TOWARDS EFFICIENT AND SCALABLE VISUAL PROCESSING IN IMAGES AND VIDEOS

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science

by

Mihir Jain
200707014
mihir@research.iiit.ac.in

Center of Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2010

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled “Towards Efficient and Scalable Visual Processing in Images And Videos” by Mihir Jain, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. C. V. Jawahar

न हि ज्ञानेन सदृशं पवित्रमिह विद्यते।
तत्स्वयं योगसंसिद्धः कालेनात्मनि विन्दति॥

In this world there is no purifier as great as Knowledge; he who has attained purity of heart through prolonged practice of Karmayoga, automatically sees the light of Truth in the self in course of time.

Shrimad Bhagvad Gita (chapter 4, verse 38)

To my loving parents and dear brother

Acknowledgments

I would like to thank Dr. C. V. Jawahar for his support and guidance during the past 3 years. He has been a great influence on me as a researcher and has guided me in academic as well as non academic pursuits. I gratefully acknowledge all his help, patience, advice and effort to make me a better researcher and more accomplished individual. I am also thankful to Dr. Andrew Zisserman for his support, guidance and encouragement.

I would also like to thank Dr. K. Madhav Krishna, Dr. Anoop Namboodiri, Dr. Jayanthi Sivaswamy and Dr. P. J. Narayanan for their references and guidance in different subjects related to the stream.

I would also like to thank all my lab mates at CVIT for creating a friendly working environment.

My thanks must go to Rahul, Avinash, Sreekanth, Chandrika, Raman, Maneesh, Gopal, Gururaj, Pratyush and Kartheeka for their invaluable help and suggestions during this work. I thank all my friends for their support and love.

Above all I am thankful to my parents and my brother, Himalaya, for their support, understanding, unconditional love and inspiration.

Abstract

The amount of multimedia content produced and made available on Internet and in professional and personal collections is constantly growing. Equally increasing are the needs in terms of efficient and effective ways to manage it. This has led to a great amount of research into content based retrieval and visual recognition. In this thesis, we focus on efficient visual content analysis in images and videos. Efficiency has emerged as one of the key issues with increase in quantity of data. Understanding of a visual content has several aspects associated with it. One can concentrate on recognizing the inherent characteristics of image (independent or from a video) like objects, scene and context. Searching for a sequence of images based on similarity or characterizing the video based on its visual content could be some other aspects.

We investigate three different approaches for visual content analysis in this thesis. In the first, we target the detection and classification of different object and scene classes in images and videos. The task of classification is to predict the presence of an object or a specific scene of interest in the test image. Object detection further involves localizing each instance of the object present. We do extensive experimentation over very large and challenging datasets with large number of object and scene categories in it. Our detection as well as classification are based on Random Forest combined with combinations of different visual features describing shape, appearance and color. We exploited the computational efficiency in both training and testing, and other properties of Random Forest for detection and classification. We also proposed enhancements over our baseline model of object detector. Our main contribution here is that we achieve fast object detection with accuracy comparable to the state of art.

The second approach is based on processing continuous stream of videos to detect video segments of interest. Our method is example-based where visual content to be detected or filtered is characterized by a set of examples available *apriori*. We approach the problem of video processing in a manner complimentary to that of video retrieval. We begin with a set of examples (used as queries in retrieval) and index them in the database. The larger video collection, which needs to be processed, is unseen during the off-line indexing phase. We propose an architecture based on trie data structure and bag of words model to simultaneously match multiple example videos in the database with the input large video stream. We demonstrate the application of our architecture for the task of content based copy detection (CBCD).

In our third and final approach we apply pattern mining algorithms in videos to characterize the visual content. They are derived out of data mining schemes for efficient analysis of the content in video databases. Two different video mining schemes are employed; both aimed at detecting frequent and representative patterns. For one of our mining approaches, we use an efficient frequent pattern mining algorithm over a quantized feature space. Our second approach uses random forest to represent video data as sequences, and mine the frequent sequences. We experiment on broadcast news videos to detect what we define as *video stop-words* and extract the contents which are more important such as breaking news. We are also able to characterize the movie videos by automatically identifying the characteristic scenes and main actors of the movie.

