

Executive Summary

The project tries to align the has two main components: image classification and object detection.

1. Implementation of an image classifier using the Bag-of-Words model with SIFT, dense SIFT and oppSIFT descriptors combined with a Support Vector Machine (SVM) with linear and histogram intersection kernels.
2. Implementation of an object detector using a selective search method suggested by [30] for the above image classifiers.

Contents

1	Introduction	8
1.1	Aims and Objectives	9
1.2	Related Research	9
1.2.1	Image Classification	10
1.2.2	Object detection	11
1.2.3	Training and testing datasets	11
2	Feature representation	12
2.1	Scale Invariant Feature Transform (SIFT)	12
2.1.1	Scale space extrema detection	13
2.1.2	Accurate keypoint localisation	15
2.1.3	Orientation assignment	15
2.1.4	SIFT descriptor	16
2.2	Dense SIFT (dSIFT)	17
2.3	Colour SIFT	18
2.3.1	Opponent SIFT	19
2.4	Feature descriptor	20
3	Image representation	21
3.1	Bag-of-Words Model	21
3.2	Implementation	22
3.3	Limitations of the model	23
4	Image Classification	24
4.1	Support Vector Machine	24
4.2	Training and classification model	26
5	Object detection	28
5.1	Segmentation as Selective Search	29
5.2	Training and classification model	31
6	Analysis and Evaluation	33
6.1	Image classification	33
6.1.1	Desirable codebook size	34

6.1.2	Desirable feature descriptor	34
6.1.3	Comparison with the state-of-the-art methods	35
6.2	Object detection in the whole system	36
7	Conclusion and Future Work	39
A	How to run the code	44
B	Description of the state-of-the-art methods	46
C	Source code	48

List of Figures

1.1	An overview of our system.	10
2.1	Example of matching SIFT features.	14
2.2	Difference of Gaussian scale space.	15
2.3	Local extrema and its neighbour points.	16
2.4	Example of keypoints and actual extrema.	17
2.5	SIFT descriptor computed from image gradients.	17
2.6	The dense SIFT descriptors.	18
2.7	Two different descriptors are made similar as the result of discarding colour information.	19
3.1	Process of making a visual codebook.	22
3.2	Representation of an image into a Bag-of-Words features. . .	22
3.3	An example showing limitation of BoW model.	23
4.1	Maximum-margin hyperplane as shown with two classes of data points.	26
4.2	Training and classification model for SVM classifier.	27
5.1	Examples of correct prediction with threshold σ of 50% and 87.5%.	28
5.2	Object detection training and classification model.	29
5.3	Potential windows found by Selective Search software.	32
6.1	Performance of linear SVM clasifier with different types of features.	36

Chapter 1

Introduction

Artificial Intelligence (AI) has been an active research area since the first computer was invented. As defined by McCarthy, artificial intelligence is “the science and engineering of making intelligent machines” [23]. In this sense, we want to have a program that is capable of doing similar tasks to those performed by the human brain. This is a difficult challenge and so far, no program has been able to pass the Turing test ¹ over 60 years of AI research. To overcome this, AI is divided into smaller and more specific problems such as natural language processing, human perception or robotics, and computer vision. In fact, good progress has been made in some of these sub-fields in the last several decades and they have become mature enough to be separate branches of research on their own. Among those, computer vision is a field that is currently drawing a lot of attention from AI research community.

Computer Vision is concerned with understanding human vision and making programs with similar capabilities to that of the human eye. Although the camera can already capture scenes like the eye does, the performance of program agents is still nowhere near that of the human brain. Human beings can recognise objects in different colours or faces from different angles with little effort but this remains a challenging task in computer vision. Fortunately, a large number of existing methods in artificial intelligence, machine learning, and natural language processing contributed significantly to the research development and expansion of this field.

The rapid growth of the Internet makes the data it contains become an unlimited resource. For example, billions of images are freely available online and constitute a large sampling of the visual world while millions of video sequences are available to watch. As a result, this huge resource is an excellent source of input for many machine learning problems.

¹Turing test is a test for a machine to see if it matches human intelligence capability. The test is as follows: a human judge engages in many conversations with either a machine or another human; if the judge is unable to distinguish the other human and the machine reliably, then the machine is said to have passed the Turing test [28].

The outline of this research is as follows:

Chapter 1 introduces the overall aims and objectives (section 1.1) and background research (section 1.2).

Chapters 2 and 3 present different methods for feature and image representation.

Chapters 4 and 5 give the details of the work for image classification and object detection carried in this project.

Chapter 6 evaluates the performance of the classifiers and object detectors with different settings and with the state-of-the-art methods.

Chapter 7 concludes and proposes some future work for the project.

1.1 Aims and Objectives

The overall aim of the project is to understand the contents of a video by exploiting its accompanying text description. The pairing of video and text data are very popular on the Internet and can be found as a movie and its screenplay or subtitle, a Youtube video and its associated description or a video embedded in a website and the surrounding text within the website, etc. In order to simplify the task and concentrate on the computer vision tasks, we assume that the list of keywords is given. The list of keywords, where we will mostly consider nouns and noun phrases, describes objects such as “red car”, “silver wrist watch”, “green apple”, etc. We want to develop a system that is capable of determining such objects in the video sequence.

The input of the system will be a short video sequence and a set of keywords describing the video. The keywords are assumed to be relevant to the object in the video sequence. The output of the system will be bounding boxes appearing in video frames covering the keyword objects. To address our aims, the objectives are taken as follows:

- For each of the keywords k , we collect a set of referenced images from the Internet (manually or automatically using available online search engines).
- Each video frame will be classified and the object in the frame will be detected by presenting a bounding box covering the object.

1.2 Related Research

The objectives raises two questions to be addressed:

1. Given an image, we want to know the content that an image contains.
2. Given an image and we know that an object in the image, we want to know where the object is.

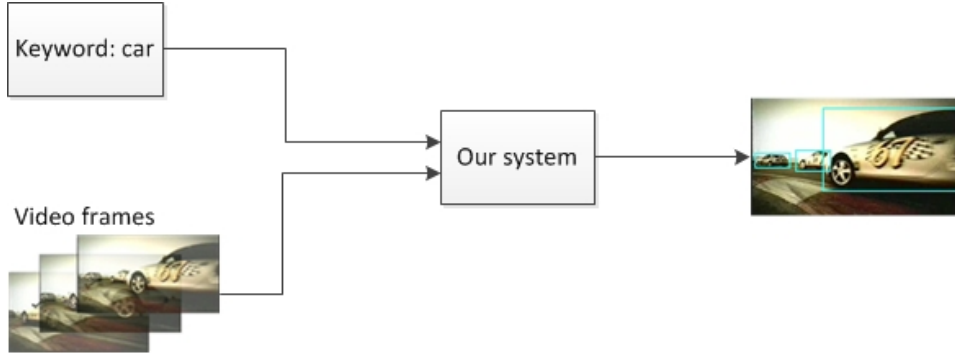


Figure 1.1: An overview of our system.

In computer vision, the former question is Image Classification and the latter one is Object Detection. In this section, we will cover the related research around these two areas.

1.2.1 Image Classification

Image classification is a common task in computer vision that determines whether an example of a specific category of objects appears in a given image. Research methods addressing this problem usually come with some way of detecting interesting descriptors or features. Both the training images and test images will be processed by a feature detector to find a set of interesting distinctive points. Similar appearance of features denotes similar representation of images and hence their class.

The extensive background of this area can be seen by many feature-based methods which have been developed in recent years. A corner detector was introduced by Moravec [24], improved by Harris and Stephens [15] to allow small image variations and effectively known as Harris corner detector which not only select corners but also other locations with large gradients in all directions at a given scale. In 1997, Schmid and Mohr [25] demonstrated an astonishing result that invariant features can be matched against a database in order to allow image recognition. Since then, Lowe extended the local feature approach to generate SIFT descriptors which are robust to scale, rotation and translation invariance [18, 19] and allow partial invariance to distortion. Many more other fast and robust feature detectors have been developed in recent years such as SURF [3] which claims to outperform SIFT in some cases with substantial reduction in running time or BRIEF feature detector [6] which also claims to outperform SIFT by reducing the memory space to store descriptors.

In this project, due to the rich existing research around the SIFT descriptors, we are also going to use SIFT [18, 19] for image classification and object recognition algorithm. This will be explored in more details in chapter 2.

1.2.2 Object detection

Object detection has been at the centre of computer vision research for many years with many practical applications such as face detection or optical character recognition. Essentially, object recognition is the task of searching for a given object in a scene image or a video frame. In order to align with existing benchmarks for evaluation, the location of an object will be represented as a bounding box and each object labelled from the target classes in the test image.

The object detection problem can be similar to image classification in the sense that we can create a classifier that trains on different windows of the images. Only the windows that tightly cover the whole object are positive examples while the windows that do not contain the object are negative examples.

A common way to detect an object is to use a sliding window to exhaustively search all possible locations and choose the best match. However, this approach is very computationally expensive. A weak classifier is usually used to complement the waste of examining non-object windows. Therefore, a selective window searching method is desirable. Vedaldi et al. [33] proposed a jumping window method which learns the relation between individual visual words and the object location. Maji and Malik [22] combine multiple relations to predict the object location using a Hough-transform. Van de Sande [30], however, proposes a different approach for selective window using segmentation. This method not only allows independence between class and object hypothesis but also gives a high recall rate and fast computation. Therefore, we will be using this selective method in our object detection.

