Object group model for scene studying

Haizi Yu
Computer Science Department
Stanford University
haiziyu@stanford.edu

Abstract

This paper [1] concerns a newly built model, namely object group (OG) model, for scene studying. The model is well suited for recognizing and classifying image content in the scene level. The first half of the paper will concentrate on introducing the OG model and coming up with a set of well developed methodologies to learn the model. The second half of the paper will introduce the techniques used to extract the OG feature of an input image, given the learned OG model. Such OG feature can be used later in various machine learning algorithms (e.g., k-NN, SVM) to do scene recognition or scene discovery.

1. Introduction

1.1. The existence of object groups

In daily life, people talk about objects individually, as well as thinking connections among objects. We observe some objects tend to co-occur as a group in natural environment. Given such observation, we hypothesize a hierarchical structure in human visual perception with individual objects lie in the bottom level, scene concepts in the top level, and the notion of object groups in between. See Figure 1.

Both a bottom-up view and a top-down view occurs naturally and usually simultaneously when people are considering this hierarchical structure. On one hand, knowing the existence of certain objects helps us get aware of the scene content. For example, it is very likely that we are in an outdoor scene if we observe the existence of sky, trees and rivers; in contrast, it is very likely that we are in an indoor scene if we observe the existence of bed, desks and chairs. On the other hand, knowing the scene helps us both recognize and localize the individual objects better. For example, if we are told that we are in an indoor scene, it would be rare to expect trees in it. If we gain more knowledge about the scene, e.g., the prior information about the relative positions of objects in it, then it would be also rare to find there is a chair on the bed.

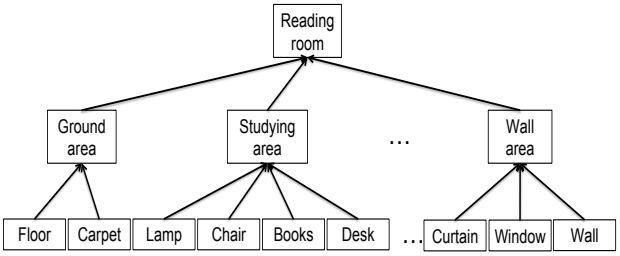

Figure 1. Hierarchical grouping.

Object detection [2] has been a hot topic in Computer Vision for a long time. However, through tons of research papers on object detection, people tend to treat the desired object individually and separately, as a single entity. In contrast, this paper will aim to detect a group of objects which tend to go together in our daily life, e.g., knife and fork, sky and cloud, in hope of exploring the relationship among objects. There comes the rough idea of object groups.

[1] This work is part of the submitting CVPR 2012 paper of Hao el al. [3]

1.2. Some statistics on object groups

Figure 2 illustrates the co-occurrence relation between objects in a subset of LabelMe [8] dataset. This shows consistently with common sense that certain object tends to go together with some other objects to form an object group, e.g, desk and lamp tend to go together to form a studying area.

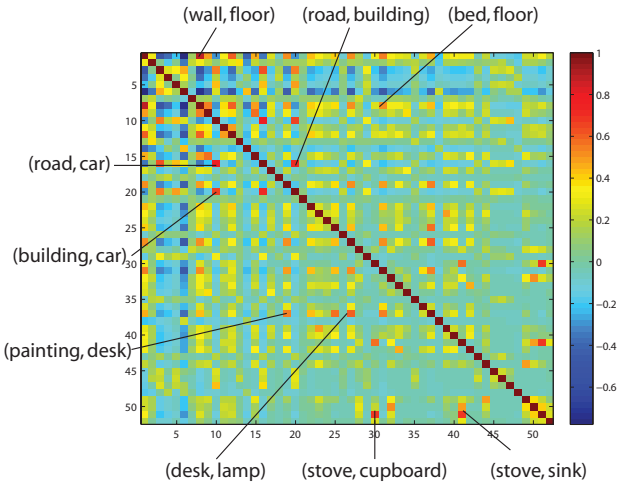

Figure 2. The co-occurrence relation between objects in a subset of LabelMe dataset. Each row (or column) corresponds to one object. The color indicates the correlation between the pair of objects. The warmer the color is, the higher the correlation is. We observe that object co-occurrence is quite common in real world images.

