An Empirical Study of Object Category Recognition: Sequential Testing with Generalized Samples

Liang Lin[1,2], Shaowu Peng[4], Jake Porway[3], Song-Chun Zhu[2,3] and Yongtian Wang[1]

[1]School of Info. Sci. and Tech., Beijing Institute of Technology, Beijing, China, {linliang,wyt}@bit.edu.cn

[2]Lotus Hill Research Institute, Ezhou, China

[3]Dept. of Statistic and Computer Science, University of California, Los Angeles, {sczhu,jporway}@stat.ucla.edu

[4]IPRAI, Huazhong University of Sci. and Tech.,Wuhan, China, {swpeng}@hust.edu.cn

Abstract

In this paper we present an empirical study of object category recognition using generalized samples and a set of sequential tests. We study 33 categories, each consisting of a small data set of 30 instances. To increase the amount of training data we have, we use a compositional object model to learn a representation for each category from which we select 30 additional templates with varied appearance from the training set. 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, 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 achieve an 81.4 % classification rate on classifying 800 testing images in 33 categories, 15.2% more accurate than a method without generalized samples.

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.

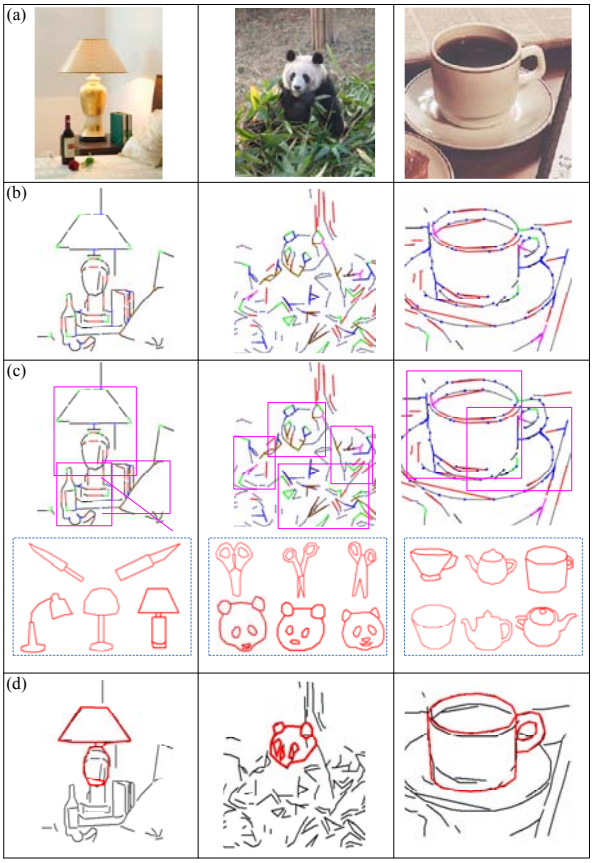

Figure 1. Illustration of object category recognition. Three typical testing images from 33 categories are shown in (a). The related sketch graphs are computed with graphlet detected in (b). A cascade of discriminative steps prune candidate categories and objects, as well as to localize the objects.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

This issue is generally resolved by solving one problem or the other - either the task becomes classification [22, 16,

17, 6, 14], 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, 13, 15] that can often only recognize classes of object that are similar in appearance and is computationally intensive to perform inference with.

To solve these 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[1, 9, 5, 18] from a few training instances. This model was first presented by Han[5], Chen[9], and Zhu and Mumford[18] extensively discussed it as a large scale knowledge representation. [1] defined a probability model on the And-Or graph 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.

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[10] to search the image for the small number of candidates remaining. Similar approaches have performed top-down matching using just a generative model, such as K-Fans or the constellation model [13, 4, 15, 19]. However, not only might these approaches alone not be able to learn large structural variations like the And-Or graph can, they are computationally expensive to perform inference with [7]. Our use of generalized samples helps us represent the object space better and our sequential tests help the efficiency of our top-down matching.

