we collect a number M^s of objects (about 50 for each category) which we assume span the configuration space. We then divide these objects into the training set Ω^a and testing set Ω^b. We then learn And–Or graph models for these categories using an increasing number M^k of training samples, i.e. the size of Ω^a keeps increasing. And we sample instances $\{G_i\}_{i=1}^{M^c}$ from the learned model at each stage and match to the objects in Ω^b. Here we simply match objects by geometric alignment. Thus, we can count how many objects in Ω^b can be generated from this model. The coverage is defined as

$$Coverage = \frac{\Sigma_{i=1}^{M^c} 1(G_i \in \Omega^b)}{M^b}. \quad (7)$$

Fig. 5 shows the results for five categories as the training size increases. We can see that all categories can learn the maximal coverage of the configuration space with as few as ten archetypal training instances.

3. Inference

We next describe how to perform inference using the augmented data set. We first select a dictionary of "graphlets" to be used as local features, and describe the approach to compute sketch graphs from raw images using graphlet detection. Then we use four sequential bottom-up tests and a top-down graph matching algorithm to classify and recognize objects from raw images over 33 categories.

3.1. Graphlets – "the Visual Words"

We now have a huge training set of synthesized instances from which to perform classification and, ultimately, recognition. We next learn a dictionary of graphlets to use as local features for candidate pruning as well as building a sketch representation of an unlabeled image.

Graphlets can be defined by a 3-tuple $g_i = (V_i^g, E_i^g, A_i^g)$, with V_i^g being a set of vertices, E_i^g a set of edges for connectivity of the nodes, and A_i^g a set of attributes for position, orientation, affine transformations, and deformations of the graphlet.

It is straightforward to extract graphlets from a perfect sketch representation using a depth-first search algorithm. More complex topologies are given a shorter coding length to encourage the algorithm to select bigger graphlets. We then cluster all the graphlets found over all training instances based on their geometry and topological structure. Graphlets are only matched to graphlets with the same topology, and a distance between these pairs is defined using a global affine transformation A_i and a TPS warping for deformation $F_i(x,y)$ on a 2D domain Λ_i covered by g_i.

$$D_{geo}(g_1,g_2) = \omega_1 E_A(g_1,g_2) + \omega_2 E_F(g_1,g_2), \quad (8)$$

where $E_A(g_1,g_2)$ and $E_F(g_1,g_2)$ measure the affine transformation and TPS bending, and ω_1 and ω_2 can be assigned empirically ($\omega_1 = 0.08$, $\omega_2 = 0.92$).

From these clusters we can further select which graphlets are the most informative for each category. In our implementation, the detectability of each graphlet g can be measured based on the

