Abstract

Object recognition has been researched and applied on many kinds of aspect, such as face recognition, pedestrian detection and vehicle detector. In this project, I will research and implement a classifier for animal recognition and image search. By comparing with ten feature extraction methods that are based on distributions of color, texture and shape cues, I will combine four appropriate methods, use K-means cluster algorithm to deal with several feature information, and achieve the classifier.

For color cue, I compare the methods of Random Subwindows, 9 blocks (8bins each channel) and 9 blocks (16bins each channel), see page 13-15. For texture cue, I try the methods of Gabor filter, LBP and a method fused with LBP and GLCM, see page 15-18. For shape cue, I compare HOG and Geometric Blur matching methods, see page 21-24. Besides the methods of these three cues, I add a sift feature to be the fourth cue and try SIFT and PCA-SIFT these two methods, see page 26-30.

The testing images are all on the web pages, after filtering, the classifier can retrieve the images in a new order based on their similarities to the animals. My project is closely related to T.L. Berg &D.A. Forsyth [1] in this area. T.L. Berg et al do an extensive work on the test cue which is associated with their collected images, but I focus more attention on the visual features and do a virtual improvement of their work.

In short, the research reported in this thesis consists of an evaluation of 10 methods and an improved classifier for re-ranking the ‘animal’ images on the web.

Contents

1. Introduction

When we type “monkey” on the Google text search, it will return about 368,000,000 results, when we type “bear” on the Yahoo text search, it will yield nearly 888,000,000 web pages. In these massive web pages, there are a large number of images. Parts of them portray the real animals, but other images may depict objects, people or other things which stand for the significance of these animals.

Figure 1: An image of monkey on the Wikipedia web site

Figure 2: An image shown on the web site of a monkey business design

Retrieving the animal images on the web is a hard work to achieve. Actually, in the first two web pages returned from the Yahoo text search “bear”, there are only 12 real bear pictures and other 46 are fake bear pictures. Most of the fake images are about Winnie the Pooh’s products. One hand, a quantity of noisy images would increase difficulty for filtering, especially some pictures of animal toys or designs. On the other hand, animal in pictures may have different pose, different scale, different rotation and different lightness. Additionally, a certain animal class would have several species, which would cause different fur or surface color. For example, polar bear and black bear both belong to bear, but they have opposite color, green tree frog and leopard frog are two species of frog, however, their skin textures are not the same.

Therefore, it needs a comprehensive classifier to determine whether a picture depicts an animal. In my project, I implement the classifier mainly based on color, shape, texture cues and the extremum point information. By combining these four cues, I could generally get a better result than that only uses one feature.

1.1 Object Recognition

Nowadays, object recognition has been a difficult problem in the area of computer vision, which attracts many scholars and experts to study and research. The main process of object recognition is to detect the interested object by applying relevant methods.

Whatever methods to be used for describing an object, it would accord with the following features [36]:

(1) Uniqueness:
Each object would have a unique description in order to distinguish other objects.
(2) Integrity:
The method can describe objects integrally.
(3) Invariance under the geometrical transformation:
The method would not be affected due to the change of position, size or rotate direction.
(4) Sensitivity:
The method could reflect the difference between similar objects.
(5) Abstractness:
The method can abstract useful feature information from a number of objects, that is to say, it would have a quality of high data compression.

There are many object recognition methods. Three different kinds are as following.

1) PS (Pictorial Structure) method, came up with P.Felzenszwalb [2], applied for evaluating the pose of people. In this method, a tree geometric structure applied to estimate the probability of body parts. Probability information about each part's position, size or orientation transfers from the leaf points to the roof point. At last the roof point can get a total probability.

2) The algorithm based on searching, used by G.Mori [3], is to detect and recognize a person. This method adopts an intelligent search strategy to find the interest object in the image.

3) The method combined with prior information and bottom feature. Prior information stands for an evaluation for the object by people. The bottom feature is on behalf of color, texture or other image features. This method needs a training set. Applying feature extracting on the training set and using the feature information to be matched by the testing set. My project is very

similar to this method.

Also, object recognition is a key challenge in many application projects like robot navigation, content-based image search, human-computer interaction and image annotation and so on. Research in this topic will have a significance prospect.

As for this project, it studies on the content-based image search. Specially, the range of content in my research is restricted to animal. In the field of image search, the mainstream technique applied by Google search engine is finding a picture through the text label. As a result, the text label is the information used to detect picture. However, in my project, I use the visual information to be the search feature of an image.

Visual information has a number of existing forms, for example, a single image, a video or the multidimensional data from graphic scanning. In order to recognition an object in the picture by visual information, an object recognition system could consist of four stages. They are as follows:

- Preprocessing.
- Feature extraction.
- Modeling.
- Matching.

1.1.1 - Preprocessing

In general, the purpose of preprocessing on an image is to make adjustment in color distribution, brightness distribution, size or noise. Although the preprocessing does not increase image information, it is very useful to suppress information which is not relevant to the analysis task. Preprocessing has many kinds, such as gray-scale transformation, geometric transformation, image smoothing and normalization [4].

Gray-scale transformation is used for transfering a color image to a grayscale image that relies on the brightness. Geometric transformation is used for rectifying some problems caused by the object position or system error. For example, if there are two different image of the same object, a geometric transformation could help them match. Image smoothing can be used for eliminating random noise of an image. Gaussian smoothing is a common method in this part and it will be used a number of times in this program. Normalization could make some features own invariance under the special conversion.

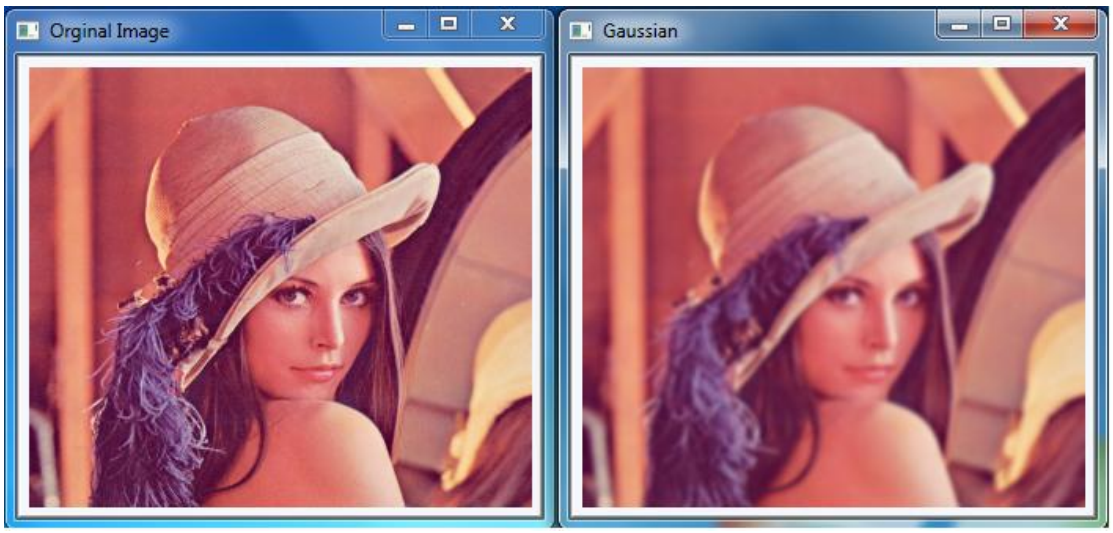


Figure 3: The effect of using Gaussian smoothing.

1.1.2 - Feature extraction

Feature extraction is the core and foundation of an object recognition system. It is also the main research orientation in this project. Object in an image would have distinct feature, which could be collected to represent the whole object.

The common extraction methods could be reduced to two types, Insert point detector and Region detector. Some detectors, such as edge detector Canny [5] and Dog (difference of Gaussian) [6], HOG descriptor [7], SIFT descriptor [8] and other feature descriptors have many application in different fields. In this project, I will compare 10 extraction methods on account of their different functions, see section 2 and 3. Generally, a good descriptor should satisfy affine invariant, which means an object under different point of views can be collected the same feature.

In recent years, some patterns in vector space have become important measure to gather feature, such as PCA [9] (Principal Component Analysis) and ICA [10] (Independent Component Analysis). These patterns apply few factors to represent high-dimensional data. The rest is treated as noise, namely irrelevant fluctuations. It can be applied to the whole image to purify feature information or to the acquired feature for obtaining the lower dimensional data. Vector space methods would have a good performance in extracting the whole image feature, but they are very strict to the training set. Images should be aligned manual in order to get a reasonable training effect.

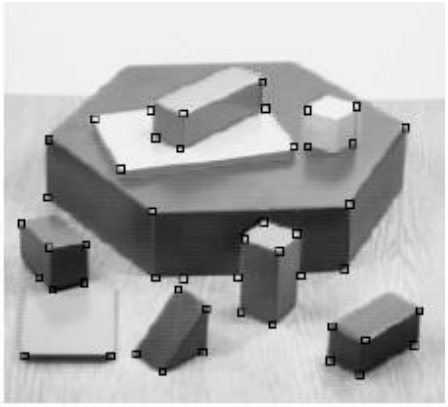


Figure 4: A corner detector, Harris algorithm. [11]

Figure 5: PCA applied on the face recognition. [12]

1.1.3 - Modeling

Some methods would include modeling. The main purpose of modeling is to find same points from the features set and to classify the features. For example, there are images of bird, cat and fish. After extracting the features from these images, it needs to distinguish which features belong to bird, cat or fish. Therefore, a good visual model is necessary. From the mathematical statistics view, visual model could be distributed into three classes, namely Descriptive Model, Discriminate Model and Generative Model.

- Descriptive Model: The model can reflect probability density between the observed features. Tree Model [13] is a typical descriptive model.
- Discriminate Model: This model focus on looking for the distinction between different classes. However, discriminate model has a disadvantage, that is, it needs to train the data set again in order to add new recognition ability to the system. SVM [14] belongs to discriminate model.
- Generative Model: This model is a little complicated, because it exist hidden variable, which is needed to compute the observed feature. A common model is constellation and trained by using EM [15] iteration.

Other models like LDA (Latent Dirichlet Allocation) [16], is a probabilistic model that blends some semantic meaning into the object. It could be used for dealing with a lot of text information. Also, it could work for the image information. An image is divided into several cells; each cell is a “word”. LDA will distribute the words into different “topics” based on their frequency.

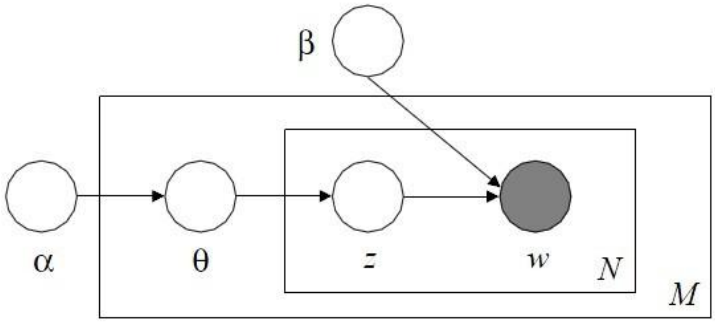


Figure 6: Graphical model representation of LDA. [16]

This project does not employ a model but a simple mathematical method to process the features, because it just needs to detect whether an image depicts animal. It re-ranks the image based on their similarities.

1.1.4 - Matching

Matching is a testing stage, including taking a test image to compare with the training data, then recognizing which object is contained in the image. In my project, each test image will be compared to the training images. After that, it will get a similarity used for re-ranking.

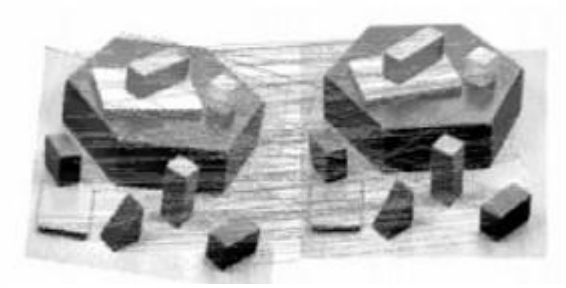


Figure 7: A matching between two images with different angle.

1.2 Aims and Objectives

T.L. Berg and D.A. Forsyth published a paper in 2006 [1] about how to find animal pictures on the web. They rely on text, color, shape and texture these four simple cues to identify images containing categories of animals. On the whole, their system is divided into four stages.

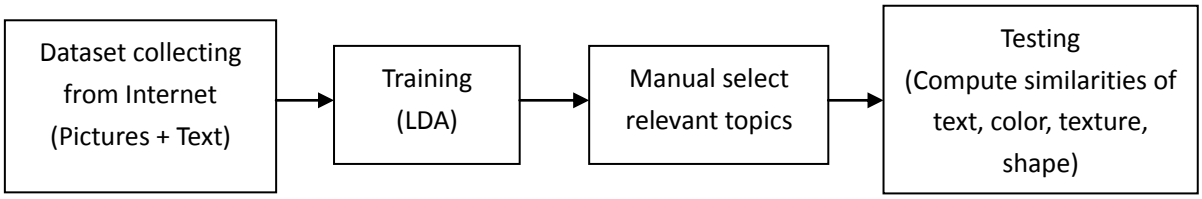


Figure 8: The flow chart of the animals search system in the previous paper

The first stage, it uses a clustering method that is applied to text on web pages to obtain images set. The collections of pictures are associated with text.

The second stage, also training stage, they use LDA [16] for filtering text information. By applying LDA, it could discover 10 latent topics for each animal category. Each topic will have a number of words and a probability distribution over the words. In order to rank the images, they select 30 exemplars for each topic based on the higher word likelihoods.

The third stage, it requires user to label each topic as relevant or background. That means, the user needs to identify which topics are the concept they want. This manual operation is necessary, because a word would have polysemy-like phenomenon. For example, “monkey” could refer to the animal as well as a band member. The pictures about band member are not the exemplars they want. Hence, to a large extent, it would reduce the precision of the images set which would be used for extracting samples features.

The fourth stage, also the testing stage, the system employs three feature extraction methods on color, texture and shape respectively. The features collected from the images of relevant topics would be considered as positive features. For each testing image, the system calculates sum of the four similarities of features matching positive features and re-ranks the testing images.

However, there are no comparisons to other extracting features methods. In addition, the operation on the features is a bit simple. Therefore, the aim of this project is to:

- Compare several features extraction methods and study their effects on the different animals set.
- Add some procedure to process the collected features.
- Explore an improved system to re-rank the animals images based on the visual information.

2. Comparison with Features Extraction Methods

The pixels distribution of a digital image can reflect features of the image context. It is an efficient approach for classifying the images based on the pixels distribution. As for the given object recognition, it can define the general feature description. For example, the main features on a person face include eyebrow, eyes, note, mouth and facial form. However, a good object recognition system should adapt as more object categories as possible. Additionally, single feature is difficult to describe an object. Taking several types of feature and integrating them is significant for a functional object recognition system. For the animals, color, texture and shape cues would be useful for depicting them. Except that, I add an extremum cue to help describe the animals.

2.1 Color Representation

Color feature is a kind of global feature, which describes a surface property of a picture. All pixels of the image would make contribution to the feature, as a consequence, it depend less on the size, orientation and angle of view than other visual features. Color feature is robust for the whole image, but it is general to detect the local feature of animals. There are many methods applied to represent the color distribution, namely color histogram, color set, color moments, color coherence vector and color correlogram.

Color histogram is widely used in many image search system [1] [17]. It describes the proportion of different colors in the whole image. A color histogram could be based on different color space, such as RGB and HSV. RGB color space means the space consist of red, green and blue. HSV color space indicates the space is made up of hue, saturation and value. Computing color histogram needs to divide the color space into several color regions and each region is a bin for the histogram. This process is called color quantization. By calculating the quantity of pixel in each bin, it can obtain the color histogram. Three feature extraction method of color compared in this project all use color histogram.

Color moments [18] are a simple color representation. It just needs nine components, that is, three color components and each color component have three parameters: mean, variance and skewness. Color moments could be used to narrow down the features.

Color set [19] is expressed to the binary feature vector. Therefore, it can use binary search tree to increase search speed, which could be efficient for massive

images set.

Color coherence vector [20] is an evolution of color histogram. The main idea of it is to split each bin (color histogram) in two parts. If the connected area of some pixels in a bin is bigger than a threshold value, all pixels in this area are coherence vector; otherwise the pixels are non-coherence vector. As a result, color coherence vector would contain some space information of color distribution.

Color correlogram [21] is a complicated representation. It not only indicates the proportion of certain color pixels, but reflects the space relevance of different color pairs.

For the animal pictures, color feature could do a reasonable favor. Not only the animals in some high-quality pictures would display a similar color, but also the colors of their surroundings are close.

