Object Retrieving from Image Database

Bao Nguyen Thien † , Yoshiaki Shirai ‡

Graduate School of Information Science and Engineering, Ritsumeikan University

Noji-Higashi 1-1-1, Kusatsu, Shiga, 525-8577 Japan

E-mail: †thienbao@i.ci.ritsumei.ac.jp, ‡ shirai@ci.ritsumei.ac.jp

Abstract: This research focuses on developing a system that can retrieval objects from large image database by exploring the types of image features necessary for recognition of common objects in scene. Then we make global representation for these features that can be used in learning. After that, we figure out a novel method for generic object detecting in still images with automatically choosing feature. Our method is simple, computationally efficient and bases on features that is easily seen by naked-eye and very close with natural detecting by human. The main advantage of this method is that it can automatically choose features which are best for detecting one type of object. We present experimental results for detecting many visual categories including side view car, front view car, bike, motorbike, train, aero plane, horse, and sheep. Results clearly demonstrate that the proposed method is robust and produces good detection accuracy rate.

Key words: Object Detection, Image Processing, Computer Vision

1. Introduction

The amount of visual information available in digital collections increases every day. Pictures, medical images, graphs and photos among others are collected for books, papers, diagnosis, reports, news, albums, etc. Due to this, Content-Based Image Retrieval (CBIR) has recently become an active research discipline. Despite the advances in content-based image retrieval, different problems remain unsolved. The main problem is the semantic interpretation of the image content [6]. Most of CBIR systems try to work on semantic of image. But the semantic is not easy to get. There are many solutions for understanding the content of image. The easiest way is to describe image with text – annotation. There are different situations in which images are surrounded by text descriptions that may provide contextual hints or semantic information about their contents. In this approach, image semantics are learned from an external source such as user's feedback and textual annotations, and of course both are provided by human beings. However, learning from external sources is limited because we don't know the annotation is correct or not. Moreover, there are different scenarios in which descriptive annotations are not available for all images in the collection or the user is not able to provide an accurate textual query. Furthermore, several works have shown that, even in the presence of text descriptions, visual features are a very useful information source to identify relevant images in a large scale CBIR system, showing important improvements in performance [8,10].

In this research, to understand the semantic of image, instead of annotation, we base on objects in the image. The more objects a system can detect, the more precise that system is. Object detection and recognition is still a difficult problem for computer vision. Most methods base on features of an image including both local and global one for detecting object. The goal of this research is to develop a necessary methodology for recognition of general object in still images by automatically choosing features. We base on features that are easily with natural recognizing by human. Most state-of-art methods focus on how to recognize object after fixing some features. In contrast, in this research, we let system choose features from a list of features, and the system determines by itself which features are good for detecting a given object.

The rest of this paper is organized as follows. Section **2** discusses previous work related to the approach taken here. Section **3** describes the set of attributes we use, and how we create attribute classifiers. In section **4** we propose an algorithm for automatically choosing best features. In our experiments (section **5**) we first evaluate the performance of both the individual attribute classifiers and combination attribute classifiers. Then we compare the attribute features with other methods. Result of automatically choosing features is also presented in this part. The last section discusses the results and suggests future directions.

2. Related Works

The most popular approach for object classification is to extract local regions from images, assign them to clusters, and use the count distribution across clusters as an input to a general classifier[4]. Some other approaches try to locate the object within the image, and to take into account the spatial relationship of the image regions which trigger potential matches[7]. These methods, however, tend to have high computational complexity. The 'bag of words' approach, instead, uses the histogram of counts across the whole image to predict whether or not the image contains the class of interest. While the image background may create confusion in recognizing object classes, the background can also provide useful cues to aid recognition [9]. Simple bag-of-words methods have shown impressive performances for object classification when used with large number of region descriptors and optimized parameters [11].

The local features used in bag-of-words methods typically lack any clear semantic meanings. The individual features do not usually have strong discriminative power, and the methods perform best when provided with very high-dimensional image descriptors. These descriptors allow the discovery of significant differences between different classes' feature distributions. Learning features with more specific meanings might help improve classification performance. Some previous work has looked at explicitly learning semantically meaningful features. For example, van de Weijer et al. [12] learnt to map from image color to the color names people use to describe objects. Ferrari and Zisserman [7] also learnt simple texture attributes such as 'stripes' and 'dots'. Recent work has embraced more complex attributes. Vogel and Schiele [13] used attributes describing scene, material, and shape to retrieve images of coasts, rivers/lakes, forests, plains, mountains, and sky/clouds. Farhadi et al. [14] used a set of semantic attributes such as 'hairy' and 'four-legged' to identity familiar objects, and to describe unfamiliar objects when an image and bounding box annotation is provided. Lampert et al. [15] showed that high-level descriptions in terms of semantic attributes can be used to recognize object classes without any example images, once semantic attribute classifiers are trained from other classes' data.