1.2.3 Training and testing datasets

In order to evaluate our system, it is necessary to have some benchmarks datasets and performance of existing methods. The PASCAL Visual Object Class Challenges (VOC) has been a good medium-size dataset for such purposes. Essentially, VOC challenges are a set of computer vision competitions set up from 2005 up until present to provide standardised databases for object recognition, a common set of tools for accessing and managing the database annotations and to run a challenge evaluating performance on object class recognition [8]. At the moment, the dataset contains about 11,000 images of 27,000 objects that belong to 20 classes. In this thesis, we are going to examine the performance of image classifiers and object detectors on the VOC2011 dataset.

Chapter 2

Feature representation

Digital images are usually represented as a three dimensional array (x, y, t) where x and y are the location of the image pixels and t has three components that correspond to the three channels in the RGB colour space. However, this representation is not that useful for machine learning problems not only because of the large memory space but also because the meaning of an image as a whole is difficult for a machine learning method to learn. For that reason, the features of an image are more commonly used for classifying an image or detecting an object. There is no precise definition of a feature but generally the reader can understand it as a small patch of the original image that has unique shape or texture such as a corner or an edge as opposed to the noisy points caused by image captured devices.

Features are extracted by applying a feature detector to the original image to find the points of interest. A feature is represented as a descriptor (i.e. an array of numbers). There have been many feature descriptors proposed in the recent years of computer vision research. Among them, the most well-known feature which has been experimentally proved to be useful for image classification is Scale-Invariant Feature Transform (SIFT) descriptor [19]. The details of this descriptor will be examined in section 2.1. Sections 2.2 and 2.3 examine other versions of SIFT descriptors. Section 2.4 concludes the chapter by introducing a method to combine descriptors.

2.1 Scale Invariant Feature Transform (SIFT)

Lowe [18, 19] describes SIFT features as distinctive points in the images that are invariant to scaling, rotation or translation transformation. He also argued that the feature extraction is done in a similar manner to how animal and human brains recognise objects [18]. SIFT extracts a collection of distinctive invariant features from a given image. These features are partially invariant to illumination changes and affine or 3D projection. The features are distinctive in the sense that each of them is correctly matched into a

database of SIFT features from training images with a high probability. The algorithm is processed in different stages where the later stages are more computationally expensive than the former ones. So, the approach is to filter as many uninteresting points as possible to minimise the cost of computation. Four major stages of SIFT are outlined as follows:

1. Scale space extrema detection: First, we construct difference-of-Gaussian (DoG) scale space from the original image of different scales and sampling levels. All image locations in the scale space will be searched efficiently to identify feature points that are invariant to scale and orientation.
2. Keypoint localisation: Each potential feature point is then fitted to a model to measure stability. More stable points have a high contrast to its neighbours and will be selected as keypoints.
3. Orientation assignment: Each keypoint is associated with one or more orientations based on its local gradient direction. The assigned orientation and scale provides invariance to rotation and scaling transformation.
4. SIFT descriptor: Finally, the local gradients are computed for each keypoint at the selected scale in the surrounding region. These gradients allow substantial levels of illumination change and local shape distortion.

These four stages are described in more detail in the following sections. Figure 2.1 gives an example of matching SIFT features to find an object in a scene image.

2.1.1 Scale space extrema detection

The first step of this stage is to construct a DoG scale space from the original image. The DoG scale space pyramid is formed by:

1. Applying Gaussian convolution to the original image at different scales and then computing the difference of successive Gaussian-blurred images; and
2. This process is repeated at different levels of sampling (see figure 2.2¹). More formally, the scale space $L(x, y, \sigma)$ of an input image $I(x, y)$ is defined as follows:

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

¹Source: David G. Lowe (2004) Distinctive image features from scale-invariant key. [19]

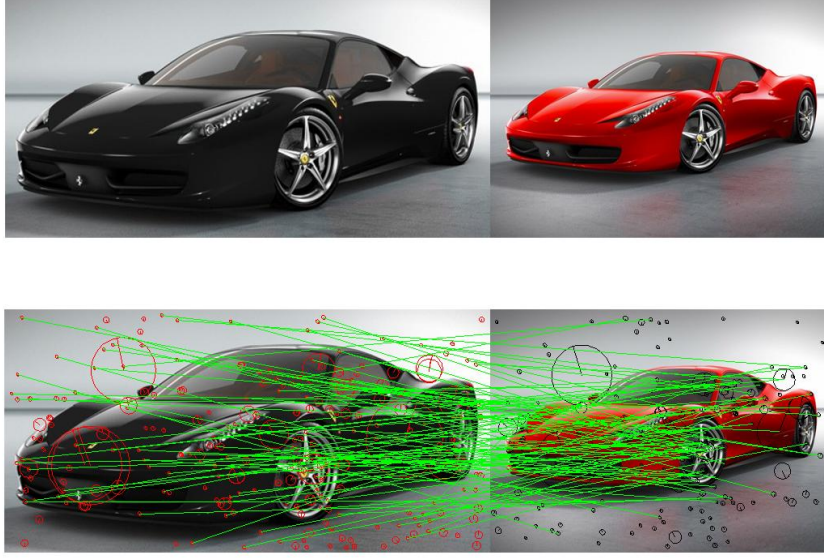


Figure 2.1: Example of matching SIFT features. In this figure, SIFT features are marked with circles and the radial lines denote the orientations. Matching features between two images are connected by the green lines.

where (x, y) is the 2D co-ordinates of a location, \star is the convolution operator and $G(x, y, \sigma)$ is a Gaussian function,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Then, the DoG image, $D(x, y, \sigma)$, is computed as the difference between two Gaussian-blurred images at scales σ and $k\sigma$,

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) \star I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

Once the DoG scale space is obtained, keypoints are identified as the local maxima/minima of all images location. A keypoint is selected if it is the maximum/minimum of its neighbouring points. There are 26 neighbours: eight on the same image and nine on each of the two adjacent scales (see figure 2.3²).

²Source: David G. Lowe (2004) Distinctive image features from scale-invariant key. [19]

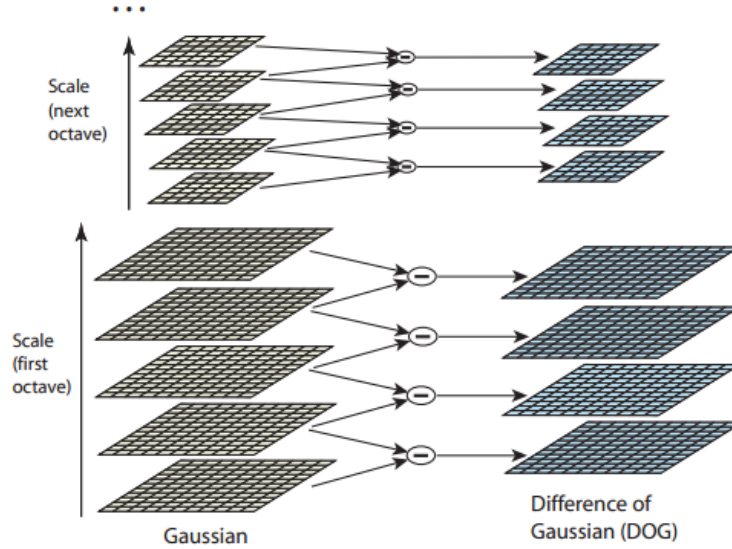


Figure 2.2: Difference of Gaussian scale space. An octave is formed by applying Gaussian function to the initial image at different scales. The DoG images are created by subtracting successive Gaussian-blurred images of the same octave. The initial image is down-sampled by a factor of 2 to make a new initial image. For the next octave, the process is repeated.

2.1.2 Accurate keypoint localisation

Note that the keypoints found in Section 2.1.1 are only approximated extrema as the actual extrema never lies exactly on a pixel (see figure 2.4³). Therefore, the next step is applying a fitted model to the keypoints to find the accurate ones. In order to achieve this, we will use Taylor expansion [5] to interpolate an extrema location in a 3D quadratic function. Formally, if $\Delta x = (x, y, \sigma)^T$ is the offset of the keypoint to the origin, the DoG is approximated :

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

2.1.3 Orientation assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. The following

³Source: <http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/4/> - Utkarsh Sinha - AI Shack - SIFT: Scale Invariant Feature Transform tutorial - as retrieved on May 14, 2012.

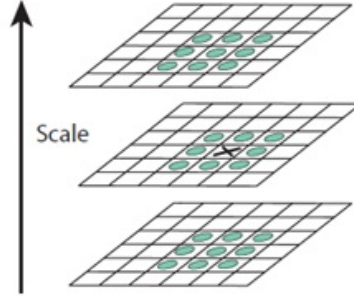


Figure 2.3: Local extrema and its neighbour points. Local extrema of the DoG scale space is identified by comparing a location (marked with X) to its 26 neighboring points in the current and adjacent scales (marked with circles).

approach was found to give the most stable result. The scale of the keypoint is used to select the Gaussian smoothed image, L , with the closest scale, so that all computations are performed in a scale-invariant manner. For each image sample, $L(x, y)$, at this scale, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, are precomputed using pixel differences:

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

$$\theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}$$

2.1.4 SIFT descriptor

The previous operations have assigned an image location, scale and orientation to each keypoint. Now we want to compute a descriptor for the local image patch that is highly distinctive and as invariant as possible to remaining variations, such as change in illumination or 3D viewpoint.