The ideas proposed in the thesis have been implemented and validated with extensive experimental results. We demonstrate the accuracy, efficiency and scalability of all the three approaches over large and standrad datasets like *VOC PASCAL, TRECVID, MUSCLE-VCD* as well as movie and news datasets.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction and Objective

The world is a rich and complex source of visual information, which is often captured as images and videos. With the development of multimedia and proliferation of cheap equipments, amount of visual information available in digital form is growing at an exponential rate. It is spreading expeditiously worldwide due to rapid distribution through internet (sites like Flickr, YouTube etc). Millions of images are indexed by Image search engines such as Google Images (claims 880 million), Picsearch (claims 2 billion). Every minute, around 13 hours of video are uploaded to YouTube. Video has become widely spread medium for serving entertainment, education, communication and other purposes. In order to cater to the needs arising from such databases and take advantage of them, advanced techniques for recognition, indexing, mining of such visual content become more and more important. There is a need to automatically extract high-level information from image or video. It can be the objects contained in an image or shot with or without its location, image and video search, semantic characterization of videos etc. And it is very important to do these tasks efficiently in order to meet the user demands. Some examples of sources of visual data are shown in Figure 1.1.

Computer vision methods and specifically visual recognition along-with different indexing techniques, can help to cope with the increasing size of video and image collections. Visual image analysis and recognition has been a subject to research by many people in last 40 years. Initially much of research in visual recognition was focused on 2D pattern classification. Gradually methods based on learning global appearance [97, 112, 126] and geometric invariance [121, 145] came up. This methods were sensitive to clutter, occlusions and object articulations and had limited applicability. Limitations of global features were overcome with the advent of local features [50, 74, 127].

Bag-of-words model has a long history of success in document retrieval, but it was not until the idea of a visual vocabulary emerged [137], that it was possible to bring this model to vision. Since then great progress can be seen in the area of visual recognition, content based retrieval, image/video understanding in general. Different Bag-of-words feature descriptors combined with modern machine

(a) Objects

(b) Scenes

(c) Movies (d) Broadcast video

Figure 1.1 Examples of different sources of visual data: images containing objects and scenes (top and middle); and movie and broadcast videos (bottom)

learning techniques led to excellent performance [29, 111, 135] for detection, retrieval and other visual recognition tasks.

Other than learning based methods for recognition, many indexing techniques have been proposed for image and video retrieval/search. Some of the successful indexing schemes for multimedia collections include LSH [63], min-hash [27, 28], pyramid match hashing [54], vocabulary forest [171], etc. Content based image and video retrieval has benefitted by incorporating better local descriptors, local invariant region detectors, indexing methods from visual recognition and text retrieval communities. Region detectors include DoG (Difference of Gaussian), LoG (Laplacian of Gaussian), MSER, Harris Affine, Hessian Affine (see [92] for details). SIFT [84], PCA-SIFT [68], SURF [14] and DAISY [146] are the more popular descriptors. Along with Bag-of-words model these feature descriptors have been used for variety of tasks. The power of Bag-of-words model to create efficient image and video retrieval systems has been explored by Sivic and Zisserman [137] as well as Nister and Stewenius [101]. Moreover, with this model it is easier to adapt many text based indexing methods for images and videos. For instance, adaptation of PLSA [58] and LDA [17] to visual bag of words has provided promising results for static image databases [19, 117, 122, 136].

1.1.1 Objective

Our goal is to efficiently process, search, filter, analyze videos and large image collections. The objective is to understand different aspects of visual information contained in the given image or video. It can be finding out what scene or object categories are present by learning models for those categories. Some concepts like demonstration, violence, party are difficult to learn. In such cases, example based approach can be taken for detecting/filtering content of interest (which are similar to examples). Automatic discovery of main actors in movies, important content in news videos and other characteristic patterns is another way to understand and analyse video content.

More specifically the following problems are addressed in this thesis:

- Object localization in cluttered environment and scene/object classification.
- On-line processing and filtering of videos.
- Mining and finding characteristic patterns in videos.

The emphasis is on achieving above with high efficiency. We now further explain our objective of understanding visual content in image and video. We take two examples one for each image and video to justify the importance of our objective.