2.1.1 – Random Subwindows

In some cases of object recognition, extracting features randomly is useful for mass training samples. Random Subwindows [17] is a feature extraction method which collects a random number of possibly overlapping square subwindows of random sizes and at random positions from training set. The minimum size of subwindows is 8*8, and the maximum size of it is the shorter side of the current image. I compute the color histogram in RGB space with 16bins each color channel. By using subwindows, it may cover a large quantity of possible pixel value. If the numbers of subwindows are enough and the training set is good, much useful information could be strengthen in the color histogram.

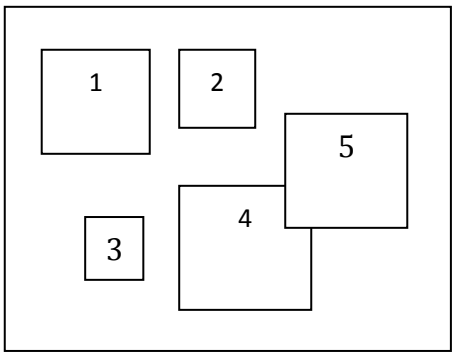


Figure 9: Five random subwindows and one overlap.

For the testing images, I set a subwindows ratio that equals to 150. The ratio used for determining the number of random subwindows, a 640*480 image would have about 1000 random windows. We could not pick random number of subwindows in testing image; they are different with training set. It would decrease precision

of ranking because the noisy information in real images.

2.1.2 – 9 blocks (8bins)

9 blocks indicates subdividing an image into 9 regions. 8 bins means calculating a normalized color histogram with 8bins per color channel. As a result, each image will extract 4608 (8*8*8*9) features. Divided into 9 blocks, the features could contain some space information. The advantage of 9 blocks is stability, it does not rely on much training exemplars. Useful color information from one or two pictures could be collected by this method.

2.1.3 – 9 blocks (16bins)

This method is so close to the just mentioned method. Only difference is that 8bins is changed to 16bins. So it will yield 36864 (16*16*16*9) features. More features, in a way, will increase precision when the training images comparing with the testing images. However, unrestricted increasing the bins per color channel would not improve precision; on the contrary, it would decrease the system efficiency and reduce the precision. In RGB space, setting the range of pixel value is 0 to 255. If each bin only contain one pixel value, the total bins is 150994944 bins (256*256*256*9), it is 4096 times more than 16bins. Additionally, more exact features will strengthen the noise influence. By my experiments, 16bins per color channel could obtain the best result.

2.2 Texture Representation

Extracting texture feature successfully is a key link of image classification and segmentation. It is a kind of global feature, different with color feature, texture feature need zones which contain a number of pixel points to calculate. This regional feature has advantage for model matching. As a result of being statistic, texture feature get rotation invariance and is robust for the noise. It would not cause a failure due to the local deviation. However, when pixel resolution ratio changes, the computed texture feature may has large deviation.

A good texture extraction should own the following properties:
- Reasonable feature dimensions.
- Good discrimination.
- Good stability.
- A small amount of calculation.
- Having a practical application.

In this part, I choose Gabor, LBP (Local Binary Patterns), and a method fused with LBP and GLCM (Gray-level Co-occurrence Matrix) to be the experimental methods.

2.2.1 – Gabor

Gabor filters [22] could be used for texture analysis. It simulates the visual perception of human visual system and is popular in the image feature extraction. Gabor method is based on the theory that texture is narrow-band signals [23]. Different textures would have different central frequency and bandwidth. Therefore, it can design one Gabor to filter the images based on their frequency and bandwidth. The Gabor filter only allows the texture with related frequency to pass, and other texture information will be restricted. Hence, collecting texture feature by Gabor filter mainly contains two steps; one is to design the filter which includes function, quantity, orientation and interval, another is to pick up effective texture feature set from the output results.

Gabor is a kind of windowed Fourier transform and the windowed function is Gauss function. The substance of Gabor transformation is to make a convolution of two-dimensional image. Two-dimensional Gabor function can be defined by

$$g_{uv}(x,y) = \frac{k^2}{\sigma^2} \exp\left(-\frac{k^2(x^2+y^2)}{2\sigma^2}\right) \left[\exp\left(ik\begin{pmatrix}x\\y\end{pmatrix}\right)-\exp\left(-\frac{\sigma^2}{2}\right)\right]$$

Where

$$k = \begin{pmatrix}k_x\\k_y\end{pmatrix} = \begin{pmatrix}k_v\cos\varphi_u\\k_v\sin\varphi_u\end{pmatrix}, \quad k_v = 2^{-\frac{v+2}{2}}\pi, \quad \varphi_u = u\frac{\pi}{K}$$

The value of v decides the frequency (scale) of Gabor filter and u means the orientation of Gabor function. K represents the total orientation numbers. The parameter σ/k determines the size of Gauss window. I set $\sigma \sqrt{2}\pi$, which is a common value. In this project, there are 4 scales. Thus, the range of v is 0, 1, 2 and 3. Also, there are 6 orientations in the project. So K equals to 6 and the range of u is 0, 1, …, 6. As a consequence of different scale and orientation, there will be 24 Gabor functions.

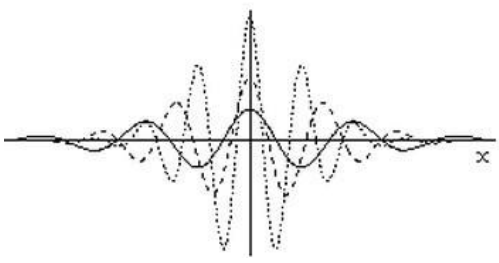


Figure 10: Gabor function with different scales

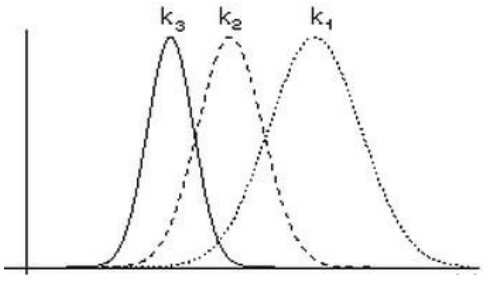


Figure 11: Gabor function with different orientations

2.2.2 – Local Binary Pattern

In general, the texture feature of a pixel point is related to its surrounding points. Local binary pattern [24] is a method which describes the texture feature based on the gray level between a pixel and its surrounding points.

It uses a 3*3 matrix to compute each pixel`s LBP operator. The detailed computational process is as follows:

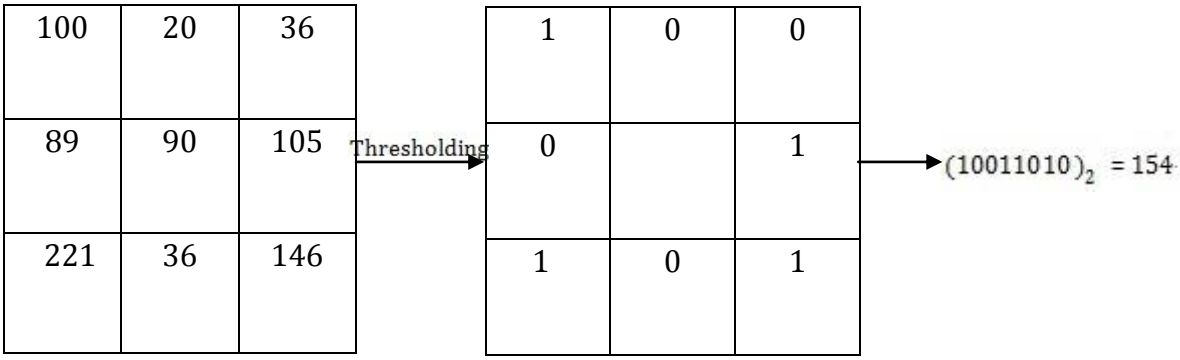


Figure 12: The computing process of LBP operator

As the figure shows, the left matrix depicts the grey level information of a pixel point and its eight surrounding pixels. Based on the threshold of central pixel, the point which grey value is larger than the central would be changed to 1; others would be changed to 0. After that, it needs to express the LBP value in a clockwise direction, which is 10011010 in the example. By calculating this binary number, we get a value between 0 and 255. So it can get a 256-dimensions gray level histogram. However, when an image rotates, the binary number will rotate also. In order to solve this problem, we collect the smallest binary number in the clockwise. For example, 10011010 will be changed to 00110101. After this step, it would collect a 36-dimensions feature.

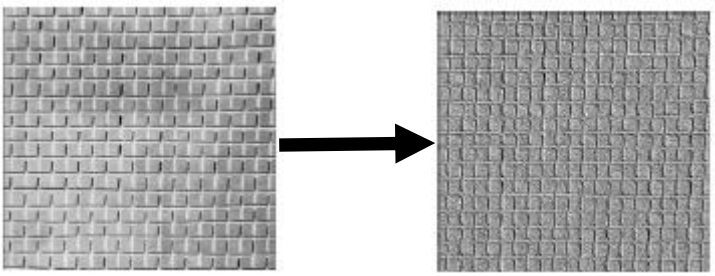


Figure 13: original texture picture

Figure 14: LBP picture

In other situation, the 3*3 block area could be replaced by a round area. LBP operator could be expressed with $LBP_{P,R}$, which means there are P pixel points in a round area with radius R. The general operators are $LBP_{8,1}$, $LBP_{16,2}$, $LBP_{24,3}$. In

my project, I use the operator $LBP_{8,1}$. In short, LBP has a good performance based on the multi-scale, brightness invariance and rotation invariance.

2.2.3 – A Method fused with LBP and GLCM

Because texture is generated with the grey-level distribution, two pixel points of a certain distance would have related grey-level information. GLCM (Gray-level Co-Occurrence Matrix) is the method which describe texture by studying the spatial correlation characteristics of grey-level. It will compute grey-level correlation of two points with different distance and direction.

It defines that the orientation of two pixels points is θ. One grey-level value is i, another is j. So, I(k,l) = i , I(m,n) = j. (k, l, m, n are coordinates of the points) The probability of occurrence of these two pixels points is defined as P (i, j, d, θ). Generally, there are four optional values of θ, namely 0°, 45°, 90°, 135°. Thus, there are four probabilities:
P (i, j, d, 0°) = # {[(k, l), (m, n)] | k-m=0, |l-n|=d}
P (i, j, d, 45°) = # {[(k, l), (m, n)] | k-m=d, l-n=d}
P (i, j, d, 90°) = # {[(k, l), (m, n)] | |k-m|=d, l-n=0}
P (i, j, d, 135°) = # {[(k, l), (m, n)] | k-m=-d, l-n=-d}
Where '#' represents amount of the set.

The following figure displays how to calculate GLCM.

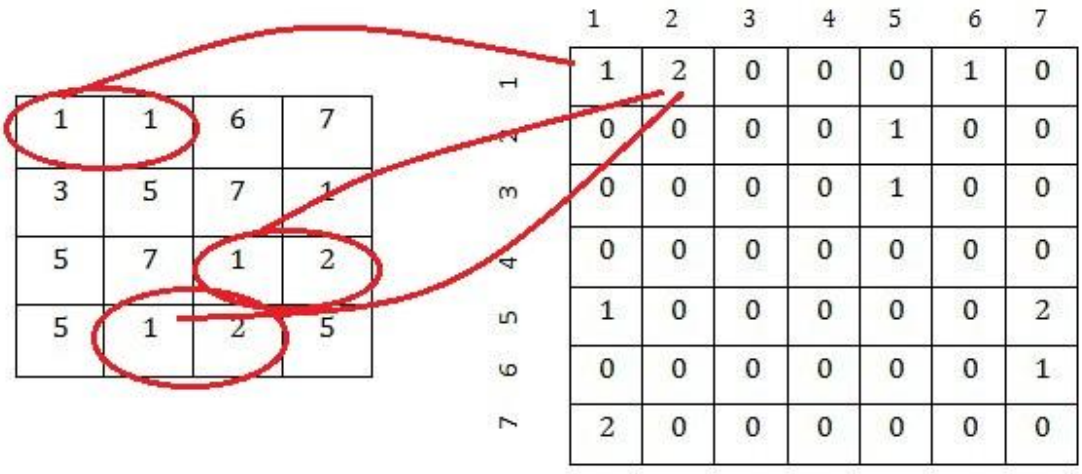


Figure 15: The GLCM of P(i, j, 1, 0°) distribution. (0<i<8 , 0<j<8)

As the figure can be seen, when computing the GLCM of a whole image, it needs a matrix with size 256*256 to store the information. It is difficult to use GLCM for extracting texture feature. Therefore, Haralick [25] addressed fourteen features used for analyzing the GLCM and collect the texture information. Based on the study of Haralick, there are only 4 features that are not relevant. They not only are easy for calculation, but also could obtain a higher precision of classification.

1. Angular Second Moment (ASM):

$$f_1 = \sum_{i,j} P^2$$

It indicates the uniformity degree of the grey-level distribution and the thickness degree of texture. If the values of all elements in the GLCM are nearly, ASM would be small. Otherwise, if some values are big and some values are small, ASM would be large. Also, if the elements of GLCM are distributed intensively, ASM would be large.

2. Contrast:

$$f_2 = \sum_{n=0}^{N-1} n^2 \left(\sum_{i=1,j=1,|i-j|=n}^{N} P \right)$$

Contrast reflects the definition of a picture and depth degree of the texture. If the texture cleft is deep, the contrast would be big and the visual effect would be clear.

3. Correlation:

$$f_3 = \frac{\sum_{i,j}(i-\mu_x)(j-\mu_y)P}{\sigma_x \sigma_y}$$

Where $\mu_x = \sum_i i \sum_j P$, $\mu_y = \sum_j j \sum_i P$, $\sigma_x = \sum_i (i-\mu_x)^2 \sum_j P$, $\sigma_x = \sum_j (j-\mu_y)^2 \sum_i P$.

Correlation measures the level of similarity in the row or column of GLCM. Therefore, it indicates the local information of a picture. When the values of elements in a row or a column are near, the correlation would be large. That means if an image has the horizontal texture, the correlation ($\theta=0°$) will be larger than other correlation of different directions.

4. Inverse Difference Moment:

$$f_4 = \sum_{i,j} \frac{1}{1+|i-j|} P$$

It reflects the homogeneity of the texture, which used for evaluating the degree of the local transformation in a texture part. If inverse difference moment is large, it demonstrates lack of change in different areas of the texture.

The shortage of LBP is that less spatial texture information is contained in the LBP image. If we use a large P and R of $LBP_{P,R}$ (P stands for the counts of points and R stands for radius of the operator), the feature dimensions would be large. The noise information would be strengthened and the calculation is more difficult. Additionally, applying feature dimension reduction method may result in loss of the texture feature. In order to extract effective LBP feature, I use a method fused with LBP and GLCM. The steps are as follows:

(1) Apply the $LBP_{8,1}$ operator to deal with the image. We can obtain a LBP image at this step.

(2) Compute GLCM of the LBP image based on d = 1, θ = 0°, 45°, 90°, 135°. We can get four GLCM at this step.
(3) Calculate Angular Second Moment, Contrast, Correlation and Inverse Difference Moment of each GLCM. Combine these four features of each GLCM to be the texture feature. We can obtain 16 texture features at this step.

The dimensions of texture feature are less than LBP histogram, it is beneficial to collect more useful information and reduce the noise influence. The detailed experiment results would be seen at part three.

2.3 Shape Representation

Shape is the cue which has a higher accuracy for recognition than color and texture. Generally, there are two types of shape representation, one is outline feature, and another one is region feature. The outline feature focuses on the edge of an object and the region feature is related to the whole shape area.

Here are several typical shape representations:

1. Edge descriptor. This method collects the shape information by depicting the edge feature of an object, such as Hough transform method [26] which is used for detecting the straight lines and edge orientation histograms method. Hough transform connects edge pixel points then combine a regional closed edge based on the global character of an image. The main concept of Hough is the allelism in point-line. Edge orientation histograms method would divide the picture into many parts then obtain the edge of object. After that, it is going to measure the histogram about edge size and orientation.

2. Fourier shape descriptors [27]. The core of this method is that using the Fourier transform of object edge to be the shape representation. This descriptor transforms a two-dimensional problem to a one-dimensional problem based on the closure and cyclicity of the regional edge. It could obtain three shape representations from the edge points. They are curvature function, distance of centroid and coordinate function.

3. Geometric parameters method. This method always uses some simple features to depict the shape, such as area, perimeter, diameter, thinness and center of gravity. In order to extract these shape factors, it needs good image segmentation.

4. Other methods. There are some other methods used for shape feature extraction, such as Finite Element Method [28], Turning Function [29] and Wavelet Descriptor [30].

In short, the shape feature is much related to the edge detection and region recognition. In this project, I use the methods Geometric Blur descriptor and HOG method for the shape feature extraction.