Figure 3 illustrates the relative location of two objects that tends to go together (i.e., in the same object group). This also shows consistently with common sense that correlated objects have a somehow "fixed" relative location to each other, e.g., it is very rare to see sky in the bottom of an image while trees are above it.

2. The object group (OG) model

We have roughly described the two important information contained in an object group. In this section we are going to formally introduce the object group model. We'll first talk about the mathematical representation of object group, called OG templates, which literally encode the co-occurrence property and spatial information mentioned above. Then, we'll discuss the methodology used to learn OG templates given a training set.

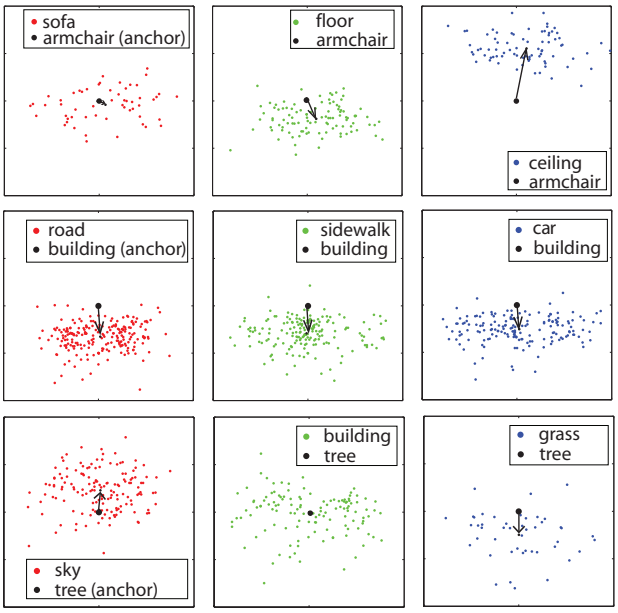

Figure 3. Relative location between positively correlated pairs of objects. In each row, one object is chosen as an anchor object. Given the anchor object, we then select positively correlated objects, which are objects tend to co-occur with the anchor object. We sample images containing both the anchor object and the correlated object and then plot the relative location between the center of the pair of objects as a cloud map. The vector drawn in each figure directs from the anchor object to the mean of the relative location of the correlated object.

2.1. Preliminary terminology

2.1.1 The Hist-image [5]

We use the notation $G \in \mathcal{H}^{m \times n}$ to denote an $m \times n$ Hist-image, which is an $m \times n$ image with each pixel replaced by a histogram[2], which we call a Hist-pixel.

A 2×2 and a 2×3 Hist-image are illustrated in Figure 4.

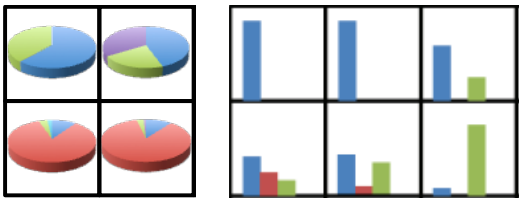

Figure 4. Two examples of Hist-images: the left is a 2×2 Hist-image, with each Hist-pixel described by a pie-shaped chart; the right is a 2×3 Hist-image, with each Hist-pixel described by a column-shaped chart.

[2]The number of the instances in the histogram is given, so as the names and orderings of the instances.

We are now ready to introduce two operations for the Hist-image we just defined.

- **The dot product between two Hist-pixels.** Given the instances of the histogram, any Hist-pixel can be treated as a vector with dimensionality the same as the number of histogram instances. Thus, we define the dot product between two Hist-pixels which is a scalar, to be the dot product of their two corresponding vectors.
  We overload the "$\cdot$" symbol to denote the dot product operation.

- **The correlation between two Hist-images.** The correlation between an input Hist-image and a kernel (mask) Hist-image is almost the same as the correlation between two images, with the only difference that we replace every occurrence of pixel-wise multiplication to be a Hist-pixel-wise dot product.
  We call the output of the correlation between an input Hist-image and a kernel Hist-image a response map of the input image.
  We overload the "*" symbol to denote the correlation operation.