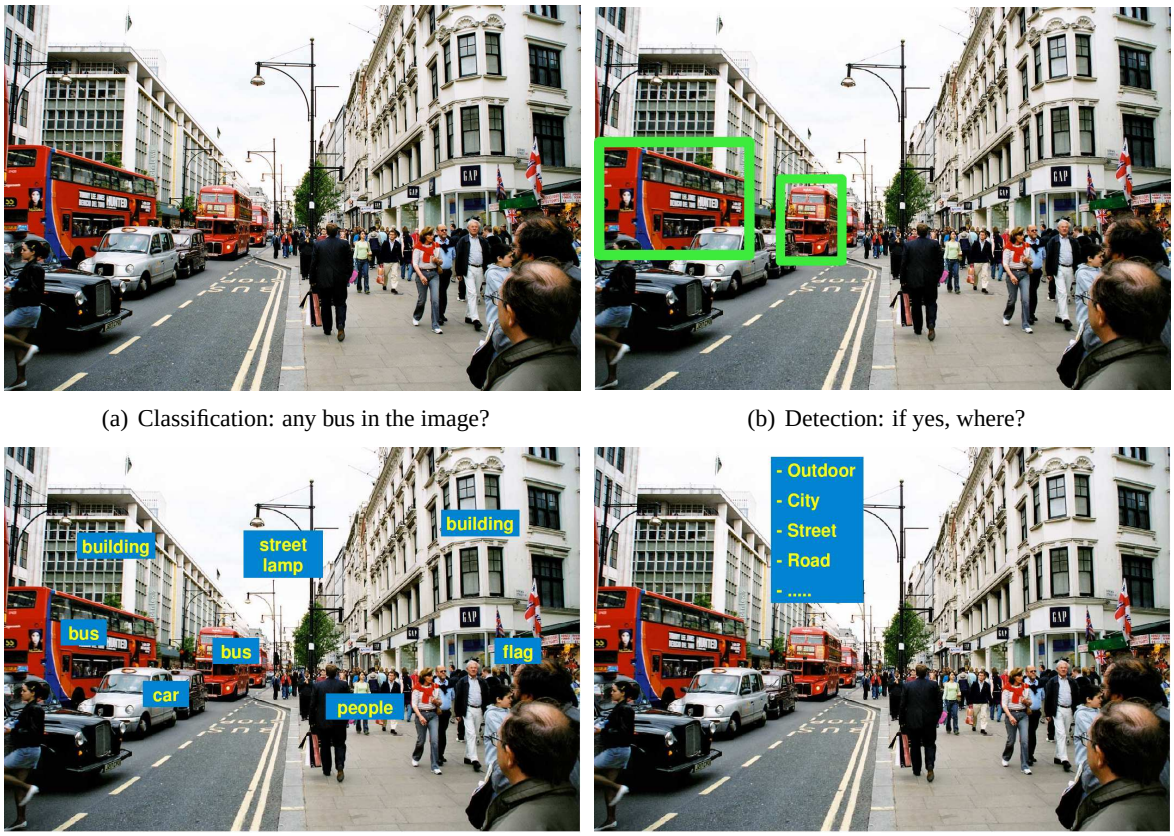
1.1.1.1 Understanding Visual Content in Images

Consider the image shown in Figure 1.2(a), one can observe a lot of things by just looking at it. Humans can easily recognize any content of the image from objects like car, bus, people and other local things to the global context of the image. The ultimate goal of visual content analysis would be to explain pixels, objects and their interactions in an image. To understand the image it is inevitable to take global as well as local information into account. The visual recognition problems: (i) classification, (ii) object detection, (iii) object categorization and (iv) scene categorization are shown in Figure 1.3. Each of them enables computers to analyse some or other aspect of the visual content of image. In chapter 3, we attempt to propose more efficient alternatives for object detection and classification.

1.1.1.2 Understanding Visual Content in Videos

Understanding visual content in video includes many approaches: (i) video retrieval or filtering, (ii) activity recognition, (iii) identifying people, (iv) video characterization via mining or other methods, (v) video summarization. Objectives of all these approaches overlap which are mainly of two types. First is to detect or search content of interest, this can be done by learned models (activity recognition) or example based methods (retrieval and filtering). Other one is aimed at characterizing or summarizing the video by analysis or mining of the video content.

Figure 1.3 shows both types of approaches. The output shown in Figure 1.3(b) shows some commercials which are blocked. This essentially would be video filtering if the content of interest, commercials,



(a) Classification: any bus in the image?