2.3.1 – Geometric Blur Descriptor

Geometric Blur Descriptor [31] is the method which is used in the previous paper. It is much similar with the geometric parameters method, but it needs more complex pre-processing and geometric distortion calculation. The descriptor can apply on determining whether an object which is similar to a training object is present in an image. This method can collect the features of its position. Also, the descriptor could find corresponding points on different objects. In the paper, it needs to obtain sparse signals first, then applied geometric blur to the sparse signals. Thus, it breaks images up into a quantity of channels, which are half-wave rectified oriented edge responses. Additionally, geometric blur in one-dimension is defined to be:

$$G_I(x_i) = \int_y I(x_i - y)K_{x_i}(y)dy$$

Where $\{x_i\}$ is a coordinate set varying in two dimension, I is a two-dimension sparse signal, y equals to T(x) that T is a geometric distortion of bounded transforms and $K_x(y)$ stands for a spatially varying kernel. Let

$$F_{I,K}(x,i) = G_I(x)$$

Finally, we can find the best match for a template in a picture by comparing the values of G_I, which means each level of geometric blur could be compared respectively using convolutions.

Figure 16: This is the effect of geometric blur about feature detection and template matching. [31]

2.3.2 – Histogram of Oriented Gradient (HOG)

HOG descriptors [7] is used for detecting pedestrian in a static picture at first, then it is applied on detecting people in the movie or video, finding vehicle and some animals in an image. It has many similarities with edge orientation histograms, but the difference is that HOG descriptor is going to calculate features in dense grid of uniformly spaced cells. Specially, it uses overlapping local contrast normalization in order to improve performance.

The most important concepts of HOG descriptor is that the appearance and shape of local object in a picture could be described well by orientation density distribution of gradients or edges. The implement method is demonstrated concisely as follows: firstly the image is divided into a number of connected areas, which are called cells. Secondly, it would collect pixel points' orientation histogram of gradients or edges. Thirdly, all the histograms would be combined to be the feature descriptor. In order to obtain a better performance, it could use contrast-normalized for the local histograms in the larger area (block). The detailed implementation is computing the density of each histogram in the block, and then normalizing each cell in this block based on the density. After normalizing, the descriptor would get a better result for the brightness and shadow change.

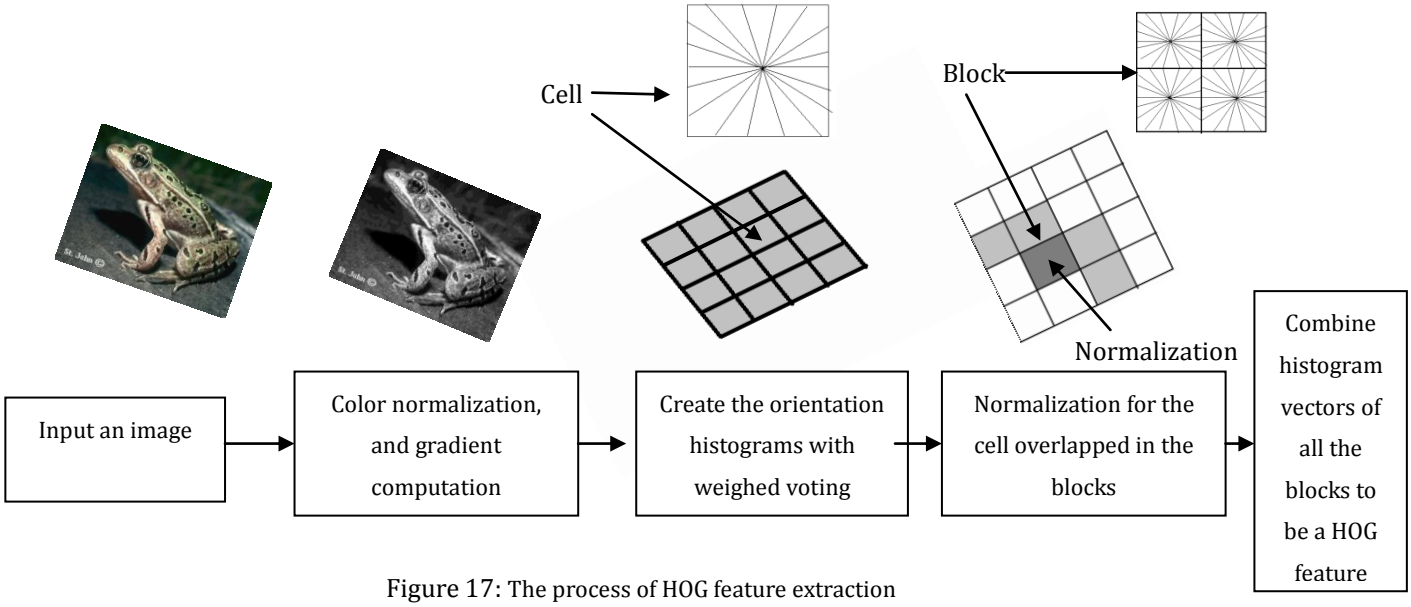


Figure 17: The process of HOG feature extraction

- Color and Gamma normalization

In order to reduce the influence of illumination, we need to normalize the whole image. One hand, because the color information is not useful in the shape detection, the color image would be transformed to a grey-scale image. In another hand, Gamma normalization could be applied to decrease the variation of illumination and shadow. The function of Gamma is as follows:

$$I(x, y) = I(x, y)^{\frac{1}{2}}$$

Where I stands for the grey-level value and (x, y) represents the point's coordinate. However, in my experiment, the results would not be changed even if I do not use Gamma for the image. Perhaps there is normalization process in the following step. So it is not essential to use Gamma at the beginning.

- Gradient computation

Gradient computation means calculating the first-order gradient of all pixel points. Derivation can not only capture some information of outline and texture, but also weaken the influence of illumination. The functions are as follows:
Gradient Size:

$$R(x,y) = \sqrt{(I(x+1,y) - I(x-1,y))^2 + (I(x,y-1) - I(x,y+1))^2}$$

Gradient orientation:

$$Ang(x,y) = \arccos(I(x+1,y)-I(x-1,y)/R)$$

- Create the orientation histograms with weighed voting

After dividing the image into a number of cells, it needs to combine all the gradient orientations in a cell and calculate its histogram. Based on the orientation-histogram channel, it uses a weighted voting method to determine which bin a pixel point belongs to.

The orientation-histogram channel is separated into 9bins, so it means each bin own 20°. Each angle of the gradient would be distributed to the appropriate bin. Assuming there is an image with size 480*360. Normally, we need to use two vectors of 480*360 dimensions to store the size and orientation separately, but we need two vectors of 480*360*2 dimensions to store these two values. Because, each angle of a gradient would have two adjacent bins, the gradient size would be resolved into two values on these two bins. The bigger one decides which bin the angle belongs to. So we use gradient size to be a weighted value and need 480*360*2 dimensions to store size and orientation respectively.

The aim of this step is to keep the basic posture and appearance sensibility.

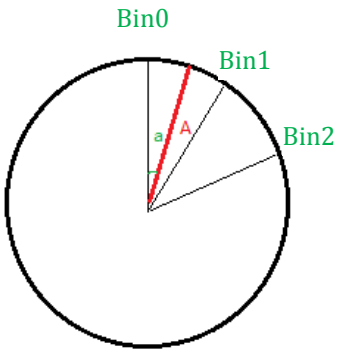


Figure 18: Weighted voting method on orientation histogram. (a stands for the gradient angle and A means the gradient size

- Group the cells together into larger blocks and Normalization

Owing to the variations of illumination and foreground-background contrast, the variation range of gradient strengths is very large. As a consequence, it needs to normalize the gradient strength. The author of HOG combines the cells into bigger areas (blocks), which are connected in the spatial space.

Actually, the author divided each cell into four parts in a block based on the pixel points' different contribution to a cell.

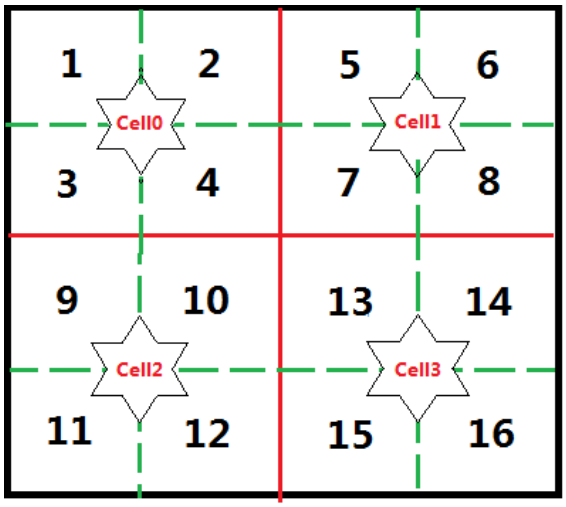


Figure 19: The different pixel points in a block have different effect on cells

As the figure shows, black box stands for a block, red box represents cells and green dotted lines separate each cell to four parts. In this figure, there are 16 parts. The author divided 16 parts into 4 groups.
The first group: 1, 6, 11, 16; which only contribute to their located cells when computing gradient histogram.
The second group: 2, 5, 12, 15; which also contribute to their left and right cell when computing gradient histogram.
The third group: 3, 9, 8, 14; which also contribute to their up and down cell when computing gradient histogram.
The fourth group: 4, 7, 10, 13; which contribute to the cells all around them when computing gradient histogram.

For example, the pixel points in 3 have influence on cell0 and cell2. When calculating the weighted voting, the gradient sizes in E should not only vote to cell0, but also to cell2. The weighted value is relevant to the distance between centers of cell0, cell2 and the pixel points' position. This process can link the cells more and keep a better geometric and photometric invariance.

- Combine histogram vectors of all the blocks to be a HOG feature

Assuming there is a picture with size 40*40, each block has 2*2 cells, and the size

of cell is 8*8. So there are 4*4 blocks in this picture and the HOG feature has 16*4*9 dimensions. In my project, I set the block size to 32*32, cell size to 16*16. In additional, I use two different methods to combine all the HOG features of samples. The first method, I standardize all the samples to the size 480*360. For the same size of training samples, it could collect same dimension of HOG features, and that is 28*20*4*9. It is convenient to compare the training samples and testing samples. The second method, I use K-means to collect all the Hog features to 100 clusters, which display a good performance. It will be detailed explained in part3.

2.4 Extremum Points Representation

An image always has a number of special points, like corner point or the points that are different in their surrounding points. Sometimes, these special points are non-visual, which means we could not realize them special by just watching the image. These special points could be used for object recognition or feature extraction, and I call them extremum points in my project.

Several feature extraction methods could be concluded to the method which aims is to find the extremum points. In this paper, I briefly introduce Harris method and Susan method, and applied sift method on my project.

Harris [32]:

a: flat point
b: edge point
c: corner point

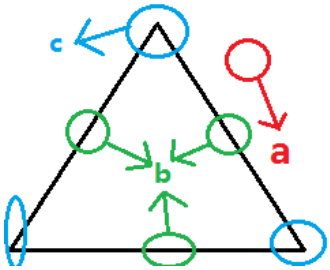


Figure 20: The types of points in a picture

If we compute the Ix and Iy of a, b, c, Ix and Iy of 'a' would be small; Ix or Iy of 'b' would be large; Ix and Iy of 'c' would be both are large. We can use Gaussian function or Prewitt operator or Sobel operator to calculate their directional derivative. Based on the values of Ix and Iy, it could find the corners to be extremum points. The performance is as follow.

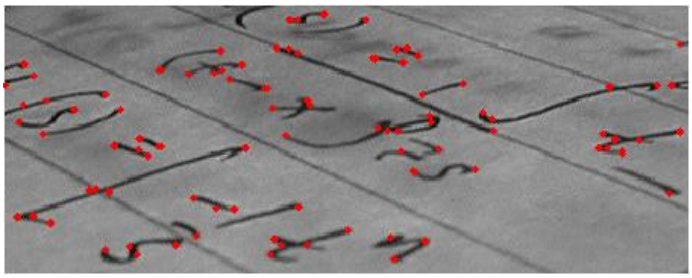


Figure 21: Harris applied on a text

Susan [33]:

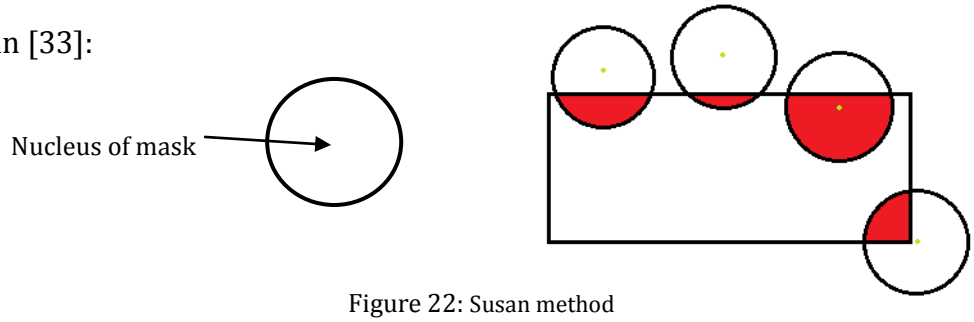


Figure 22: Susan method

It is a method which applies on the area size. It uses a circle mask to explore the image, in order to find the area illumination similar with the circle. The area is called USAN (Univalue Segment Assimilating Nucleus). As the figure shows, the red areas are USAN. The area would be biggest when the center of mask is located in the area; the center of mask is on the edge, when USAN is half of the mask; the center of mask is on the corner, when USAN is a quarter of the mask. Based on this information, we can evaluate the location of the pixel point.

Harris and Susan method always used for detecting fixed structure object and have a good performance or the edge and corner. However, for the animals on an image, they are more complex. They have more irregular extremum points and the features always have different orientation and illumination. Moreover, the background information would cause the interference when finding the features.
As a result, in my project, I choose a more stable and robust method Sift in this part.

2.4.1 – Scale–Invariant Feature Transform (Sift)

Sift [8] is a method which finds the extremum points in the spatial space and collect their position, scale and rotation invariance. The main advantages of Sift are as follows:

- Sift has a good distinctiveness, which is beneficial for a large quantity of image set. For the different categories of animals, it would make a reasonable effect to distinguish them.
- Sift could yield a mass of feature vectors even if there are few objects in the image. Sometimes, a picture may have only one animals, it also could be extracted enough features for processing.
- Sift could combined with other feature vectors conveniently. It is the reason why I take sift feature to be the fourth cue representation.
- Sift is the local feature. It could not only keep the invariance of rotation, scale and illumination, but also maintain stability of blur and affine.

Now I give a detailed introduction of Sift. There are five steps.

1) Create the scale-space

The aim of creating scale-space is to simulate multi-scale feature of image data. The kernel function used for scale transformation is Gaussian function. So the scale-space of a two-dimension image is defined as:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$, (x, y) stands for the coordinate and the σ is scale ratio. Scale ratio decides the smoothness degree of an image, which means that a big scale ratio is relevant to low resolution and a small scale ratio is related to high resolution. In order to detect the stable points in the scale-space, it used DOG scale-space to deal with the image. It could be defined as:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x,y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

From the author's opinion, the suggested value of σ is 1.6 and k is 0.5. Generally, we need to double the size of original image before creating the scale-space, because it could keep the information of original image better and increase the number of extremum points. In the Gaussian pyramid, there are two concepts, octave and interval. Each octave has several intervals. For example, the figure in the following indicates that there are two octaves and each octave has five intervals. In an octave, an interval is computed with its former by Gaussian smoothing. For different interval, the parameter of Gaussian smoothing would be $0, \sigma, k\sigma, k * k\sigma$, and so on. For the octave, the latter octave would narrow the size on the last interval of the former octave.