2.1.2 The H-transform

We use the notation $H_0(\cdot)$ to denote the H-transform, which maps an annotated input image to a single Hist-pixel, i.e., simply a histogram.

An annotated image is an image contains object annotations, i.e., ideally each pixel will be designated without ambiguity an object label which it belongs to. In real experiments, we would allow less than 10% of the pixels to be not designated. [3]

We can implement the H-transform $H_0(\cdot)$ by counting the pixels that belong to certain objects, accumulating the numbers in the corresponding bins and finally normalizing to obtain the histogram.

We use the notation $H_{m\times n}(\cdot)$ to denote an extension of the H-transform, which at first equally divides the input image into $m\times n$ sub-images and apply the H-transform $H_0(\cdot)$ to each sub-images. The output of $H_{m\times n}(\cdot)$ is an $m \times n$ Hist-image. [4]

Figure 5 shows an illustration of the extended H-transform $H_{4\times 4}(\cdot)$.

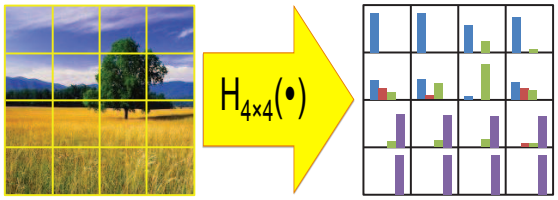

Figure 5. The extended H-transform $H_{4\times 4}(\cdot)$. The annotation of the input image is not shown. There are four instances in the histogram: sky (blue), hill (red), trees (green), grass (purple).

The extended H-transform will be widely used in the sequel.

2.2. OG templates

An OG template is the mathematical representation of an object group.

2.2.1 Notations

- $\mathcal{O} =< O_1, O_2, \ldots, O_h >$.
  The codebook which specifies the instances of a histogram. A typical codebook is a list of single objects where ordering matters. For example, $\mathcal{O} =<$ 'tree', 'chair', $\ldots$, 'bike' $>$.

- $\mathcal{I} = \{I_1, I_2, \ldots, I_p\}$.
  The set of input images.

- $\tilde{\mathcal{I}} = \{\tilde{I}_1, \tilde{I}_2, \ldots, \tilde{I}_p\} = H_{m\times n}(\mathcal{I})$.
  The set of Hist-images obtained by applying the extended H-transform to the original input images. Put it mathematically, $\forall \tilde{I}_i \in \tilde{\mathcal{I}}, \tilde{I}_i = H_{m\times n}(I_i) \in \mathcal{H}^{m\times n}$.

- $\mathcal{G} = \{G_1, G_2, \ldots, G_g\}$.
  The set of OG templates. $\forall G_i \in \mathcal{G}, G_i \in \mathcal{H}^{s\times t}$, for some predefined s and t, with the requirements that $s \leq m$ and $t \leq n$.

2.2.2 The desired OG templates

The desired OG templates is intentionally designed to encode the co-occurrence and spatial

[3] The SUN09 dataset, for example, is a good source for us to obtain such annotated images.

[4] It can be easily seen that $H_{1\times 1}(\cdot)$ is the same as $H_0(\cdot)$.

information which are mentioned in the introduction section at the same time.

As described in the notation above, an OG template $G \in \mathcal{H}^{s\times t}$ is an $s \times t$ Hist-image. Naturally, the co-occurrence properties can be encoded directly using histograms and the spatial information can be encoded as relative locations in the Hist-images.

We give a simple example to illustrate what a desired OG template might look like.

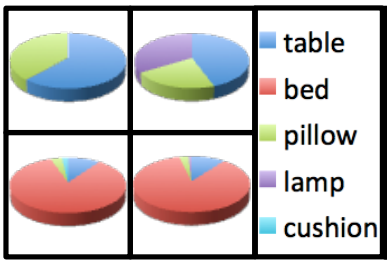

Figure 6. An example illustrating the desired OG template.