Our data consist of *sketch representations* of objects, in which perceptually important parts have been outlined in a graphical format, with each image comprised of nodes and edges. The left and right columns of Figure 3 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.5.

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 [8]. 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** [20] 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 [10]. 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.

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 PBT (Probabilistic Boosting Tree) framework[22], which is used for learning two-class and multi-class discriminative framework.

All the data used in our experiments are selected from the Lotus Hill dataset [2], 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 first briefly present the principle of synthesizing instances from the And-Or graph model 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

The And-Or graph was proposed by Chen [9] recently, and a learning and sampling algorithm is presented in [1]. It is a compositional model capable of creating novel instances from a small set of training data. By combining a stochastic context free grammar with the constraints of a Markov random field, it can represent the variability seen in many object categories yet still constrain the appearances of these objects so that they are perceptually equivalent. This allows it to generate combinatorially many instances from a small sample set.

Figure 3 shows 3 And-Or graphs, simplified for the sake of space. On the left of each we see instances from 3 categories, teapot, clock, and car, and on the right we see high

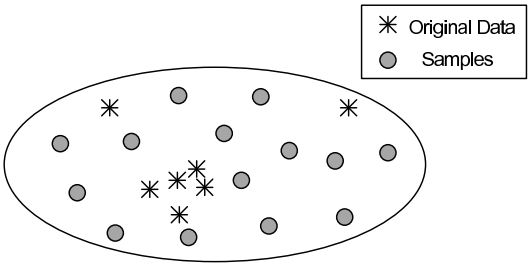

Figure 2. Instances mapped into the appearance space. The original samples are too sparse to cover the space, but samples from the And-Or graph increase coverage.

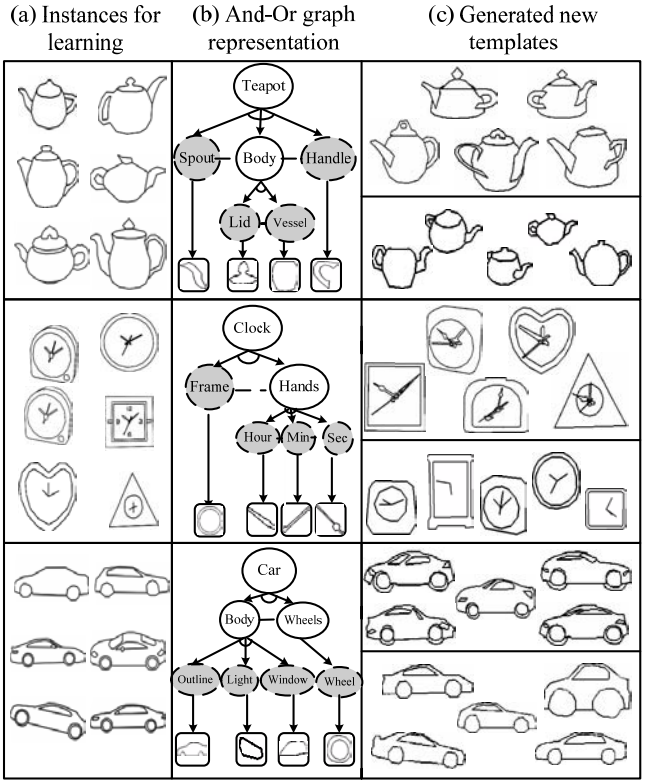

Figure 3. Examples of And-Or graphs for three categories (in (b)), and their selected training instances (in (a)) and corresponding samples (in (c)). And The samples (in (c)) contain new object configurations, compared with the training instances (in (a)). Note that, for the sake of space, the And-Or graphs (in (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.