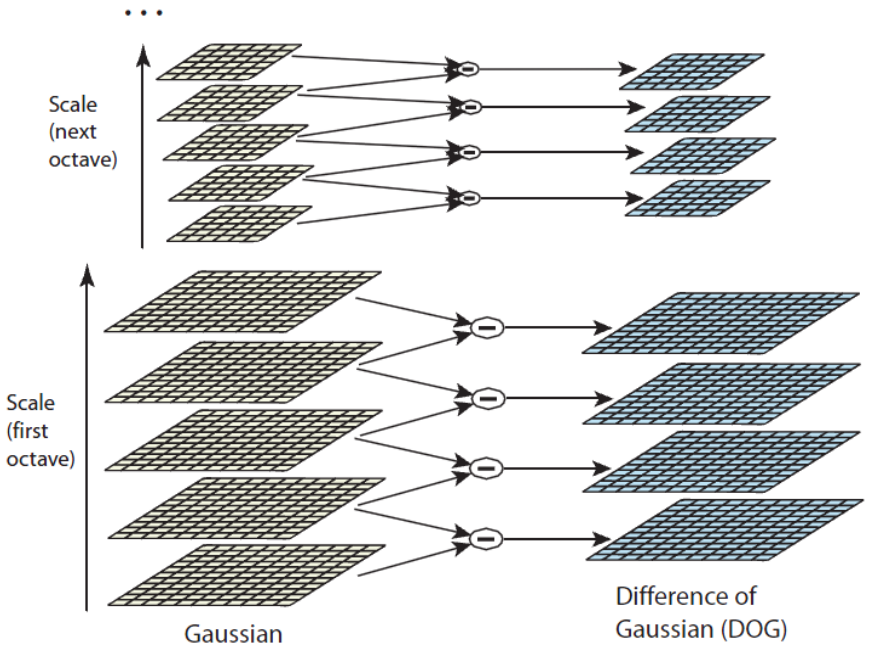


Figure 23: Gaussian Pyramid and DOG. (Each octave has five intervals) [8]

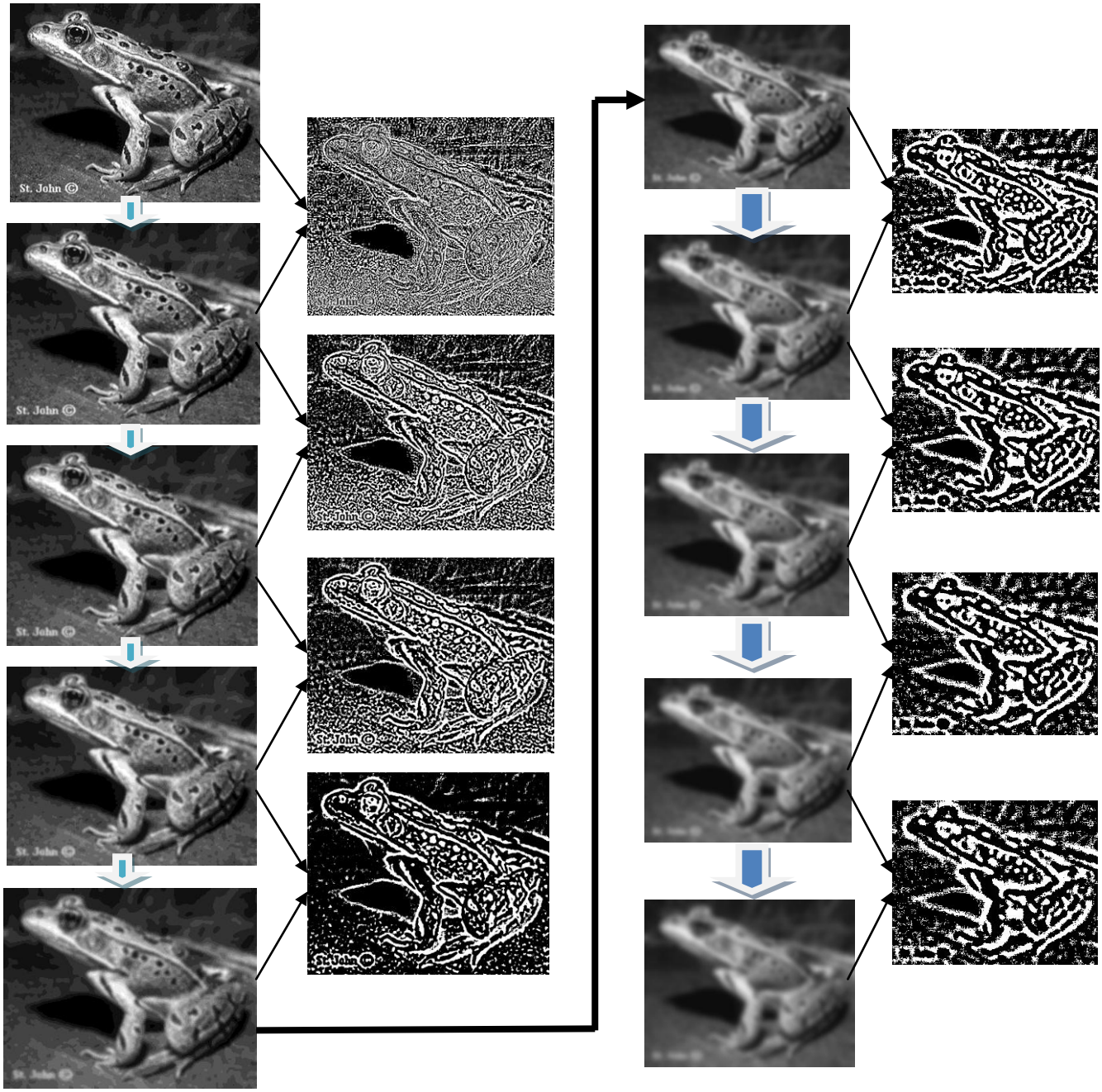


Figure 24: Gaussian pyramid and DOG applied on an image of frog

2) Detect the spatial extremum points

In order to find the extremum points of scale-space, each pixel points needs to be compared with its surrounding points, including the image zone and scale zone. As the figures shows, if a point are the biggest or smallest in their 26 surrounding points, the points could be treated as an extremum point.

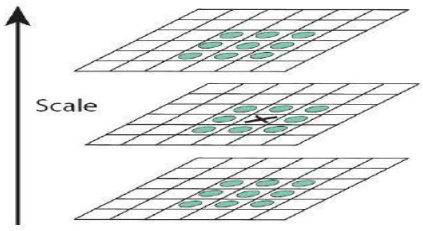


Figure 25: the black cross point is the extremum point. [8]

3) Determine the position of extremum

This step is used for find the position and scale of the extremum point exactly. Also, it can wipe out the points of instability or low contrast.

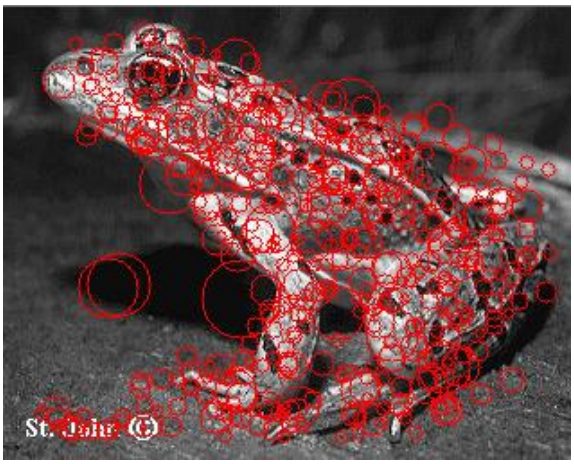


Figure 26: The extremum points on the frog image

4) Distribute the extremum points' orientations

There are a number of points adjacent to the extremum point. Based on the gradient orientation of these surrounding points, it could compute the orientation parameters of the extremum point, and make the feature vector having rotation invariance.

Gradient size: $M(x, y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$

Gradient orientation: $\theta(x,y)=\tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$

Combined of the information with step 3, each extremum point would have three values: Position (x, y), Scale and orientation. As a result, we could get a feature area in the next step based on these values.

5) Calculate the descriptor of feature point

The feature area is a window of size 8*8, the center of it is the extremum point. By computing the gradient- orientation histogram of every 4*4 subwindows, we get a keypoint descriptor. Each block would have eight directions. In my project, for the sake of higher accuracy, I will use 4*4 blocks to describe an extremum point, so there would be yield the feature vector with 4*4*8 dimensions.

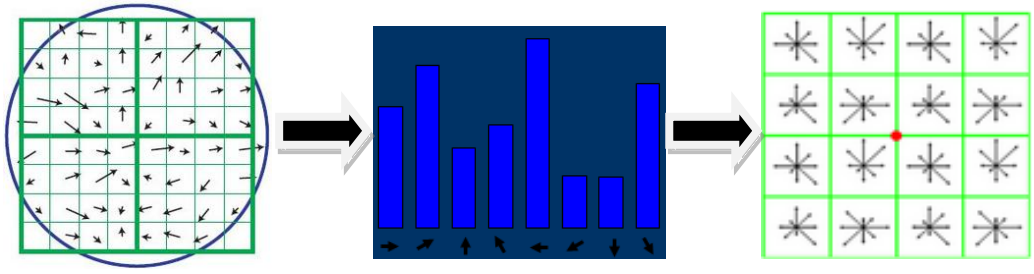


Figure 27: Use the neighbor points to produce the descriptor. [8]

2.4.2 – PCA-SIFT

PCA stands for principal component analysis, it is a method which used for collecting the main information and reducing dimension. The difference between PCA-Sift [34] and Sift is the step about computing descriptor. In PCA-Sift, it will pick an area of 41*41 size for each extremum point, and then computing the gradient of horizontal and vertical. So it would get a 3042 (39*39*2) dimensions feature. At last, the feature would be reduced to 36-dimensions by applying PCA method.

Although PCA-SIFT decrease the dimension of feature and keep the runtime of program, it is inevitable to lose amounts of useful information in the pictures. It is more reasonable for the pictures which own the special features. For sift, it could depict a picture more completely. The following figure is a comparison [35] with Sift and PCA-Sift.

Method	Time	Scale	Rotation	Blur	Illumination	Affine
Sift	Common	Best	Best	Common	Common	good
PCA-sift	Good	Good	Good	Best	Good	best

Figure 28: The comparison of SIFT and PCA-SIFT

The figure indicates that Sift do the best in Scale and Rotation. In fact, this is the most important filed in animal recognition. The experiments will be demonstrated in the following part.

3. Implementation and Results

My classifier mainly includes two stages, training and testing. In the previously paper, it uses the text cue to get the training exemplars. However, I select the training example manually in my project.

Different from the previous paper [1], I do not use the text cue to recognize the animal images because it is difficult to capture the text information related to an image on the web. Generally, in the standard format of html, each picture will have a text characteristic, which could be parsed by regular expression. However, a large number of images on the web are without the text characteristic. As a result, it is hard to collect the text information in an appropriate format. Perhaps we could find some kinds of text information on the web, but they are too noisy and not explicit to specified images. That means, it could obtain images and collect text, but we do not know which text information belongs to a certain image. So the text cue would have a bad effect on the animal recognition, when we could not build a good connection between images and text information.

It needs an advanced web crawler method applied to the web pages for obtaining exemplars with text. For my work, I would pay much more attention to the visual cues and try to improve the original methods on extracting visual features.

For the training examples, I would extract their features from four cues, color, texture, shape and extreme points. By the comparison of different methods for each cue's representation, I determine the methods of cues are 9 blocks (16bins), Gabor filter, HOG descriptor and Sift descriptor. Because the training images may have different sizes, it would yield the features of different dimension on some methods (like HOG and SIFT). It may result in inconvenience when computing similarities of testing images. There are three solutions: resizing a same size of all training exemplars, finding the nearest neighbor feature for the testing exemplars and applying the method k-means to cluster the features.

For the first solution, the common approach of resizing is linear interpolation. The feature information would lose whether resizing to a bigger size or a smaller size. Additionally, it is not easy to decide the fixed size of the training set. For the second solution, finding the nearest neighbor feature for every matching indicates that it will make a number of features not contribute to the similarity. For the third solution, k-means could make the same dimensional feature, decrease the large dimensions and improve the feature information. So, I use k-means to cluster the features from training set. By my experiments, I set 100 clusters for each HOG descriptor and 300 clusters for each Sift descriptor.

K-means is a method which is based on the distance between cluster points and other points. It uses distance to be the similarity factor, that is, the closer two points are, the more their similarities are. To be brief, the main steps of K-means include:

1 Choose K vectors $V_1, V_2, ..., V_k$ randomly, which would be treated cluster points at the initialize step. (each point is a multi-dimensional vector)

2 Use the following formula to compute the distance between the remaining vectors $x_1, x_2, \ldots, x_n$ and the K cluster vectors

$$D_j = || x_i - V_j ||$$

3 For x_i , select the nearest cluster vector by comparing the values of all the D_j , and then class x_i to this cluster.

4 Determine the new K cluster points, by computing the average of all vectors in their clusters separated.

5 Repeat step2 to step4 until the new K cluster vectors are equal to the former K cluster vectors or their deviation is smaller than a designed threshold.

In this project, the final K cluster vectors would be the new feature vectors of some methods.

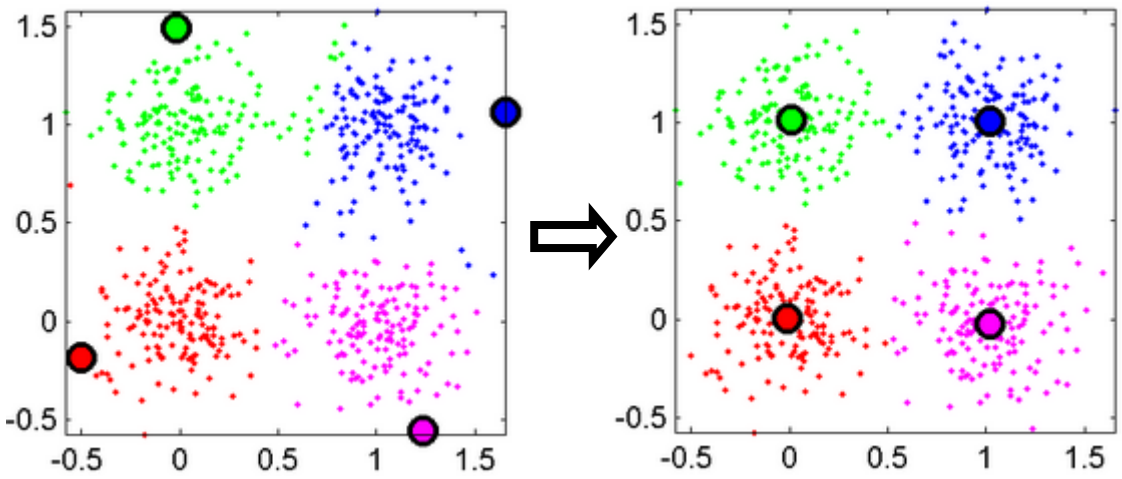


Figure 29: K-means applied on a two-dimension point set. (K=4)

For each visual cue, I compute the sum of the similarities of features matching training set. For each picture, I would normalize the similarity of each cue by dividing the maximum similarity of color, texture, shape or extremum points. After that, I would combine 4 independent similarities by using a linear combination. The four similarities are equally weighted in my program.

At Last, re-ranking the total similarities and output the re-ranked images.

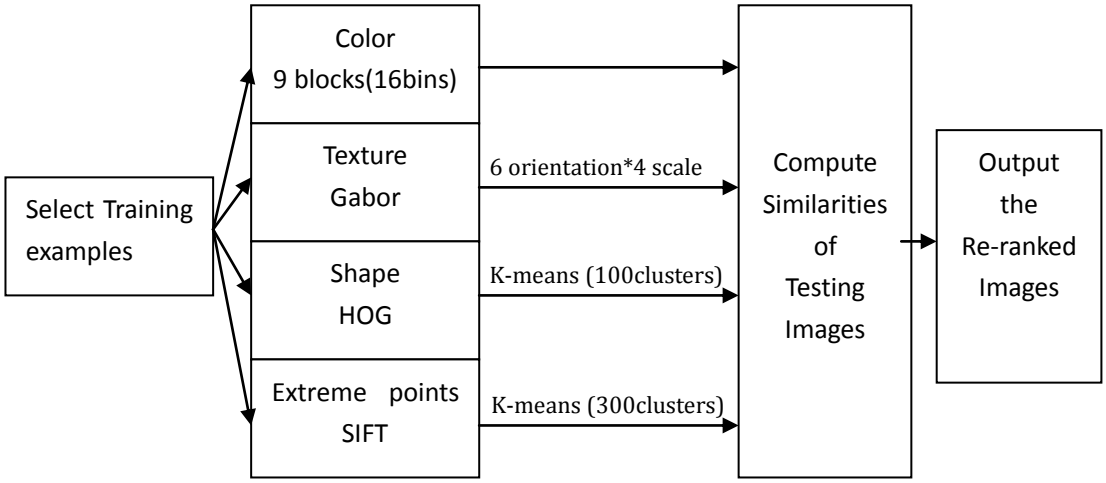


Figure 30: The main process of the program

3.1 Data Set

As a result of neglecting the text cue, we need to select the training images artificial. It indicates that the positive exemplars in the previous paper are all the real animal images. For this reason, I decrease the percentage of training set, in order to display a good performance of my classifier in the visual features extraction.

For the training set, I collect 150 images on five animal categories: “alligator”, “bear”, “frog”, “leopard” and “monkey”. Each category has 30 images. For the testing set, which is different with the training set, I collect about one thousand pictures for each category on the web. Because runtime of the program is too long, which classifies one category on 300 testing images would spend nearly three hours, I randomly select 100 real animals images and 200 fake images to be the testing images. It would be convenient to do the experiments and evaluate the methods’ performance. At last, I would use my classifier to re-rank 1514 pictures of “monkey”.