We also use diverse semantic attributes with explicit meanings to describe the visual content of images. Notable differences include that most of these approaches based on both local and global features

difficult to see with human eyes. In our method, we base on features that are easily seen and very close with human. Moreover, these attributes provide contextual information which is important for object classification. Beside, we do not use any manual semantic attribute labels for images, instead collecting a separate set of representative images. Our system automatically decides which feature is good to recognize for a specific object. The classifiers learnt in this way are more general than all other approaches which fix features before detecting.

3. Object Detection Based on Combination of Multi Features

In the first stage, we try to detect object by combination many features including both global and local features. Up to present, we have tested with seven features, such as: edge, corner, HoG, line, circle, SURF, and color. This section describes how to extract and use these features for recognize object.

3.1 Edge, Corner and HoG Feature

It is easy to see that the most important features for recognition object are the overall shape of that object and all boundaries which separate main parts of object. Shape or boundary is basically composed by *edges*. These edges are arranged in a specific order and meet each other at some points called *corners*. Beside, edge orientation also plays a prominent role for figuring object. With the same number of edges, but in different order and different orientation, it makes viewer imagine different objects. The description of how these principal components make object imagination is in Fig. 1.

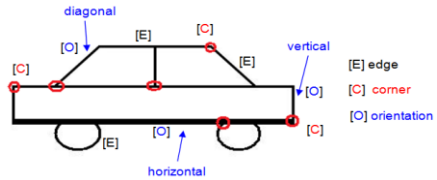

Fig. 1. *Three main components make object be figured.*

Based on this, we propose an overall approach for object detecting which focuses on edge, corner and edge orientation (HoG). More detail is presented in the next diagram in Fig. 2. But there is still an issue in this model. That is how do we know that edge/corner is at right position or not? In order to answer this question, we must point out the place where edge/corner must belong to before detecting object. Actually, there are many way to specify location of edge/corner, such as using prior knowledge, or machine learning…

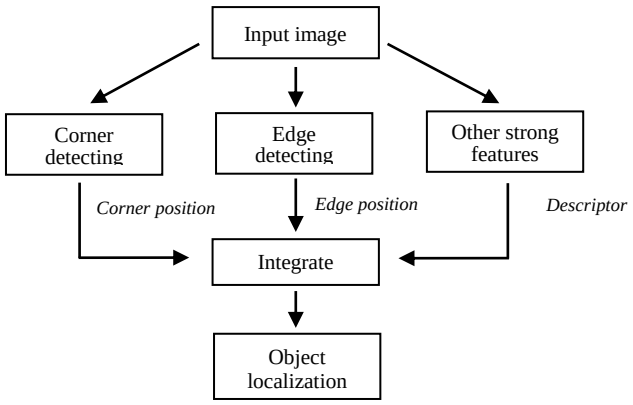

Fig. 2. *Model for object detecting based on multi-feature.*

In our implement, we choose machine learning approach because this method is more general for a lot of objects and one most important reason is that when using this method, user doesn't have to know more detail about object that they want to detect.

Edge and corner detection

Up to present, there are a lot of implementations of edge/corner detection and get the result with a high accuracy rate. Many of them have been reported in public. Zuniga and Haralick fit a continuous surface over a small neighborhood of each point and consider the rate of change in gradient direction. Moravec defined "points of interest" as points where there is a large intensity variation in every direction. Harris and Stephens used image derivatives to estimate the autocorrelation of the image. Rangarajan, Shah and Brackle found an optimal function representing the corner detector which when convolved with the gray level function yields a maximum at the corner point. Rafajlowicz, E. uses the idea of vertically weighted regression and in its simplest form it leads to interpreting SUSAN in terms of a box sliding on the surface of an image. This modification of the SUSAN algorithm is still simple, robust against errors and provides thinner edges, without further efforts on additional thinning. Canny introduces the notion of non-maximum suppression, which means that given the pre-smoothing filters, edge points are defined as points where the gradient magnitude assumes a local maximum in the gradient direction. And in 2007, Sonya Coleman *et al* [16] proposed a new method for enabling edge and corner detection to be integrated with a significantly reduced computation time.

In our performance, we have tried with many method for edge/corner detecting. And it shows that Canny detector gives the best result for edge detection, while Harris detector works well with corner detection.