We do this by considering the neighbouring gradient around the keypoints. First, a set of orientation histogram is computed for 16×16 pixels with 8 bins for each direction. The magnitude and orientation of each gradient is weighted by a Gaussian function over the distance to the keypoint. Each of the 4×4 neighbour gradients are sampled into one gradient histogram. This process is illustrated in figure 2.5⁴. Note that the figure demonstrates 2×2 descriptors computed from 8×8 neighbour samples. In the original implementation, 4×4 descriptors are computed from 16×16

⁴Source: David G. Lowe (2004) Distinctive image features from scale-invariant key. [19]

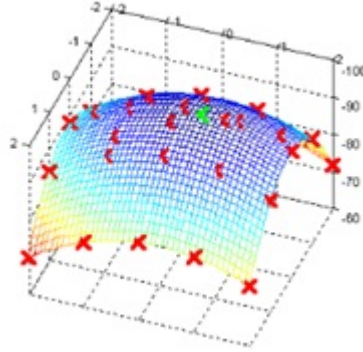


Figure 2.4: Example of keypoints and actual extrema. Keypoints found as local extrema are marked as by red crosses while the actual extrema is marked by the green cross. The actual extrema is approximated by using Taylor expansion.

neighbour samples. Finally, a SIFT descriptor is presented as a 128-vector, where each component is the magnitude of 8 bins of 16 neighbouring gradient histograms.

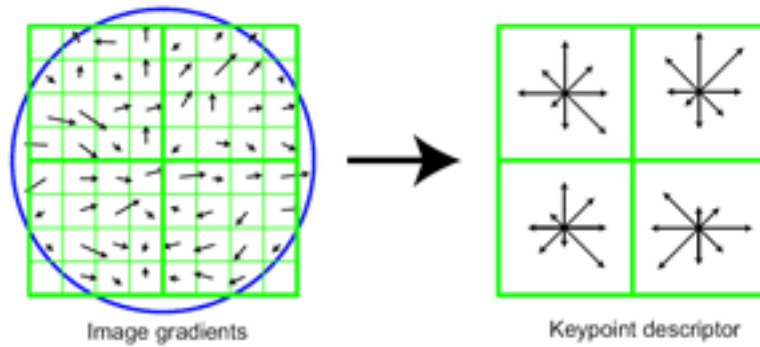


Figure 2.5: SIFT descriptor computed from image gradients. Gradients of neighbourhoods around the keypoint are computed. The magnitude and orientation of each gradient is weighted by a Gaussian window indicated by the blue circle. This figure demonstrates 2×2 descriptors computed from 8×8 neighbour samples. In the implementation, 4×4 descriptors are computed from 16×16 neighbour samples to produce a 128-vector.

2.2 Dense SIFT (dSIFT)

Dense SIFT (dSIFT) descriptors are essentially the same as SIFT descriptors except that instead of computing the keypoint descriptors on the sparse,

highly contrasting keypoints detected by the method described in section 2.1.2, we compute the descriptors on a dense regular grid. This approach has been taken by Fei-Fei and Perona in [9], who have shown that dense features work better for scene classification. The explanation for this is that a set of image descriptors computed over a dense grid usually provide more global information than the corresponding descriptors evaluated at a sparser set of local points [16]. Therefore, we want to see if this approach also works well for classification of images containing objects.

In the implementation, we first resize the original image to the size of 300 pixels, and then compute SIFT descriptors on 20×20 overlapping patches with a spacing of 10 pixels. To make the descriptors scalable, we sample the SIFTs at three scales: 1, 0.5 and 0.25 of the original image. This approach to sample dense SIFT descriptors aligns with the set of dense features pre-computed in the Image-Net database [7]. Figure 2.6 illustrates the process that we sample SIFT descriptors on a dense regular grid.

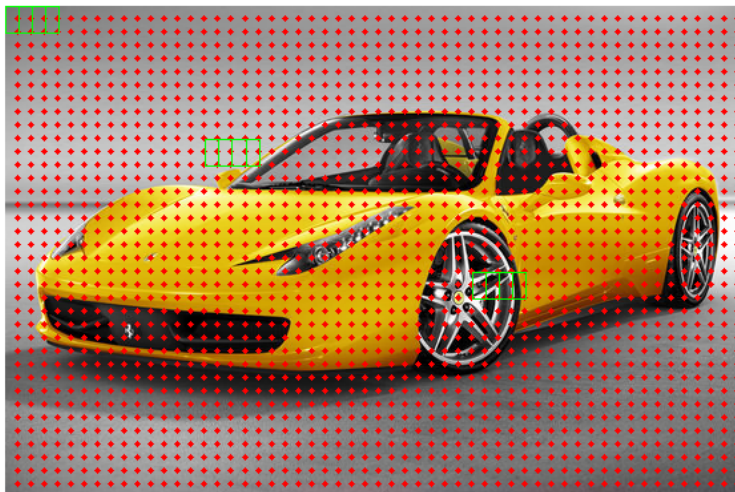


Figure 2.6: The dense SIFT descriptors. The descriptors are denoted as red crosses while the patches are denoted as green windows (mostly omitted for brevity).

2.3 Colour SIFT

As SIFT descriptors are performed on the grayscale, the colour channels are discarded. However, the colour information around a keypoint might



Figure 2.7: In this figure, two different descriptors are made similar as the result of discarding colour information.

be useful in some cases (see figure 2.7⁵). As a result, we can think of incorporating the colour information into our descriptors. Fortunately, there have been many research papers about colour descriptors [2, 31]. Among them, van de Sande et al. [29] make a comparison about the variance and invariance properties of the colour descriptors. In another paper [31], the authors also suggested the use of two colour descriptors on opponent colour space and normalised RGB colour space. From the colour spaces, the colour SIFT descriptors are computed and performed well as the features for image classification. In this section, we are going to briefly introduce the opponent colour spaces and how the colour SIFT descriptors in this colour space are computed.

2.3.1 Opponent SIFT

Opponent histogram van de Sande et al. in [30] have experimented and concluded that Opponent SIFT descriptors are recommended if only a single

⁵Source: Picture taken from "CSIFT: A SIFT Descriptor with Color Invariant Characteristics", by Abdel-Hakim, A.E. and Farag, A.A., in Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on vol. 2, (, 2006), pp. 1978 - 1983.

colour descriptor is used for image classification. Therefore, we are going to use opponent SIFT descriptors as part of our feature vector later (see section 2.4).

Formally, the opponent histogram is a combination of three 1-D histograms based on the channels of the opponent colour space:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \quad (2.1)$$

In this colour space, channel O_3 represents the intensity information while the other two channels represent the colour information. A white light source will be represented when the three channels are equal. The O_1 and O_2 are also shift-invariant with respect to light intensity. The intensity channel O_3 has no invariance properties. The verification of these properties can be found in [31].

OpponentSIFT Opponent SIFT (oppSIFT) describes all the channels in the opponent colour space (equation 2.1) using SIFT descriptors. The information in the O_3 channel is equal to the intensity information, while the other channels describe the colour information in the image. The other channels do contain some intensity information but due to the normalisation of the SIFT descriptor they are invariant to changes in light intensity.

2.4 Feature descriptor

We have had three types of descriptors in hand: SIFT, dSIFT and oppSIFT. Except for the original SIFT descriptors, the other two SIFT descriptors are computed at exactly the same location points. As we have multiple descriptors, we want to try to combine these descriptors into a large feature descriptor so that we can obtain more information at each sample point. One popular way to do this is, at each key point, the feature descriptor will be the concatenation of the two SIFT descriptors. The set of all possible feature vectors establishes the feature space. The size of dSIFT and oppSIFT are 128 and 384, respectively. So, a combined feature descriptor will be a vector of the length $128 + 384 = 512$.

Chapter 3

Image representation

As we have finished examining the small feature representations, it is essential to consider the original image representation. There have been many methods proposed for image representation in the past decade of computer vision research. Most of the state-of-the-art methods in image classification use Bag-of-Words (BoW) model to quantify together the feature descriptors into histograms. Our implementation will follow this model which is presented in section 3.1. Section 3.2 briefly mentions the parameters in the implementation and finally, section 3.3 notes the limitation of the BoW model.

3.1 Bag-of-Words Model

Bag-of-Words (BoW) model is a popular model used in natural language processing and information retrieval to simplify document representation. The key assumption is that a document is considered as an unordered collection of words; we discard any information about grammar or word order. The BoW model is now used for many research papers in image classification in computer vision such as [7, 31, 12].