For the testing images on the web, there are a lot of typical pictures that are the fake. They would be shown in the following figure.

Some characteristics of animal are hard to spot. For example, different color of bear’ fur, the leopard’s stripe, stains of the frog and flexible limbs of monkey. My classifier would show a substantial contribution to the performance based on the visual information.

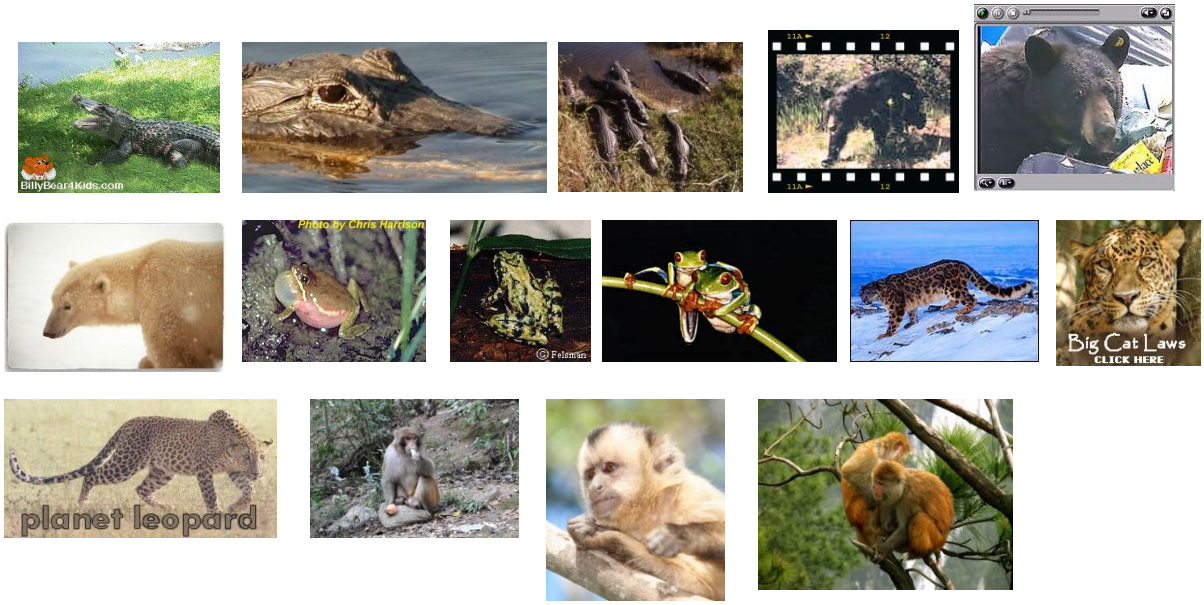


Figure 31: The training exemplars on five categories

Figure 32: The testing exemplars on five categories

As the figure of testing exemplars displays, the typical fake images on the web include toys, sketch, leather and the people who is relevant to the animal. They are frequent to be searched when people want to find the animals images. Some objects in the pictures have similar looks in color, shape or texture with real animals. By combining 4 cues, my classifier displays a good performance in getting rid of these pictures.

3.2 Explanation on Some Details

Grey-level images

In my project, I convert the color image to the grey-level image on most methods, except for the color representation. The cues of shape, texture and extremum points would not be changed when applying the transformation. It is more effective to process a grey-level image than a color image.

Normalization

For color representation and Gabor filter, I used scale conversion and normalization for the features of histogram, in order to strengthen invariance of the geometric transformation. As for HOG and SIFT, I do not use normalization specifically, because their process on feature descriptor contain this part.

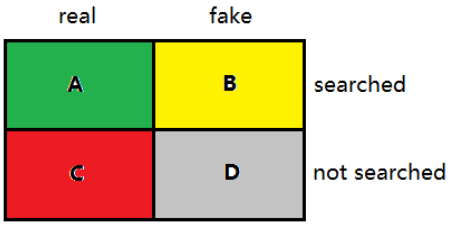
Similarity

Each testing picture would be compared with all the training samples one by one. By computing the difference of feature between the testing image and one training image, it would obtain a deviation. After that, get summation of all the deviations, we could collect a total deviation. It indicates that, the smaller this deviation is, the higher similarity of the testing picture is.

Results

I choose precision and recall to be the assessment standard. For the figure as following, the x-coordinator stands for recall and the y-coordinator stands for precision.

(A) Real images having searched
(B) Fake images having searched
(C) Real images not having searched
(D) Fake images not having searched



Precision = A/ (A+C); Recall = A/ (A+B)

Combining precision and recall could reflect the performance of a method well. For the result figures, because there are 100 real images and 200 fakes images in a testing set, precision would be 1/3 when recall reaches 1. We should pay much more attention to the images of top similarities and the general result of precision before recall reach 0.5. The former could reflect the whether the top images of re-ranking are real images; the latter could reflect the number of animal images

collected in the higher ranking. Both of them are significant for a good classifier.

3.3 Results on cues

Perhaps there exists the situation that a cue representation is superior to another cue representation on a certain category, but it is inferior in the others categories. The characteristic of different animal may have different effects on different categories. Also, the quality of their training samples would affect their performance, because there are only 30 training sample instead of 300 training sample in the previously paper. So, I would choose a reasonable method from the overall achievement.

3.3.1 – Results on Color

For color, I tried the methods of random subwindows, 9 blocks (8bins) and 9 blocks (16bins).

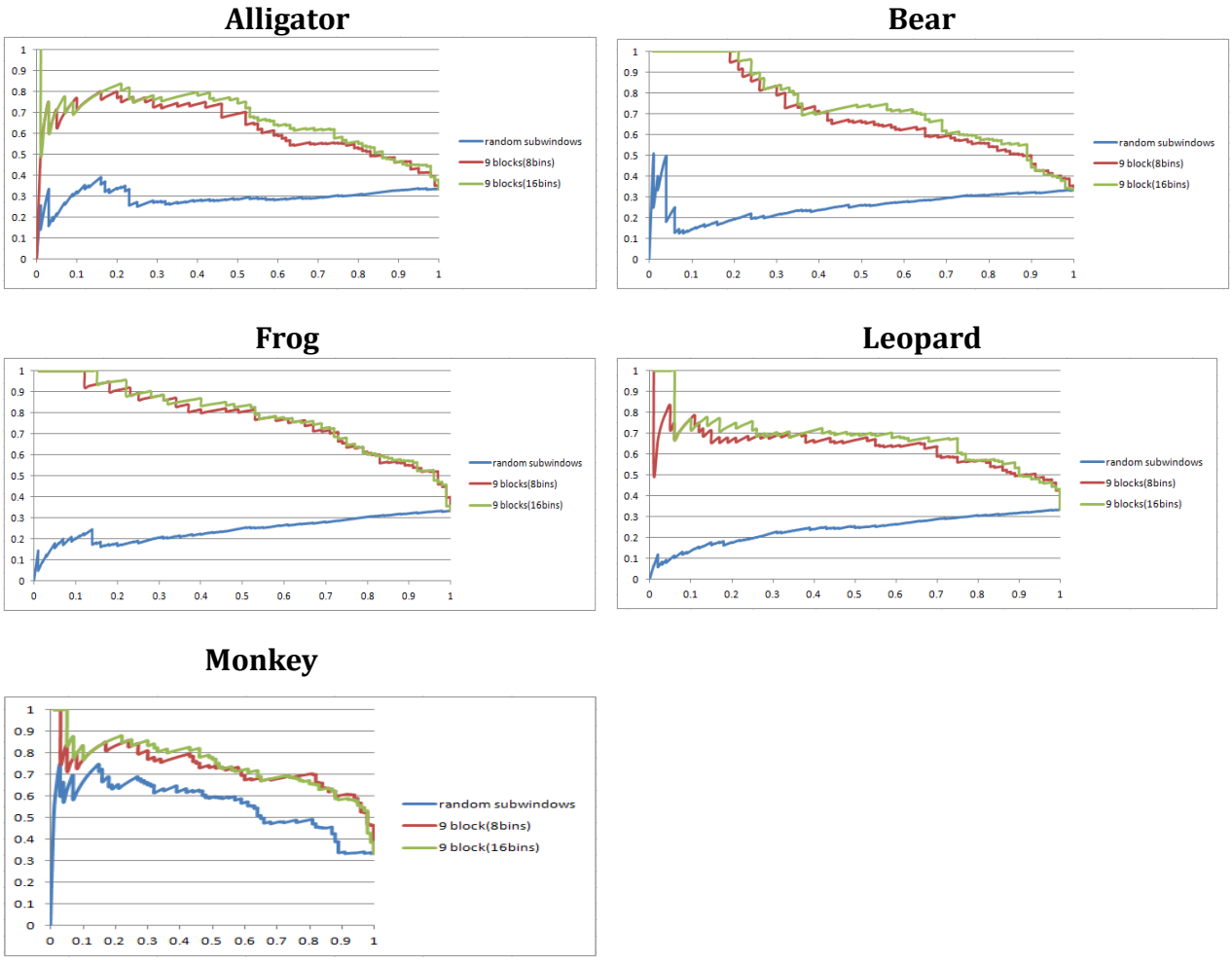


Figure 33: The precision and recall of color representation

As the figures can be seen, random subwindows methods display the worst performance. It is not appropriate to the pictures containing much noisy background information. Especially, the colors of training exemplars do not focus on a few bins of color histogram. So the aim of subwindows, which is going to cover the color more specific, seems not an easy task. Also, it needs a large number of training set to collect the feature because of its step of selecting subwindows with random size and random position. In short, a simpler image and more training examples may get a better achievement.

For the 9 block methods, the performance of them is very close. But, 16bins also display a better precision than 8bins; it is more accurate on the top-rank images, which could be seen obviously in the figure of leopard. Moreover, 16bins is the best division of color histogram. In my experiments, when I set the bins number to 32, it do not get a better result but cause a sharp decrease of precision on bear. It is because dividing a channel more accurate, it would magnify the influence of noisy information and background information. It may result in interference when computing the similarity for testing set.

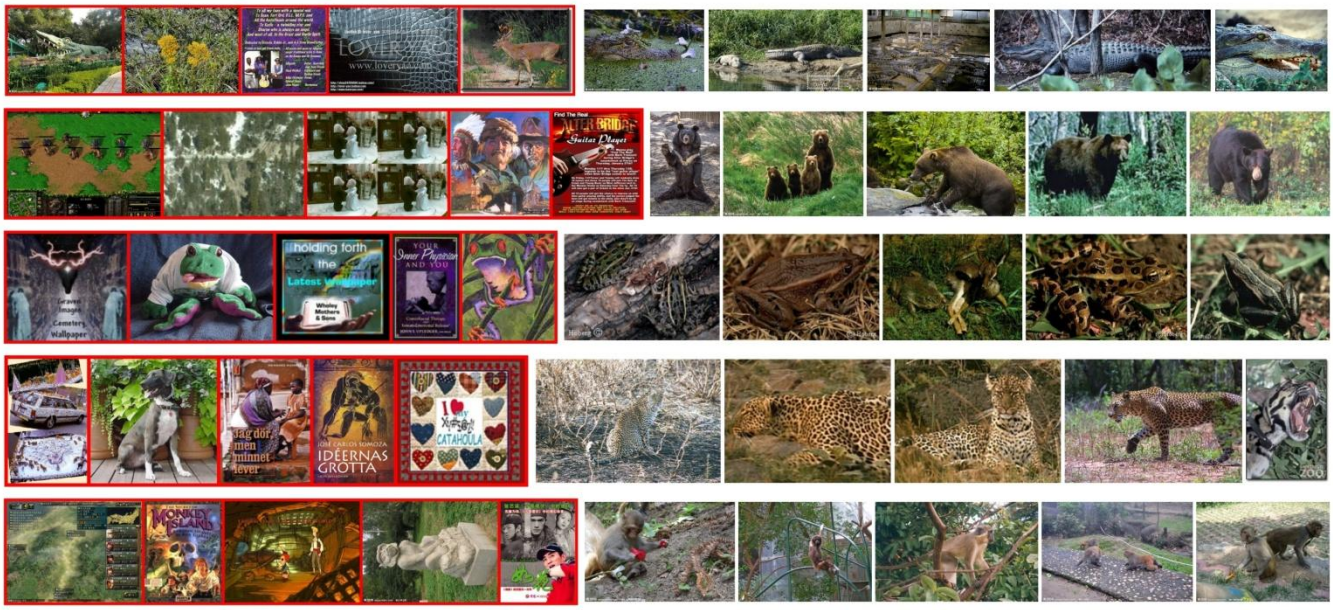


Figure 34: Top five fake and real images of 9 block (16bins)

The above images are part of re-ranking result by 9 blocks (16bins). They are the top-five ranked of real and fake images respectively. The fake images are boarded in red. The color representation could recognize the pictures which have a common environment surrounding the animals. Also, the animals in the picture have a specific color distribution. However, for the pictures of similar color distribution, it could not distinguish them. For example, the second picture in row 1, it is just several flowers with green grass, but it is similar with the alligator's habitat; the second picture in row 3, it is a frog toy with lifelike color. If only recognizing animals by color cue, it would be difficult.

3.3.2 – Results on Texture

For texture, I tried Gabor filter, LBP descriptor and a method fused with LBP and GLCM.

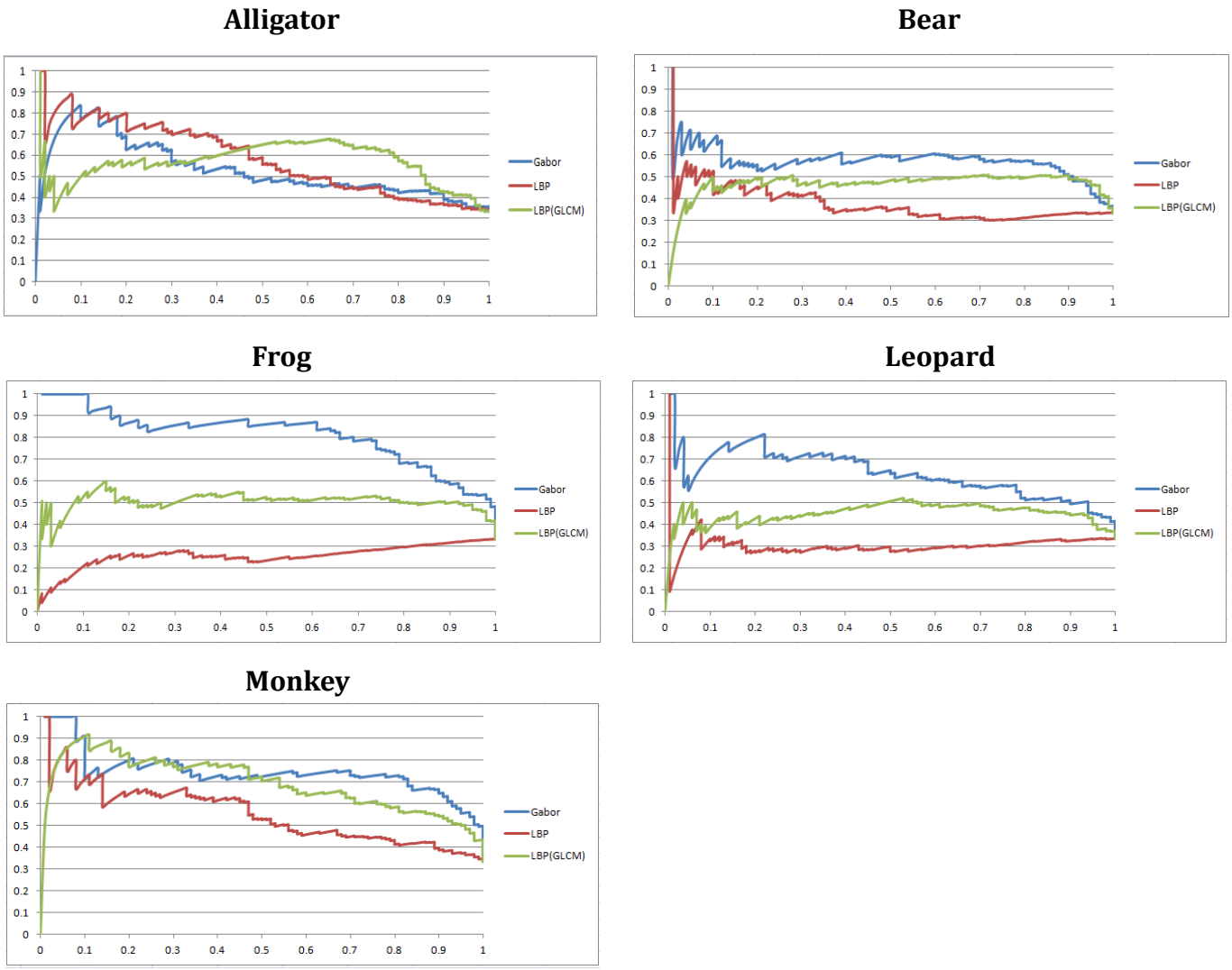


Figure 35: The precision and recall of texture representation

From the figures can be seen, we could discover that LBP does the best for alligator but the worst for frog, leopard and monkey. Gabor does the best for bear, frog, leopard and monkey. That is to say, for the single type of texture, LBP could display its superiority, because the descriptor is generated by each point with its eight surrounding points. The benefit of this descriptor is very sensitive for texture of the same structure. The textures of alligator in most pictures are close. However, there are some animals, which would have different texture, in this situation, the precision of LBP decrease sharply.