Edge orientation detection

Specifying orientation of each edge is not an easy task. Because, with all edge detection method above, we only get the total edge image not each separate edge in image. So, it is almost impossible to know the orientation of each edge due to we don't specify each edge individually. Instead of that, we can calculate the domain orientation of all edge in a specific sub-image. From this, we can define relatively the edge orientation. The domain orientation of a region can be calculated with HOG descriptor method (Histogram of Oriented Gradient). HOG descriptor is a local statistic of the orientations of the image gradients around a keypoint. HOG descriptor was initially proposed by Lowe in his Scale Invariant Feature Transform (SIFT) [17]. Several HOG-based algorithms have been recently presented [2] and combined with technologies such as boosted classifiers [19]. Navneet Dalal and Bill Triggs (INRIA) have used this method for human detection with a high accuracy rate [19]. David Monzo *et al* [18] compare HOG-EBGM vs. Gabor-EBGM and show the result that HOG has a better performance.

Making edge map and corner map

Edge map/corner map is used to know if edge/corner is at right position or not. To make edge map and corner map, we use the same method as Zhenfeng Zhu *et al* in [3]. For all images in the positive training-image set **($\mathbf{T_{pos}}$)** (the rest of training-image set is negative, **$\mathbf{T_{neg}}$**), after detecting edge/corner we accumulate them in one temp image T_e or T_c. Then edge map and corner map will be made from these temp images respectively. However, [3] makes edge map and corner map be binary images. Our performance shows that with gray scale image, the result will be better. With a binary map, the location of edge or corner must be fixed at that position. But, due to variant of scale or view point, the location of edge or corner cannot be fixed at one specific point, other while it can be swung in a small region. So, gray scale map can work better in this case.

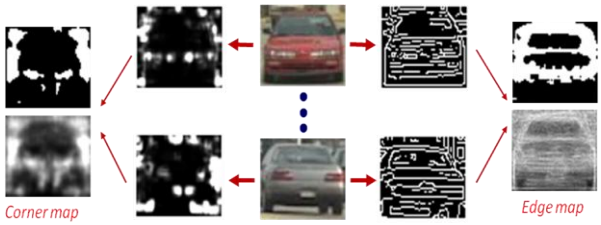

Fig. 3. *Making edge map and corner map from training set.*

The edge map $M_E(x,y)$ and corner map $M_C(x,y)$ are constructed from T_e and T_c as

$$M_E(x,y) = \frac{1}{\theta_1 * N} T_e(x,y), \qquad M_C(x,y) = \frac{1}{\theta_2 * N} T_c(x,y) \quad (1)$$

where N is the number of images in the training set, and θ_1, θ_2 are specific thresholds. Here, threshold θ_1 equals to 25 and value 35 is for θ_2. $M_E(x,y)$ and $M_C(x,y)$ denote for edge map and corner map respectively.

3.2 SURF Feature

In order increase the accuracy rate, after a candidate satisfies edge map, corner map and orientation of edge, that candidate continue to be passed to SURF [2] checking stage. We choose SURF because it is invariant with scale, illumination, and similar but faster than SIFT.

Fig. 4. *Making vocabulary for matching SURF feature.*

Method of matching between two set of SURF descriptors as Gabriella *et al* in [20, 21] is used. Bag of keypoints or vocabulary are made by quantizing all descriptors from training image set to K bins. Once descriptors have been assigned to form feature vectors, we use Naïve Bayes classifier to determine whether an input image belongs to one category or not by taking the largest posterior score, argmax{P(C_j| I)}

$$P(C_j \mid I) \quad \alpha \quad P(C_j)P(I \mid C_j) = P(C_j)\prod_{t=1}^{k} N^{(t,I)} P(v_t \mid C_j) \quad (2)$$

where I is new image, C_j is category class (object/non-object), v_t represents for keypoint or cluster center, and $N^{(t,I)}$ is the number of times that keypoint v_t occurs in image I.

3.3 Color Feature

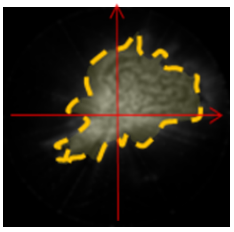

Fig. 5. *Horse color map*

With some objects such as horse, sheep, their color changes not much. Color is then a good point to distinguish object and non-object. We mark which color belongs to object by making color map from training images after converting to HSV space. Follow is an example of horse color map from all HSV images. Inside the yellow boundary is horse-color region, other while outside is non-horse color region. Color matching is calculated by

$$M_{color} = \sum_{x,y} CM(x,y) * H(x,y) \quad (3)$$

which CM is color map and H is histogram of image in HSV space. After extract feature, in order to determine feature is used for detecting a specific object or not, we should know if that feature *satisfies* object. Value of feature *f* in an image *i* is written as *val(f,i)*.