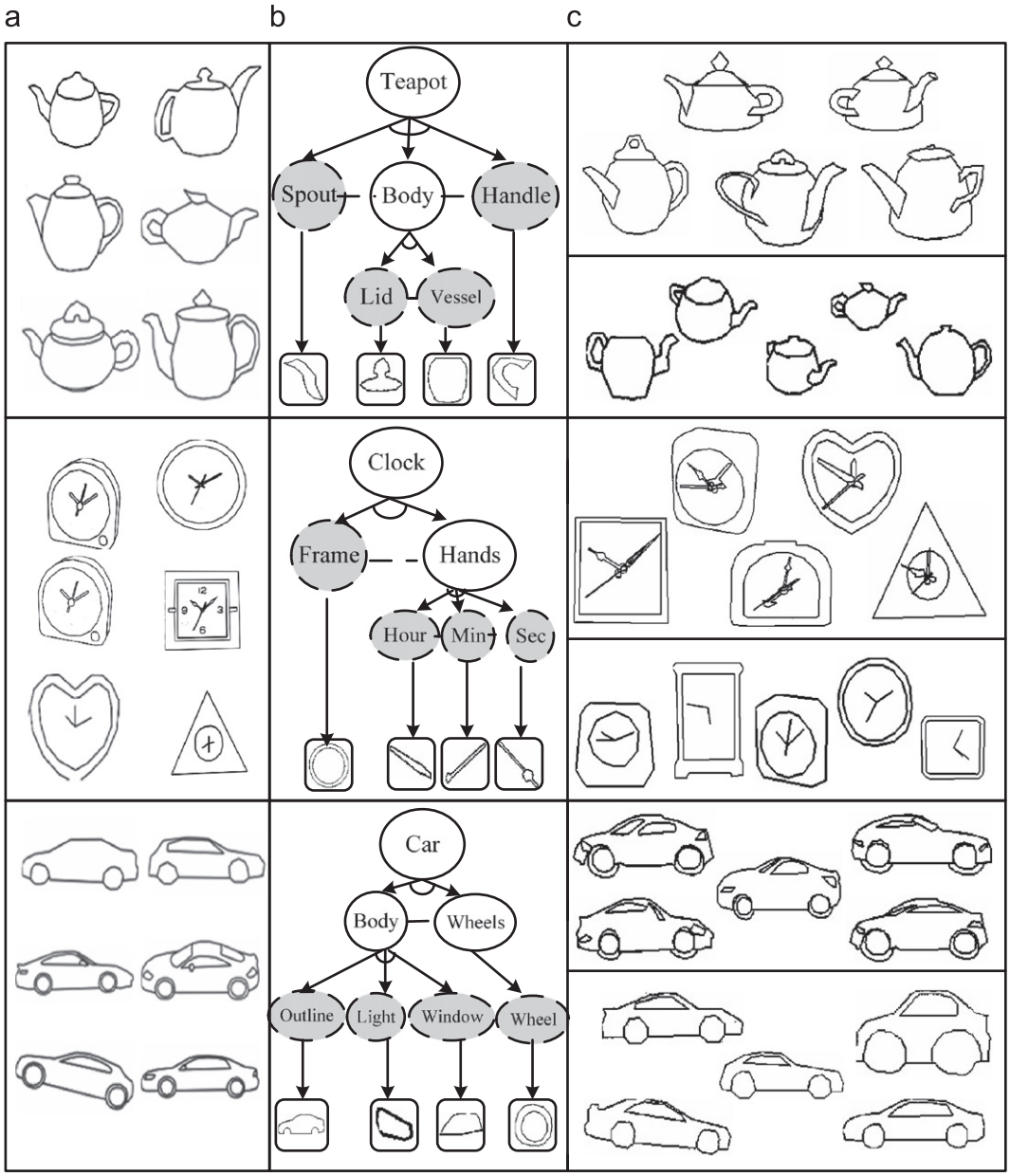

Fig. 4. Examples of And–Or graphs for three categories (b), and their selected training instances (a) and corresponding samples (c). The samples (c) contain new object configurations, compared with the training instances (a). Note that, for the sake of space, the And–Or graphs (b) have been abbreviated. For example, the terminal nodes show only one of the many templates that could have been chosen by the Or Node above. (a) Instances for learning, (b) And-Or graph representation and (c) generated new templates.

mutual information between it and one given category c:

$$MI(g,c) = \sum_{g \in G} \sum_{c \in C} p(g,c) \log \frac{p(g,c)}{p(g)p(c)}, \tag{9}$$

where $p(g)$ denotes the graphlet's frequency in one category, $p(c)$ denotes the category frequency, and $p(g,c)$ denotes the joint probability of occurrence of the graphlet and category. According to the average mutual information in all categories, we select the top 14 detectable graphlets, shown in Fig. 6 along with examples of raw image patches they arise in. The distributions of each of these 14 graphlets over 15 categories are shown in Fig. 7.

3.2. Sketch graph computation

To compare our target image to our training data, we must also convert it into a sketch representation. To compute a sketch graph G from a testing image I, we first compute an edge probability map using a learning-based BEL detector [6], which incorporates approximately 50 000 low-level features. A few representative examples of BEL detection results are shown in Fig. 8(b), where darker pixels denote higher probability of an edge. The primal sketch algorithm [12] is then run on the edge probability map to obtain a sketch graph S (Fig. 8(c)). S is an attributed graph and can reconstruct the original image using a dictionary Δ_{SK} of small image patches for edges and texture in the remaining areas. The sketch graph can be decomposed into graphlets according to the learned graphlet dictionary:

$$S = \bigcup_{i=1}^{N} g_i \cup g_0, \tag{10}$$

where g_0 is the remaining line segments in S.

A compositional boosting [27] algorithm is then utilized for graphlet detection, and the sketch graph G is decomposed into a

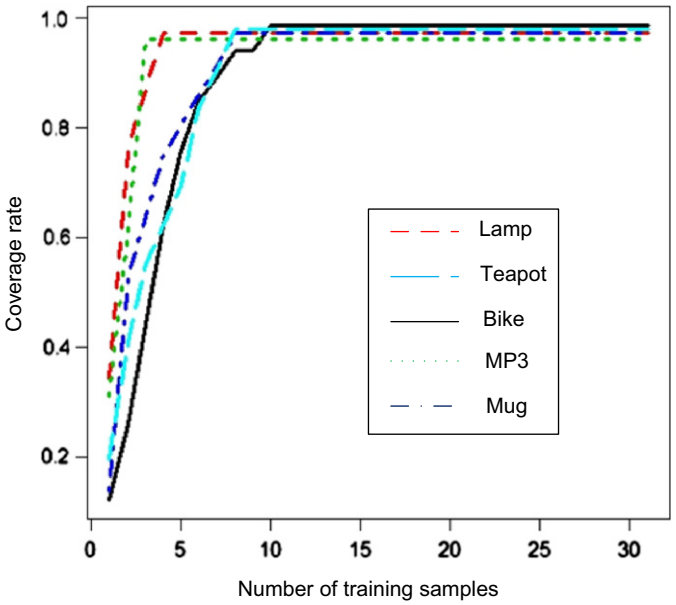

Fig. 5. Plot showing the percentage of the testing set producible by an And–Or graph model learned with an increasing number of training samples. We see that, with very few samples, the And–Or graph can learn a model that covers nearly the entire appearance space.

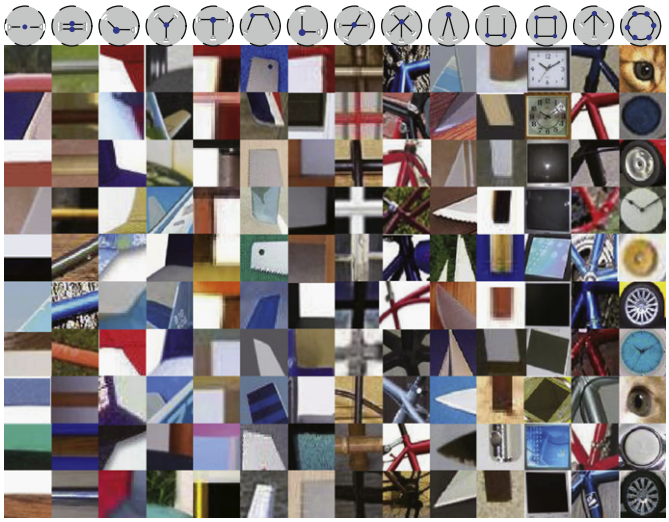

Fig. 6. The top detectable graphlets with typical examples from raw images.

number of graphlets. A few typical results of computed sketch graphs with their underlying graphlets are shown in Fig. 8(d).

3.3. Sequential test pruning

We collected 800 raw images at multiple scales from 33 categories to be used as testing images from the Lotus Hill Institute's image database [29]. These were converted to sketch graphs, as described above, comprising our test set $\Omega_{test}=\{T_1,T_2,\ldots,T_N\}$. We also selected between 30 and 50 annotated objects to learn the And–Or graph from, which then generated 30 new samples in sketch representation for each category. A combination of raw and synthesized data formed the full training set $\Omega_{train}=\{G_1,G_2,\ldots,G_N\}$ of 1980 (60×33) objects over 33 categories.