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

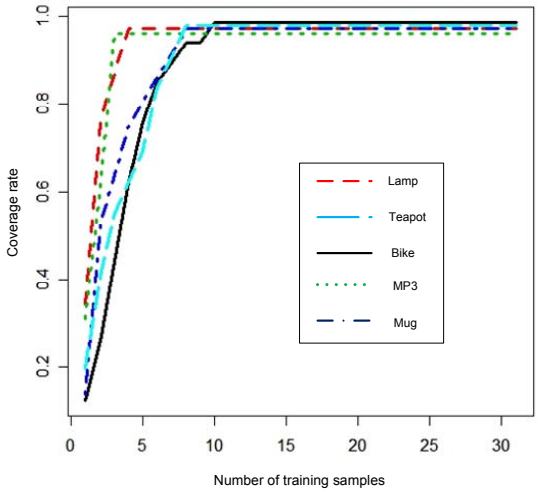

Figure 4. Plot showing the percentage of the testing set producible by an And-Or graph model learned with an increasing number of training samples. We see that, with very few samples, the And-Or graph can learn a model that covers nearly the entire appearance space.

appearance space of an object category, thus more accurately representing it in our discriminative tests and providing more candidates for our top-down match. Figure 2 illustrates this concept. The asterisks represent our initial sample set, which doesn't 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. Figure 3 (c) shows examples of these samples at both high and low resolution, along with the corresponding And-Or graph for that category (Figure 3 (b)). Note that, a few new configurations are synthesized, as compared to the training instances (Figure 3 (a)).

To quantitatively illustrate the And-Or graph's ability to synthesize new object configurations, we collect a number of objects (about 50 for each category) which we assume span the configuration space. We then learn And-Or graph models for these categories using an increasing number of training samples. We then sample instances from the learned model at each stage and measure how many instances in our testing set are producible from this model. Fig.4, shows the results for 5 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

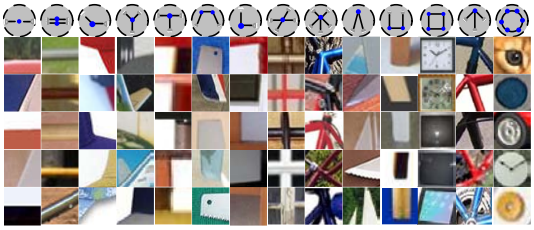

Figure 5. The top detectable graphlets with typical examples from raw images

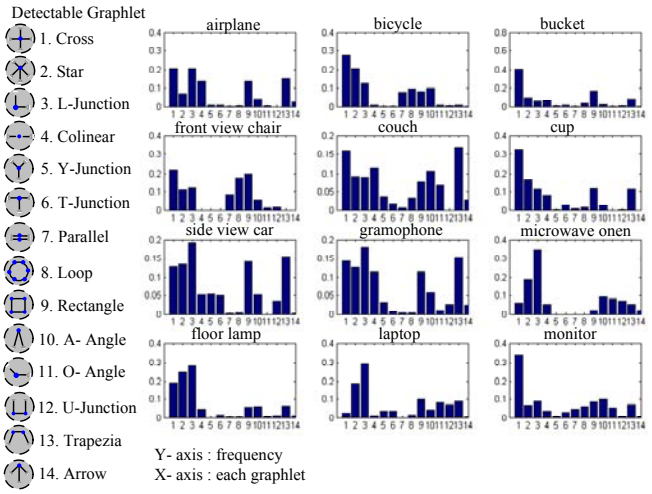

Figure 6. The graphlet distributions in 12 categories for illustration. The graphlets are listed in order according to detectability.

"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 $g_i = (V_i, E_i, A_i)$, with V_i being a set of vertices (nodes), E_i a set of edges for connectivity of the nodes, and A_i 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.

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

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 mutual information between it and one given category c.

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

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.5 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.6.

3.2. Computing the Sketch Graph

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[12], which incorporates approximately 50000 low-level features. A few representative examples of BEL detection results are shown in Fig.7 (b), where darker pixels denote higher probability of an edge. The primal sketch algorithm[3] is then run on the edge probability map to obtain a sketch graph S (Fig. Fig.7 (c)). S is an attributed graph and can reconstruct the original image using a dictionary $\triangle_{SK}$ 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.

$$S = \cup_{i=1}^{N} g_i \cup g_0$$

where g_0 is the remaining line segments in S.