We first describe the model by giving an example in natural language processing terminology and then we give an analogy to our computer vision application. Suppose we have two simple text documents:

1. Jim likes an apple but Ann likes an orange.
2. Ann likes a movie.

The dictionary is constructed from all the words appearing in the texts: {"Jim", "likes", "an", "apple", "but", "Ann", "orange", "a", "movie"}, which has 9 distinct words. Using the indices in the dictionary, each document is represented by a 9-vector:

- [1, 2, 2, 1, 1, 1, 1, 0, 0], and

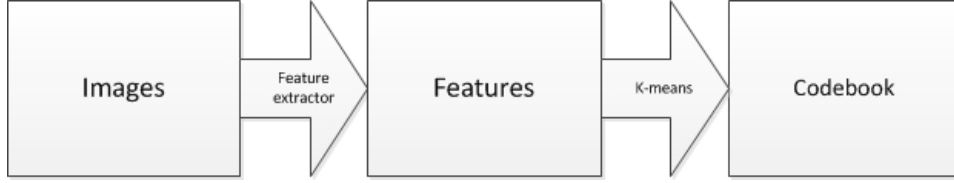


Figure 3.1: Process of making a visual codebook in Bag-of-Words model, as described in section 3.1.

- [0, 1, 0, 0, 0, 1, 0, 1, 1],

which is the histogram representation of frequency count.

To use BoW model for image representation, we can consider each image as a histogram of *visual words* where each *visual word* can be a local image feature that we have examined in chapter 2.

The final step of the BoW model is to generate the “codebook” of visual words (analogy to the word vocabulary). One simple method is performing the popular *k*-means clustering [20] over all feature descriptors from all training images to find clusters of similar features. The visual words are chosen as the centroids of these clusters. In this way, each feature descriptor in an image can be quantified into a cluster and hence an image can be represented by a histogram of visual words. Figure 3.1 presents a flow char that describes the process of making a visual codebook.



Figure 3.2: Representation of an image into a Bag-of-Words features.

3.2 Implementation

We will use the MATLAB function “kmeans” as provided in the Statistics Toolbox. It should also be noted that the complexity of this implementation of *k*-means algorithm is $\mathcal{O}(kmw)$ where *k* is the number of iterations, *m* is the number of descriptors to be clustered and *w* is the number of visual words. Therefore, generating a codebook is provisioned to be the bottle neck of our system. In order to complete the process, we will examine a less sophisticated codebook of size $w = 100$ and 1000 and $m = 10000$. This is much smaller than the codebooks found in the literature where *m* is up to 10 million [7] and *w* is usually from 1000 to 4000[29, 30, 31].

3.3 Limitations of the model

One of the notorious limitation of the Bag-of-Words model is that it removes the spatial layout. This removal increases the invariance to scale, translation and formation. Figure 3.3¹ illustrates the limitation of the BoW model where images in different classes are classified as the same class using Bag-of-Words model because they contain the same number of features.

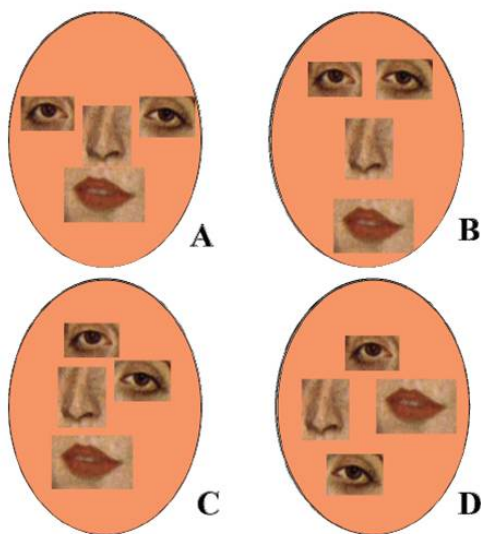


Figure 3.3: An example showing limitation of BoW model. Four bags A, B, C and D contain the same features but C and D should not be in the same class with A and B. In the BoW model, these four bags are classified as the same classes because the spatial information is discarded.

It is also worth noting that to incorporate the spatial information, several other models have been developed and proven successful in PASCAL VOC challenges such as Spatial Pyramid Matching model [16] or Deformable Part-based Models [10]. However, to keep the project within a reasonable size, we decided to work only with the Bag-of-Words model and the extension for spatial information is subject to future research.

¹Source: image taken from slide eccv10_tutorial_part1.ppt, ECCV-2010 Tutorial: Feature Learning for Image Classification, Kai Yu and Andrew Ng [1].

Chapter 4

Image Classification

Chapter 2 and 3 have demonstrated the method of quantifying an original image into a bag of feature descriptors, a format that is easier for a classifier to learn. To address the objectives mentioned in section 1.1, we now proceed to the image classification stage.

In computer vision, image classification is the task of determining whether or not an image contains some specific category of objects. The research of machine learning over 50 years has produced hundreds of classifiers, many of which have been implemented in the Weka suite [14]. Since a popular question of image classification is whether or not a specific object is in the image, a binary classifier is preferred over a scoring regression classifier. Therefore, computer vision research usually utilises the Support Vector Machine which is well-known and proved extremely successful at binary classification tasks [32, 30]. Section 4.1 presents the robust Support Vector Machine and the use of kernel to achieve high accuracy.

4.1 Support Vector Machine

Motivation In machine learning, Support Vector Machine (SVM) is a supervised learning model used for classification and regression analysis. A SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification. Intuitively, we want a hyperplane that has the largest distance to the nearest training data point of any class (or functional margin). In general, the larger the margin, the lower the generalisation error of the classifier.

It is often the case that the data sets to be classified are not linearly separable in that space. For that reason, the original finite-dimensional space can be mapped into a much higher dimension to find a separation. SVM uses a kernel function $k(x, y)$ to serve the mapping purpose while maintaining the linear dot product function in the original space.

Linear SVM Given a training data D of n data points: $D = \{(x_i, y_i) | x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n$, where y_i is either 1 or -1 which are the classes of the data point x_i . Each x_i is a p -dimensional vector. We want to find the maximum-margin hyperplane that divides the points of $y_i = 1$ and $y_i = -1$ (see figure 4.1¹). Any hyperplane can be written as the set of points x satisfying:

$$w \cdot x - b = 0,$$

where \cdot denotes the dot product and w is the normal vector to the hyperplane.

If the training data are linearly separable, we can choose two parallel hyperplanes in a way that they separate the data and there are no points between them, and then try to maximise their distance. The region bounded by them is called the “the margin”. These hyperplanes can be described by the equations:

$$w \cdot x - b = 1 \text{ and } w \cdot x - b = -1$$

As we can see from figure 4.1, the distance between these two hyperplanes is $2/||w||$, so we want to minimise $||w||$. As we also have to prevent data points from falling into the margin, we have the following constraint for each data point x_i either:

$$w \cdot x_i - b \geq 1 \text{ for } x_i\text{'s of the class } y_i = 1, \text{ or}$$

$$w \cdot x_i - b \leq -1 \text{ for } x_i\text{'s of the class } y_i = -1.$$

These inequalities can be rewritten as $y_i(w \cdot x_i - b) \geq 1$ for all $1 \leq i \leq n$.

The optimisation problem of the SVM is:

Minimise $||w||$, subject to

$$y_i(w \cdot x_i - b) \geq 1 \text{ for all } 1 \leq i \leq n.$$

Nonlinear SVM The original optimal hyperplane algorithm was a linear classifier. However, Boser et al. [4] suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. The resulting algorithm is similar, except that every dot product is replaced by a non-linear kernel function. The optimisation can be rewritten as follows:

Minimise $||w||$, subject to

$$y_i(k(w, x_i) - b) \geq 1 \text{ for all } 1 \leq i \leq n.$$

In the implementation, we are going to use a robust, popular kernel-based SVM called Least Square SVM (LSSVM). The details of LSSVM can be found in [26].

¹Source: Image taken from http://en.wikipedia.org/wiki/Support_vector_machine on September 30, 2012 [34]

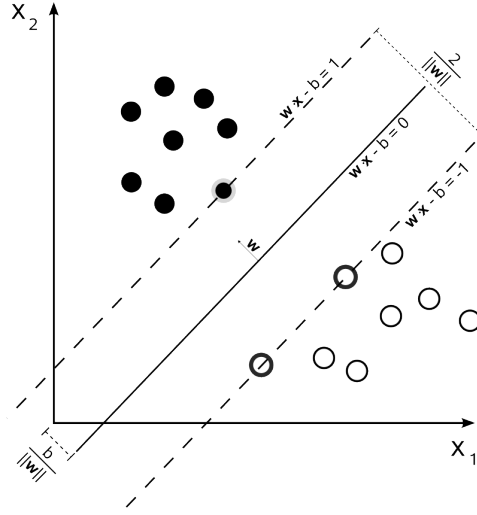


Figure 4.1: Maximum-margin hyperplane as shown with two classes of data points.

4.2 Training and classification model

As we have had the histogram representation of the image, the training model is trivial. We assume that the dataset of positive and negative examples has been obtained. An image is a positive example if it contains the object of interest and a negative example if it does not. Then, the input of the SVM classifier is the histogram vector of the positive and negative example images along with their labels.

A classifier is trained using such data is able to classify a test image by following the similar process: extract the features of the test image, vector quantify the features into a histogram vector and finally fetch the histogram vector to the classifier to get the prediction of the label. Figure 4.2 illustrates the training and classification model of image classification.