For each $T_i\in\Omega_{test}$, we search over windows at multiple scales and locations $W_i=\{w_{i1},w_{i2},\ldots,w_{im}\}$ looking for objects from our 33 categories. Our goal is to find the sketch graph $G^*\in\Omega_{train}$ that best matches each window w_{ij}, if any. While our graph matching algorithm [16] is powerful for matching one graph to another, even under large deformations, it is far too inefficient to exhaustively match each training instance to w_{ij}. We thus use our set of sequential tests to map the training set into increasingly complex spaces, keeping only the best matching subset of Ω_{test} at each stage for the next test:

$$\Omega_{test}\supseteq\Omega_1\supseteq\Omega_2\supseteq\Omega_3\supseteq\Omega_4=\Omega_c, \tag{11}$$

where each Ω_i is a new test space and Ω_c is the final candidate set we use for graph matching. The best matching $G^*\in\Omega_c$ is selected as our final match.

The number N of candidates we keep at each stage is empirically determined as the minimum subset that produces 100% true positives. This ensures we never discard possible matches, yet likely reduces our candidate set size. We also plot a confusion matrix of the top N candidate categories at each stage, which describes whether the true category is in the top N candidate categories. From step 3 on, we not only prune categories, but also candidate templates from each category.

For ease in later notation, let us define the sketch graph within our window of interest $w_{ij}\in W_i,T_i\in\Omega_{test}$ as G'. Each training sketch graph will be represented as $G_k\in\Omega_{train}$, and any graph G is comprised of a set of graphlets $\Gamma(G)=\{g_1,g_2,\ldots,g_{n(g)}\}$.

Step 1: *Category histogram*. At this stage, the frequencies of graphlets in G' and each template graph G_k are pooled over each category C_i to be used as our data points H_i:

$$H_i(z)=\frac{\sum_{j\in C_i}\sum_{k=1}^{n(\Gamma(G_j))}1(g_k==z)}{\sum_{j\in C_i}\sum_{k=1}^{n(\Gamma(G_j))}1}.$$

The likelihood between $H(G')$ and each $H(G_k)$ is then calculated, and the top N candidates are selected to keep the true positive rate at 100%. In our experiments, 15 categories remained, the results of which are shown in Fig. 9(a). The training instances from these 15 categories are carried to the next stage as Ω_1.

Step 2: *Nearest Neighbor of Single Graphlet*. We next convert each $G\in\Omega_1$ and G' into a sparse vector V_j of graphlet weights. Each vector represents the frequency of each graphlet along with its weight(detectability) ω_i, computed when we first formed the dictionary. Suppose we have M training templates in each remaining category, then each vector is

$$V_i=\{C(g_1^i),\ldots,C(g_{n(g)}^i)\}=\{\omega_1N_1^i,\ldots,\omega_kN_{n(g)}^i\},$$

where N_k denotes frequency of each graphlet.

Then we use a parameterizing Hamming distance as our nearest neighbor metric for classification:

$$Hamming(V_i,V_j)=\sum_{k=1}^{n(g)}|\omega_k^iN_k^i-\omega_k^jN_k^j|.$$

We first calculate the distance between our testing instance and each training instance from the 15 candidate categories kept from step 1. We then find the 20 shortest distances within each category and calculate the average distance between them. We keep the N closest candidate categories for step 3 as Ω_2, which in our experiments consisted of eight categories with 30 candidate templates in each category. The results of step 2 are shown in Fig. 9(b).

Step 3: *Nearest Neighborhood via Graphlet Pairs*. We next introduce spatial information into our features. To capture spatial information in the sketch graph, adjacent graphlets in each G are composed into pairs of graphlets. From these pairs a dictionary of composite graphlets can be learned as in Section 3.1, and the top 20 detectable composite graphlets are shown in Fig. 10. We then vectorize each G just as in step 2, but using these graphlet pairs instead of single graphlets. The distance metric used is again Hamming distance. In this stage, however, we not only prune the candidate sets, we also prune candidate graphs from each of the

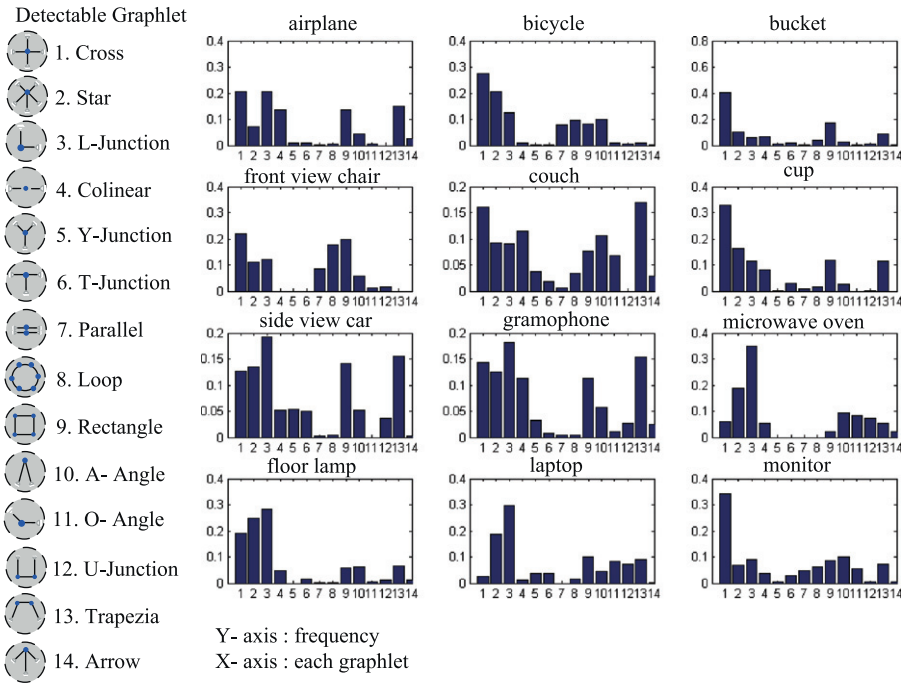

Fig. 7. The graphlet distributions in 12 categories for illustration. The graphlets are listed in order according to detectability.

top N candidate sets. We keep the top five candidate categories and top 15 candidate templates from each of the three categories to comprise Ω_3. Results from this step are shown in Fig. 9(c).

