Unsupervised Modeling of Object Categories Using Link Analysis Techniques

Gunhee Kim   Christos Faloutsos   Martial Hebert
School of Computer Science, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA
{gunhee,christos,hebert}@cs.cmu.edu

Abstract

We propose an approach for learning visual models of object categories in an unsupervised manner in which we first build a large-scale complex network which captures the interactions of all unit visual features across the entire training set and we infer information, such as which features are in which categories, directly from the graph by using link analysis techniques. The link analysis techniques are based on well-established graph mining techniques used in diverse applications such as WWW, bioinformatics, and social networks. The techniques operate directly on the patterns of connections between features in the graph rather than on statistical properties, e.g., from clustering in feature space. We argue that the resulting techniques are simpler, and we show that they perform similarly or better compared to state of the art techniques on common data sets. We also show results on more challenging data sets than those that have been used in prior work on unsupervised modeling.

1. Introduction

1.1. Overview

Unsupervised visual modeling of object categories involves the extraction of parts (regions or sets of features) from unlabeled images corresponding to different, unknown object categories. This problem, typically addressed with statistical machine leaning tools, remains one of the most challenging visual perception problems. Here, we propose a novel representation and formulation of the problem inspired by the tools commonly used for the analysis of complex networks such as the WWW and social networks.

Specifically, we construct a single large graph which captures the interactions of all visual features in the training set, called the *visual similarity network*. The basic idea of the approach is that (1) if a feature i has significant number of consistent matching from others, then the feature i should be important and (2) if feature i from one image matches features in the other images that are mostly the same as those matched by another feature j, then i and j are more likely to belong to the same object. This type of reasoning relies only on the configuration of the graph of features (*i.e.* which feature matches which features in the entire training set) and powerful link analysis techniques can be brought to bear to mine global matching consistency from the graph for the inference of visual models.

Our approach is analogous to the Web search engines, which are successful examples of extracting information by analyzing the interactions of data elements with each other. In this case, despite the large amount of data, it is possible to retrieve accurate information within a very short time by analyzing the interactions between the web pages (specifically, which page *links* which other pages) without explicitly describing their content. In particular, we will extract visual models from unlabeled data by using primarily the PageRank [5] and the vertex similarity algorithms [4] as link analysis techniques.

Because we need to represent in principle all pairwise relations between all visual features, the representation could rapidly grow to become intractable. In practice, however, the resulting representation is compact and efficient because the graphs involved are very sparse and, like other complex networks such as those found in social networks, WWW, or epidemiology, the graph used in our approach roughly follows the power law (*i.e.* it is a *scale-free* network) [1]. In other words, in practice, there is some small number of features acting as *hubs* compared to the total number of features, and most features are less important. Intuitively, when we gather images of object classes and generate a graph of all the features, there is a small number of class representative visual information, and most of the features are likely to be irrelevant (such as background) to the discovery of category models. Therefore, by applying link analysis techniques, we can quickly filter out large amount of non-essential visual information. Another reason for the efficiency of the approach is that, once the network is constructed, we do not need to access the feature data. All inference for unsupervised modeling is based on link analysis of the graph, and thus the computation is quite fast.

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

1.2. Related Work

Our approach is unique in describing all of the visual interactions explicitly in a single view, by formulating low-level visual information as a complex network and by applying link analysis techniques for visual modeling. Here, we introduce some previous work which is closely related to ours.

A common way to approach the unsupervised modeling problem is to first quantize the input into a discrete set of words, which are then used for extracting groups of features corresponding to categories [8, 9, 10, 21]. However, unlike the text analysis domain from which it originated, this identification process is not straightforward. Since, unlike words in text, there are no natural boundaries, orders, and clear semantic meaning of visual words, the definition of visual words and their assignments to instances of local features are in themselves challenging. For this reason, there is no dominant methods for dictionary formation (*e.g*. hierarchical agglomerative clustering [19] or k-means [6, 21]), the optimal selection of dictionary sizes, and the assignment of codewords to each feature instance (*e.g*. soft or hard assignment). In contrast, we do not try to identify each visual entity (*i.e*. the definition and assignment to *codewords*) but focus on its *interactions with others*. In this approach, the data describing each feature, such as its position, orientation, scale, and descriptor, is used only for defining its relationship with other features.

Grauman and Darrell [11] applied the pyramid match kernels to unsupervised modeling. Their work is similar to ours in that they use image-based matching and spectral clustering for final classification results. However, their algorithm relies on the results of image matching but it does not take advantage of explicit interactions between individual low-level features. We compare experimentally with their approach.

Todorovic and Ahuja [22] proposed an unsupervised modeling method based on tree matching. Segmented regions are represented as nodes in a tree, inference of models is performed by tree matching. Although the tree structure can support complex hierarchical relationships, the complexity of the approach is on the order of the fourth power of the number of nodes.