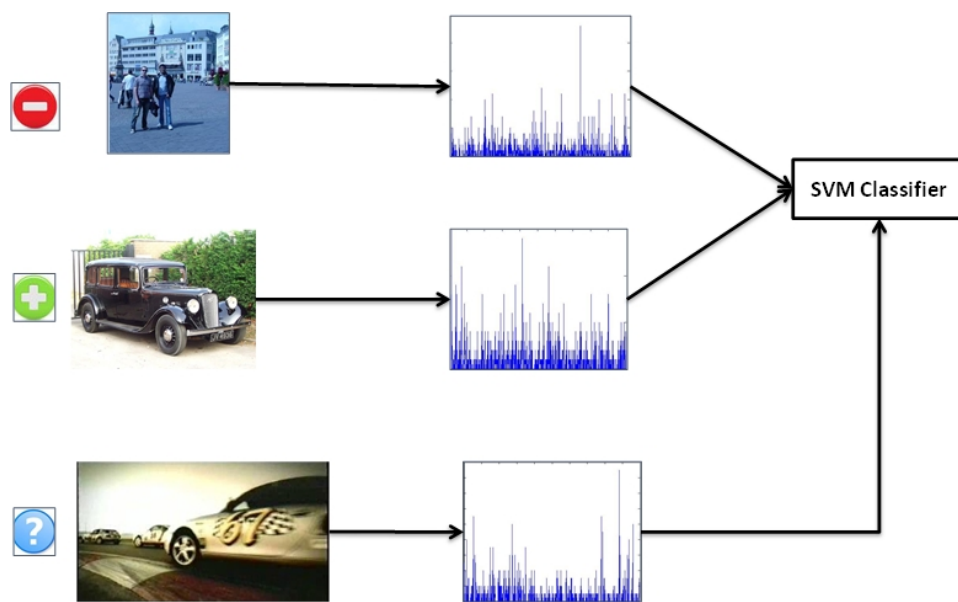


Figure 4.2: Training and classification model for SVM classifier. In this figure, the images in the first two rows are the positive and negative examples while the image in the third row is a test image and has not been seen by the SVM classifier.

Chapter 5

Object detection

Object detection is the task of finding a given object in an image or a video sequence. In order to make our system consistent with the VOC2011 challenge, we find an object by predicting the bounding box covering the target object in the test image. A bounding box is specified by a rectangle with the top-left and bottom-right corner. The ground truth of bounding boxes in the dataset have been manually done by human. A detection is considered as correct if the area of overlap α_0 between the predicted bounding box B_p and ground truth bounding box B_{gt} exceed a certain threshold σ by the formula:

$$\alpha_0 = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (5.1)$$

Figure 5.1 (left) shows examples of correct predictions with threshold σ of 50% as originally proposed in the VOC challenges. However, van de Sande et al. noted in [30] that the threshold σ should be changed to 87.6% to reflect a better prediction. Figure 5.1(right) shows examples of correct predictions with threshold σ of 87.6%.

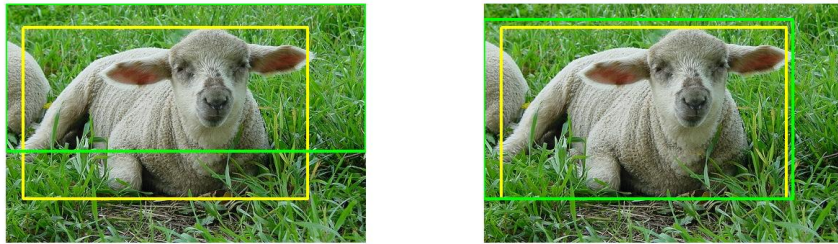


Figure 5.1: Examples of correct prediction with threshold σ of 50% (left) and 87.5% (right). The ground truth is represented by yellow boxes while the predictions are represented by green boxes.

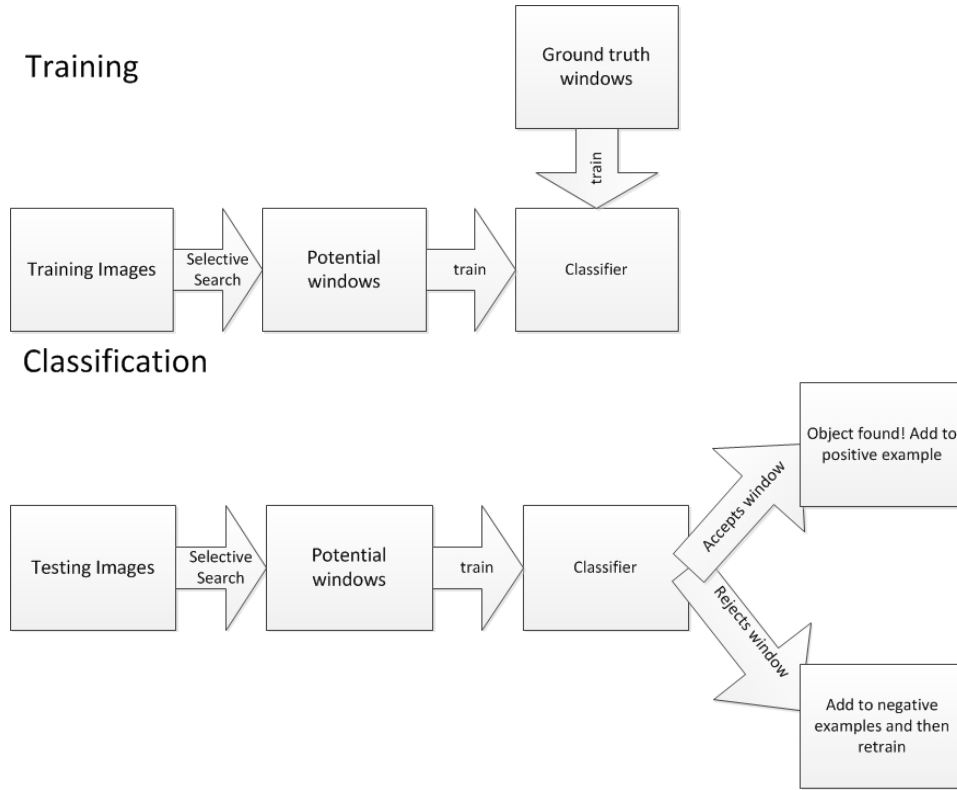


Figure 5.2: Object detection training and classification model.

Having an image classifier as described in chapter 4, the idea of detecting an object is simple. We train an image classifier with sub-image of good and bad locations; extract potential locations for the object in a novel image; give the sub-image of this location to the classifier to determine the result. Figure 5.2 illustrates this idea.

As we have considered various methods for building classifiers in chapter 4, we want to consider a method that is able to efficiently generate possible locations. In this project, we follow a method selective search using segmentation, proposed by van de Sande et al. [30]. The details of this method is described in section 5.1. Then, section 5.2 expands the idea in figure 5.2 in a more details. We follow the training and classification model as described in chapter 4.

5.1 Segmentation as Selective Search

Van de Sande et al. have developed a very robust method in detecting potential object locations in an image [30]. This method uses a hierarchical segmentation in combination with texture features to generate object win-

dows. The method starts with an oversegmentation (i.e. pixels that belong to the same regions do not belong to different objects) such as [11]. The authors made 4 experiments, of that only the first, second and the fourth will be relevant to our project.

In the first experiment, they have shown that selective search using a hierarchical segmentation algorithm outperforms the use of multiple flat segmentations in both measures of object recall rate and number of potential windows. In the experiment, they combined two hierarchical segmentations using a threshold of 100 and 200 in RGB colour space. However, it was not clear why the two thresholds $k_1 = 100$ and $k_2 = 200$ were chosen. Therefore, we are going to experiment their programs and our version of Selective Search with different parameter tuning.

Suppose now the formula of k_i is computed as $k_i = x_i \times \text{length}(\text{image})/5$ where $\text{length}(\text{image})$ is the maximum size of the image. There are two reasons for computing k_i depending on the maximal size of the image. First, from this paper [27], we realise that objects can still be recognised by humans up to a scaling down to 32×32 and still greatly outperforms the state-of-the-art algorithms in object detection. Second, in the preliminary experiments with size-reduced images in VOC 2011 dataset, we realise that using the original tuning parameters (i.e. $k_1 = 100$ and $k_2 = 200$) will result in no object coverage in some cases. The explanation is that the objects become smaller than the initial segmentation that k_i specify, so obviously we need a different set of starting k_i 's. By using the minimum size of starting regions this way, we can resize the original images VOC2011 to a smaller suitable scale and still be able to utilise the potential windows generated by selective search.

To aid the reader's understanding of the details of Selective Search using hierarchical segmentation, we include here the goals and algorithm used in [30]. Segmentation was adapted as selective search for object detection.

Goals

- High recall rate: The idea is that objects whose locations are not generated can never be recognised [30]. Although we are not able to tell exactly which potential location is the desired object location, we want to make sure that at least one of the detected potential location is the desired one. To obtain high recall rate, the authors observe that (1) objects can appear in different scale and (2) there is no single best segmentation strategy to group regions together: An edge may represent an object boundary in one image but it can also be the result of shading or illumination difference. So, instead of trying to create a best segmentation of an object, we try to diversify the segmentations.
- Easy to compute: with the set of 17000 images in VOC2011 waiting and the time frame constraint of the project, it is desirable for the

potential windows to be quickly generated. Suppose a set of windows is created in 1 second per image, it will take more than 4 hours in total to compute just the potential windows. We do not want to make this process a bottle neck for object detection.