For Gabor filter, it has a bandwidth to distinguish different frequency. So, it is not

required to match the texture exactly, for the large quantity of data set, Gabor filter could fit animal's texture feature better.

The method fused with LBP+GLCM mainly uses four different features (angular second moment, contrast, correlation and inverse difference moment), the aim of this method is going to reduce the noisy information of LBP image and apply four independent properties for describing spatial relation of pixel points. The method shows ordinary performance in all the animal categories. Generally, it is better than LBP, but worse than Gabor.

Figure 36: The left group is texture of alligator, it is so similar. The right group is texture of frog, it show three different kinds of texture.

Figure 37: Top five fake and real images of Gabor Filter

The above pictures are obtained by using Gabor filter. In these five categories, alligator, frog and leopard have specific textures. Comparatively, texture of bear and monkey are much smoother. Their fur does not show much distinct spots. As a result, the effect of bear seems not very well. Additionally, some leather products have a close texture of animal, like the first, third and fifth pictures in row 1. They would disturb the ranking, when only applying Gabor filter.

3.3.3 – Results on Shape

For shape, I tried Geometric Blur descriptor and HOG descriptor.

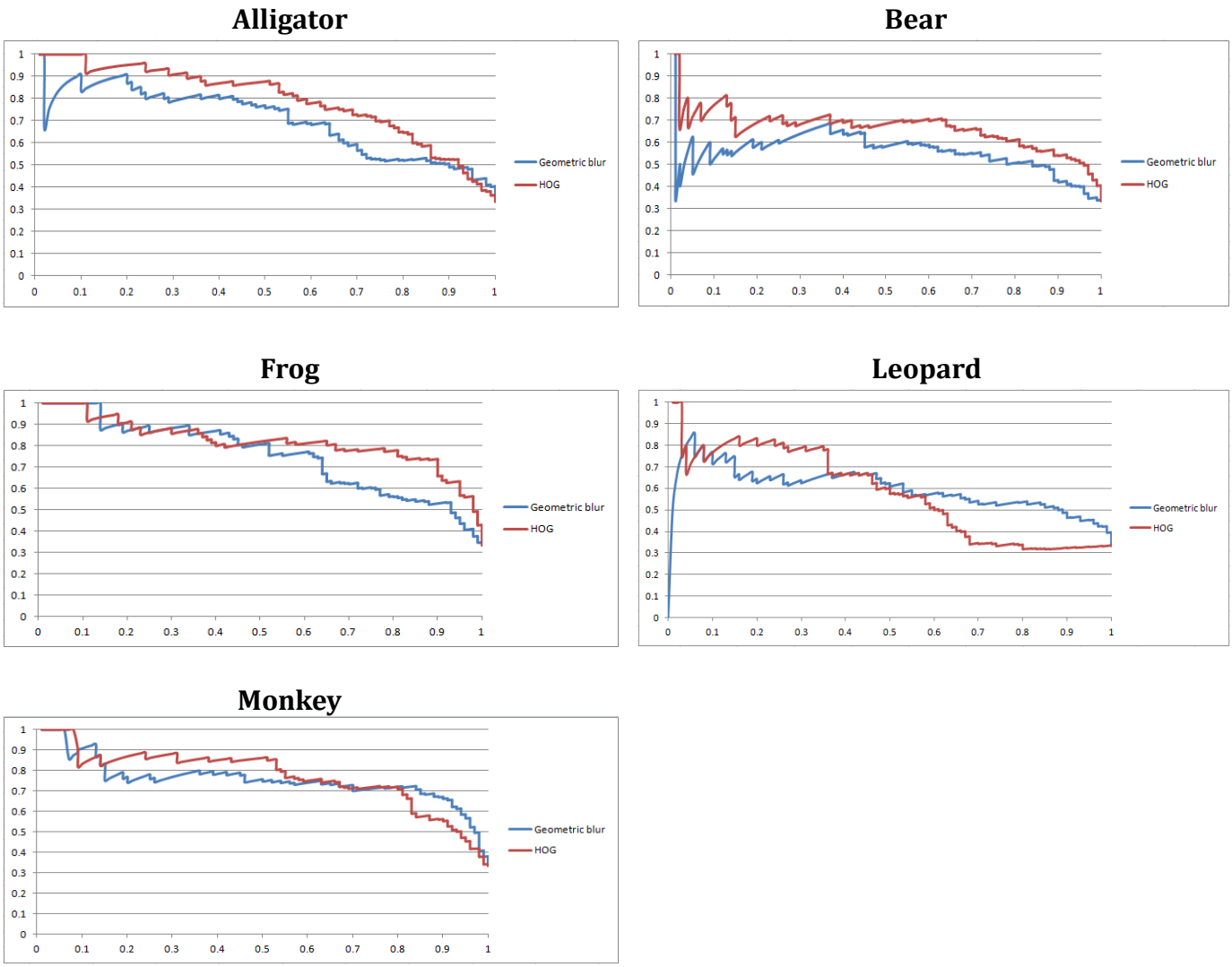


Figure 38: The precision and recall of shape representation

As the figure above demonstrates, HOG get a better achievement than Geometric Blur (GB). For alligator and bear, the precision of HOG surpasses GB totally. HOG exceeds at least 10 percent on alligator before the recall arrives 0.8, and exceeds nearly 30 percent on bear when the recall reaches 0.1. For frog and monkey, the curves of them are close and constant crossover. For leopard, HOG takes the lead in almost part before the recall intersects at about 0.45. As the previous part mentions, the top results and the precision which is shown before recall reaches 0.5 are more significant. In these comprehensive situations, a better ranking is achieved by HOG.

Different with color and texture representation, shape features seems more robust for animal recognition. It is because the shape of real animals would display a specific appearance. HOG descriptor could extract amounts of orientation information and it is very useful for animal recognition.

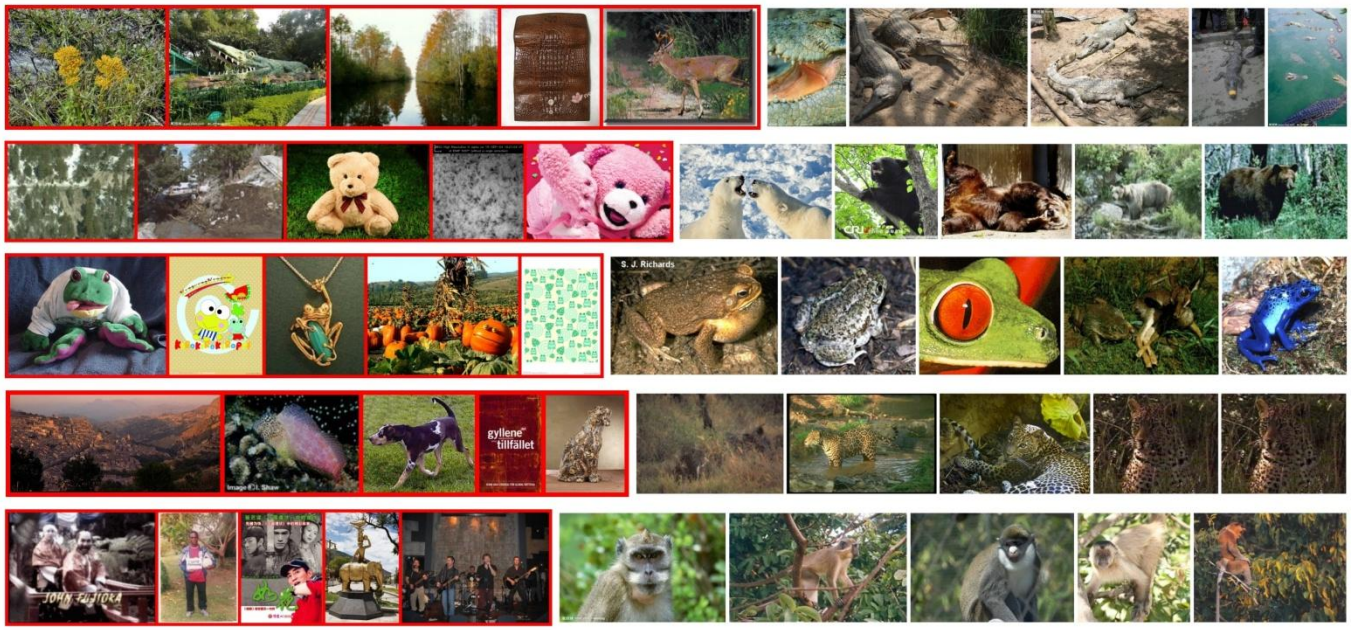


Figure 39: Top five fake and real images of HOG

Although HOG do better than texture and color extraction methods, it is also has a number of weakness. First, too much background information would increase deviation, such as the first and the third picture in row 1. The background information would be treated as positive features even if there is no any animal in the picture.

Second, HOG descriptor would treat animal toys as positive images, which is common in the collected images on the web, like the third and the fifth pictures in row 2.

Third, other animals or human might be misunderstood to positive samples. The original application of HOG is used for detecting [7]. Hence, if an object has a similar outline, it would be identified to the positive sample, such as the third image in row 4 and all the fake images in row 5.

Regardless of these disadvantages, HOG shows a valuable performance on the animals detecting.

3.3.4 – Results on Extremum Points

For the extremum points, I use Sift to be the main descriptor in this part. SIFT descriptor has 128-dimension features, PCA-SIFT has 36-dimension features. For PCA-Sift, after processing the initial features, the dimensions could be descended and the main components of feature would be collected.

However, it is not useful for the animal recognition. As the following figure indicates, the curve of PCA-Sift has a sharp decline at the beginning on the category of bear. Sift descriptor is widely used for matching. When it needs to search the most similar picture with the original one, Sift could achieve it well. Based on this application, extracting the main features information is reasonable by PCA-Sift, because it just needs to find the most suitable picture. However, for this project, it needs to re-rank a lot of images collected on the web, not to detect the best one.

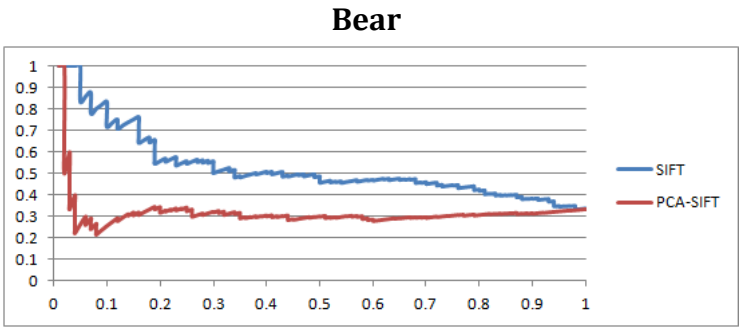


Figure 40: The performance of SIFT and PCA-SIFT on bear

Therefore, Sift seems more reasonable and robust for animal recognition. The following figure is the result of five categories re-ranked by Sift.

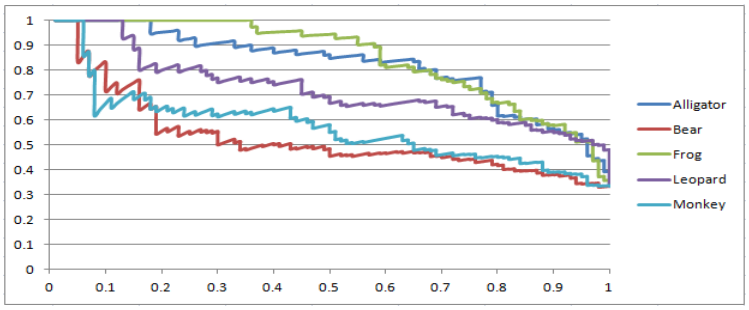


Figure 40: The performance of SIFT and PCA-SIFT on bear

For the category of frog, Sift displays an excellent performance, its precision continue to keep 100 percent until the recall reaches about 0.4. However, for the categories of monkey and leopard, it does not achieve a similar result of frog. Frog has a more fixed structure than monkey and leopard. Their limbs are flexible, so they can make a number of postures. In short, for the animals with fixed structure,

Sift demonstrates a high accuracy.

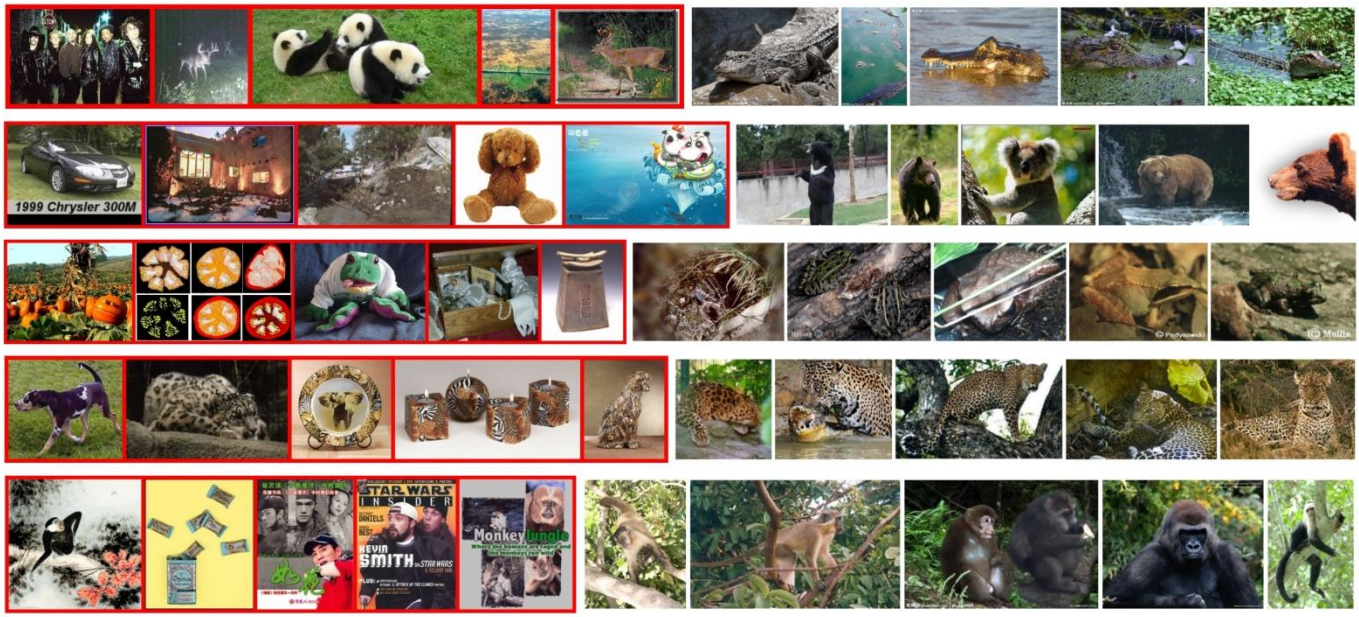


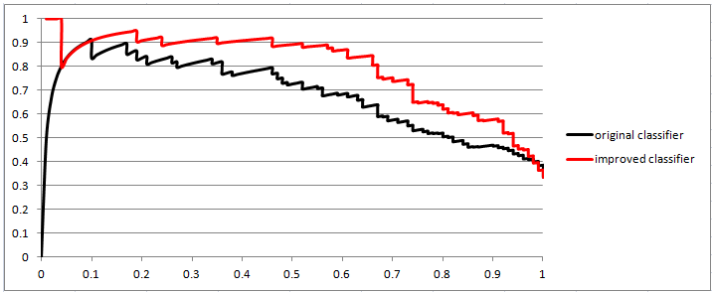
Figure 41: Top five fake and real images of Sift

There also many fake images of others animals or people. But these pictures are not ranked in the fore part. For the first picture in row 1, it ranks nineteenth; for the first picture in row 4, it ranks fourteenth. By comparison with HOG, it is more robust to avoid these interferences.