Data mining and link analysis techniques have been used in computer vision tasks. Quack *et al*.'s work [19] applies well-known data mining techniques, termed *frequent itemsets* and *association rules*, to the feature selection. However, their work differs from ours in that it requires bounding box annotations, do not use any networks, and only applied two-class cases (positive and negative image sets). Pan *et al*.'s work [18] applies the PageRank algorithm to image data. Their task is auto-captioning in which, given a novel image, the most probable caption words are assigned by using the PageRank algorithm. However, their work requires labeled training sets (*i.e*. caption words should be annotated to the segmented regions in training images). Also, their task is a labeling problem rather than a visual perception task.

2. Construction of *Visual Similarity Networks*

The basic representation on which we will operate is a weighted directed graph, termed the *visual similarity network* (VSN). The nodes of the VSN are the features extracted from all of the training images, and the edges of the graph link features that have been matched across images.

We denote the set of training images by $\mathbb{I} = \{I_a\}_{a=1,\ldots,m}$, from which we wish to extract K categories. We denote image indices by $a, b, \ldots$ and feature indices by $i, j, \ldots$. The VSN is a graph $G = (V, E, W)$, where V is the set of vertices, E is the set of edges, and W is the set of edge weights. We also denote the adjacency matrix of the VSN by M. Each node in V is denoted by a_i, representing the i-th feature in image I_a. We denote the total number of features by n, and the number of features in image a by n_a.

In general, a node in the VSN can be any unit of local visual information. Here, we use the standard Harris-Affine interest point detector [17] and the SIFT descriptor [16]. That is, all affine covariant regions extracted from all the training images form the set of nodes (V) of the VSN. We now describe the procedure used to form the links in E, and finally the way the weights in W are computed.

2.1. Establishing edges in the VSN

The links are established by finding matches between features in different images. In our case, we apply the spectral matching of [13, 14] to each pair of images (I_a, I_b). This approach combines matching based on local data with geometric consistency constraints by finding a combination of correspondences that is globally most consistent, based on pairwise relations between features. The algorithm requires an initial set of potential correspondences based on local appearance, which we obtain by using the L_2 distance between SIFT descriptors. The second-order geometric affinity is calculated by the over-complete set of translation invariant geometric relations proposed in [15]. The advantage of this particular matching technique is that it is fast and simple and it has been shown to have good limit properties in the context of inference in problems defined by first- and second-order potentials. Other similar matching techniques such as the deformable matching of [2] or pyramid matching of [11] could be used as well.

After matching all the pairs of images, each correspondence between features a_i and b_j forms a new edge between the two corresponding nodes in V: $e = (a_i, b_j)$. It is important to note that, at this stage, we do not require the matching to be accurate. The resulting graph may be quite noisy, especially because of extra edges between images that do not contain any common objects. The link analysis algorithm will be responsible for dealing with the (possibly large number of) incorrect correspondences. In fact, we do not want the matching to be *too strict*, in which case it might find very few correspondences which will not be enough to populate the graph. To further relax the matching, we allow many-to-many correspondences between a pair of images, by iterating the one-to-one algorithm of [13]. Also, we use a fixed model of the pairwise geometric relations, rather than a more accurate model learned from training data as in [15]. The final output of this process is a set of potential correspondences E.

The resulting graph is not necessarily symmetric: If I_a (the *query* image) is matched to I_b (the *reference* image), then the initial correspondences are obtained by retrieving the k-nearest features b_j to each feature a_i (by using the L_2 distance between SIFT descriptors). If the order of the query and reference images is reversed, we will instead retrieve the k-nearest features to each feature in I_b, which will yield a different set of initial correspondences. As a result, the edge (a_i, b_j) does not necessarily exist if the (b_j, a_i) exists. If they both exist, their weights (described in the next section) will be different in general.

Figure 1 shows an example of edge construction by comparing the same image on the left to two different images from different object classes on the right. There is a substantial number of wrong correspondences in the bottom pair because the object classes are different between the two images (*i.e.* an *giraffe* on the left and a *car* on the right.) However, if the matching behavior is consistent, the link analysis techniques introduced below will be able to extract the major trends from noisy correspondences.

2.2. Computing the edge weights

The weight w_e of the edge $e = (a_i, b_j)$ should reflect how consistent that correspondence is with all the other correspondences obtained in matching I_a and I_b. A higher weight would indicate more confidence in the correspondence, meaning that many other correspondences would agree with it. In order to describe how w_e is computed, we need to look more closely at the spectral matching approach [13]: For the matching of a pair of images, a matrix Q is first created with one row and one column for each potential correspondences (a_i, b_j). Here, we abbreviate the notation (a_i, b_j) to simply ij since we are dealing with a single pair of images I_a and I_b in this paragraph. $Q(ij, i'j')$ contains the pairwise geometric consistency between correspondences ij and $i'j'$ such that the more deformation is needed to map the pair ij to the pair $i'j'$, the lower $Q(ij, i'j')$ is. The solution is found essentially by computing the principal eigenvector of Q and binarizing it by following the procedure described in [13]. We denote by $\mathbb{C}^*$ the set of correspondences that are selected at the end of the matching. An estimate of the confidence of $(a_i, b_j) \in \mathbb{C}^*$ is given by: $C_{ij} = \sum_{i'j' \in \mathbb{C}^*} Q(ij, i'j')/|\mathbb{C}^*|$. C_{ij} measures how well the correspondence ij agrees with all the other correspondences. In practice, Q is constructed such that $0 \leq C \leq 1$ for all the correspondences.