(b) Detection: if yes, where?

(c) Object Categorization: objects present

(d) Scene Categorization: scene descirption

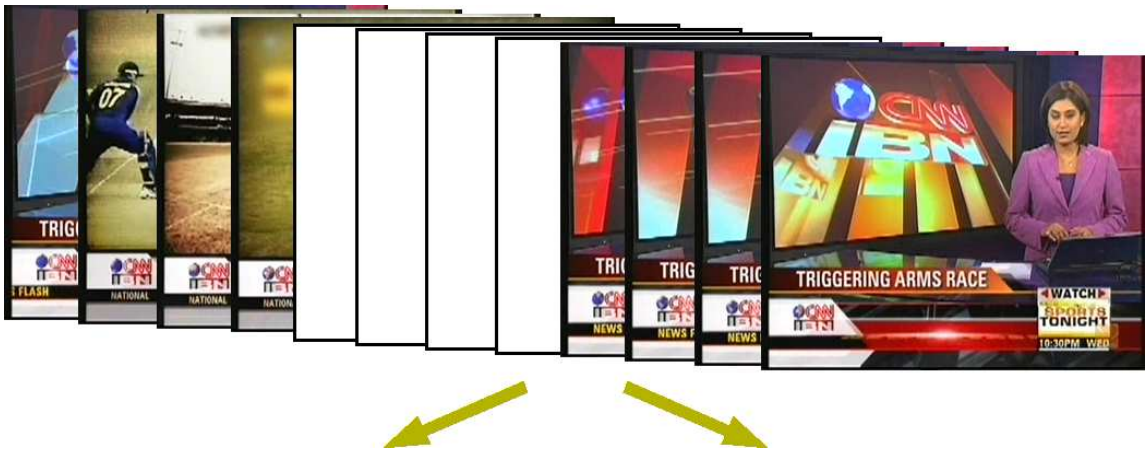
Figure 1.2 Visual Recognition Problems: different approaches to understand the visual content of an image.

was detected by already learned model or based on its similarity to some example in the database. Figure 1.3(c) shows output which is the more informative or important content of the news video such as breaking news. This can be done by mining the video or by applying some video summarization method.

In this thesis our approach for visual content processing in videos is based on filtering (chapter 4) and mining (chapter 5).

1.2 Motivation

Given the amount of easily accessible visual information the significance of its fast processing is very much apparent. Achieving the objectives mentioned in the last section would serve a large number of applications. Possible categories of applications include:



(a) News video

(b) Blocked content

(c) Informative content

Figure 1.3 (a) News video which is to be processed, (b) Detected commercials and (c) Breaking news retrieved.

- Semantic search in images and videos

  An automatic searching tool which would immediately return all images or frames containing instance of given semantic concept. It can be scenes like mountain, cityscape or objects like dog, hand, bus or activities like dancing, demonstration. Robust and efficient classification and detection methods can greatly help for such application. Also example based approach would be useful in searching complex activities like 'people applauding' by detecting sequence of frames similar to example sequences.

- Video characterization by automatic labeling

  Automatic labeling of characteristic scenes, main actors and other aspects in movies or sitcom videos can help in genre prediction, automatic annotation to build large-scale, highly varied datasets. Information obtained through finding such patterns could convey a lot about the visual content and major theme of the video. This can be used to mine patterns in movies for sociological studies, for teaching cinematography students the proper theories and practices of film

aesthetics. Other applications could be to implement automatic movie recommendation systems, video summary or intelligent fast-forward. One could for example jump automatically between all shots with characteristic scenes in the movie, or fast-forward to the next appearance of main character of the movie.

- Content based video filtering
  Content-based filtering of images and videos includes many applications like removal of commercials, event detection, content based copy detection, adult content removal, detecting occurrence of a particular object. Many methods formulate this problem as an object/scene recognition or detection by using an appropriate classifier and many others use examples to match with for complex categories. Sequence information can be very useful for event detection. Content based copy detection is another possible application which has become very popular recently. The constant struggle to identify and remove copyright multimedia content has been evident on popular video sites like YouTube. A efficient and scalable system for Content Based Copy Detection is required to deal effectively with huge amount of visual data.