- Coarse locations are sufficient: the purpose is to roughly determine the location of objects instead of segmenting object boundaries, the evaluation should focus on finding reasonable approximations of the object locations, such as those measured by the PASCAL overlapping criterion (equation 5.1).

Algorithm

- As a region yields more information than a pixel, we start with an oversegmentation of the original image. A fast method such as [11], is found well-suited for generating oversegmentation.
- A greedy algorithm is used such that at each turn, we group the two most similar regions to become one. The algorithm stops when the whole image becomes a single region. The tight bounding box of each region will be considered as a potential location for the objects.
- Finally, we need to consider the similarity measure between two neighbouring regions a and b , given by $S(a, b) = S_{size}(a, b) + S_{texture}(a, b)$. Both components are computed in range $[0..1]$ and weighted equally.
 - $S_{size}(a, b)$ is defined as one minus the fraction of the image that the segment a and b jointly occupy. This measure encourages small regions to merge early and prevents a single region from grabbing up all others one by one.
 - $S_{texture}(a, b)$ is defined as the fraction of histogram intersection between SIFT-like texture measurements. We compute the SIFT descriptor with no Gaussian weighting in a grid in each region. As we use colour, each colour channel is computed separately and the result is concatenated.

Figure 5.3 presents set of locations found by the Selective Search software [30].

5.2 Training and classification model

Once we are able to find the possible locations, we will need to provide the training data for the classifiers as described in chapter 4. For positive examples, we extract the sub-image of the ground truth provided by VOC2011 and the potential boxes that intersect with the ground truth above the threshold

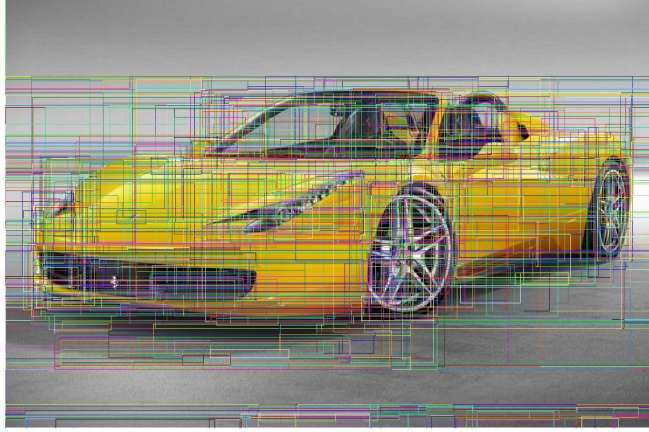


Figure 5.3: Potential windows found by Selective Search software [30]. In this figure, the bounding boxes concentrated on the main object, which is the yellow car rather than the background points. This helps us to focus on classifying the actual object.

σ . On the other hand, the potential boxes that intersect with the ground truth from 20% to 50% are considered as negative examples. We avoid the duplicate negative examples by removing the negatives examples that overlap each other for more than 70%.

Retrain: on validation data, once a bounding box is classified, compare the overlapping region with the ground truth, if the overlapping result is greater than 87%, we add it to the possible example set, if it ranges from 20% to 50% then we add it to the negative example set. The number of retrain iterations is at most 2.

Chapter 6

Analysis and Evaluation

This chapter demonstrates two goals: evaluation of the methods described in the previous chapters on a standard datasets and an analysis of the method's performance.

There are a few computer vision benchmarks being focused on the field, such as PASCAL VOC, and the idea is that a “significant” improvement is if a classifier can produce a statistic measure significantly larger than by another method. In the VOC contests, the measures are the average precision (AP) over each category and the mean AP over all categories. So, an increase in AP points would mean an improvement in accuracy. However, there have been no reports on the time performance of the classifier so there will be no comparison of this criterion.

PASCAL VOC 2011 [8] contains three non-overlapping datasets called *train*, *val* and *test*. The *train* and *val* datasets are fully annotated and given for users to test their classifier locally. The two datasets contains 5717 and 5823 images of 20 classes of objects altogether. The *test* dataset is provided for user without any annotation or labelling. Competitors are required to train their classifier on *train+val* dataset and submit the labellings (for classification task) or bounding boxes (for detection task) of the images on *test* dataset.

6.1 Image classification

In order to evaluate the image classification performance, we propose the following three experiments:

1. Desirable codebook size: we are going to use the same dSIFT descriptors computed on the training set of VOC 2011. The codebooks generated have the size of 100 and 1000 visual words. We want to see if it is better to have more visual words for training and classifying images.

2. Desirable feature descriptor: we want to see the performance of the single descriptors SIFT, dSIFT, dense oppSIFT as opposed to the large combined feature descriptor in image classification. The same SVM classifier is used and the same for all four type of descriptors.
3. Our classifier vs. state-of-the-art methods: we want to compare the performance of our best classifier against the state-of-the-art methods.

Before we go into the experiments, it should be noted that we will be using the MATLAB function “svmtrain” and “svmclassify” for SVM classifier. The linear SVM version using this function does not converge for medium-size dataset such as *test* dataset, so we opted to use LSSVM (svmtrain with “least square” option) for our classifiers.

6.1.1 Desirable codebook size

The following table demonstrates the AP of three image classifiers using Bag-of-Words model with dense SIFT descriptors trained and classified with codebooks of size 100 and 1000. As the classifier is LSSVM, we name the three classifiers as dSIFT+100+LSSVM, dSIFT+200+LSSVM and dSIFT+1000+LSSVM, respectively. We will use *train* dataset for training and *val* dataset for testing and swap the two dataset to perform 2-fold cross validation. The result is presented as the AP points on image dataset.

The table 6.1 comes with an unexpected result. For a codebook size of 10 times larger, the mean AP performance slightly decreases. However, when we consider the individuals cases, dSIFT+1000+LSSVM still performs better than dSIFT+100+LSSVM in 7 out of 20 classes (comparing train-val). One of the shortcomings in the experiment is that the use of the LSSVM has not been examined in details and also the performance of the built-in SVM classifier from the Bioinformatics ToolBox was not well examined.

However, as both programs used the same classifiers the relative meaning of the result is reliable. From this result, we may argue that dense SIFT descriptors have already contains the global information of the image. Therefore, the increase in size of the BoW codebook is unnecessary.

6.1.2 Desirable feature descriptor

In the second experiment, we are going to compare the performance of the four single feature descriptors: SIFT, dSIFT, oppSIFT and combined SIFT in combination with linear SVM on two selected categories of the ImageNet dataset [7]. In particular, we are going to examine the discriminative performance on classes “car” and “dog”. The size of the training set is just 20 so it is possible to perform using a linear SVM in this case. The number of of testing images are 100, with the split of 50/50 positive/negative examples.

	Class	dSIFT+100+LSSVM		dSIFT+1000+LSSVM	
		train-val	val-train	train-val	val-train
1	aero	26.6	25.8	31.1	30.0
2	bike	10.8	10.4	9.7	9.8
3	bird	11.7	13.7	11.9	12.8
4	boat	16.0	15.4	17.5	18.5
5	bottle	10.1	11.7	8.4	10.5
6	bus	13.6	13.2	15.2	14.1
7	car	23.9	24.1	19.3	20.4
8	cat	21.4	20.8	21.3	21.3
9	chair	19.3	21.4	17.6	17.6
10	cow	5.5	6.2	4.6	6.7
11	table	11.5	10.5	7.9	7.9
12	dog	19.2	20.6	18.8	15.9
13	horse	12.4	10.1	10.0	10.8
14	mbike	10.0	9.5	10.9	11.5
15	person	60.2	57.7	59.4	58.1
16	plant	6.9	7.8	6.4	7.5
17	sheep	8.6	9.1	8.1	7.8
18	sofa	10.4	9.3	8.4	10.2
19	train	14.2	13.0	15.3	14.3
20	tv	12.3	14.6	14.9	12.9
	Mean	16.24+/-0.01		15.87+/-0.06	

Table 6.1: Image classification performance on different codebook sizes. The table illustrates the performance of three classifiers: dSIFT+100+LSSVM and dSIFT+1000+LSSVM on VOC 2011, training and testing on *train* and *val* dataset.

Figure 6.1 reports the RoC of three classifiers SIFT+SVM, dSIFT+SVM and oppSIFT+SVM. From the table, we can see that oppSIFT+SVM does not performs as well as it was claimed. The original SIFT descriptor is the best and recommended for image classification.

6.1.3 Comparison with the state-of-the-art methods

In this section we are going to show the performance of our classifier with the state-of-the-art classification methods in the VOC2011 challenge. We have selected here three classifiers with the best performance over 20 classes that have also influenced our work. The description of their methods are in appendix B. We present here briefly the methods used in those classifiers and their references. However the details of them are beyond the scope of this thesis.