Definition 1: With a given image *i*, one feature *f* is called "***satisfy"***, written ***sat(f,i)***, if value of feature *f* in image *i* falls into range of all values of positive images(T_{pos})

$$\textbf{sat(f,i)}: \arg\min_{i_{pos} \in T_{pos}}(val(f, i_{pos})) \le val(f,i) \le \arg\max_{i_{pos} \in T_{pos}}(val(f, i_{pos}))$$

Definition 2: A given image *i* is called ***"satisfy"*** a feature set F, written ***sat(F,i)***, if image *i* "*satisfy"* all features of set F

$$\textbf{sat(F,i)} \ : \ \forall f \in F, sat(f,i)$$

Definition 3: With a given image set I and a specific feature set F, ***sat(F,I)*** is a set of image, member of T that "satisfy" all features of feature set F

$$\textbf{sat(F |I)} \ = \ \{i \in I, sat(F,i)\}$$

We only describe Edge, Corner, HoG, SURF and Color feature. With Circle feature, Line feature we use Hough Transform to extract.

In next section, an algorithm for automatically determining which feature to be good to recognize a given object is presented.

4. Automatically Choosing Feature Algorithm

After detecting many object categories based on features which were chosen manually, we try to let system automatically find which feature is suitable for every type of object. Automatically choosing feature is the most advatage of our system. In most of the other state-of-art recognizing object systems, the designer must define features which are used in their system before detecting. This is a limitation because the system has to work with one or some specific features, while other strong features for object cannot be used. Due to this, automatically choosing feature is a great development of this research. We work on seven features: edge, corner, line, circle, HoG, SURF, and color. Following is our proposed algorithm to let system determine features to be useful for detecting object.

Init: $F_C = \Phi$, set of chosen features
F ={f_i} = {edge, corner, line, circle, HoG, SURF, color }

Step 1: for every feature $f_i \in F$,
$g(f_i,F_C) = \omega_p * pos(f_i,F_C) + \omega_n * neg(f_i,F_C)$
where: $pos(f_i,F_C) = |sat(F_C \cup \{f_i\})|T_{pos})|*1/|T_{pos}|$
$neg(f_i,F_C) = (|T_{neg}| - |sat((F_C \cup \{f_i\})|T_{neg})|)*1/|T_{neg}|$
|A|: cardinality of set A.
ω_p, ω_n *:* bias of positive and negative score.

Step 2: $F_C = F_C \cup \{f_j\}$, with $f_j = \underset{f_i}{\text{argmax}}\{g(f_i, F_C)\}$
$F = F \setminus \{f_j\}$

Step 3: if ($F = \Phi$ || argmax{$g(f_i,F_C)$>δ) stop
else: goto step 1

Fig. 6. *Algorithm for automatically choosing feature*

The larger threshold δ is, the more precise system is, and also the more time it takes for training. Bias ω_p and ω_n are weight of positive and negative score in the total satisfied score for every feature. The basic idea of this algorithm is "best-fixed" approach. From the set of feature, we calculate the score for every feature, and then we choose the one with highest score. The chosen feature will be put in F_C (set of chosen features). After that, we will again estimate the satisfy-score for every feature left. This time the satisfy-score bases on combination of every feature left with all features of F_C. It means that all the best features of the previous running are fixed for this running time. It will be looped until there is no any feature left or until we get a good result (the argmax is larger than threshold δ). According to our experiment, it is difficult to run out of feature set F, we should stop when it reaches a peak of satisfied score.

5. Experimental Results

In this section we first evaluate the performance of the object detecting by combination multi-features, and show how they can be used to predict class for new images (Section 5.1), then we compare our method with other detecting methods base on the state-of-art result of PASCAL 2009. Finally, the evaluation of automatically choosing feature for detecting new object is presented in section 5.2.

5.1 Detection Result

With an input image, we will make edge image- I_E and corner image-I_C. To extract edges for making edge image, we use canny edge detector with 3x3-structure. And to consider the position of corners, Harris corner detector is used with 5x5 structure. Beside, Gaussian filter is applied on input image before extracting edge and corner features to subtract background.

Edge map satisfaction

Edge image after detecting edge from input one is denoted as I_E. Let's define two parameters:

n_1^e : number of edges in the input edge image I_E.

n_2^e : number of edges matching between I_E and M_E.