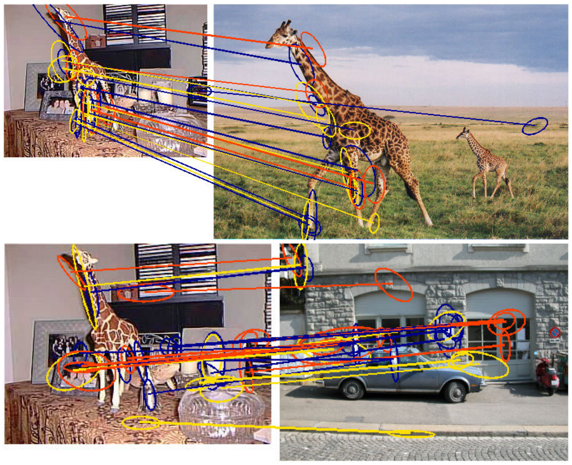

Figure 1. An example of link generation on two pairs of images. The features are matched by using a liberal spectral matcher. The jet colormap is used from red (strong) to blue (weak geometric consistency). (These figures are best viewed in color.)

One problem is that we cannot simply set w_{ij} to be equal to C_{ij} since we do not know in advance what, in absolute terms, is a *good* value for C. To address this problem, we take into account the histogram of the confidence values and the rank ordering of each C_{ij} rather than the absolute values. More precisely, let $\{t_l\}, l = 1, \ldots, N_t$ be a set of thresholds such that $t_1 < \ldots < t_{N_t}$. We define $\mathbb{C}_l$ as the set of surviving correspondences ij such that $C_{ij} > t_l$. Clearly, $\mathbb{C}_{N_t} \subset \mathbb{C}_{N_t - 1}, \ldots, \subset \mathbb{C}_1 \subset \mathbb{C}^*$. In practice, we use five thresholds regularly spaced between 0.8 and 0.4. For example, in Fig.1, the red features belong to a subset of strongly geometrically consistent features for $t = 0.8$ while the blue ones have weaker consistency for $t = 0.4$.

Finally, the weight w_e of the edge $e = (a_i, b_j)$ is defined as:

$$w_e = \frac{1}{n_a} \sum_{l=1}^{N_t} S_l \mathbf{1}_{\mathbb{C}_l}(ij) \tag{1}$$

where S_l is the normalized score $S_l = \frac{\sum_{ij \in \mathbb{C}_l} C_{ij}}{|\mathbb{C}_l|}$ which measures the global consistency of the set of correspondences that are over a particular threshold. $\mathbf{1}_{\mathbb{C}_l}(ij)$ is the value of the indicator function which is defined by 1 if the correspondence ij is in the set $\mathbb{C}_l$. The division by n_a is intended for the normalization of irregular number of features in each image. Intuitively, ten correspondences from an image with a total of 50 features should be weighted more than ten votes by an image with 100 features.

Consequently, by iterating the matches between all pairs of image, we obtain a sparse $n \times n$ matrix M (*i.e.* the adjacency matrix of the VSN G) where $M(a_i, b_j)$ is the value of the weight w_e of the edge $e = (a_i, b_j)$.

3. Inference of Object Models from Networks

3.1. Ranking of Visual Information

Since we use all the features in the training images, and since the links are very noisy, we need a mechanism to estimate the relative importance of nodes in the graph. This can be done by treating the weights associated with the links in the VSN as *votes* for the importance cast by other nodes. Even though there will be a lot of false links from different classes or even background, they are highly likely to have higher variations in those linking behaviors than the links between the nodes of the same objects, which will vote more consistently. In order words, *hubs* in a given class are likely to be formed through consistent matches with features in the same class.

Well-known ranking algorithms such as PageRank [5] and Kleinberg's HITS algorithms [12] can estimate the ranked importance of nodes in a graph using only the graph connectivity. In our experiments, PageRank slightly outperforms the Kleinberg's HITS and it is the one that we use as the baseline algorithm. In its most general form, the ranking algorithm generates the $n \times 1$ PageRank vector P by solving the equation [5]:

$$P = (1-\alpha)(M+D)P + \alpha v, \tag{2}$$

where M is the weight matrix of the graph (the VSN in our case), α is a constant close to zero (in all of our experiments $\alpha = 0.1$), v is the *transport* vector ($= [\frac{1}{n}]_{n\times 1}$, uniform probability distribution over all nodes), and $D = vd^{\mathrm{T}}$ (d is the n-dimensional indicator vector identifying the nodes with outdegree 0). Intuitively, the definition of ranking can be viewed recursively in the sense that components of P with high values are nodes connected to many nodes with high values.