- Applications in robotics
  Autonomous mobile robotics is another important area, where recognition of objects is critical for robot localization and navigation. Real-time processing of visual content would enable robot to quickly infer the world around and make it useful for variety of situations. In this way a completely autonomous robot specialized to recognize certain objects of interest will be able to substitute humans in dangerous situations such as underwater exploration, fireman help etc.

- Surveillance and Security
  Automated surveillance is another aspect of security, where identification of objects and events play an important role. Advanced systems to monitor a large set of security cameras and signal the presence of specific objects/people or unusual events can be crucial to a good surveillance system. The methods have to be efficient and robust in-order to perform well in crowded environments, for example cricket stadium.

We have a long history of partially successful and encouraging attempts to analyse visual information. However there is no perfect solution for the situations mentioned above, because they are still very challenging.

1.3 Challenges

Whether the problem is object detection or retrieval, the challenge lies often in describing the visual content well inspite of many kinds of variations. We present the challenges and our solution for indexing, learning and mining in chapters respective to these problems (Chapter 3, 4 and 5). Here we list some of the prominent challenges in the area of visual recognition:

(a) View-point/Pose

(b) Occlusion/Truncation

(c) Scale/Size

Figure 1.4 Variation caused due to (a) View-point/Pose, (b) Occlusion/Truncation and (c) Scale/Size

- Illumination Variation
  Lighting change have a major influence on the appearance and is one of the common problems in recognition. Variation in environmental illumination causes large variations in the intensity values of pixels. Due to different lighting and the occurrence of shadows objects can appear completely different and become difficult to recognise. It can also add strong gradient edges to the image, which can increase background-clutter and create confusion.

- Viewpoint and Pose Variation
  Viewpoint or position of the camera relative to the object can significantly change the appearance of an object. Objects occur in different poses and can have completely different appearances as a result. For example, the different views of cars shown in Figure 1.4(a).

- Occlusion and Truncation
  Visibility of some part of object can be hindered due to some other object in vicinity of the current object or due to other parts of same object. The latter phenomenon is known as self-occlusion. Sometimes object gets truncated by the image border. All these cases complicate the recognition task significantly as the visual model either needs to explicitly model the possibility of missing

(a) Articulation

(b) Intra-class (c) Inter-class

Figure 1.5 (a) Examples of variations caused by object articulation, (b) two instances of same object class, dog, with very different appearances, and (c) very similar looking dog (left) and cat (right).

parts or needs to be sufficiently robust to it. Figure 1.4(b) shows few examples of occlusion and truncation.

- Scale and Size Variation
  The scale and size of objects can significantly influence the similarity to other object-classes and increase the variance within one object-class. Some examples given in Figure 1.4(c) illustrates the effect of scale and size of object.

- Background Clutter
  Highly complex background can result in confusion between foreground objects and background. The chance of finding object features in the background increases thereby producing false-positives.

- Articulation
  Articulation describes the variation of appearance caused by different positions of parts of the object relative to each other. It mostly happens with living objects but also applies to other object classes such as bicycles, bike. This causes large variations among the samples of same class and increases intra-class variation. Figure 1.5(a) shows the different articulated positions of human, horse and bike.

- Intra and Inter Class Variations
  Many classes have high intra-class variation that is variation between objects belonging to the same class. For example different breeds of dog (shown in Figure 1.5(b). High inter class similarity in many object-classes is another problem which makes it challenging not to confuse those classes. Figure 1.5(c) shows an example where dog and cat look very similar.

We need approach that can address most of these challenges at the same time keeping the solution computationally efficient.

1.4 Our Contributions

In this thesis, we have proposed solutions for problems mentioned in Section 1.1.1. Brief description of each of the contributions is given below:

- Rapid Classification and Localization with Random Forests
  State of art object detection and scene classification is often achieved by support vector machines (SVM). We have employed collection of randomized trees for efficient classification. The advantage of Random Forest over SVM and other classifiers is that it is fast to train and test and still almost as accurate as non-linear SVMs. We show that Random Forest classifier can be used for fast and accurate classification and object localization. We have used a combination of different visual features with random forest for the high-level feature extraction task of TRECVID'08 [3]. Random forest is presented as a rapid object detector with results on challenging datasets like VOC PASCAL and TRECVID09. We achieved results comparable to the best in VOC'07 [45] for object detection.