Degree of satisfying M_E of I_E is determined by some conditions as bellow:

$$C_1^E(I_E) = \begin{cases} 1 & n_2^e > N_1^e \\ 0 & else \end{cases} \quad (4)$$

$$C_2^E(I_E) = \begin{cases} 1 & \frac{n_2^e}{n_1^e} > \theta^e \\ 0 & else \end{cases} \qquad C_3^E(I_E) = \begin{cases} 1 & n_1^e < N_2^e \\ 0 & else \end{cases} \quad (5)$$

where N_1^e , N_2^e are constants and θ^e is a given threshold. $C_1^E(I_E)$ guaranties that edges in the input image must be in correct position to construct object like edges in M_E. While both $C_2^E(I_E)$ and $C_3^E(I_E)$ is to avoid situation that there are so many edges in the input image. So, if $Q_E = C_1^E(I_E) \cap C_2^E(I_E) \cap C_3^E(I_E)$ (6) is true, then I_E satisfies M_E, and the image will be passed to the next corner test step.

But, as we mention in section 3.1, after detecting edge we cannot know exactly individual edge. It also means that we cannot calculate n_1^e and n_2^e. However, instead of counting number of edges, we can sum of pixels which are marked as edge. Thus, n_1^e and n_2^e can be known approximately as

$$n_1^e \approx \frac{1}{255}\sum I_E(x,y) \qquad n_2^e \approx \frac{1}{255\times 255}\sum I_E(x,y)\cdot M_E(x,y) \quad (7)$$

here, because both I_E and M_E are gray scale, we should divide the sum to 255 for each image.

Corner map satisfaction

Similarity to the edge test step, in this stage we also define three conditions:

$$C_1^C(I_C) = \begin{cases} 1 & n_2^c > N_1^c \\ 0 & else \end{cases} \quad (8)$$

$$C_2^C(I_C) = \begin{cases} 1 & \frac{n_2^c}{n_1^c} > \theta^c \\ 0 & else \end{cases} \qquad C_3^C(I_C) = \begin{cases} 1 & n_1^c < N_2^c \\ 0 & else \end{cases} \quad (9)$$

where N_1^c , N_2^c are constants, θ^c is a given threshold, n_1^c is the number of corners in the input corner image I_C and n_2^c is the number of corner matching between I_C and the corner map image M_C.

$$n_1^c \approx \frac{1}{255}\sum I_C(x,y) \qquad n_2^c \approx \frac{1}{255\times 255}\sum I_C(x,y)\cdot M_C(x,y) \quad (10)$$

Then, qualification of whether the input image satisfies corner map or not is $Q_C = C_1^C(I_C) \cap C_2^C(I_C) \cap C_3^C(I_C)$ (11)

Orientation satisfaction

From the input image I(x,y), we find the orientation of each pixel by using HOG descriptor method as in [2]. The magnitude m(x, y) and the orientation θ(x, y) are computed by

$$m(x,y) = \sqrt{g_x(x,y)^2 + g_y(x,y)^2}$$
$$\theta(x,y) = \tan^{-1}(g_y(x,y)/g_x(x,y)) \quad (12)$$

where $g_x(x, y)$ and $g_y(x, y)$ denotes the x and y components of the image gradient in x and y direction calculated by 1-D centered mask [−1, 0, 1]

$$g_x(x, y) = I(x + 1, y) - I(x - 1, y) \quad (13)$$

$$g_y(x, y) = I(x + 1, y) - I(x - 1, y)$$

After that, we specify some sub regions of object where the orientation is so strong such as horizontal, vertical and diagonal.

Fig.7 *Sub regions with strong orientation of front/rear view car, from left to right: edge image, horizontal, diagonal and vertical.*

The orientation is ranged from [0 − 2π], divided into 16 bins: *h[0]*... *h[15]*. For each sub regions, we calculate histogram of orientation in that region by quantizing the orientation $\theta(x, y)$ for all pixels falling to *h[i]* bins, $i = \overline{0,15}$, weighted by its magnitude $m(x, y)$.

Condition for input image satisfies orientaiton is that

$$H(S) = \begin{cases} 1 & \frac{h[0]+h[15]+h[7]+h[8]}{\sum_{i=0}^{15} h[i]} > \sigma_1 \\ 0 & else \end{cases} \quad (14)$$

$$V(S) = \begin{cases} 1 & \frac{h[3]+h[4]+h[11]+h[12]}{\sum_{i=0}^{15} h[i]} > \sigma_1 \\ 0 & else \end{cases} \quad (15)$$

$$D(S) = \begin{cases} 1 & \frac{h[1]+h[2]+h[9]+h[10]}{\sum_{i=0}^{15} h[i]} > \sigma_1 \\ 0 & else \end{cases} \quad (16)$$

where σ_1, σ_2, and σ_3 are specific thresholds. If all sub-regions satisfy condition of the strong orientation respectively then input image satisfies orientation.