We obtain the PageRank vector P_a for each image I_a in $\mathbb{I}$ by considering the portion of the VSN M obtained by considering the links between the nodes a_i and all of the other nodes from the other images. In other words, when computing P_a, we use the modified M_a for Eq.2 by enforcing $M_{ij} = 0$ *if* $i \notin I_a$ *and* $j \notin I_a$. This eliminates the interactions between the features irrelevant of the image I_a for the computation of P_a. Intuitively, a large $P_a(i)$ means (1) if $i \in I_a$, i is an relatively important feature in the image I_a (*i.e.* this information is valuable for *localization* of an object in the image) or (2) if $i \notin I_a$, i is an highly relevant feature with respect to I_a (*i.e.* useful to clustering of the images according to object classes).

3.2. Structural similarity

In constructing the edges of the VSN (Section 2.1), we already consider two types of similarities, appearance similarity and geometric consistency. However, once we have a global representation of all the interactions between the features, we can infer another type of similarity termed *structural similarity*. The underlying observation is that similar nodes are highly likely to exhibit similar link structures in the graph. This observation is illustrated in Figure 2, in which both node i and node j are *wheel* features. Both nodes are highly likely to point out to and to be pointed to similar sets of features (*e.g.*, other *wheel* nodes), and at the same time both are highly unlikely to be linked to the same set of entities from different objects or background clutters.

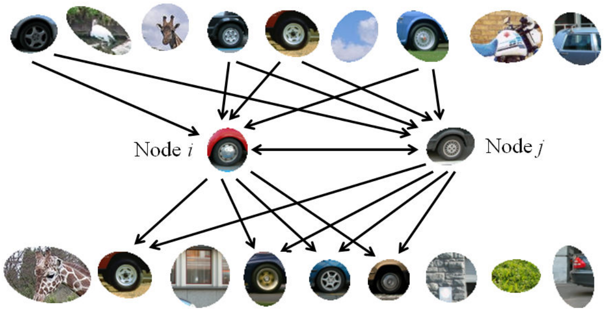

Figure 2. An example of structural similarity in a small part of the VSN G. Each link with its weight represents its local similarity measure (appearance and geometric consistency). The structural similarity captures the degree of similarity of the link structures of two nodes. The structural similarity is high if the two nodes match similar nodes like the *wheel* patches in this example.

Brondel *et al.* propose an algorithm which provides a generalized method to compute structural similarities between vertices of two directed graphs by using only link analysis [4]. The simplified version of the algorithm which we use here is the same as the algorithm used for automatic discovery of synonyms in a dictionary [4]. Given the VSN G, the *neighborhood graph* G_{ai} of a node a_i is the subgraph of G whose vertices are pointed to by a_i or are pointing to a_i. Let M_{ai} be the adjacency matrix of G_{ai}. M_{ai} is of dimension $N_{ai} \times N_{ai}$, where N_{ai} is the number of nodes in G_{ai}.

The algorithm of [4] defines the *central score* which is the similarity scores between the vertexes of G_{ai} and the vertex 2 of the path graph of length 3 of Eq.3. B is the incident matrix of the path graph. Intuitively, if a vertex $b_j \in G_{ai}$ has a high *central score*, then b_j and a_i are likely synonyms such that they contain the same words in their definitions and at the same time they are included in the definitions of the same words.

Operationally, the structural similarity values between G_{ai} and the graph of Eq.3 are computed by iterating Eq.4.

$$1 \rightarrow 2 \rightarrow 3, \quad B = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3}$$

$$U_{k+1} = \frac{BU_kM_{ai}^T + B^TU_kM_{ai}}{\|BU_kM_{ai}^T + B^TU_kM_{ai}\|_F}, \tag{4}$$

where U_k is a $3 \times N_{ai}$ matrix, initially set to $\mathbf{1}$, and $\| \cdot \|_F$ is the Frobenius norm. Upon convergence, $U_{ai} = \lim_{k \to \infty} U_k$ has the property that $U_{ai}(2, b_j)$ is the structural similarity value for each node b_j in the neighborhood of a_i [4]. In other words, a large value $U_{ai}(2, b_j)$ indicates that b_j and a_i share a lot of common nodes both in the incoming and outgoing directions.

This structural similarity algorithm is applied to each node a_i independently and the resulting similarity values are combined in a single $n \times n$ matrix Z, such that $Z(a_i, b_j)$ is the structural similarity of node b_j to a_i: $Z(a_i, b_j) = U_{ai}(2, b_j)$. Although n can be large, Z is very sparse in practice. We row-normalize Z to make the sum of vertex similarities of all the other features with respect to a feature to 1.