- On-line Video Processing
  We proposed an architecture for efficient online content based processing of continuous stream of videos to detect segments (or sequences) which are similar to a given set of examples. An indexing technique is developed using *Trie* data structure with bag-of-words model. It does simultaneous on-line spotting of multiple examples in a video stream which makes it possible to process large amount of unseen video. This video filtering is complementary to video retrieval where query is an example video and the retrieved results are similar clips from already indexed large database.

- Video Mining
  We apply pattern mining in videos to characterize the visual content. Two different video mining schemes are employed; both aimed at detecting frequent and representative patterns. For one of our mining approaches, we use an efficient frequent pattern mining algorithm over a quantized feature space. Our second approach uses random forest to represent video data as sequences, and mine the frequent sequences. Experiments are done with boadcast News and Movie videos.

1.5 Thesis Overview

The remainder of the thesis is organized as follows. Chapter 3 presents Random Forest as fast and robust object detector. After survey on object detection and Random Forest, we experimentally compare RF with SVM. It is evaluated on VOC PASCAL and TRECVID datasets for detection and

classification. In chapter 4, we have presented a detailed survey of existing approach towards video processing and different indexing techniques. Our example based method for online processing and filtering of videos is then presented with results for Content Based Copy Detection. Our video mining approach is presented in Chapter 5 with experiments on movie and news videos. Finally, in Chapter 6 we draw conclusions from this thesis and also explore some of the possible avenues for future work.

Chapter 2

Background

In this chapter we introduce some basic tools and algorithms that we use throughout in this thesis. They include methods for image representation, clustering algorithms, visual vocabulary construction, classifiers and pattern mining methods.

2.1 Image Representation

In visual recognition literature many techniques have been used to represent the content of an image. All object/image classification and content based retrieval systems require an appropriate representation of the input images. In this section the basic techniques for representing and describing images and the objects therein are described. One can represent an image globally or locally. In the case of local models we have *sparse* and *dense* representations. Sparse as well as dense image representations are local as they focus on specific image regions instead of describing the image as a whole. In the next subsection we describe global representation. Then we give a detailed overview of sparse representation techniques, which only represent interesting areas of an image. This is followed by the presentation of methods that provide a dense representation of the image in the sense that each pixel contributes to the feature description of the image. We then describe different feature descriptors used to represent image content. In the last subsection we discuss Bag-of-words model for image representation.

2.1.1 Global Representation

Global features describe the entirety of an image with a single feature vector capturing information from the whole image. For example, variations of global color or gradient histograms, texture features. Attention is not paid to the constituents of the image, such as individual regions or parts of objects. Once each images feature is computed, we can classify each image or measure the similarity between any pair of images using some distance metric. The appeal of this approach lies in its simplicity. The drawbacks of global appearance representation are:

- problems with partial occlusion and background clutter.

- the large amount of training data required to achieve viewpoint and lighting invariance.

Many recent works in computer vision has highlighted the importance of global image representations for scene recognition [47, 75, 104] and as a source of contextual information [60, 99, 150]. These representations are based on computing statistics of low level features (similar to representations available in early visual areas such as oriented edges, vector quantized image patches, etc.) over fixed image regions. One example of a global image representation is the gist descriptor [104]. The gist descriptor is explained in section 2.1.4.3. Some of the earlier works that used this approach for recognition are [40, 98, 126].

2.1.2 Local Representation: Sparse

Local appearance methods are at the heart of some of the most successful object recognition systems. The introduction of very powerful local visual features in the late 90s is one of the main reasons for the astonishing progress the field of computer vision has made in recent years. Unlike global features, local features decompose the image into localized image patch descriptors around interest points (sparse) or on a regular grid (dense).

Selecting the right image patches and describing them in meaninigful way is very important for sparse image representation. In general this is carried out in two steps: (i) detecting interest points/regions in the image; (ii) extracting a feature descriptor from each region, which describes that specific image region. Since only a subset of image regions is represented by these feature descriptors, this provides a sparse representation of the image.

A good interest point detector locates points, that can be detected repeatedly, even if the original image is modified or the same scene is shown under varying conditions. Such variations include for example viewpoint changes (angle, zoom, etc.), illumination changes, or image compression. We now give an overview of a few interest point detectors that are used throughout the computer vision community to model images sparsely. A detailed overview and comparison of some of the most well-known interest point detectors can be found in [92, 93, 156].