A compositional boosting[21] algorithm is then utilized for graphlet detection, and the sketch graph G is decomposed into a number of graphlets. A few typical results of computed sketch graphs with their underlying graphlets are shown in Fig.7 (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 [2]. These were converted to sketch graphs, as described above, comprising

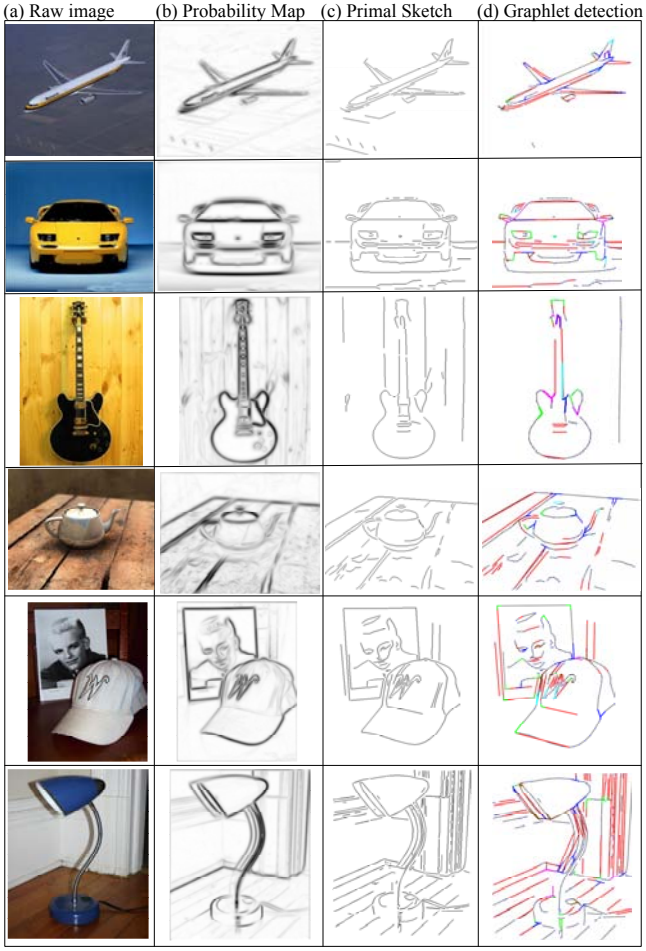

Figure 7. The sketch graphs computed from raw images. (a) Raw images. (b) Edge probability maps by BEL detector[12]. (c) Primal sketch results by [3]. (d) Refined sketch graphs with graphlet detection (A few selected graphlets are shown in colors).

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 $W_i = \{w_{i1}, w_{i2}, \ldots, w_{im}\}$ to recognize 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 [10] 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.

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

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. 8 (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

$$V_i = \{C(g_1^i), \ldots, C(g_{n(g)}^i)\} = \{\omega_1 N_1^i, \ldots, \omega_k N_{n(g)}^i\}$$

where N_k denotes frequency of each graphlet.