Suppose we want to hand design a desired OG template used for describing an object group for bed area. In this example, we take $\mathcal{O} =<$ table, bed, pillow, lamp, cushion, ... $>$, $G \in \mathcal{H}^{2\times 2}$ is a desired OG template in $\mathcal{G}$ (i.e., $s = t = 2$).

Figure 6 shows one possible such G. It says, 1) a typical object group for bed area contains objects like table, bed, pillow, lamp and cushion, but does not contain other things like trees or buildings; 2) inside this object group, a bed is more likely to be observed in the bottom half, while a pillow on contrast is more likely in the top half, since a pillow is usually on the bed rather than under the bed.

One thing should be aware of is that Figure 6 merely illustrates one possible and reasonable example for an OG template that we desire to have. An OG template is consider to be reasonable if it is consistent with our common sense, more accurately, it is consistent with most cases that we would see in various images existed in the world.

Therefore, it makes no sense to talk about the notion of a "perfect" or "optimal" OG template for an object group in general. However, it does make sense to find a set of OG templates that is the most consistent with a training set of images.

This implies we can learn a set $\mathcal{G}$ of OG templates given a training set of images by formulating and solving an optimization problem.

2.2.3 OG templates learning

We put an OG templates learning problem into an optimization problem illustrated as follows.

$$\text{minimize} \quad \sum_{i=1}^{p} \frac{\|s_i\|_1}{\|s_i\|_2} - \frac{\|\bar{s}\|_1}{\|\bar{s}\|_2} \tag{1}$$

$$\text{subject to} \quad s_i(j) = \max\{\tilde{I}_i * G_j\}; \tag{2}$$

$$\bar{s} = \frac{1}{p}\sum_{i=1}^{p} s_i; \tag{3}$$

$$\sum_{k=1}^{h} G_j(:,:,k) = 1; \tag{4}$$

$$G_j(:,:,k) \geq 0. \tag{5}$$

In the formulation mentioned above, $\mathcal{G} = \{G_1, G_2, \ldots, G_g\}$ and $\bar{s}$, $s_1, \cdots, s_p$ are the optimization variables, $\mathcal{O} =< O_1, O_2, \ldots, O_h >$ and $\tilde{\mathcal{I}} = \{\tilde{I}_1, \tilde{I}_2, \ldots, \tilde{I}_p\}$ are the known parameters of the optimization problem. $\tilde{\mathcal{I}}$ is also known as the training set of learning problem.

Constraint (2) and (3) simply give the definitions of the optimization variables $\bar{s}$, $s_1, \cdots, s_p$; while constraints (4) and (5) give the straight forward requirements for a histogram (i.e., nonnegative components summing to 1), since each OG template is a Hist-image.

The objective (1) is a little bit tricker which heavily inherits the idea of Sparse Filtering [4] . We'll explain this next in detail.

The quotient of L_1 norm and L_2 norm plays the key role in the objective (1). $\forall x \in \mathcal{R}^n$, the quotient $\|x\|_1/\|x\|_2 = \|x/\|x\|_2\|_1$ measures the element-wise sparsity of the vector x normalized by its L_2 norm. Thus, minimizing the quotient $\|x\|_1/\|x\|_2$ drives the vector x to be sparse, i.e., only a small number of components in x is nonzero. In contrast, maximizing the quotient will encourage most of the components in x to be nonzero.

The response vector s_i defined by constraint (2) takes its jth component to be the maximum of the response map $\tilde{I}_i * G_j$. One major property of the correlation operator is that it finds one patch in the input image, which is most similar to the kernel. So in this case, $s_i(j)$ finds the most similar patch to the OG template G_j in the input image $\tilde{I}_i$. The larger $s_i(j)$ is, the more likely that object group G_j is contained somewhere in the input image $\tilde{I}_i$. For an input image $\tilde{I}_i$, our goal is to find what object groups are contained in it. It's obvious that most images/scenes we see contains only a small number (possibly 1-4) of object groups. Therefore, we require the response vector s_i to be active only on a small number of its components and response strongly on those active components. Put it mathematically, we want to minimize the quotient $\|s_i\|_1/\|s_i\|_2$ for each i. That is exactly where the first term of the objective (1) comes from.