4. Unsupervised Modeling

From the link analysis described above, we have now two pieces of information: the PageRank vectors P_a for all the images $I_a \in \mathbb{I}$, which characterize how strongly the features of each image is related to all the other features from the other images, and the vertex similarity matrix Z, which characterizes how structurally similar the nodes are with respect to each other. We will now use these two pieces of information in a two-step approach to unsupervised modeling. First, we will estimate which image belongs to which category. Roughly speaking, this step is the counterpart of the *topic discovery* step used in other approaches [21]. Second, for each category, we will estimate which features from the training images are relevant to that category. This is similar to the localization step used in other approaches with the critical difference that we do not attempt any clustering of the original features; we use directly the original features and we merely assess which feature is important for a given category. We argue that this can be done in large part by direct link analysis of the VSN without any clustering, statistical modeling, or other difficult manipulation of the actual feature values.

4.1. Category discovery

Our first objective is to partition $\mathbb{I}$ into K groups corresponding to the K categories. Of course, this is not optimal because it prevents the correct handling of images containing multiple categories (a case that is generally not handled by unsupervised techniques) and because it would be better to not make a hard decision on the partition of $\mathbb{I}$ before the next step. However, we feel that this is still an effective approach for demonstrating the feasibility of using the link analysis techniques for this problem.

The basic idea is to combine the m PageRank vector P_a and the $n \times n$ matrix Z into a single $m \times m$ affinity matrix A. $A(a, b)$ measures the affinity of I_b with respect to I_a and by combining 1) the total sum of $P_a(b_j)$ for the features in I_b, and 2) the total sum of the $P_a(a_i)$ of the features in I_a distributed proportionally to vertex similarities. $A(a, b)$ takes into account the entire graph structure to evaluate how confident we are that I_a and I_b contain a consistent collection of features corresponding to the same category. Intuitively, if they do, then they should be linked to similar groups of images, and many of their features should be structurally similar. In practice, A is computed as:

$$A(a, b) = \sum_{b_j \in I_b} P_a(b_j) + \sum_{a_i \in I_a, b_j \in I_b} P_a(a_i) Z(a_i, b_j), \quad (5)$$

for all pairs of images I_a and I_b. The groups of images corresponding to the K categories are estimated by partitioning A by using spectral clustering. Following [23], we use the Shi and Malik's Normalized spectral clustering [20] on the k-nearest neighbor graph. The k-nearest neighbor graph is easy to work with because it is a sparse matrix, and known to be less sensitive parameter settings [23]. In practice, we use $k = 10 \log(m)$ since [23] recommends that k be chosen in the order of $\log(m)$. After measuring the accuracy for different values of k, the variation of performance is less than 2% over the experiments reported below. This means that the matrix has strongly separated blocks in practice and that, therefore, the affinity measure is effective at separating the different categories. Figure 3 shows the affinity matrix computed from one dataset. This example shows that the groups of images corresponding to different categories are clearly separated by using the definition of affinity above.

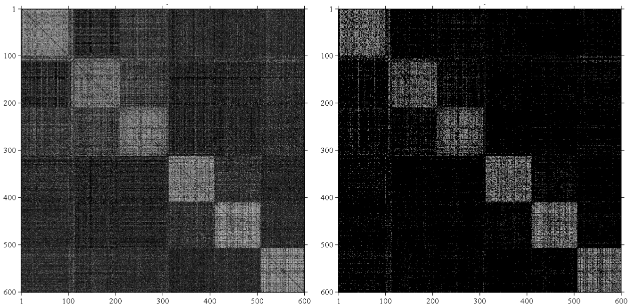

Figure 3. The raw affinity matrix A computed from 600 images of 6 categories of the Caltech-101 dataset (left) and the same matrix after retaining the $10 \log(m)$ largest values for each node (right). The rows and columns have been ordered so that images in the same category are grouped together.

4.2. Localization

The objective of the localization step is to establish which features are relevant to each category. To do this, we first apply again the page rank algorithm, but this time to all the features in each category. More precisely, for each class c, we compute the page rank matrix P_c based on Eq. 2, replacing M by M_c, that is, the graph matrix obtained by using only the features from images in category c. The relative importance of each feature a_i from an image I_a in category c should be estimated by combining its own rank, $P_c(a_i)$ with the sum of all the other features' rank, weighted by their structural similarity to a_i, so that the importance of a feature will increase if many other features agree with it:

$$I_c(a_i) = P_c(a_i) + \sum_{b_j \in c} P_c(b_j) Z(b_j, a_i). \qquad (6)$$

$I_c(a_i)$ can be interpreted as a confidence measure that each feature a_i belongs to the class c. As the last final localization step, we select the features whose importance are close enough to the maximum in the image: $I_c(a_i) \geq \rho \times max_{a_i} I_c(a_i)$. Different operating points are obtained by varying ρ as shown in the localization experiments below. All the examplar results shown later in this paper used $\rho = 0.8$ to be consistent with the top 20% rule used in [19].