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

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

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 dis-

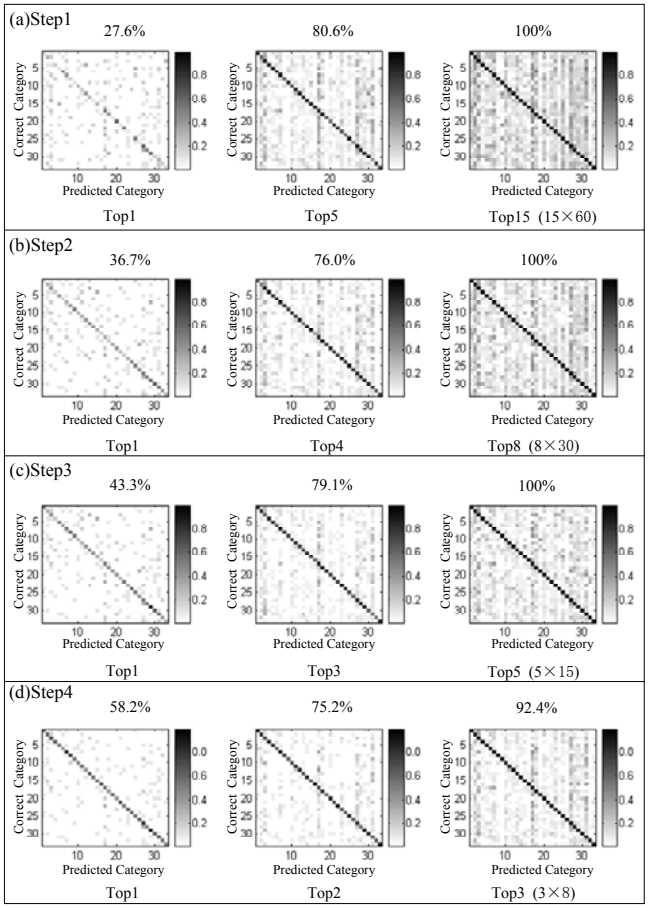

Figure 8. Four steps of cascaded prune. For each step, three confusion matrix show results with the top N candidate categories, and candidate templates are also pruned for next step. The $\omega_i\{i = 1...4\}$ denotes the pruned templates space.

tances 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 8 categories with 30 candidate templates in each category. The results of step 2 are shown in Fig. 8 (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. 9. 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 top N candidate sets. We keep the top 5 candidate categories and top 15 candidate templates from each of the 3 categories to comprise Ω_3. Results from this step are shown

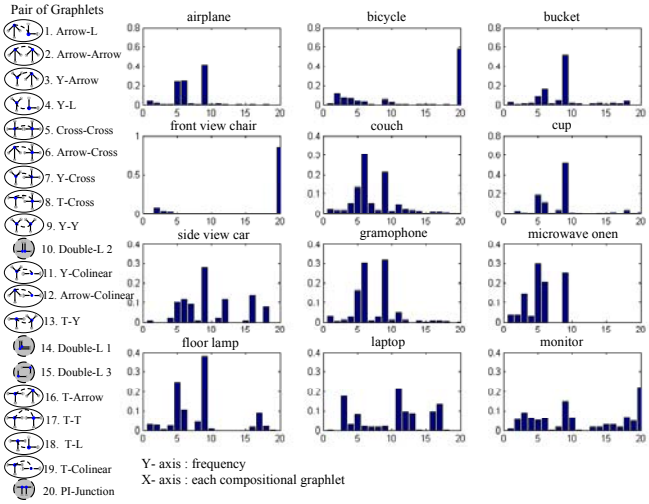

Figure 9. Top 20 detectable composite graphlets and their distributions of selected 16 categories.

in 8 (c).

Step 4 Nearest Neighborhood via Shape Context We next incorporate additional spatial information into each G by modeling it using Shape Context[20]. 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 3 categories with lowest average energy, and 8 candidate templates within those categories with lowest energy, forming our final candidate set Ω_4. Results are shown in Fig. 8 (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 [10]. Following [10], 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. 10 and top-down matching is illustrated. The input testing graph matches to candidate templates and 6 selected matching instances are shown in Fig. 10.