Our proposed scheme is evaluated on eight object-categories, including front/rear view car, side view car, bicycle, train, aeroplane, motorbike, sheep and horse. We use CALTECH256, UIUC and PASCAL-VOC 2010 database. Beside, in order to evaluate the result more exactly, an additional about 1500 negative examples are collected randomly and added to the database. Thus with each object-category, the whole database contains about more than 1500 negative images and 1500 positive images (including 1000 images for training, and 500 images for testing).

The database contains natural images that are taken from several sources and include occlusion and cluttered backgrounds. With side view car, images used for training have the same size 40x100 as in original database. Otherwhile, all images in the training set of front/rear view car, we resize manually into 60x80 to eliminate the influence from background. Similar to other object, when training, we have to choose the basic size for one object. Then in detecting phase, with an input image, we preprocess it by resizing it into four times as size of training image.

Fig. 8 *Example of correct detection with front/rear view car*

After detecting at this scale, we reduce it size and continue detect until it's size is equal to training iamge. This work is to be able to detect all object in the input image even if the size of object in it is smaller or larger than the basic size, although it consumes more time than detecting at a specific scale.

The final result of our system for car detection is shown in Table 1. The performance of Zhenfeng Zhu [3] (with the same database: Caltech and UIUC) reaches the highest accurate rate at 90.5%, while the lowest accurate of our sytem is about 91.67%. This shows that

our schema has better perfomance. For space consideration only some examples of correct/incorrect detection are shown in Fig.9 and Fig. 10. The last image in Fig. 9 presents the positive false of bicycle, while in Fig. 10 it is the negative false of areo plane.

Set	No. image	Correct	In-correct	False rate	Accuracy
Calt 101	504	481	23	4.57%	95.43%
Calt 256	526	475	41	8.95%	92.05%
Non- car	665	624	41	6.17%	93.83%
Detection result of front/rear view car					
UIUC1	300	275	25	8.33%	91.67%
UIUC2	278	264	14	5.04%	94.96%
Non-car	665	638	27	4.06%	95.94%
Detection result of side view car					

Table 1. *Result of our car detecting system*

We use two ways to check if edge/corner pixel is at right position or not. One is that if no-pixel in map and no-pixel in mage, then result is not right (0x0=0); other is that no-pixel in map, no-pixel in image, the result is between right and not-right (0x0=0.5). Because if there is no-pixel in map and no-pixel in testing image, it mean that there is no edge/corner at that position. That is also a good hint for regconize object. The final average precision (AV), average recall (AR) of these two methods is shown in table 2.

Fig. 9. *Example of bicycle detection result*

Fig. 10. *Example of areoplane detection result*

If there are many edge/corner inside object (such as aero plane, motorbike), then the 0x0=0 method is better. Otherwhile, if object doesn't include a lot of edge/corner (bike, horse, sheep or animal is example), the 0x0=0.5 method will get the higher perfomance. In the case there are a few of edge/corner on the surface of object (ex. train), both 0x0=0 and 0x0=0.5 do the same function, and the result is not different so much.

Object	0x0=0		0x0=0.5	
	AP	AR	AP	AR
F/r car	97.40%	89.71%		
Side car	95.83%	92.38%		
Bike	83.76%	72.64%	83.82%	78.46%
Train	84.05%	74.10%	81.57%	76.43%
Aero plane	85.59%	87.56%	81.15%	86.54%
Motorbike	95.63%	85.35%	95.35%	84.47%
Horse	88.64%	68.78%	88.37%	70.72%
Sheep	74.84%	65.96%	86.13%	71.93%

Table 2*. Comparison between 0x0=0 and 0x0=0.5*

5.2 Automatically choosing feature for detecting

In our experiments below we evaluate the performance of automatically choosing semantic attribute features on the large data set used in the 2009, 2010 PASCAL VOC challenge. We believe this is the first time that the performance of automatically choosing features has been tested on a standard object classification benchmark.

In previous section, our system can detect eight categories with satisfactory result. We combine many features for every object. But which one category, we must decide which feature is used for detecting by ourselves. For example, to detect a bike, circle feature is a good choice, because most of bikes have two wheels with circle shape. With horse or sheep, we can't work with circle, but color is a helpful hint, due to there is not much change in color space between this horse and the other horse. But, how about SURF or HoG feature? Because these features are not visual, it is not easy to determine they are suitable for one object or not. In this part, we will evaluate our proposed method in section 4 for choosing feature without handling. We first test with seven features to describe images: 'edge', 'corner', 'HoG', 'circle', 'line', 'SURF', and 'color'. Bias ω_p and ω_n are equal to 0.6 and 0.4 respectively.