The average response vector $\bar{s}$ defined by constraint (3) is obtained by literally taking the average over $s_1, \cdots, s_p$. It can be easily shown that each response vector s_i is component-wise nonnegative. Therefore, if the jth component of $\bar{s}$ is nearly 0, i.e, $\bar{s}(j) < \epsilon$, where ϵ is small positive number, then $s_i(j) < \epsilon, \forall i = 1, \cdots, p$. This implies that the jth object group G_j is not likely to be contained in any of the input images. One can therefore, argue the necessity of the existence of G_j in the OG templates set $\mathcal{G}$. To avoid such phenomenon, we hope there does not exist a component in $\bar{s}$ which is inactive or active with very weak response relative to other components.[5] Put it mathematically, we want to maximize the quotient $\|\bar{s}\|_1/\|\bar{s}\|_2$, which is equivalent to minimize the negative quotient $-\|\bar{s}\|_1/\|\bar{s}\|_2$. That is exactly where the second term of the objective (1) comes from.

[5] In this course project, we predefined the number of OG templates in $\mathcal{G}$, i.e., we fix g at the beginning. How to choose such number g is a remaining issue to be discussed in the future research. One possible way of tackling this problem is always to try a big g first, if $\bar{s}^*$ which is the solution to the optimization problem, has k components that response weakly, we can decrease the value of g by k.

2.3. OG features

Once we have obtained or learned a set $\mathcal{G}$ of desired OG templates, we want to extract a feature vector from an image using $\mathcal{G}$ and make use of the feature vector to do scene studying.

2.3.1 Feature vector as s_i: spatial information lost

A natural way of extracting a feature vector from an image I_i by using $\mathcal{G}$ is simply using the response vector s_i. It can be obtained directly from s_i^* which is the solution to the optimization problem mentioned in the previous section.

Feature vector extracted in this way retains the information of object group existences. In other words, whenever an object group G_j is contained with high probability in image I_i, we would expect a large value of $s_i(j)$. However, we have no idea where the object group G_j is located in the image I_i, i.e., we lose the spatial information of the particular object group completely.

One reason for losing the spatial information is that we obtain s_i by simply solving the optimization problem. Note that constraint (2) of the optimization problem just takes the maximum value of the response map, but drops the information of where the maximum comes from (i.e., we doesn't care about the "argmax" of the response map $\tilde{I}_i * G_j$).

2.3.2 OG features: retain the spatial information via pyramid max pooling

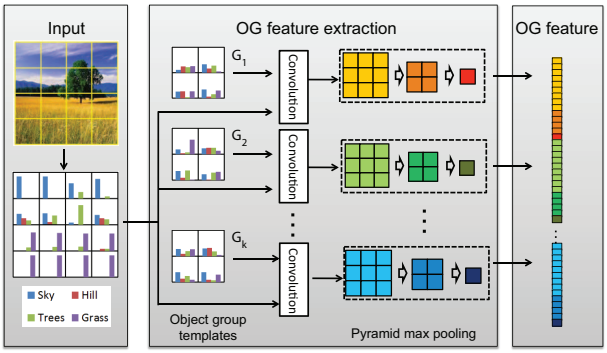

Figure 7. Extracting an OG feature for an image using the set of learned OG templates.

In order to come up with a feature vector that is capable in answering both the question of what object group is contained in the image and the question of where the object group is located, we use a pyramid max pooling [9] procedure after we get all the response maps. We concatenate everything obtained during pyramid max pooling to form a long vector, which is called the **OG feature** of an image.

Figure 7 gives a concrete example of how to extract OG features for an image by using learned $\mathcal{G}$. In this example, $m = n = 4$, $s = t = 2$, i.e., $\forall i = 1, \cdots, p, \tilde{I}_i \in \mathcal{H}^{4\times4}$; $\forall j = 1, \cdots, g, G_j \in \mathcal{H}^{2\times2}$. Therefore, the response map of the input image and each OG template is a 3×3 matrix. The right half of Figure 7 illustrates the procedure of pyramid max pooling.