2.1.2.1 Harris corner detector

Harris corner detector [56] is one of the first introduced interest point detectors. It is very basic but still influential and compared to the newer achievements. It is based on detecting corners as areas with low self similarity, i.e. small shifts of an image patch result in a large sum of squared differences.

2.1.2.2 Hessian-Affine

Hessian Affine interest point detectors [92] belong to a class of so called affine-covariant detectors, which are not only invariant to scale and rotation, but can even cope with affine changes. The main concept of these detectors is to find first a stable interest point in scale-space as with the methods

described above, but afterwards to fit an elliptical region around the interest point. (Instead of a square or circle). This ellipse adapts - i.e. is covariant - with affine changes of the underlying image structures. For Hessian-Affine detectors, the shape of this ellipse is determined with the second moment matrix of the intensity gradient.

The descriptor is extracted on a normalized region for all interest points, e.g. the ellipses are transformed into a circle, before the descriptor is calculated on the pixels within this circle.

2.1.2.3 Difference-of-Gaussians (DoG)

This involves convolving the image with a Gaussian at several scales, creating a so called scale space pyramid of convolved images. Interest points are now detected by selecting points in the image, which are stable across scales. For Difference-of-Gaussians (DoG) approach the convolved images at subsequent scales are subtracted from each other. The DoG approach is in fact simply an approximation of the Laplacian. Stable points are searched in these DoG images by determining local maxima, which appear at the same pixel across scales.

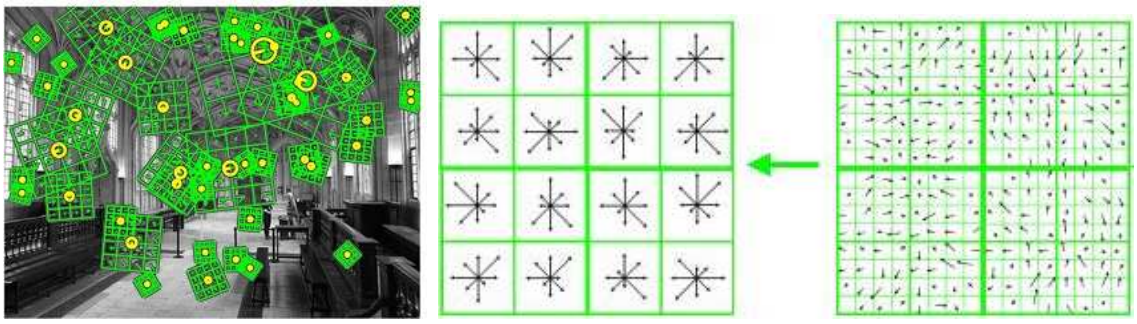
2.1.2.4 MSER

MSER (Maximally Stable Extrema Regions) [90] also belong to the class of affine-covariant detectors. They are not based on one of the standard Gaussian scale space methods, but are based on connected components of an appropriately thresholded image. The word extremal refers to the property that all pixels inside the MSER have either higher (bright extremal regions) or lower (dark extremal regions) intensity than all the pixels on its outer boundary. The maximally stable in MSER describes the objective optimized during the threshold selection process: while changing the threshold value, these regions binarization stays stable over a range of threshold values. “Maximally stable” is defined as the local minimum of the relative area change as a function of relative change of threshold.

Just as with the Hessian-Affine detectors, an ellipse can be fitted to the output regions of the detector, and after normalization, a region descriptor such as SIFT can be calculated on the pixels in the region [93].