The final confusion matrix on 800 testing images is shown in Fig. 11,

3.4. Analysis and Comparison

The results in the previous section prove that we can both 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

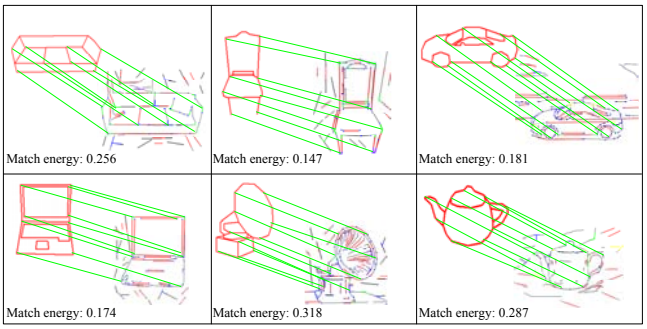

Figure 10. The top-down graph matching verification. Each testing graph matches with the candidate templates, and obtain the best candidate with lowest matching energy.

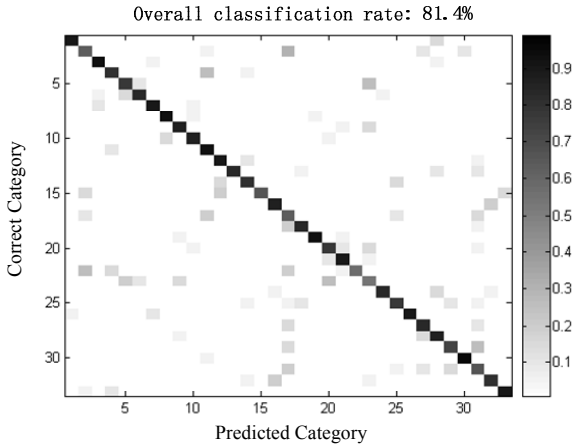

Figure 11. The final confusion matrix on 800 testing images.

	step1	step2	step3	step4	final
with generalized samples	27.6%	36.7%	43.8%	59.2%	81.4%
without generalized samples	21.5%	27.4%	39.5%	42.3%	66.2%

Table 1. 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.

rate.

The top-down graph matching results show that we can identify objects even if they are slightly dissimilar from our candidate matches. 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. The combination of these two algorithms, along with the set of sequential tests, is what allows us to recognize objects with highly varied appearances.

Table 1 shows the importance and benefit of using synthesized samples to augment our training set. We ran the algorithm using both a set of original data plus synthesized samples as well as with a sample set of just hand-collected data (without generalized samples). In this experiment, we did not prune in the final stage to ensure 100% true positive rate. Thus, to get the classification rate for step 3, we pruned in steps 1 and 2, then measured the classification accuracy of labeling using step 3. The top 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.

To show the usefulness of this algorithm with small sample sets, we compare our performance to another classification framework, PBT[22]. As the number of training samples increases, the PBT performs well in learning a discriminative model while keeping the hierarchical tree structure, in which each node combines a number of weak classifiers (evidence, knowledge) into a strong classifier (a conditional posterior probability). However, like typical discriminative approaches, the number of training data it needs exponentially increases for the multi-class classification task. As the red curve shows in Fig.12, PBT shows poor classification results for small sets of training images (from 10 to 60 for each category). As illustrated in Fig.12, our framework on synthesized templates appears to have better classification power than the PBT framework.

The results above show that And-Or graph models, learned from annotated parsing graphs, have greater ability to capture object category variability, and the synthesized objects from those And-Or graph models thus are more representative than just the original data alone. Additionally, combining the And-Or graph model with the sequential prune is indeed an effective approach for multi-class classification task on small sets of original training data, outperforming discriminative methods that often rely on large amounts of data to work effectively.

4. Summary

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 classification and 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 improve-

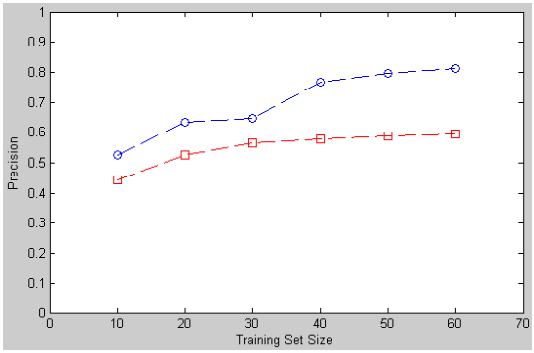

Figure 12. Classification rate with the training data increasing. We use synthesized templates as training data for our framework, and use collecting images as training data for PBT framework. The red curve denotes the performance of PBT, while green denotes our approach.

ment over PBT[22]. 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.