Step 4: *Nearest Neighborhood via Shape Context*. We next incorporate additional spatial information into each G by modeling it using Shape Context [1]. The points comprising each G are used as the input to the algorithm, and matching each possible G_k to G' returns a warping energy. This energy is our distance metric, and we keep the three categories with lowest average energy, and eight candidate templates within those categories with lowest energy, forming our final candidate set Ω_4. Results are shown in Fig. 9(d).

Step 5: *Top-down verification with Graph Matching*. Given the pruned set Ω_c from our sequential tests, we can now perform the last stage of recognition, top-down graph matching. The matching algorithm we adopt is a stochastic layered graph matching algorithm with topology editing that tolerates geometric deformations and some structural differences [16]. Following [16], we can use the underlying graphlets from each G as the initial seed graphs, which greatly improve the matching accuracy and efficiency. Some final candidate categories are shown in Fig. 11 and top-down matching is illustrated. The input testing graph matches to candidate templates and six selected matching instances are shown in Fig. 11. The final confusion matrix on 800 testing images is shown in Fig. 12(a), and the classification rate is 81.7%.

The presented results prove that we can reduce the size of our sample set at every stage, yet keep classification rate high. As the confusion matrices at each step show the diagonal is darkened as we keep more and more categories, as expected. In addition, at each step, the off-diagonal elements get lighter. Thus we are keeping a high true positive rate, while diminishing our false positive rate.

The top-down graph matching results show that we can identify objects even if they are slightly dissimilar from our candidate match. The flexibility of this algorithm lets us identify objects that have undergone slight pose or appearance changes. Note that the power of this match relies on us having a representative set of candidates with different appearances and poses to match with, which were created via the And–Or graph. However, the first four pruning steps are important as well to effectively narrow the searching space. We present an experiment of only using the graph matching method for categorization. As shown in Fig. 12(b), the correct rate is 74.6%, and it is 7.1% lower than the complete framework.

Our system is implemented in the Matlab environment with a common desktop PC, and has large room for improvement. It takes around 3–5 h for training an And–Or graph model and generalizing samples. In the inference, the first four steps of cascaded pruning are very fast while the top-down step needs around 8–15 s for a matching. Therefore, for one testing, it will cost around 5–10 h to match with all the candidates (1980 templates). In contrast, after the sequential pruning, there are much fewer candidates, i.e. less than 30, and the final verification thus takes less than 10 min. Hence, we obtain significant improvement for computation efficiency.

3.4. Comparisons and discussions

In the following, we further demonstrate the advantages of our system by comparing with other methods.

First, we show the importance and benefit of using synthesized samples to augment our training set, as reported in Table 2. We ran the algorithm using both a set of original data plus synthesized samples and with a sample set of just hand-collected data (without generalized samples). In this experiment, we kept the top 1 prediction for calculating classification accuracy. The bottom row, representing the results for the hand-collected data, shows markedly worse performance than the bottom row, which includes synthesized instances. This is evidence that synthesized samples better cover the appearance space and produce better recognition and classification results.

Then we compare our approach with three state-of-the-arts object categorization methods, particularly with small sample sets. The tree-based classifier (i.e. PBT [23]), part-based latent SVM [9], and generative basis model [30] are adopted, and we directly use their original codes and settings for fair evaluation. Since the image features they proposed are different with ours, e.g., Histogram of Gradients [5], or Gabor filters, we take the raw images as inputs for these methods. For each object category, we still put 60 instances (i.e. 30 original and 30 generalized) as