5. Experiments

The input of our algorithm is a set of m unlabeled images with a single piece of information (*i.e.* the number of object categories K). The outputs are the classification of images according to object classes, and the ranked importance of all features with respect to their object categories, from which we can easily estimate most probable locations of the objects in the images.

We evaluate the proposed unsupervised modeling method using two different datasets, which are Caltech101-dataset [7] and TUD/ETHZ dataset [1]{ETHZ *Giraffes*, TUD *Motorbikes*, TUD *Cars*}. By following the experimental setup proposed by Grauman and Darrell [11], we iterate the same experiment ten times, in which 100 and 75 images per object are randomly picked in each object class for the Caltech-101 and TUD/ETHZ dataset, respectively. We select only 75 images for the TUD/ETHZ experiments because there are only 83 images for the *giraffe* class.

5.1. Category discovery

For Caltech-101, we selected six object classes which have more than 100 training images – {*airplane*, *rear cars*, *faces*, *motorbikes*, *watches*, *ketches*}. We measure how well the unlabeled training images are clustered according to their categories by measuring the agreement of topic discovery with ground truth labels. Table.1 shows the confusion matrices for Caltech-101 classes. As shown in the results, our performance is competitive compared to previous work. In the case of four object classes, our results achieve 98.55% success ratio (compared to 98% in [21]). We outperform the Grauman and Darrell [11]'s method (86%) by more than 10% by using the same experimental as theirs. While related prior work generally goes up to four classes, we show that we can increase the number of classes with only a slow degradation in performance: 97.30% and 95.42% for five and six object classes, respectively. Also, for the TUD/ETHZ dataset, our method achieved 95.47% classification success ratio. Unlike the Caltech-101, this dataset has a lot of class variations and clutter in the background.

	A	C	F	M
A	**98.4**±0.82	**1.0**±0.9	**0.1**±0.3	**0.5**±0.7
C	**0.2**±0.4	**99.8**±0.4	**0.0**	**0.0**
F	**1.9**±1.3	**0.1**±0.3	**98.0**±1.2	**0.0**
M	**1.4**±1.2	**0.6**±1.0	**0.0**	**98.0**±1.5

	A	C	F	M	W
A	**98.2**±1.2	**0.7**±0.8	**0.1**±0.3	**0.8**±0.4	**0.2**±0.4
C	**0.6**±0.7	**99.3**±0.8	**0.0**	**0.0**	**0.1**±0.3
F	**2.2**±1.3	**0.1**±0.3	**96.2**±1.7	**0.0**	**1.5**±1.5
M	**1.3**±0.8	**0.9**±1.1	**0.0**	**97.5**±1.6	**0.3**±0.7
W	**2.7**±2.1	**0.8**±0.4	**0.0**	**1.2**±1.0	**95.3**±1.9

	A	C	F	M	W	K
A	**94.5**±4.2	**0.5**±0.7	**0.0**	**0.5**±0.5	**0.3**±0.5	**4.2**±3.8
C	**1.1**±2.2	**97.1**±3.2	**0.0**	**0.0**	**0.0**	**1.8**±2.1
F	**1.5**±1.2	**0.0**	**95.6**±2.5	**0.0**	**1.8**±1.8	**1.1**±1.0
M	**1.4**±1.6	**0.4**±0.7	**0.0**	**93.5**±3.3	**0.1**±0.3	**4.6**±3.3
W	**2.2**±1.0	**0.3**±0.5	**0.0**	**0.3**±0.7	**93.4**±2.7	**3.8**±2.3
K	**1.5**±1.2	**0.0**	**0.1**±0.3	**0.0**	**0.0**	**98.4**±1.3

Table 1. Confusion tables for the Caltech-101 data set for increasing number of objects from four to six. The means and standard deviations of 10 runs for each are shown. The modeling accuracies (*i.e.* the averages of the diagonals) of four to six object categories are **98.55%** **97.30%**, **95.42%**, respectively. (A:*Airplanes*, C: *Cars*, F: *Faces*, M: *Motorbikes*, W: *Watches*, K: *Ketches*)

	M	C	G
M	**93.3**±2.7	**0.0**	**6.7**±2.7
C	**4.8**±2.6	**95.2**±2.6	**0.0**
G	**2.0**±1.1	**0.1**±0.4	**97.9**±1.4

Table 2. Confusion tables for the TUD/ETHZ shape dataset. The means and standard deviation values of 10 runs for each are shown. The classification accuracies (*i.e.* the averages of the diagonals) are **95.47%**. (M:*Motorbikes*, C: *Cars*, G: *Giraffes*)

5.2. Localization

Localization is in general harder to measure and, in fact, most prior work evaluates classification and localization performance in separate experiments. For example, [6, 21, 22] designed simpler experimental setups for evaluating localization performance such as limiting the experiments to two category cases. Here, we evaluate the localization on the same setup as we used for evaluating classification, including up to six categories in the training set.