Note that, in Figure 7, if we collect the red-most, green-most and blue-most square and concatenate them together, we get exactly the response vector s_i for an input image I_i. So the OG feature has a much larger dimensionality than the response vector does and now we get the spatial information back.

3. Scene studying using the object group model

So far, we have talked about the way of representing an input image as an OG feature in our newly built object group model. Therefore, it's straight forward to use the OG feature together with the existing machine learning algorithms to do some classification and recognition tasks in scene studying.

Figure 8 shows a typical system chart when performing scene studying tasks in the framework of the OG model.

4. Experiments and results

The **dataset** used for this set of experiments is a subset of the SUN09 dataset [1]. Each image in the dataset is an annotated image as required (i.e., more than 90% of the pixels are assigned to a certain object label). There are 573 object classes. The dataset has 2500 images with each image having a scene label from 17 predefined

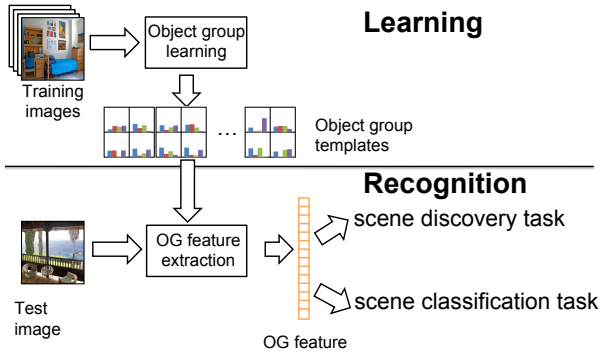

Figure 8. System chart for using OG model to scene studying tasks. Given training images, we first learn a set $\mathcal{G}$ of OG templates and then use $\mathcal{G}$ to extract an OG feature for a test image. We then use the OG feature for scene studying tasks such as scene discovery and scene classification.

scene classes. To assure enough examples in each of the scene classes, every scene class includes at least 50 images.

In a word, using the notation defined earlier, we have the number of images $p = 2500$, the number of objects $h = 573$ and the object codebook $\mathcal{O}$ is completely given.

4.1. Learned OG templates

To learn the set $\mathcal{G}$ of OG templates, we solve the optimization problem (1)-(5).

Before solving the problem itself, we first hand design several problem parameters as follows. Choose $m = n = 4$, $s = t = 2$, i.e., each input image will be transformed into a 4×4 Hist-image and each OG template is designed as a 2×2 Hist-image; choose $g = 32$, i.e., there are 32 OG templates in $\mathcal{G}$.

We first tried an iterative method called projected sub-gradient descent to solve the optimization problem. The method did not work quite well in two aspects: 1) the method is very sensitive to the choice of starting point and also very easily trapped around undesired the local minima due to the problem's high non-convexity; 2) the method progress extremely slowly and might not converge at all for some unfortunate starting points.

An alternative and more efficient method is called **projected Quasi-Newton (PQN) method** which is implemented by Hao et al [3]. The solver

is used in the following manner.

- **Starting point selection for $\mathcal{G}$.**
  We extract all of the $s \times t$ Hist-image patches from $\tilde{\mathcal{I}} = \{\tilde{I}_1, \tilde{I}_2, \ldots, \tilde{I}_p\}$. We perform a k-means clustering algorithm, where $k = g = 32$, to the extracted Hist-image patches. The output 32 cluster centers will then be chosen as the starting point for $\mathcal{G} = \{G_1, G_2, \ldots, G_g\}$, respectively.

- **Run the PQN based solver to solve the optimization problem.**
  The typical run time for solving the problem for $\mathcal{G}$ is around 20-30 minutes with GPU support, which is far more efficient and reliable than simple sub-gradient descent method.

- **Learned OG templates visualization** It is helpful to evaluate the learning task by outputting the learned G_j's together with the image patches on which the corresponding G_j has a high response. Figure 9 shows three typical examples.