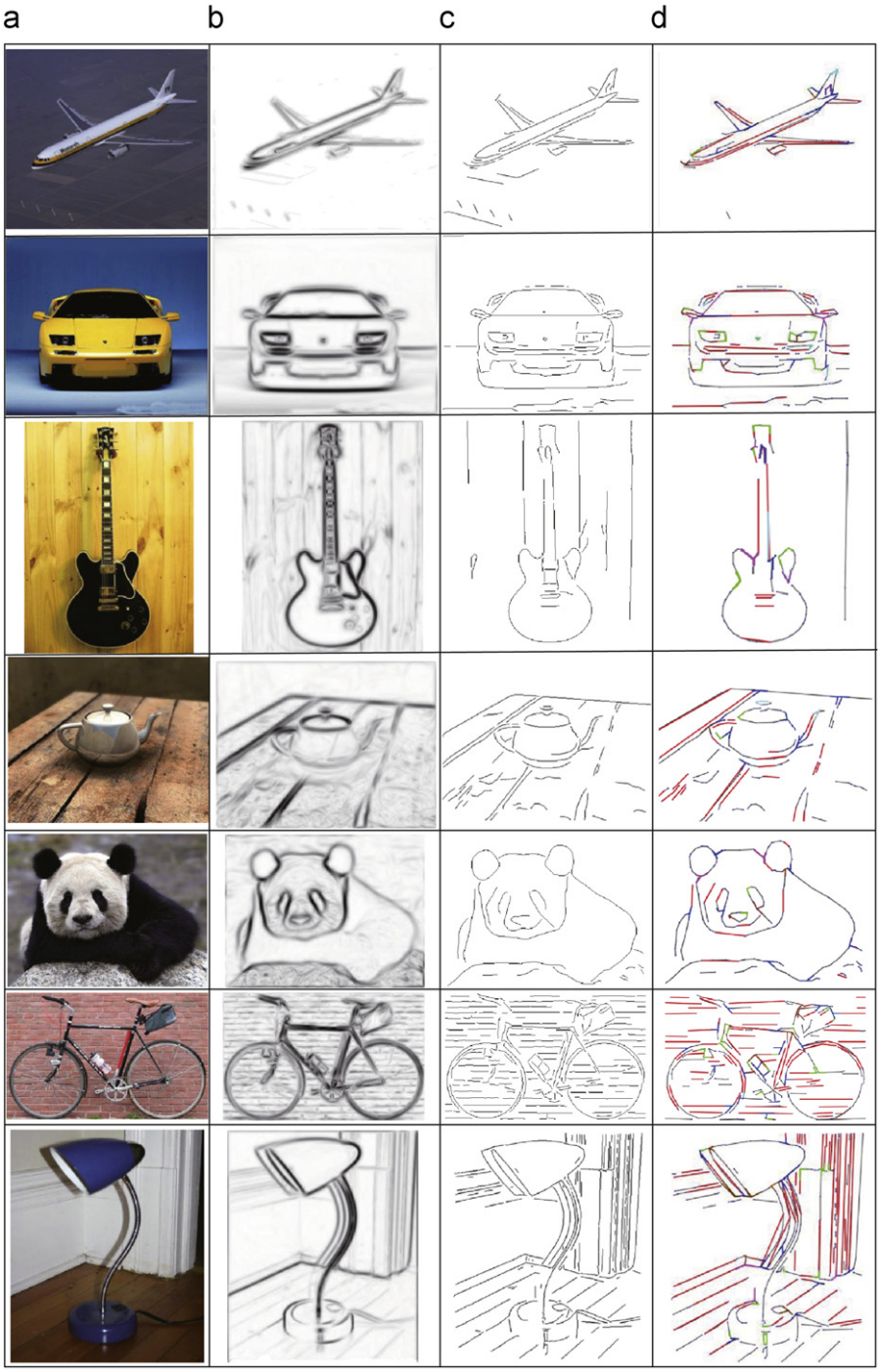

Fig. 8. The sketch graphs computed from raw images. (a) Raw images. (b) Edge probability maps by BEL detector [6]. (c) Primal sketch results by [12]. (d) Refined sketch graphs with graphlet detection (a few selected graphlets are shown in colors). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

training data, and collect additional 50 original images for the three comparing methods. To well evaluate the performances with different data amount, we repeat the experiment of object categorization as the number of training samples increases. Fig. 13 reports the results of this experiment, from which we can observe that our method outperform the others.

The advantages of our framework lie in the following aspects. First, the three comparing approaches basically ignore object semantics and construct classifiers based on image features; thus the number of training data they needs exponentially increases for the multi-class classification task. By contrast, we introduce the And–Or graph learned from annotated data, can be viewed as a template of semantics (e.g., part decompositions and relationship definitions); the semantics are quite useful to guide the learning with small sample sets. Second, the effective generalized instances allow us to simply perform Nearest-Neighbor (NN) based image classification and achieve very good results, rather than carefully selecting reference examples. Last, the hierarchy that we construct for each object class is currently defined by hand. This is not terribly time consuming, as many of the object classes only consist of a small number of different parts. We can therefore define the high-level parts we are interested in