Acknowledgement

This work is done at the Lotus Hill Institute and is supported by China 863 program (Grant No.2006AA01Z339 and No.2006AA01Z121), NSFC (Grant No.60673198), NSF (Grant No.0413214 and No.0713652), and ONR (Grant No.00014-05-0543). The data used in this paper were provided by the Lotus Hill Annotation project[2], which was partially supported by a subaward from the W.M. Keck Foundation. The author would like to thank Zhuowen Tu for providing PBT and BEL codes, and thank Zhuowen Tu and Zijian Xu for extensive discussion.

References

[1] J. Porway, B. Yao, and S.C. Zhu, Learning An And-Or graph for Modeling and Recognizing Object Categories, Technical Report, Department of Statistics, UCLA, 2007. 2

[2] B. Yao, X. Yang, and S.C. Zhu. Introduction to A Large Scale General Purpose Groundtruth Dataset: Methodology, Annotation tool, and Benchmarks, EMMCVPR, 2007. 2, 4, 8

[3] C. E. Guo, S. C. Zhu, and Y. N. Wu. A Mathematical Theory of Primal Sketch and Sketchability, ICCV, 2003. 4, 5

[4] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial Priors for Part-Based Recognition Using Statistical Models, CVPR, 2005. 2

[5] F. Han and S.C. Zhu. Bottom-up/Top-down Image Parsing by Attribute Graph Grammar, ICCV, 2005. 2

[6] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition, ICCV, 2005. 2

[7] I. Ulusoy, C. Bishop. Generative Versus Discriminative Methods for Object Recognition, CVPR, 2005. 2

[8] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories, CVPR, 2005 2

[9] H. Chen, Z. Xu, Z. Liu, and S.C. Zhu. Composite Templates for Cloth Modeling and Sketching, CVPR, 2006. 2

[10] L. Lin, S.C. Zhu, and Y. Wang. Layered Graph Match with Graph Editing, CVPR, 2007. 2, 5, 6

[11] M. P. Kumar, P. H. S. Torr, and A. Zisserman, Extending Pictorial Structures for Object Recognition, BMVC, 2004.

[12] P. Dollar, Z. Tu, and S. Belongie. Supervised Learning of Edges and Object Boundaries, ICCV, 2005. 4, 5

[13] P. Felzenszwalb and D. Hut tenlocher. Pictorial Structures for Object Recognition, IJCV 61(1):55-79, 2005. 2

[14] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features, CVPR, 2001. 2

[15] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-invariant Learning, CVPR, 2003. 2

[16] R. Maree, P. Geurts, J. Piater, L. Wehenkel. Random Subwindows for Robust Image Classification, CVPR, 2005. 2

[17] S. Ali and M. Shah. An Integrated Approach for Generic Object Detection Using Kernel PCA and Boosting, ICME, 2005. 2

[18] S.C. Zhu and D. Mumford. A Stochastic Grammar of Images, Foundations and Trends in Computer Graphics and Vision, 2007(To appear). 2

[19] S. Ullman, E. Sali, and M. Vidal-Naquet. A Fragment-Based Approach to Object Representation and Classification, Proc. 4th Intl Workshop on Visual Form, Capri, Italy, 2001. 2

[20] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. PAMI, 24(4):509-522, 2002. 2, 6

[21] T.F. Wu, G.S. Xia, and S.C. Zhu. Compositional Boosting for Computing Hierarchical Image Structures, CVPR, 2007. 4

[22] Z. Tu: Probabilistic Boosting Tree. Learning Discriminative Models for Classification, Recognition, and Clustering, ICCV, 2005. 2, 7, 8