Figure 9. OG templates examples. Each row represents one case of a specified OG template. The first column illustrates 3 OG templates represented by pie-shaped histogram. The rest part of each row lists a set of image patches on which the corresponding OG template has high responses. It can be seen that each learned OG template faithfully exhibits the set of objects that co-occur in the corresponding object group and moreover, the spatial information (top-left, top-right, bottom-left, bottom-right) is also retained in the OG template.

4.2. Automatic scene discovery [7]

Have the learned set $\mathcal{G}$ of OG templates at hand, it is ready to construct an OG feature as described in Figure 7 for an input image.

Using the extracted OG features, we perform a task named automatic scene discovery, which is described as follows.

Using the dataset we introduced at the beginning of this section, where each image is assigned to one of the 17 scene labels, we carry out a k-means (k = 17) clustering. We use three different representations, i.e., object bank (OB) [6] feature, OB feature with principle component analysis (PCA), OG feature, for images in the clustering task. After applying the k-means clustering algorithm, we'll get $k = 17$ cluster centers and their corresponding clusters. For each of the 17 groups, find an image I_c (represented by its feature vector) that is closest to the corresponding cluster center and predict every image in the cluster a scene label that is the same as the scene label of I_c.

Figure 10 illustrates several examples for some selected clusters.

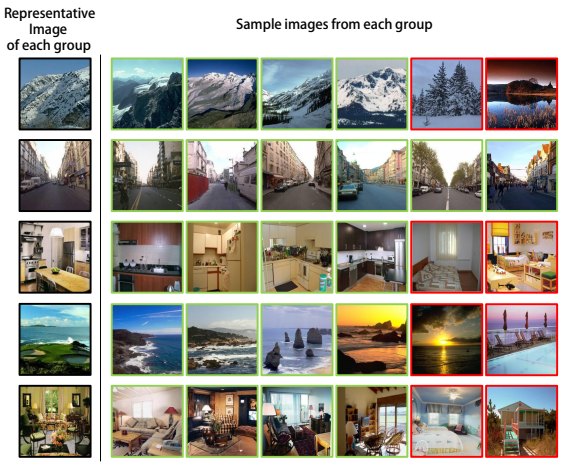

Figure 10. Automatic scene discovery example. Each row denotes one cluster, with the representative image which is closest to the cluster center listed on the left. The rest part of each row lists several other images in the same cluster, which are predicted the same scene label as that of the representative image. Most of the images in a cluster get the correct scene label prediction (in a green frame), a few of them get the wrong predictions (in a red frame).

Now, we can compute an accuracy of the test set, which compares the predicted scene label to its ground truth for each image. Table 1 gives a comparison among results obtained by using different feature vector of an image. We can see the

Used feature vector	Scene label prediction accuracy
OB feature	68.5%
OB feature with PCA	62.2%
OG feature	75.1%

Table 1. Accuracies for scene label prediction via different feature vector representations.

improvements made by using the OG feature representation of the input images, compared to classical OB feature representation and modified OB feature with PCA.

5. Conclusion

In this paper, we introduced an object group (OG) model for scene studying. Empirical results show that the learned OG templates are consistent with human's common sense. Using OG features produced by OG templates, we achieve state-of-the-art performance in scene discovery task.

References

[1] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer VIsion and Pattern Recognition (CVPR)*, 2010.

[2] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *Journal of Artificial Intelligence Research*, 29, 2007.

[3] F.-F. L. Hao Su, Haizi Yu. Learning object group for scene recognition. 2012.

[4] Z. C. S. B. A. N. J. Ngiam, P. Koh. Sparse filtering. 2011.

[5] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. 2006.

[6] E. P. X. Li-Jia Li, Hao Su and L. Fei-Fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[7] N. Loeff and A. Farhadi. Scene discovery by matrix factorization. 2008.

[8] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. 2005.

[9] S. Z. Yuanqing Lin, Tong Zhang and K. Yu. Deep coding networks. In *Advances in Neural Information Processing Systems*, 2010.