Object	Edge	Corner	Line	Circle	HoG	SURF	Color
F/r car	1	1	0	0	1	1	0
Side car	1	1	0	1	1	1	0
Bike	1	1	1	1	0	0	0
Train	1	1	0	0	1	1	0
Aero plane	1	1	1	0	0	1	0
Motorbike	1	1	0	0	0	1	1
Horse	1	1	1	0	0	1	1
Sheep	1	1	0	0	0	1	1

Table 3*. Object with automatically chosen features*

Result of automatically choosing features is in table 3, where zero(0) means that feature is not good for object, and one(1) represents we should use this feature to detect object. The image database for testing automatically choosing feature is same with the image database for object detecting in section 5.1. The outcome of this algorithm is predictable, it is almost fit with what we have done by hand. Only one thing is not as we though is that color feature is useful for motorbike object (value 1 at the cell of column Color and row Motorbike). After checking the motorbike database, we recognize that most of motorbike-images from CALTECH 256 have white background and their colors only vary in green, brown, red. That is the reason why with this database, color is helpful for detecting motorbike object.

Fig. 11. *Satisfied scored for automatically choosing feature*

In the fig. 11, fig. 12, detail of automatically choosing feature for horse object and side view car object is described. From set of features, we choose one feature with the highest satisfied score (calculated as in section 4). We combine every feature with the first chosen feature, and then we recalculate the satisfied score. We will pick the feature which makes the satisfied score maximum. This feature then is moved to chosen feature set. Next running time, both two chosen features are integrated with new one to compute new satisfied score. By doing this way, the satisfied score increases slowly until it rises to a superior value, which is the maximum value. After that, if we combine more features, result will be worse. If the score is lower than previous in three running times, we will stop. Our algorithm will reverse to the peak, and only these features before gaining the top will be chosen for that object. For example, with horse object these are chosen-features: CL (color), C (corner), L (line), E (edge) and S (SURF). Other while, with side-view car, H (HoG) is the best feature following by E (edge), CR (circle), E (edge) and C (corner).

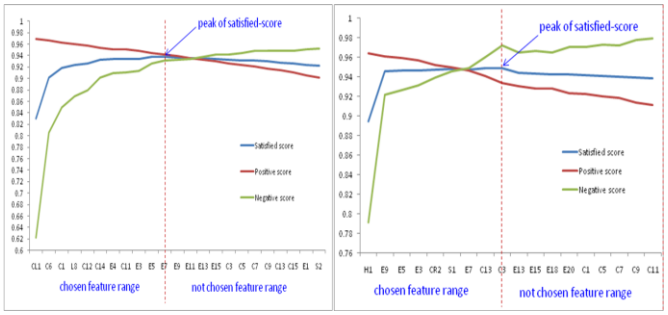

Fig. 12. *Satisfied, positive and negative score of (left) horse object and (right) side view car*

With the first feature, generally *positive score* $pos(f_i,F_C) = |sat(F_C \cup \{f_i\})|T_{pos}|*1/|T_{pos}|$ is very large because most of image in T_{pos} satisfy this feature. Other while, the *negative score* $neg(f_i,F_C) = (|T_{neg}|-|sat((F_C \cup \{f_i\})|T_{neg})|)*1/|T_{neg}|$ is small. When the more features are added to chosen feature set, the smaller positive score is, and the larger negative score becomes. Totally, the *satisfied score* $g(f_i,F_C) = \omega_p*pos(f_i,F_C) + \omega_n*neg(f_i,F_C)$ (with $\omega_p = 0.6$ and $\omega_n = 0.4$), get larger until it reach the prime value, represented by red line. After this peak, if more features are integrated into F_C, the satisfied score gets lower. After the satisfied score becomes worse k times (in our evaluation, k = 3), the system stops and reverses to get the prominent value (at red interrupted-line). Left side of this red line are chosen features and these features not good for object are on the right side.

6. Evaluation and Future Work

Taking into account that both local and global features may provide useful information to recognize object, the combination of multi features to construct a system gets a good performance. We evaluate a large image database and the outcome is satisfactory. Compared with PASCAL2009 result, the consequence of our research is better. However, there are some restrictions. If the input image is blurring or object is occuluded, it cannot be detected. If there are edges and corners arranged similar to the object, our shema knows that it is the object to detect.

Between specific size object (car, bike, ...) and unspecific size object (train, ...), detecting result of the unchanged-size object is better. In table 2, train detecting result is the lowest, because train has no specific size. It can not make properly Edge map and Corner map. But when combining with SURF, it gets better result. SURF feature works well with object having "large surface" (plane, train). In contrast, for example with bike object, almost SURF features fall into background and do not belong to object. With such object, SURF feature is not good. Color feature is only suitable for natural objects with small change in color such as sheep, horse, or animal.