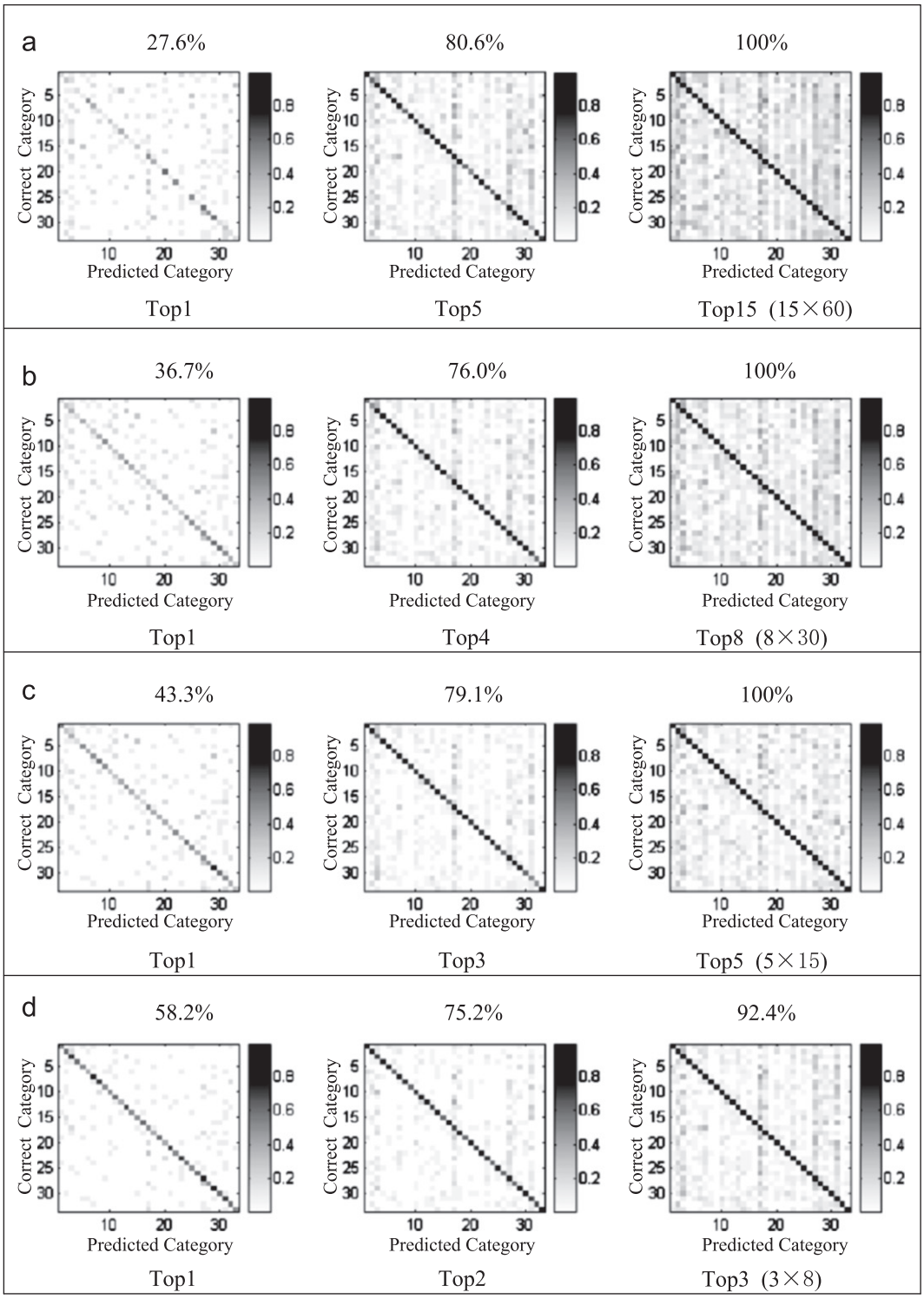

Fig. 9. Four steps of cascaded prune. For each step, three confusion matrices show results with the top N candidate categories, and candidate templates are also pruned for next step. The $\omega_i\{i=1,\ldots,4\}$ denotes the pruned templates space. (a) Step 1, (b) step 2, (c) step 3, (d) step 4.

(e.g., clock hands and clock frames) and arrange them hierarchically. We are considering techniques to help automate this process and learn the hierarchy automatically.

4. Conclusion

In this paper, an empirical study of a system composed of a compositional object model, a sequential testing scheme, and a top-down matching method for object category recognition was presented. We exhibited the And–Or graph model's ability to span the object configuration space and showed a method to compute sketch graphs using graphlets detection. In the inference process, four stages of sequential pruning were adopted to narrow down possible matches for a target image, and a graph matching algorithm was used on the final candidates for verification. We showed how our classification rate varied as the number of synthesized samples increased, and showed its improvement

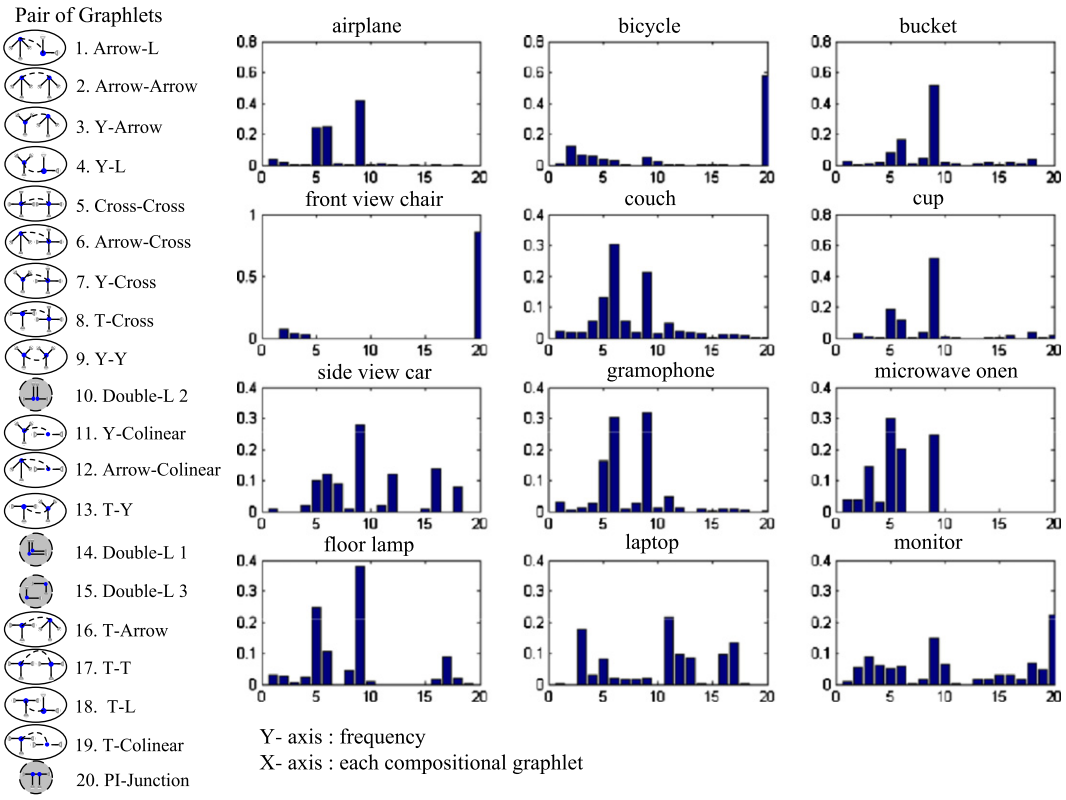

Fig. 10. Top 20 detectable composite graphlets and their distributions of selected 12 categories.

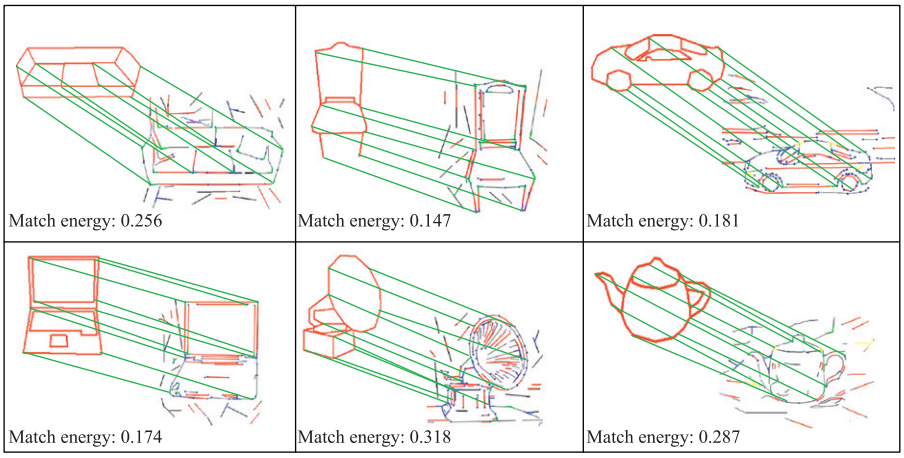

Fig. 11. The top-down graph matching verification. Each testing graph matches with the candidate templates, and obtain the best candidate with lowest matching energy.