We use two metrics proposed by [19] - bounding box hit rates (BBHR) and false positive rates (FPR). Some papers use the segmentation ratios of intersections of detected regions and ground truth [6, 21] for the localization. But we feel BBHR and FPR would be better because we use Harris-Affine interest regions as our unit visual information instead of segmented patches that are more amenable to pixel-wise

[1] The TUD *Motorbikes* and *Cars* dataset is available at http://www.pascal-network.org/challenges/VOC/ and ETHZ *Giraffes* at http://www.vision.ee.ethz.ch/datasets.

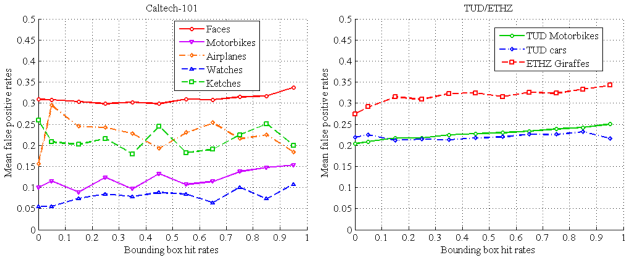

Figure 4. BBHR-FPR plots for Caltech-101(Left) and TUD/ ETHZ (Right) dataset.

error. The bounding box hit (BBH) number is incremented for every ground truth box in which more than h features fall. We use $h = 5$, following [19]. The BBHR is the number of BBH divided by the total number of object instances in the dataset (BBHR=1 for perfect localization). The FPR is defined as the number of selected features lying outside the bounding box, divided by the total number of selected features (FPR=0 for perfect localization). In general, the FPR is a fairly severe measure because it counts the number of features without accounting for their spatial extent. For example, 10 misclassified features may give a high FPR even though they are clustered in a very small region. Unfortunately, for feature-based approaches, there is no absolutely fair measure, unlike patch-based methods for which a pixelwise error rate can be defined easily.

As proposed in [19], we generate BBHR-FPR curves by varying the relative threshold ρ (Fig.4). The plots show that our methods achieve reasonably low FPRs across the BBHRs. For some objects of caltech-101 dataset such as *watch* and *motorbikes*, the FPRs are fairly low since the objects are generally quite large in the image and only one object instance exist in most cases. The principal remaining source of errors is that, although our unsupervised classification is quite accurate, the misclassified images might produce the wrong localization result. For example, if a *face* image is misclassified into a *airplane*, the matched regions are unlikely to be on the correct object, which leads to localization errors. On the other hand, *faces* in Caltech-101 and *giraffes* in ETHZ dataset generate higher FPR values. This is primarily because there is relatively little background variation across some of the training images. For example, since trees in grassy plain are very often observed along with *giraffes* across the training images, it is natural that trees are also considered as important visual information for the *giraffes* class. In the case of *faces*, the higher FPRs are mainly due to the fact that the upper bodies (especially, shoulders) are always in the image with the faces, but the bounding boxes are located on the faces only.

Fig.5 shows some examples of the localization. Even though there are a lot of features in the background, the high confidence features are mostly on the objects. Some selected features on the background are low-ranked (colored blue). At the same time, the class representative features are fairly selected as *hubs* such as reddish *wheels* in the *car* class and *eyes* in the *face* class.

5.3. Computational issues

The VSN is represented by a $n \times n$ matrix, where n is the total number of features. However, in practice, the VSN is very sparse. For example, in the case of six object classes in Caltech-101, the number of nodes in the VSN is about 90,000. The VSN is quite sparse since the ratio of nonzero elements is about 5×10^{-4}. The sparseness of the vertex similarity matrix Z is about 0.002. However, since most of the non-zero elements have very low values, we could use an even sparser matrix by thresholding it.

The basic operation used in the algorithms is the power iteration on matrices. Owing to the sparseness of the matrices involved, the complexity of the power iteration grows roughly linearly with n. This is similar to the behavior observed in other applications of the link analysis techniques [3]. In addition, motivated by the very large size of the matrices involved in Web applications, there has been a lot of work in optimizing the power method by taking advantage, among other things, of latent block structure and convergence acceleration techniques [3]. Although we did not use them in this work, these methods would enable scaling to much larger graphs.

6. Conclusions

We proposed an approach for extracting object models from unlabeled training data. Unlike prior methods, this approach extracts categories and features within the categories by analyzing a visual similarity graph, without clustering or statistical modeling. This representation provides a global view of the interactions between all features which allows us to use different types of information - ranked importance of each feature with respect to an image or an object category and structural similarity between any pair of nodes. This approach yields better results on the Caltech-101 examples used in prior work in unsupervised modeling, with a larger number of classes. We also showed competitive results for the TUD/ETHZ dataset.