Under the proposed framework, different with other state-of-art image retrieve systems, we also set up an algorithm for automatically choosing feature. This algorithm does not only determine which feature is good for which object, but it also unintentional compute the threshold value for each feature. Up to present, this is the first proposed algorithm that can do both of these tasks. But in this research, we only use *"and"* operator $(f_i \wedge f_{i+1} \wedge \ldots \wedge f_{i+k})$ for combining features. What does happen if both *"and"* and *"or"* operator $((((f_i \wedge f_{i+1}) \vee f_{i+2}) \wedge \ldots \wedge (f_{i+k-1} \vee f_{i+k}))$ take part in detecting? With *n* features and only two fundamental operators $(\wedge,\vee)$, there are about 2^{n-1} combinations. It is really a huge number, and we cannot check all the cases due to large running time. We should find a suitable way to combine with both two basic operators, but in a limited running time. This is truly a great challenge for automatically choosing feature. If we can solve this problem, the accuracy of our method will become more believable.

In this study, we combine many features to detect object. We also build an automatically choosing features system. It needs more experiments to appraise the accuracy of our method. Especially, in automatically choosing feature algorithm, we only use one operator (*"and"*) for combining features. How to use both *"and"* and *"or"* operators in feature combination is also one important future task.

7. References

[1] H. Harzallah, F. Jurie, C. Schmid, "Combining efficient object localization and image classification," *ICCV09*, Kyoto, Japan, 2009.

[2] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *ECCV06*, Graz, Austria, May 2006.

[3] Zhenfeng Zhu., J. Uchimura, "Car detection based on multi-cues integration," *ICPR04*, Cambridge, UK, Aug. 2004.

[4] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: a comprehensive study," *IJC V07*, 73(2), 2007.

[5] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, 106(1), 2007.

[6] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 12, pp. 1349-1380, 2000.

[7] V. Ferrari and A. Zisserman, "Learning visual attributes," *Advances in Neural Information Processing Systems*, 2008.

[8] Y. Jing, S. Baluja, and H. Rowley, "Canonical image selection from the web," *the 6th ACM international conference on Image and video retrieval*, New York, NY, USA, 2007, pp. 280-287.

[9] J. Uijlings, A. Smeulders, and R. Scha, "What is the spatial extent of an object?," *IEEE Computer Vision and Pattern Recognition*, 2009.

[10] T. Arni, P. Clough, M. Sanderson, and M. Grubinger, "Overview of the ImageCLEFphoto 2008 photographic retrieval task," *Working Notes for the CLEF 2008 Workshop*, 2008.

[11] Marcin Marszałek, Cordelia Schmid, Hedi Harzallah, and Joost van de Weijer, "Learning object representations for visual object class recognition," *ICCV07*, October 2007.

[12] J. Van Deweijer, C. Schmid, and J. Verbeek, "Learning color names from real-world images," *IEEE Computer Vision and Pattern Recognition*, 2007.

[13] J. Vogel and B. Schiele, "Natural scene retrieval based on a semantic modeling step," *International Conference Image and Video Retrieval*, 2004.

[14] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," *IEEE CV PR*, 2009.

[15] C. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[16] Sonya Coleman, Bryan Scotney, and Dermot Kerr, "Integrated Edge and Corner Detection," *International Conference on Image Analysis and Processing*, Modena, Italy, Sep 2007.

[17] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *IJ CV, 60(2)*, pages 91–110, 2004.

[18] David Monzo, Alberto Albiol, Jorge Sastre, Antonio Albiol, "HOG-EBGM VS. GABOR-EBGM," *IEEE International Conference on Image Processing*, San Diego, CA, Oct. 2008.

[19] Navneet Dalal and Bill Triggs, "Histograms of Oriented Gradients for Human Detection," *IEEE Computer Vision and Pattern Recognition*, CA, USA, June 2005.

[20] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *IEEE Computer Vision and Pattern Recognition*, 2006.

[21] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray, "Visual Categorization with Bags of Keypoints," *Xerox Research Centre Europe*, 2004.

[22] J. C. Caicedo, A. Cruz, and F. Gonzalez, "Histopathology image classification using bag of features and kernel functions," *Artificial Intelligence in Medicine, AIME09*, vol. LNAI 5651, 2009.

[23] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. Multimedia Comput. Commun.* Appl., vol. 2, no. 1, pp.1-19, February 2006.

[24] J. C. Caicedo, A. Cruz, and F. Gonzalez, Histopathology image classification using bag of features and kernel functions," *Artificial Intelligence in Medicine Conference, AIME 2009*, vol. LNAI 5651,pp. 126-135, 2009.