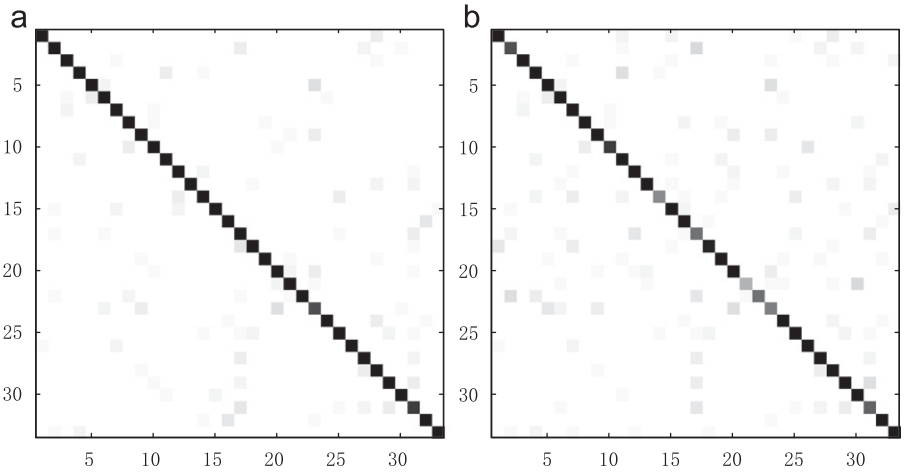

Fig. 12. The final confusion matrix and comparison to classifying without sequential pruning. As shown in (a), we achieve an 81.7% classification rate on classifying 800 testing images in 33 categories. In (b), we show the result of performing top-down matching directly with candidates, i.e. neglecting the first four steps, and achieve 74.6%.

Table 2
Results of classification rate in each step with generalized samples and without generalized samples. In each step, the classification rate is calculated by cascaded pruning.

Classification rates	Step1	Step2	Step3	Step4	Final
With generalized samples (%)	27.6	36.7	43.3	58.2	81.7
Without generalized samples (%)	21.5	27.4	39.5	42.3	66.2

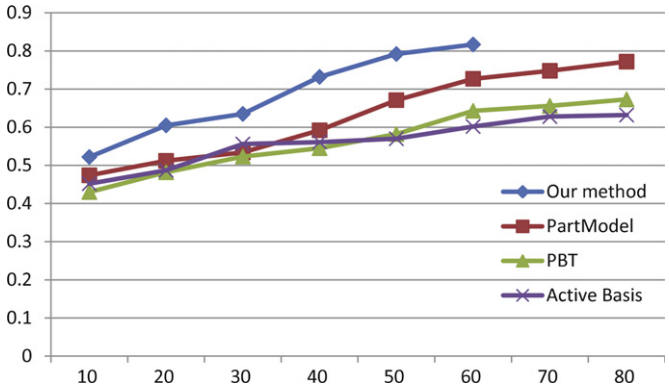

Fig. 13. Comparisons with other learning-based object categorization methods. The experiment is performed as the training data increases. The vertical and horizontal axes, respectively, represent the categorization rate and the number of training examples. We use synthesized templates as training data for our framework, and use collecting images as training data for other comparing methods. The blue curve represents the performances of our approach, and the other curves denote the results by PBT [23], part-based latent SVM [9], and generative basis model [30]. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

over the state-of-the-arts object categorization methods. The experimental results and comparisons show that the use of synthesized instances allows us to better represent objects with large variability in appearance, and that our bottom-up discriminative prune combined with top-down matching is an efficient and accurate way to perform classification and recognition.

In the future we plan to study more general approach to unsupervisedly discover object category models with the And–Or graph representation, and adopt more informative appearance features in the framework.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant nos. 90920009 and 60970156), the Hi-Tech Research and Development Program of China (National 863 Program, Grant no. 2012AA011504), and the Guangdong Natural Science Foundation (Grant no. S2011010001378). This work was also supported in part by the Guangdong Science and Technology Program (Grant no. 2011B040300029), and SYSU-Sugon high performance computing typical application projects (Grant no. 62000-1132001).