We believe much remains to be done for this approach to be used in other visual tasks. In particular, even though we have a rich representation which describes all interactions between low-level visual information, we can certainly improve the way we integrate it in Eq.5 and Eq.6, which are first-order sums of two types of information estimated from link analysis. More detailed investigation of feature level interactions is needed, including handling a set of images containing multiple categories and instances. Finally, the sparseness of the data observed in these experiments, together with the fact that the link analysis tools are routinely used in far larger applications suggest that it is possible to scale the algorithms up to a much large of classes.

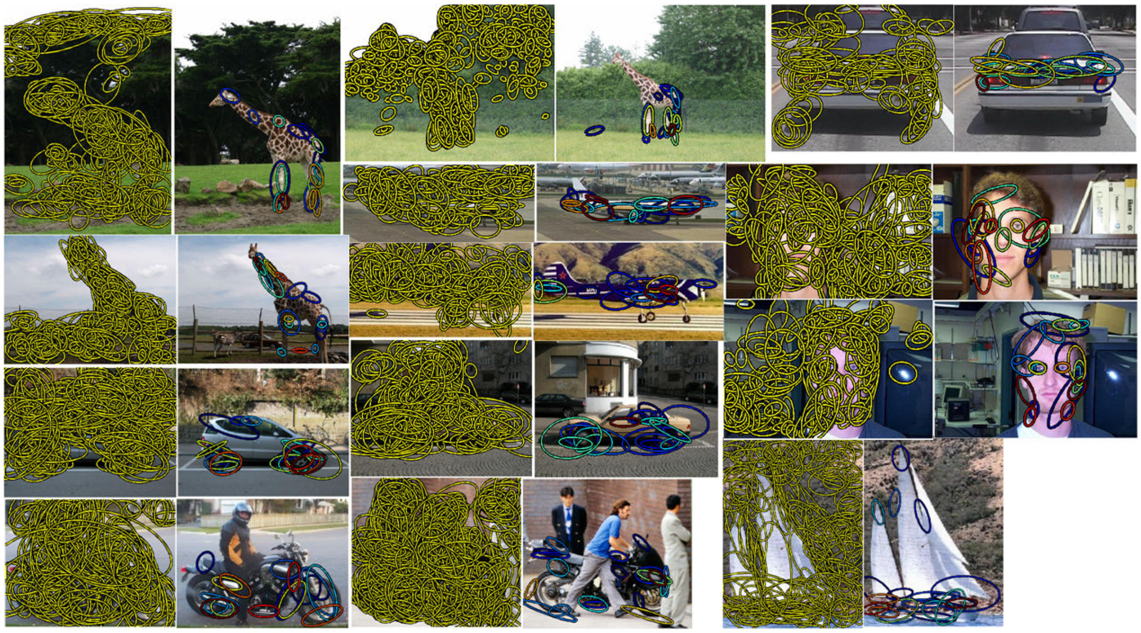

Figure 5. Some examples of localization for the Caltech-101 and TUD/ETHZ dataset. In each image pair, the left image represents original extracted features with yellow, and the right image shows top 20% high-ranked features with color variance according to the importance weights. The jet colormap is used from red(high) to blue(low). (These figures are best viewed in color.)

Acknowledgement. This research was performed in part for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Program funded by the Ministry of Commerce, Industry and Energy of Korea.

References

[1] A.-L. Barabási. Scale-free networks. *Scientific American*, 288:60–69, 2003.

[2] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence, 2005. CVPR.

[3] P. Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2(1):73–120, 2005.

[4] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. V. Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Review*, 46(4):647–666, 2004.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine, 1998. WWW.

[6] L. Cao and L. Fei-Fei. Spatial coherent latent topic model for concurrent object segmentation and classification, 2007. ICCV.

[7] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *PAMI*, 28(4):594–611, 2006.

[8] L. Fei-Fei, R. Fergus, and A. Torralba. Recognizing and learning object categories, 2007. Short Courses for CVPR.

[9] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search, 2005. ICCV.

[10] M. Fritz and B. Schiele. Towards unsupervised discovery of visual categories, 2006. DAGM-Symposium.

[11] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features, 2006. CVPR.

[12] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[13] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints, 2005. ICCV.

[14] M. Leordeanu and M. Hebert. Efficient map approximation for dense energy functions, 2006. ICML.

[15] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features, 2007. CVPR.

[16] D. Lowe. Distinctive image features from scale invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[17] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[18] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery, 2004. KDD.

[19] T. Quack, V. Ferrari, B. Leibe, and L. V. Gool. Efficient mining of frequent and distinctive feature configurations, 2007.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[21] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images image features, 2005. ICCV.

[22] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images, 2006. CVPR.

[23] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.