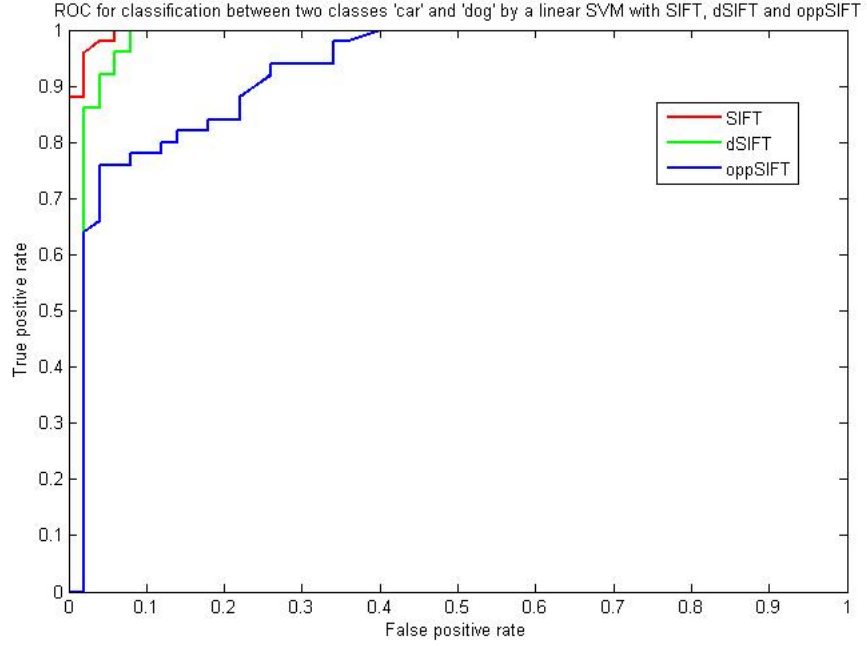


Figure 6.1: Performance of linear SVM clasifier with different types of features.

1. NLPR_SS_VW_PLS: Performed best for 8 out of 20 classes.
2. NUSPSL_CTX_GPM: Performed best for 11 out of 20 classes.

6.2 Object detection in the whole system

We have been concentrating on image classification in this project so far. However, let us not forget the overall system to be developed in this project. Table 6.3 presents several examples of video sequences for object detection to prove that the proposed system is capable of detecting objects in the videos.

	class	dSIFT+1000+LSSVM	dSIFT+100+LSSVM	NLPR	NUSPSL
1	aero	31.1	25.8	94.5	<u>95.5</u>
2	bike	9.7	10.4	<u>82.6</u>	81.1
3	bird	11.9	13.7	<u>79.4</u>	79.4
4	boat	17.5	15.4	<u>80.7</u>	<u>82.5</u>
5	bottle	8.4	11.7	57.8	58.2
6	bus	15.2	13.2	<u>87.8</u>	87.7
7	car	19.3	24.1	<u>85.5</u>	84.1
8	cat	21.3	20.8	83.9	83.1
9	chair	17.6	21.4	66.6	<u>68.5</u>
10	cow	4.6	6.2	<u>74.2</u>	<u>72.8</u>
11	table	7.9	10.5	69.4	68.5
12	dog	18.8	20.6	75.2	<u>76.4</u>
13	horse	10.0	10.1	83.0	<u>83.3</u>
14	mbike	10.9	9.5	88.1	87.5
15	person	59.4	57.7	<u>93.5</u>	92.8
16	plant	6.4	7.8	56.2	<u>56.5</u>
17	sheep	8.1	9.1	75.5	<u>77.7</u>
18	sofa	8.4	9.3	64.1	<u>67.0</u>
19	train	15.3	13.0	90.0	<u>91.2</u>
20	tv	14.9	14.6	76.6	<u>77.5</u>
	Average	15.8	16.25	78.2	78.6

Table 6.2: Comparison between our classifier and the state-of-the-art methods.





	Good examples	Bad examples
Dog		
Car		

Table 6.3: Examples of successful detection and misclassification in classes “dog” and “car”.

Chapter 7

Conclusion and Future Work

Overall, we have demonstrated in section 6.2 that a system that is able to detect objects in a video sequence by plotting tight bounding boxes covering the objects. However, the performance of the method, as a comparison with the state-of-the-art methods 6.1.3 is still subject to further research.

Future work may be carried in several different directions:

1. Improve the feature representation: although there was consideration of incorporating multiple feature descriptors into one, the state-of-the-art methods show that the use of 4-6 feature descriptors are commonly done. Therefore one future direction is to compute more feature descriptors, make a comparison among themselves and when combining with each other.
2. Improve the image representation: as provisioned in section 3.2 and experimentally proved in section 6.1.1, the use of small codebooks reduces the performance of classification substantially. Moreover, due to the limitation of BoW model addressed in section 3.3, we may want to use a codebook of larger size of 2000 to 4000 and incorporating the spatial information such as in Spatial Pyramid Matching [16] or Deformable Part-based model [10].
3. Improve classifier kernels: the use of linear SVM results in a fast but weak classifier. Another well recommended kernel in the literature such as Histogram Intersection Kernel [13] with efficient approximation [21] and sparse coding such as [17] is said to give near the best performance in VOC challenges.

Bibliography

- [1]
- [2] Alaa E. Abdel-Hakim and Aly A. Farag. Csift: A sift descriptor with color invariant characteristics. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1978–1983, Washington, DC, USA, 2006. IEEE Computer Society.
- [3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.
- [4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.
- [5] Matthew Brown and David G. Lowe. Invariant features from interest point groups. In *In British Machine Vision Conference*, pages 656–665, 2002.
- [6] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.
- [7] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248 –255, june 2009.
- [8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. <http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html>.

- [9] Li Fei-fei. A bayesian hierarchical model for learning natural scene categories. In *In CVPR*, pages 524–531, 2005.
- [10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.
- [11] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.
- [12] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, aug. 2010.
- [13] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *In ICCV*, pages 1458–1465, 2005.
- [14] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
- [15] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conference*, pages 147–151, 1988.
- [16] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.
- [17] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *In NIPS*, pages 801–808. NIPS, 2007.
- [18] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision - Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.
- [19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.
- [20] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

- [21] Subhransu Maji, Alexander C. Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.
- [22] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *CVPR*, pages 1038–1045. IEEE, 2009.
- [23] J. McCarthy. What is artificial intelligence? *Int. J. Comput. Vision*, November 2007.
- [24] Hans P. Moravec. Rover visual obstacle avoidance. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI’81, pages 785–790, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.
- [25] Cordelia Schmid and Roger Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.
- [26] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Process. Lett.*, 9(3):293–300, June 1999.
- [27] A. Torralba, R. Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, November 2008.
- [28] A. M. Turing. Computers & thought. chapter Computing machinery and intelligence, pages 11–35. MIT Press, Cambridge, MA, USA, 1995.
- [29] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Empowering visual categorization with the gpu. *IEEE Transactions on Multimedia*, 13(1):60–70, 2011.
- [30] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *IEEE International Conference on Computer Vision*, 2011.
- [31] Koen van de Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1582–1596, September 2010.
- [32] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
- [33] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

- [34] Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2012. [Online; accessed 30-September-2012].

Appendix A

How to run the code

Most of the source code files were written in Matlab and some were written in C++ and imported as a mex file to Matlab in order to improve running performance.

The function API is as follow:

Function name	Usage
compute_dsift	compute dSIFT descriptors of an image
compute_hog	compute HOG descriptors of an image
compute_oppSIFT	compute opponent SIFT descriptors of an image
compute_comdesc	compute combined SIFT descriptors of an image
compute_codebook_HOG	compute HOG codebook via k-means
compute_codebook_oppSIFT	compute oppSIFT codebook via k-means
selectiveSearch	compute boundingbox for an image

First, we need to run the file “startup.m” to load the VFleat library.

- To load an image, use the default Matlab function:

```
img = imread( 'file_name' );
```

- To compute dSIFT descriptor of image *img*:

```
[f, d] = compute_vl_dsift(img);
```

where f is an array of feature points and d is an array of the corresponding dSIFT descriptors.

- To compute HoG descriptors of image *img*:

```
hogs = compute_vl_hog(img);
```

where hogs is an array of HoG descriptors.

- To compute oppSIFT descriptors of image *img*:

```
oppSIFT = compute_vl_oppSIFT(img);
```

- To compute combined descriptors of image *img*:
`desc = compute_combined_desc(img);`

Appendix B

Description of the state-of-the-art methods

The appendix is here to give more details of the state-of-the-art methods that we have compared in section 6.1.3. The descriptions are taken directly from the result page of [8]. These descriptions, written by their respective authors, are required for the entry to the VOC competitions.