References

[1] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on PAMI 24 (4) (2002) 509–522.
[2] H. Chen, Z. Xu, Z. Liu, S.C. Zhu, Composite templates for cloth modeling and sketching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006.
[3] Z. Chi, S. Geman, Estimation of probabilistic context-free grammars, Computational Linguistics 24 (2) (1998).
[4] Liang Lin, Ping Luo, Xiaowu Chen, Kun Zeng, Representing and recognizing objects with massive local image patches, Pattern Recognition 45 (1) (2012) 231–240.
[5] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2005.
[6] P. Dollar, Z.W. Tu, S. Belongie, Supervised learning of edges and object boundaries, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006.
[7] L. Fei-Fei, P. Perona, A Bayesian hierarchical model for learning natural scene categories, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005.
[8] P. Felzenszwalb, D. Huttenlocher, Pictorial structures for object recognition, Intel' Journal of Computer Vision 61 (1) (2005) 55–79.
[9] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.
[10] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2003.
[11] G. Guo, C.R. Dyer, Learning from examples in the small sample case: face expression recognition, IEEE Transactions on SMC, Part B 35 (3) (2005) 477–489.
[12] C.E. Guo, S.C. Zhu, Y.N. Wu, A mathematical theory of primal sketch and sketchability, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2003.
[13] F. Han, S.C. Zhu, Bottom-up/top-down image parsing by attribute graph grammar, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2005.
[14] F. Jurie, B. Triggs, Creating efficient codebooks for visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2005.
[16] L. Lin, X. Liu, S.C. Zhu, Layered graph matching with composite cluster sampling, IEEE Transactions on PAMI 32 (8) (2010) 1426–1442.
[17] L. Lin, S. Peng, J. Porway, S.C. Zhu, Y. Wang, An empirical study of object category recognition: sequential testing with generalized samples, in: Proceedings of the of International Conference on Computer Vision (ICCV), 2007.
[18] L. Lin, T. Wu, J. Porway, Z. Xu, A stochastic graph grammar for compositional object representation and recognition, Pattern Recognition 42 (7) (2009) 1297–1307.
[19] R. Maree, P. Geurts, J. Piater, L. Wehenkel, Random subwindows for robust image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005.
[20] J. Porway, B. Yao, S.C. Zhu, Learning compositional models for object categories from small sample sets, in: S. Dickinson, et al., (Eds.), Object Categorization: Computer and Human Vision Perspectives, Cambridge University Press, 2009.
[21] G. Schneider, H. Wersing, B. Sendhoff, E. Korner, Evolutionary optimization of a hierarchical object recognition model, IEEE Transactions on SMC Part B 35 (3) (2005) 426–437.
[22] N.H. Trinh, B.B. Kimia, Category-specific object recognition and segmentation using a skeletal shape model, in: Proceedings of the British Machine Vision Conference (BMVC), 2009.
[23] Z. Tu, Probabilistic boosting tree: Learning discriminative models for classification, recognition, and clustering, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2005.
[24] Liang Lin, Xiaolong Wang, Wei Yang, Jianhuang Lai, Learning contour-fragment-based shape model with and-or tree representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
[25] I. Ulusoy, C. Bishop, Generative versus discriminative methods for object recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005.
[26] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2001.
[27] T. Wu, G. Xia, S.C. Zhu, Compositional boosting for computing hierarchical image structures, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007.
[28] J. Yang, D. Zhang, J.Y. Yang, Constructing PCA baseline algorithms to reevaluate ICA-based face-recognition performance, IEEE Transactions on SMC, Part B 37 (4) (2007) 1015–1021.
[29] B. Yao, X. Yang, L. Lin, M.W. Lee, S.C. Zhu, I2T: Image parsing to text description, Proceedings of IEEE 98 (8) (2010) 1485–1508.
[30] Y.N. Wu, Z. Si, H. Gong, S.C. Zhu, Learning active basis model for object detection and recognition, International Journal of Computer Vision 90 (2) (2010) 198–235.

Liang Lin received the B.S. and Ph.D. degrees from the Beijing Institute of Technology (BIT), Beijing, China, in 1999 and 2008, respectively. From 2006 to 2007, he was a joint Ph.D. student with the Department of Statistics, University of California, Los Angeles (UCLA). He was a Post-Doctoral Research Fellow with the Center for Image and Vision Science, UCLA. From 2007 to 2009, he was a Senior Research Scientist with the Lotus Hill Research Institute, Hubei, China. He is currently an Associate Professor in

Sun Yat-Sen University, Guangzhou, China. His current research interests include but are not limited to computer vision, machine learning, and multimedia technology. He has authored or co-authored over 40 academic papers over a wide range of research topics. He has received a number of honors, including several scholarships during pursuing the Ph.D. degree, the Beijing Excellent Students Award in 2007, China National Excellent Ph.D. Thesis Award Honorable Mention in 2010, and the Best Paper Runners-Up Award in ACM NPAR 2010.

Xiaobai Liu has been pursuing the Ph.D. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, since September 2006. Since December 2008, he has been a Research Associate with Prof. S. Yan, Learning and Vision Group, National University of Singapore, Singapore. From September 2007 to December 2008, he was a Research Associate with the Lotus Hill Research Institute, Ezhou, China, under the supervision of Prof. S.C. Zhu. He has published more than ten articles over a series of research topics. His current research interests include computer vision, machine learning, and large-scale image retrieval.

Shaowu Peng received the B.S. degree and M.S. degree in Applied Mathematics from South China University of Science and Technology, Guangzhou, China 2001 and 2004 respectively. He received the Ph.D. degree from Huazhong University of Science and Technology (HUST) Wuhan, China. He is currently an assistant professor in South China University of Science and Technology, Guangzhou, China. His research interests are computer vision, image understanding, object categorization and shape metric.

Hongyang Chao received her Ph.D. degree in Computational Mathematics from Sun Yat-Sen University in 1987. She is a full professor of School of Software at Sun Yat-Sen University (SYSU) in Guangzhou of China. She was awarded by the "Hundred Talents Program" of the University in 2004. Her current research interests include the areas of image and video compression, image and video processing, massive multimedia data analysis and understanding, content based image (video) retrieval, etc. Her research works have been fully funded by US Navy, NSFC, 863 High Tech Program of China, and so on. Between June 1994 and May 1995, she was a visiting researcher at the Department of Computer Science at Stanford University. From 1996 to 2003, she worked for a software company named Infinop (later acquired by Vianet/ESPRE) in the US, where she was also a founding researcher. She has published extensively in the area of image/video processing (including TCSVT, ECCV, SigGIS, etc.) and holds three US patents and 1 China patent. Currently she is also Deputy Dean in School of Software at SYSU.

Yongtian Wang received his B.Sc. degree in precision instrumentation from Tianjin University, China, in 1982, and his Ph.D. degree in optics from the University of Reading, England, in 1986. He is currently a professor of optics and applied computer science in the Beijing Institute of Technology, and a Yangtze River Scholar appointed by the Chinese Ministry of Education. His research interests include optical design and CAD, optical instrumentation, image processing, 3D display, virtual reality (VR) and augmented reality (AR) technologies and applications. Dr. Wang is a Fellow of SPIE, a Fellow of IET and a director of the Chinese Optics Society.