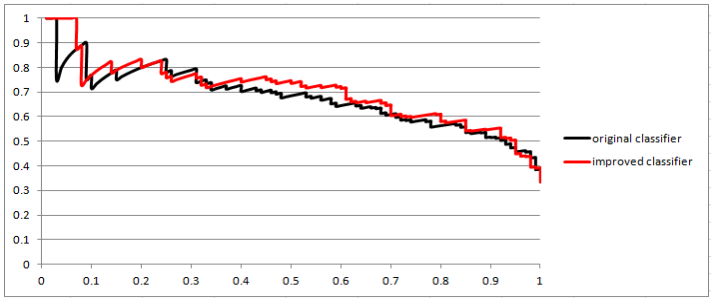
3.3.5 – Results on Combined Cues

In the previous paper, it uses 9 blocks (8 bins) for color representation, Gabor filter for texture representation and Geometric Blur descriptor for shape representation. In my project, I combine 9 block (16bins), Gabor filter, HOG and Sift for the visual cues' representation.

Alligator



Bear



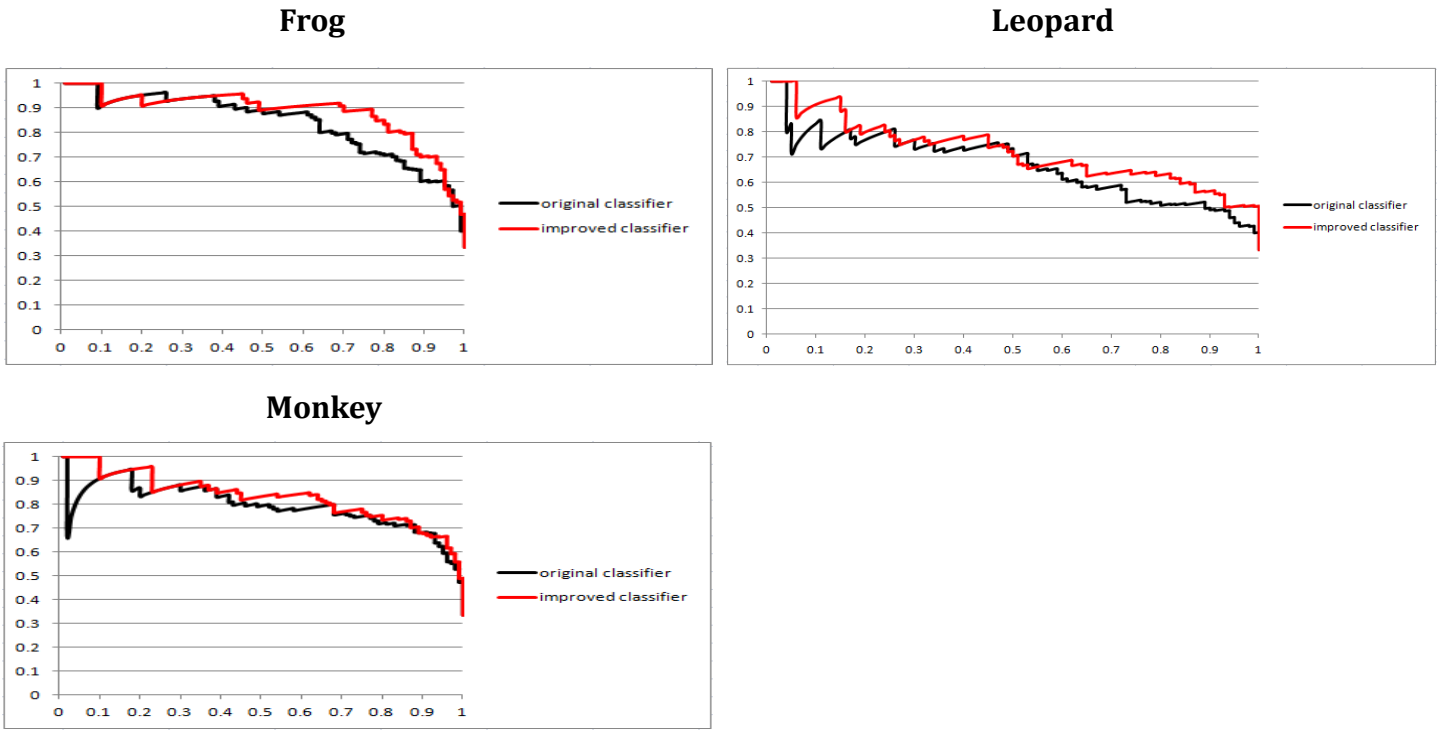


Figure 42: The performance of original and improved classifier

For five categories of animals, the improved classifier shows a better achievement than the original classifier on extracting visual information. Especially, it obtains higher precision of the foremost results, which is very significant indicator for evaluating property of the classifier. For color, it merely improves a little; for texture, the methods are the same. The main improvement is contributed by Hog, Sift and k-means applied on them. As a result, Hog and sift descriptor is useful for animals recognition.

At last, I applied my classifier to re-rank the “monkey” pictures on the web. There are 1514 images, 203 of them are real images. After re-ranking, precision of the top 30 pictures reaches 83.33 percent and precision of the top 100 pictures arrives 69 percent. It displays a good achievement of my classifier used for animal recognition on visual cues.

Figure 43: The top 30 pictures after applying the classifier

4. Discussion

This section is mainly about some problems and obstacles encountered in the project.

1) Text cue

In the previous paper, text cue play an important role, including obtaining the training exemplars by LDA and selecting the positive examples by asking the user to identify. At the last step, it is also computed the similarity for combination. In this project, the function of text cue is replaced by selecting the training set manually. There is a negative effect that the program is less intelligent and it loses a valuable cue for computing the totally similarity. From the figure of previous paper, the text cue could contribute much for the final results. However, ignoring text cue could make the developed direction focus on the visual cues. There are only 30 training images compared with 300 training images in the previous paper. Although, the number of training images is small, it also could make a good performance by the improved visual representation. So, when adding the text cue, it would obtain better results.

2) Calculation of similarity

The main calculation of similarity in this project is to compute the feature distance of Euclidean geometry between training and testing set. It may use another advanced method to measure the similarity based on different feature dimensions. If the dimension of a feature is low, distance of Euclidean geometry could not reflect the similarity exactly. A big difference in few component vectors would influence the similarity more.

3) Weight of different cue

In my project, I use a linear combination with equal weights for the four cues. However, a certain cue would contribute more to a specific category. For example, Hog does better than other three on leopard, the weight of shape should be increased in order to get a better result. Additionally, the precision would change irregularly as the recall increases. So, it seems that a changeable weight for each cue could develop the classifier. In my opinions, to achieve this function should be based on the similarity. The following figure reflects the deviation value of each method on different animal category. That is to say, when deviation becomes lower, similarity would become higher. We can find that, the similarity of color is low generally. Similarity of Gabor, HOG and SIFT are close. So perhaps we could compute a weight ratio when processing training set, and then combined with the

similarity in order to set flexible weights for different category.

	Alligator	Bear	Frog	Leopard	Monkey
Subwindow	0.985142	0.989243	0.987123	0.980428	0.998954
8bins	0.685032	0.773097	0.717176	0.719496	0.690913
16bins	0.770492	0.846039	0.805806	0.819723	0.749762
Gabor	0.247989	0.344695	0.289556	0.248812	0.256591
LBP	0.246508	0.403159	0.211427	0.23176	0.465441
LBP+GLCM	0.125986	0.042001	0.11672	0.009198	0.039934
GB	0.13011	0.244943	0.294723	0.313345	0.278825
HOG	0.312058	0.330757	0.361143	0.316893	0.293714
SIFT	0.268583	0.396083	0.308156	0.337263	0.169778

Figure 44: The deviation value of each category on different methods

4) Runtime of the program

When the data set becomes larger, we need to take the runtime into account. Sift could be replaced by Surf method, which could be treat as accelerated sift. However, it could not perform well like sift does on the invariance of rotation and scale. [35]

5. Evaluation and Conclusion

Object recognition is a popular and complicated problem in the field of computer vision. At the beginning, this paper reviews the general application of object recognition and analysis the main procedures of object recognition. Based on object recognition and a previous thesis, this paper come up with a new combined classifier for re-ranking animals pictures on the web.

The main idea of the classifier is to combine four different cues, color, texture, shape and extremum points, for recognizing the pictures with animals and re-ranking the pictures based on their similarities.

In order to make a better classifier, I compare 10 feature extraction methods for different cues. For color, I tried random subwindows, 9 blocks (8bins) and 9 blocks (16bins). For texture, I tried Gabor filter, LBP descriptor and LBP+GLCM. For shape, I tried HOG and Geometric Blur descriptor. For extremum points, I tried Sift and PCA-Sift descriptor. Additionally, I applies k-means cluster on collecting a same dimension feature, which would be convenient to compute similarity and cluster the features. Through the experiments, I set 100 clusters for random subwindows, geometric blur and hog descriptor; 300 clusters for Sift and PCA-Sift descriptor.

For the training set, I select 150 images on five categories of animals. Each animal has 30 images. For the testing set, I randomly select 100 real animal images and 200 fake images from about 1000 images collected on the web.

Through experiments, I find that 9 blocks (16bins), Gabor filter, Hog descriptor and Sift descriptor display a better performance for their cues. By combining these four methods, I obtain a new classifier of the visual information. From the results of experiment, it gets a better achievement than the visual method in the previous paper. Additionally, it manifests effectiveness and feasibility in re-ranking and collecting monkey images in 1514 testing exemplars.

Although, this program accomplishes and researches the mentioned task, it also needs a lot of work to complete its function.

6. Future Work

Based on the concepts in the paper, we could make a deeper research on the following aspects:

1) For the fake images of other animals or human, they are easy to be treated as real images. Only using HOG or SIFT is not a very effective method, we could tried a new method fused with the color information and HOG (or SIFT) descriptor.

2) Collect the data set with test information. By adding the test cue, I would help the classifier more intelligence.

3) Try more pictures with high resolution, for the sake of finding more valuable problems during the experiments.

4) Extend animal recognition to general object recognition. If the classifier is effective in re-ranking animal pictures, it may get a success in other object re-ranking. Try to research a more general classifier for object recognition.

7. Reference

[1]. T.L. Berg & D.A. Forsyth. (2006). *Animals on the Web.* Proc CVPR 2006.
[2]. D. Ramanan & D.A. Forsyth & A. Zisserman. *Strike a pose: Tracking people by finding stylized poses.* CVPR 2005.
[3]. G. Mori & X. Ren & A.A. Efros & J.Malik. *Recovering human body configuration: Combining segmentation and recognition*. CVPR 2004.
[4]. M. Sonka & V. Hlavac & R. Boyle (2008). *Image Processing, Analysis, and Machine Vision, Second Edition.* USA: Thomson. Pages 57-69.
[5]. J. Canny. *A computational approach to edge detection.* IEEE(Transaction on Pattern Analysis and Machine Intelligence), 1986.
[6]. D.G. Lowe. *Object recognition from local scale-invariant features*. International Conference on Computer Vision, Kerkyra Greece, 1999.
[7]. N. Dalal & B. Triggs. *Histograms of Oriented Gradients for Human Detection.* CVPR 2005.
[8]. D.G. Lowe. *Distinctive Image Features from Scale-invariant Keypoints.* International Journal of Computer Vision, 2004.
[9]. I.T. Jolliffe. *Principal Component Analysis*. Springer New York, 2002.
[10]. A. Hyvarinen & J. Karhunen & E. Oja. *Independent Component Analysis*. IEEE Transactions on Biomedical Engineering, 14:21-30, 2000.
[11]. C. Guo & X. Li & L. Zhong & X. Luo. *A Fast and Accurate Corner Detector Based on Harris Algorithm*. IEEE Third International Symposium on Intelligent Information Technology Application. 2009.
[12]. M.S. Bartlett & J.R. Movellan & T.J. Sejnowski. *Face Recognition by Independent Component Analysis.* IEEE Transactions on Neural Networks, Vol. 12, No.6. 2002.
[13]. S. Ullman & E. Sali. *Object Classification Using a Fragment-based Representation*. Lecture Notes in Computer Science, pages 73-87,2000.
[14]. C.J.C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery, 2(2):121-167, 1998.
[15]. A.P. Dempster, N.M. Laird, D.B. Rubin, et al. *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society, 39(1): 1-38, 1977.
[16]. D.M. Blei & A.Y. Ng & M.I. Jordan. *Latent Dirichlet Allocation*. Journal of Machine Learning Research 3, 993-1002. 2003.
[17]. R. Maree & P. Geurts & J. Piater & L. Wehenkel. *Decision Trees and Random Subwindows for Object Recognition.* ICML workshop on Machine Learning Techniques for Processing Multimedia Content. 2005.
[18]. M. Stricker & M. Orengo. *Similarity of Color Images.* In In SPIE Conference on Storage and Retrieval for Image and Video Databases III, volume 2420, page 381-392, Feb. 1995.
[19]. J.R. Smith & S.F. Chang. *Tools and Techniques for Color Image Retrieval.* 1996.
[20]. G. Pass & R. Zabih & J. Miller. *Comparing Images Using Color Coherence*

Vectors. 1996.
[21]. J. Huang & S.R. Kumar & M. Mitra & W.J. Zhu & R. Zabih. *Image Indexing Using Color Correlograms.* Computer Vision and Pattern Recognition. 1997.
[22]. Y. Rubner & J. Puzicha & C. Tomasi & J.M. Buhmann. *Empirical Evaluation of Dissimilarity Measures for Color and Texture*. In ICCV. 1999
[23]. A.C. Bovik & M. Clark & W.S. Geisler. *Multichannel texture analysis using localized spatial filters[J]*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, 12(1): 55-73.
[24]. T. Ojala & M. Pietkainen. *Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns.* IEEE Transactions on Pattern Analysis and Machine Intellgence, Vol 24, No7, July 2002.
[25]. R.M. Haralick & K. Shanmugam & I.H. Dinstein. *Texture Features for Image Classification[J]*. IEEE Transactions on Systems, Man, and Cybernetics, 1973, 3(6): 610-621.
[26]. J. Jeppe. *Hough Transform for Straight Lines.* Mini-project in Image Processing, 7th semester 2007.
[27]. D. Zhang & G. Lu. *A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures*.
[28].U.S. Dixit. *Finite Element Method: An Introduction.*
[29]. C.F.S. Volotao & R.D.C. Santos &G.J. Erthal & L.V. Dutra. *Shape Characterization with Turning Functions.* IWSSIP 2010 - 17th International Conference on Systems, Signals and Image Processing. 2010.
[30]. D. Shen & H.H.S. IP. *Discriminative Wavelet Shape Descriptors for recognition of 2-D patterns.* Pattern Recognition 32(1999) 151 – 165.
[31]. A.C. Berg and J. Malik. *Geometric blur for template matching.* In CVRP, June 2001.
[32]. C. Harris and M. Stephens. *A combined edge and corner detector.* Proceedings of the 4th Alvey Vision Conference. 1988: pp. pages 147--151.
[33]. S.M. Smith & J.M. Brady. *Susan – a new approach to low level image processing.* International Journal of Computer Vision 23 (1): 45-78. 1997.
[34]. Y. Ke & R. Sukthankar. *PCA-SIFT: A More Distinctive Representation for Local Image Descriptors.* Computer Vision and Pattern Recognition. 2004.
[35]. J. LUO & O. Gwun. *A Comparison of Sift, PCA-SIFT and SURF.*
[36]. C.The & R.Chin, *on image analysis by the method of moments,* IEEE Trans. Pattern Anal. Mach,1998, (10)

8. Appendix: Source Code

I only paste the methods of 16bins, Gabor, Hog and sift. Other methods could be seen in the electronic submission of my code.