- **NUSPSL_CTX_GPM** : The whole solution for object classification is based on BoW framework. On image level, Dense-SIFT, HOG², LBP and color moments features are extracted. VQ and Fisher vectors are utilized for feature coding. Traditional SPM and novel spatial-free pyramid matching scheme are then performed to generate image representations. Context-aware features are also extracted based on detection result [1]. The classification models are learnt via kernel SVM. The final classification scores are refined with kernel mapping [2]. The key novelty of the new solution is the pooling strategy (which well handles the spatial mismatching as well as noise feature issues), and considerable improvement has been achieved as shown in other offline experiments. [1] Zheng Song, Qiang Chen, Zhongyang Huang, Yang Hua, and Shuicheng Yan. Contextualizing Object Detection and Classification, CVPR 2011. [2] <http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/workshop/nuspsl.pdf>
- **NLPR_SS_VW_PLS**: The framework is based on the classical Bag-of-Words model. The system consists of: 1) In feature level, we use Semi-Semantic and non-semantic Visual Words Learning, with Fast Feature Extraction (Salient Coding, Super-Vector Coding and Visual Co-occurrence) and multiple features; 2) To learn class model, we employ an alternative multiple-linear-kernel learning for intra-class feature combination, after using Partial Least Squares analysis, which projects the extremely high-dimensional features into a low-dimensional space; 3) The combination of 20 categories scores and detections scores

generate a high-level semantic representation of image, we use non-linear-kernel learning to extract inter-class contextual information, which further improve the performance. Besides, all the parameters are decided by cross-validation and prior knowledge on VOC2007 and VOC2010 trainval sets. The motivation and novelty of our algorithm: The traditional codebook describes the distribution of feature space, containing less semantic information of the interesting objects, and a semantic codebook may benefit the performance. We observe that the Deformable-Part-Based Model [Felz TPAMI 2010] describes the object by “object parts”, which can be seen as the semi-semantic visual words. Based on this idea, we propose to use the semi-semantic and non-semantic visual words based Bag-of-Words model for image classification. We analyze the recent image classification algorithms, finding that the feature “distribution”, “reconstruction” and “saliency” is three fundamental issues in coding and image description. However, these methods usually lead to an extremely high-dimensional description, especially with multiple features. In order to learn these features by MKL, we find Partial Least Square is a reliable method for dimensionality reduction. The compression ratio of PLS is over 10000, while the discrimination can be preserved.

Appendix C

Source code

```

function sig = compute_dSIFT_signature(img)
    global visualWord; % Codebook of dSIFT
    % visualWord = importdata('Codebooks\codebook_dSIFT_1000.txt');

    %% compute signature of this image - as the histogram of the codebook
    [f, desc] = compute_dSIFT(img);
    desc = double(desc');

    sig = mex_convert_vldsift2hist(desc, visualWord);

end

function compute_files_dSIFT
    %startup;
    dir_path = 'VOCdevkit/VOC2012/JPEGImages/';
    if (exist('VOCallFiles.mat', 'file'))
        load('VOCallFiles.mat', 'allFiles')
    else
        allFiles = getAllFiles(dir_path);
        save VOCallFiles allFiles;
    end

    MAX_SIZE = 10^4;

    if (exist('dSIFT_desc_10000.mat', 'file'))
        desc = importdata('dSIFT_desc_10000.mat');
    else
        desc = [];
    end

    if (~exist('dSIFT', 'dir'))
        mkdir dSIFT
    end
    for i = 1:size(allFiles, 1)+1

        if (i == size(allFiles, 1)+1)
            save('dSIFT_desc_10000.mat', 'desc');
            break;
        end
        if (rand > 0.01)
            continue;
        end
        [p, n, e] = fileparts(allFiles{i});
        filename = sprintf('dSIFT/%s_dSIFT_fd.mat', n);
        if exist(filename, 'file')
            dsift = importdata(filename);
        else
            img = imread(allFiles{i});
            [f, dsift] = compute_dSIFT(img);
            dsift = dsift';
            save(filename, 'dsift');
        end

        desc = [desc; dsift];

        if (size(desc, 1) > MAX_SIZE)
            ind = zeros(1, size(desc,1) - MAX_SIZE);
            for j = 1:size(desc, 1) - MAX_SIZE
                ok = false;
                while ~ok
                    ind(j) = round(rand * size(desc, 1) + 0.5);

                    if (ind(j) <= 0) || (ind(j) > size(desc,1))
                        ok = false;
                        continue;
                    end

                    ok = true;
                    for k = 1:j-1
                        if (ind(j) == ind(k))
                            ok = false;
                            break;
                        end
                    end
                end
            end
            [desc] = removeverrows(desc, 'ind', ind);
        end
    end
end

```

```

        end

        fprintf('%d file %s, number = %d \n', i, n, size(desc,1));
    end
end

function [f, d] = compute_dSIFT(img)
    % Compute the vl_dsift as in ImageNet for a new image
    % ***** remember to call vl_setup before running this function

    global MAX_LENGTH;
    MAX_LENGTH = 300;

    if (length(img) > MAX_LENGTH)
        resize_ratio = MAX_LENGTH/length(img);
        img = imresize(img, resize_ratio);
    end

    if (length(size(img)) == 3)
        img = rgb2gray(img);
    end
    I1 = single(img);
    I2 = imresize(I1, 0.5);
    I3 = imresize(I1, 0.25);

    [f1, d1] = vl_dsift(I1, 'step', 10, 'size', 5);
    for i = 1:size(f1, 2)
        f1(3, i) = 0;
    end
    [f2, d2] = vl_dsift(I2, 'step', 10, 'size', 5);
    for i = 1:size(f2, 2)
        f2(3, i) = 1;
    end
    [f3, d3] = vl_dsift(I3, 'step', 10, 'size', 5);
    for i = 1:size(f3, 2)
        f3(3, i) = 2;
    end

    f = [f1,f2,f3];
    d = [d1,d2,d3];

end

function compute_codebooks_dSIFT

dSIFTs = importdata('dSIFT_desc_10000.mat');
opts = statset('Display', 'final');

dSIFTs = double(dSIFTs);
tic;
[D, C] = kmeans(dSIFTs(1:10000,:), 100, 'emptyaction', 'singleton', 'Options', opts,
'replicates', 2);
save codebook_dSIFT_200.mat C;
toc;

dSIFTs = double(dSIFTs);
tic;
[D, C] = kmeans(dSIFTs(1:10000,:), 200, 'emptyaction', 'singleton', 'Options', opts,
'replicates', 2);
save codebook_dSIFT_200.mat C;
toc;

dSIFTs = double(dSIFTs);
tic;
[D, C] = kmeans(dSIFTs(1:10000,:), 400, 'emptyaction', 'singleton', 'Options', opts,
'replicates', 2);
save codebook_dSIFT_400.mat C;
toc;

end

function dSIFT_classifier

% change this path if you install the VOC code elsewhere
addpath([cd 'VOCdevkit/VOCcode']);
global visualWord;
visualWord = importdata('Codebooks\codebook_dSIFT_400.mat');

```

```

% initialize VOC options
VOCinit;

fid = fopen('result_dSIFT_400_train_val.txt', 'w');
% train and test classifier for each class
tic;
for i=1:VOCopts.nclasses
    cls=VOCopts.classes{i};
    classifier=train(VOCopts,cls); % train classifier
    test(VOCopts,cls,classifier); % test classifier
    [recall,prec,ap]=VOCevalcls(VOCopts,'comp1',cls,true); % compute and display PR

    %save(sprintf('classifier_dSIFT_100_%s', cls), 'classifier');
    fprintf(fid, '%s %f\n', cls, ap);

    if i<VOCopts.nclasses
        fprintf('press any key to continue with next class...\n');
        %drawnow;
        %hold on;
        %pause;
    end
end
toc;
fclose(fid);

% train classifier
function classifier = train(VOCopts,cls)

% load training set for class
[ids,classifier.gt]=textread(sprintf(VOCopts.clsimgsetpath,cls,VOCopts.trainset),'%s %d');

% extract features for each image
classifier.FD=zeros(0,length(ids));

tic;
for i=1:length(ids)

    if (classifier.gt(i) == 0)
        classifier.gt(i) = 1;
    end
    % display progress
    if toc>1
        fprintf('%s: train: %d/%d\n',cls,i,length(ids));
        %drawnow;

        tic;
    end

    fdp=sprintf(VOCopts.exfdpath,ids{i});
    if exist(fdp,'file')
        % load features
        load(fdp,'fd');
    else
        % compute and save features
        I=imread(sprintf(VOCopts.imgpath,ids{i}));
        fd=extractfd(VOCopts,I);
        save(fdp,'fd');
    end

    classifier.FD(1:length(fd),i)=fd;
end

%opts = optimset('Algorithm', 'interior-point-convex');

classifier.SVMStruct = svmtrain(classifier.FD', classifier.gt, 'method', 'LS');

% run classifier on test images
function test(VOCopts,cls,classifier)

% load test set ('val' for development kit)
[ids,gt]=textread(sprintf(VOCopts.imgsetpath,VOCopts.testset),'%s %d');

% create results file
fid=fopen(sprintf(VOCopts.clsrespath,'comp1',cls),'w');

% classify each image
tic;

```



```

for i=1:length(ids)
    % display progress
    if toc>1
        fprintf('%s: test: %d/%d\n',cls,i,length(ids));
        %drawnow;
        tic;
    end

    fdp=sprintf(VOCopts.exfdpath,ids{i});
    if exist(fdp,'file')
        % load features
        load(fdp,'fd');
    else
        % compute and save features
        I=imread(sprintf(VOCopts.imgpath,ids{i}));
        fd=extractfd(VOCopts,I);
        save(fdp,'fd');
    end

    % compute confidence of positive classification
    c=classify(VOCopts,classifier,fd);

    % write to results file
    fprintf(fid,'%s %f\n',ids{i},c);
end

% close results file
fclose(fid);

% trivial feature extractor: compute mean RGB
function fd = extractfd(VOCopts,I)

    fd=compute_signature_dSIFT(I);

% trivial classifier: compute ratio of L2 distance between
% nearest positive (class) feature vector and nearest negative (non-class)
% feature vector
function c = classify(VOCopts,classifier,fd)

    c = svmclassify(classifier.SVMStruct, fd);

```