Initial Parameters

```
char dirname[100];
	int TrainNum=30;
	int TestNum=300;    // 1-100 are real images, 101-300 are fake images
	int ShowNum=100;
	int KmeansK1=100; // used for gb\hog\random subwindows    (cluster numbers, could be
100/200/300/500 and so on)
	int KmeansK2=300; // used for sift\pca-sift
	int *h = new int[KmeansK1]; // used to calculate histogram
	int *s = new int[KmeansK2]; // used to calculate histogram
	CvvImage pSrcImg;
	IplImage *pSrcImCopy=NULL; // the copy of image
	IplImage *pSrcGrayImg=NULL; // the gray image
// 9 blocks color parameters
	float *ColorHistData=new float[16*16*16*9];
	CColorHist *pColorHist=new CColorHist();
// gabor parameters
	double Sigma = 0.1*PI;
	double F = sqrt(2.0);
	double dPhi[6]={0,PI/6,PI/3,PI/2,PI*2/3,PI*4/6}; // 6 orientations
	int iNu[4]={1,2,3,4};    // 4 scales
	int size=64; // bins of histogram
	float range[]={0,255};
	float max_val;
	float* ranges[]={range};
	float *GaborHistData=new float[size*6*4];
	IplImage *GaborImg=NULL;
	CvGabor *gabor1 = new CvGabor();
// Hog    parameters
	HOGDescriptor *hog=new
   HOGDescriptor(cvSize(64,64),cvSize(32,32),cvSize(16,16),cvSize(16,16),9);
	float *HogHistData=new float[KmeansK1];
	int AllHogfeaturesNum=0;
	int *pHogImgNum=new int[TrainNum];
	vector<float>Hogdescriptors;// result array
// sift feature parameters
	int intvls = SIFT_INTVLS;
```

```
double sigma = SIFT_SIGMA;
double contr_thr = SIFT_CONTR_THR;
int curv_thr = SIFT_CURV_THR;
int img_dbl = SIFT_IMG_DBL;
int descr_width = SIFT_DESCR_WIDTH;
int descr_hist_bins = SIFT_DESCR_HIST_BINS;
struct feature* siftfeatures;
struct feature* siftAllfeatures;
int SiftNum=0;
int SiftDim=128;
float *SiftHistData=new float[KmeansK2];
int *psiftImgNum=new int[TrainNum];
```

Training

```
for(int di=0;di<TrainNum;di++)
    {
        cout<<"Training:"<<di+1<<endl;

        sprintf(dirname,"C:/Users/Alex_Lin/Desktop/image/aliigator/%d.jpg",di+1);
        pSrcImg.Load(dirname,1);
        pSrcImCopy=pSrcImg.GetImage();
        pSrcGrayImg = cvCreateImage(cvGetSize(pSrcImCopy),IPL_DEPTH_8U,1);    // create an image with the same size as training image
        cvCvtColor(pSrcImCopy,pSrcGrayImg,CV_BGR2GRAY); //color space conversion, change the color image to gray image

// 9 blocks
        pColorHist->ComputerColorHist(pSrcImCopy,ColorHistData);
        for(int m=0;m<16*16*16*9;m++)
        {
            gColorHistDFile<<ColorHistData[m]<<" ";
        }
        gColorHistDFile<<endl;

// gabor
        GaborImg = cvCreateImage(cvSize(pSrcGrayImg->width,pSrcGrayImg->height), IPL_DEPTH_8U, 1);
        IplImage *planes[] = {GaborImg};
        CvHistogram* hist=cvCreateHist(1,&size,CV_HIST_ARRAY,ranges,1);

        for(int m=0;m<6;m++)
            for(int n=0;n<4;n++)
            {
```

```
                    gabor1->Init(dPhi[m], iNu[n], Sigma, F); // initialize the parameters
                    gabor1->conv_img(pSrcGrayImg, GaborImg, CV_GABOR_MAG); // use gabor filter
to process the image
                    cvCalcHist(planes,hist,0,0); // calculate the histogram
                    cvGetMinMaxHistValue(hist,0,&max_val,0,0);    // find the max value
                    cvConvertScale(hist->bins,hist->bins,max_val?255/max_val:0.,0); // shrink the bin into
zone [0,255]
                    cvNormalizeHist(hist,1.0);
                    for(int r=0;r<size;r++)
                    {
                        float bin_val = (float)(hist->mat.data.fl[r]);
                        gGaborHistDFile<<bin_val<<" ";
                    }
                }
                gGaborHistDFile<<endl;
                cvReleaseHist (&hist);

// Hog
            hog->compute(pSrcGrayImg, Hogdescriptors,Size(64,64), Size(0,0));
            writeHogFloat2Bin(hogfeaturefile,Hogdescriptors);
            AllHogfeaturesNum+=Hogdescriptors.size();
            pHogImgNum[di]=Hogdescriptors.size();
            Hogdescriptors.clear();

// sift feature
            int n = _sift_features(pSrcGrayImg, &siftfeatures, intvls, sigma, contr_thr, curv_thr,
                img_dbl, descr_width, descr_hist_bins ); // find sift features in an image and store in
&siftfeatures
            SiftNum+=n;
            export_features((char*)Siftfeaturefile.c_str(),siftfeatures,n); // export the feature to a file
            psiftImgNum[di]=n;
}
```

K-means of Hog and Sift

```
//k-means of Hog
    CvMat *Hogsamples=cvCreateMat(AllHogfeaturesNum,1, CV_32FC1);
    CvMat *Hogclusters=cvCreateMat(AllHogfeaturesNum, 1, CV_32SC1);
    CvMat *Hogcenters=cvCreateMat(KmeansK1, 1, CV_32FC1);
    cvSetZero(Hogclusters);
    cvSetZero(Hogcenters);
    readHogFloat2Bin(hogfeaturefile,AllHogfeaturesNum, Hogsamples);
    cvKMeans2(Hogsamples, KmeansK1, Hogclusters,cvTermCriteria(CV_TERMCRIT_EPS,10,1.0), 3,
(CvRNG *)0, KMEANS_USE_INITIAL_LABELS, Hogcenters);
```

```
cvReleaseMat(&Hogsamples);

int **Hogp=new int *[TrainNum];
for(int di=0;di<TrainNum;di++)
{
	Hogp[di]=new int[pHogImgNum[di]];
	for(int K=0;K<pHogImgNum[di];K++)
		Hogp[di][K]=0;
}
int hh=0;
for(int di=0;di<TrainNum;di++)
{
	for(int K=0;K<pHogImgNum[di];K++)
	{
		Hogp[di][K] = Hogclusters->data.i[hh];
		hh++;
	}
	for (int j=0;j<KmeansK1;j++)
		h[j]=0;
	vl_binsum(h,Hogp[di],pHogImgNum[di]);
	int MaxHog=0;
	for (int j=0;j<KmeansK1;j++)
		if(h[j]>MaxHog)
			MaxHog=h[j];
	for (int j=0;j<KmeansK1;j++)
	{
		float Ht=(float)h[j]/(float)MaxHog;
		gHogHistDFile<<Ht<<" ";
	}
	gHogHistDFile<<endl;
}
gHogHistDFile.close();
for(int di=0;di<TrainNum;di++)
{
	delete Hogp[di];
}
delete []Hogp;
cvReleaseMat(&Hogclusters);

// k-means of sift
	int SiftTrainNum=SiftNum/50+1;
	CvMat *Siftsamples=cvCreateMat(SiftTrainNum, SiftDim, CV_32FC1);
	CvMat *Siftclusters=cvCreateMat(SiftTrainNum, 1, CV_32SC1);
	CvMat *Siftcenters=cvCreateMat(KmeansK2, SiftDim, CV_32FC1);
```

```
        cvSetZero(Siftclusters);
        cvSetZero(Siftcenters);
        import_features((char *)Siftfeaturefile.c_str(), FEATURE_LOWE, &siftAllfeatures,SiftNum,SiftDim);
        int temp=0;
        for(int i = 0; i < SiftNum; i=i+50)
        {
            for(int j = 0; j < SiftDim; j++)
            {
                Siftsamples->data.fl[temp++]=(float)siftAllfeatures[i].descr[j];    // store the feature
informations in the matrix
            }
        }
        cvKMeans2(Siftsamples, KmeansK2, Siftclusters,cvTermCriteria(CV_TERMCRIT_EPS,10,1.0), 3,
(CvRNG *)0, KMEANS_USE_INITIAL_LABELS, Siftcenters);
        cvReleaseMat(&Siftsamples);

        int **Siftp=new int *[TrainNum];
        for(int di=0;di<TrainNum;di++)
        {
            Siftp[di]=new int[psiftImgNum[di]];
            for(int K=0;K<psiftImgNum[di];K++)
                Siftp[di][K]=0;
        }
        int AFT=0;
        int labelT=0;
        double *SiftTDistVecter=new double[SiftDim];
        for(int di=0;di<TrainNum;di++)
        {
            for(int K=0;K<psiftImgNum[di];K++)
            {
                double SDistMin=10000.0;
                for(int Km=0;Km<KmeansK2;Km++)
                {
                    for(int m=0;m<SiftDim;m++)
                    {
                SiftTDistVecter[m]=siftAllfeatures[AFT+K].descr[m]-Siftcenters->data.fl[Km*SiftDim+m];
                    }
                    CvMat *Matrix=cvCreateMat(1,SiftDim,CV_32FC1);
                    cvSetData(Matrix,SiftTDistVecter,Matrix->step);
                    double dist=cvNorm(Matrix,NULL,CV_L2);
                    if(dist<SDistMin)
                    {
                        labelT=Km;
                        SDistMin=dist;
```

```
                }
                cvReleaseMat(&Matrix);
            }
            Siftp[di][K]=labelT;
        }
        AFT+=psiftImgNum[di];
        for (int j=0;j<KmeansK2;j++)
            s[j]=0;
        vl_binsum(s,Siftp[di],psiftImgNum[di]);
        int SiftMaxGb=0;
        for (int j=0;j<KmeansK2;j++)
            if(s[j]>SiftMaxGb)
                SiftMaxGb=s[j];
        for (int j=0;j<KmeansK2;j++)
        {
            float Ht=(float)s[j]/(float)SiftMaxGb;      // store the sift features to the txt file
            gSiftHistDFile<<Ht<<" ";
        }
        gSiftHistDFile<<endl;
    }
    gSiftHistDFile.close();

    for(int di=0;di<TrainNum;di++)
    {
        delete Siftp[di];
    }
    delete []Siftp;
    cvReleaseMat(&Siftclusters);
```

Testing

```
for(int qi=0;qi<TestNum;qi++)
    {
        cout<<"Testing:"<<qi+1<<endl;

        sprintf(dirname,"C:/Users/Alex_Lin/Desktop/image/T_aliigator/%d.jpg",qi+1);
        pSrcImg.Load(dirname,1);
        pSrcImCopy=pSrcImg.GetImage();
        pSrcGrayImg = cvCreateImage(cvGetSize(pSrcImCopy),IPL_DEPTH_8U,1);
        cvCvtColor(pSrcImCopy,pSrcGrayImg,CV_BGR2GRAY);
        // color

// 9 blocks
        pColorHist->ComputerColorHist(pSrcImCopy,SearchColorHistData);
```

```
// gabor
        CvHistogram* hist=cvCreateHist(1,&size,CV_HIST_ARRAY,ranges,1);
        int GaborBin=0;
        for(int m=0;m<6;m++)
            for(int n=0;n<4;n++)
            {
                GaborImg = cvCreateImage(cvSize(pSrcGrayImg->width,pSrcGrayImg->height),
IPL_DEPTH_8U, 1);
                IplImage *planes[] = {GaborImg};
                gabor1->Init(dPhi[m], iNu[n], Sigma, F);
                gabor1->conv_img(pSrcGrayImg, GaborImg, CV_GABOR_MAG);
                cvCalcHist(planes,hist,0,0);
                cvGetMinMaxHistValue(hist,0,&max_val,0,0);
                cvConvertScale(hist->bins,hist->bins,max_val?255/max_val:0.,0);
                cvNormalizeHist(hist,1.0);
                for(int r=0;r<size;r++)
                {
                    float bin_val = (float)(hist->mat.data.fl[r]);
                    SearchGaborHistData[GaborBin]=bin_val;
                    GaborBin++;
                }
                cvReleaseImage (&GaborImg);
            }
        cvReleaseHist (&hist);

// Hog
        hog->compute(pSrcGrayImg, Hogdescriptors,Size(64,64), Size(0,0));
        int *pSearchHog=new int[Hogdescriptors.size()];
        float HogDistVecter;
        int HogLabel=0;
        for(int i=0;i<Hogdescriptors.size();i++)
        {
            float HogDistMin=10000.0;
            for(int Km=0;Km<KmeansK1;Km++)
            {
                HogDistVecter=Hogdescriptors[i]-Hogcenters->data.fl[Km];
                CvMat *Matrix=cvCreateMat(1,1,CV_32FC1);
                cvSetData(Matrix,&HogDistVecter,Matrix->step);
                float dist=cvNorm(Matrix,NULL,CV_L2);
                if(dist<HogDistMin)
                {
                    HogLabel=Km;
                    HogDistMin=dist;
                }
```

```
                cvReleaseMat(&Matrix);
            }
            pSearchHog[i]=HogLabel;
        }
        for (int j=0;j<KmeansK1;j++)
            h[j]=0;
        vl_binsum(h,pSearchHog,Hogdescriptors.size());
        delete []pSearchHog;
        float MaxHog=0.0;
        for (int j=0;j<KmeansK1;j++)
            if(h[j]>MaxHog)
                MaxHog=h[j];
        for(int j=0;j<KmeansK1;j++)
            SearchHogHistData[j]=(float)h[j]/(float)MaxHog;
        Hogdescriptors.clear();

// sift
        int Siftn = _sift_features(pSrcGrayImg, &siftfeatures, intvls, sigma, contr_thr, curv_thr,
            img_dbl, descr_width, descr_hist_bins );

        double *siftDistVecter=new double[SiftDim];
        int *pSearchSift=new int[Siftn];
        int Label2=0;
        for(int i=0;i<Siftn;i++)
        {
            double SiftDistMin=10000.0;
            for(int Km=0;Km<KmeansK2;Km++)
            {
                for(int m=0;m<SiftDim;m++)
                {
                    siftDistVecter[m]=siftfeatures[i].descr[m]-Siftcenters->data.fl[Km*SiftDim+m];
                }
                CvMat *SiftMatrix=cvCreateMat(1,SiftDim,CV_32FC1);
                cvSetData(SiftMatrix,siftDistVecter,SiftMatrix->step);
                double dist=cvNorm(SiftMatrix,NULL,CV_L2);
                if(dist<SiftDistMin)
                {
                    Label2=Km;
                    SiftDistMin=dist;
                }
                cvReleaseMat(&SiftMatrix);
            }
            pSearchSift[i]=Label2;
        }
```

```
for (int j=0;j<KmeansK2;j++)
        s[j]=0;
vl_binsum(s,pSearchSift,Siftn);
delete []pSearchSift;

float SiftMaxGb=0.0;
for (int j=0;j<KmeansK2;j++)
        if(s[j]>SiftMaxGb)
                SiftMaxGb=(float)s[j];
for(int j=0;j<KmeansK2;j++)
        SearchSiftHistData[j]=(float)s[j]/(float)SiftMaxGb;
```

Computing the deviation

```
for(int Imgi=0;Imgi<TrainNum;Imgi++)
                {
                    for(int FCi=0;FCi<16*16*16*9;FCi++)
                    {
                        ginColorHistDFile>>ColorHistData[FCi] ;
                        ColorDelt=abs(SearchColorHistData[FCi]-ColorHistData[FCi]); //color
                        ColorDeltSum[qi]+=ColorDelt;
                    }
                     for(int FGi=0;FGi<size*6*4;FGi++)
                    {
                        ginGaborHistDFile>>GaborHistData[FGi] ;
                        GaborDelt=abs(SearchGaborHistData[FGi]-GaborHistData[FGi]); // texture
                        GaborDeltSum[qi]+=GaborDelt;
                    }
                    for(int FHi=0;FHi<KmeansK1;FHi++)
                    {
                        ginHogHistDFile>>HogHistData[FHi] ;
                        HogDelt=abs(SearchHogHistData[FHi]-HogHistData[FHi]);
                        HogDeltSum[qi]+=HogDelt;
                    }
                    for(int FSifti=0;FSifti<KmeansK2;FSifti++)
                    {
                        ginSiftHistDFile>>SiftHistData[FSifti] ;
                        SiftDelt=abs(SearchSiftHistData[FSifti]-SiftHistData[FSifti]);    // sift
                        SiftDeltSum[qi]+=SiftDelt;
                    }
                }
```

```
for(int i=0;i<TestNum;i++)
		{
			if(ColorDeltSum[i]>MaxColor)
				MaxColor=ColorDeltSum[i];
			if(GaborDeltSum[i]>MaxGabor)
				MaxGabor=GaborDeltSum[i];
			if(HogDeltSum[i]>MaxHog)
				MaxHog=HogDeltSum[i];
			if(SiftDeltSum[i]>MaxSift)
				MaxSift=SiftDeltSum[i];
for(int i=0;i<TestNum;i++)
		{
			ColorDeltSum[i]=ColorDeltSum[i]/(MaxColor); // color similarity
			GaborDeltSum[i]=GaborDeltSum[i]/(MaxGabor); // texture similarity
			HogDeltSum[i]=HogDeltSum[i]/(MaxHog);
			SiftDeltSum[i]=SiftDeltSum[i]/(MaxSift); // sift similarity
/		}
for(int i=0;i<TestNum;i++)
		{
	TotalDeltSum[i]=ColorDeltSum[i]*Wi[0]+GaborDeltSum[i]*Wi[1]+HogDeltSum[i]*Wi[2]+SiftDeltSu
m[i]*Wi[3]; // the total similarity of each image
		}
```

Re-ranking

```
UINT32 *Index_L1=new UINT32 [TestNum];

		for(UINT32 i = 0; i<(unsigned int)TestNum; i++)
		{
			Index_L1[i]=i;
		}
		sort(TotalDeltSum,Index_L1,TestNum); // re-rank testing images
```