2.1.3 Local Representation: Dense

Dense features are a widely used for many recognition tasks as an alternative to region detectors. By dense it means that the features are not extracted at the detected interest points, but feature descriptors are computed for each sampled region/pixel on a dense grid. One advantage of dense representations over sparse ones can be the fact that regions with uniform texture, which usually are not returned by interest point detectors, will be represented equally well. The preferred method depends on the application and computational constraints. There is no general rule stating clear advantages of sparse versus dense image representations. In [66], Jurie and Triggs compare these two ideas on the object categorisation task and



(a) Interest points (b) SIFT descriptor computation

Figure 2.1 (a) Example of detected key-points and (b) SIFT descriptor computation: On the right are the gradients of an image patch around a key-point. These gradients are then accumulated over 4×4 sub-regions, as shown on the left, the length of the arrow corresponding to the sum of the gradient magnitudes in that direction.

conclude that dense features perform better there. Dense sampling is also used in [20, 161] which boosted image classification and object detection results. However, due to computational constraints a combination of sparse representations and dense sampling can be useful. In [78], Leibe and Schiele use a sparse representation of interest points as a first step and refine the initial object detection by further sampling of dense features around the initial hypothesis. Thereby, dense sampling in the whole image is avoided and only applied to the potential candidate regions.

2.1.4 Feature Descriptors

Now we briefly present a few feature descriptors that we use to describe the detected regions of interest or regions from dense grid. We also discuss global descriptor Gist. It is very crucial for good performance in visual recognition that the features are robust. The most important quality criteria for descriptors are a compact representation and high precision and recall while matching descriptors from a database of images. Below we summarize the properties of some of the feature descriptors used in this thesis.

2.1.4.1 SIFT

SIFT (Scale Invariant Feature Transform) proposed by Lowe [84] is scale and rotation invariant. Originally SIFT consists of both an interest point detector and descriptor. It refers to an implementation which uses a scale invariant region detector based on the difference of Gaussians. The descriptor is however used stand alone as well in combination with various interest point detectors.

For the descriptor, around each interest point a region is defined, divided into orientation histograms on (4×4) pixel neighborhoods. The orientation histograms are relative to the keypoint orientation.

Histograms contain 8 bins each, and each descriptor contains a 4×4 array of 16 histograms around the keypoint. This leads to a SIFT feature vector with ($4 \times 4 \times 8 = 128$ elements) (Figure 2.1). This vector is normalized to enhance invariance to changes in illumination. The gradient histograms seem to contribute significantly to this performance by representing local shape. One disadvantage of SIFT is its high dimensionality and one way to reduce it is using PCA-SIFT [68] by performing Principal Component Analysis (PCA) on the raw 128 dimensional SIFT vector.

In original version after detecting interest points, several refinement steps are applied, to select the most robust points (e.g. eliminating edge responses etc.). Finally, the most dominant orientations are determined, by creating a radial histogram of gradients in a circular neighborhood of the detected point. The maxima from this histogram determine the orientation of the point, and thus enable rotation invariance.

2.1.4.2 SURF

SURF [14] is a particularly fast and compact method. Just like SIFT, SURF is also scale and rotation invariant. The interest point detector used by SURF is based on the Determinant-of-Hessian (DoH) blob detector. However, just as SIFT uses DoG as an approximation of the Laplacian, SURF uses a more efficient approximation of the Hessian. This is done using a approximation of the Gaussian second order derivatives of the Hessian detector with simple box filters. Using box filters allows using integral images [164] for efficient computation.

Like its detector, the SURF descriptor is also tuned for efficiency. It calculates a set of simple Haar-like features in sub-regions of a rectangular neighborhood around an interest point. As in the case of SIFT, this is done after determining a dominant orientation and expressing the descriptor in relation to that orientation to achieve rotation invariance. The Haar-like feature responses can again be calculated very efficiently using integral images.

2.1.4.3 Gist

Gist is a global descriptor, initially proposed in [104]. The idea is to develop a low dimensional representation of the scene. A set of perceptual dimensions (naturalness, openness, roughness, expansion, ruggedness) are proposed to represent the dominant spatial structure of a scene. Authors show that these dimensions can be reliably estimated using spectral and coarsely localized information.

The descriptor is a vector of features, g, where each individual feature g_k is computed as:

$$g_k = \sum_{x,y} \mathcal{W}_k(x,y) \times |I(x,y) \otimes h_k(x,y)|^2$$

where $\otimes$ denotes image convolution and $\times$ is a pixel-wise multiplication. $I(x,y)$ is the luminance channel of the input image, $h_k(x,y)$ is a filter from a bank of multiscale oriented Gabor filters (β orientations and four scales), and $w_k(x,y)$ is a spatial window that will compute the average output energy of each filter at different image locations. The windows $\mathcal{W}_k(x,y)$ divide the image into a grid of

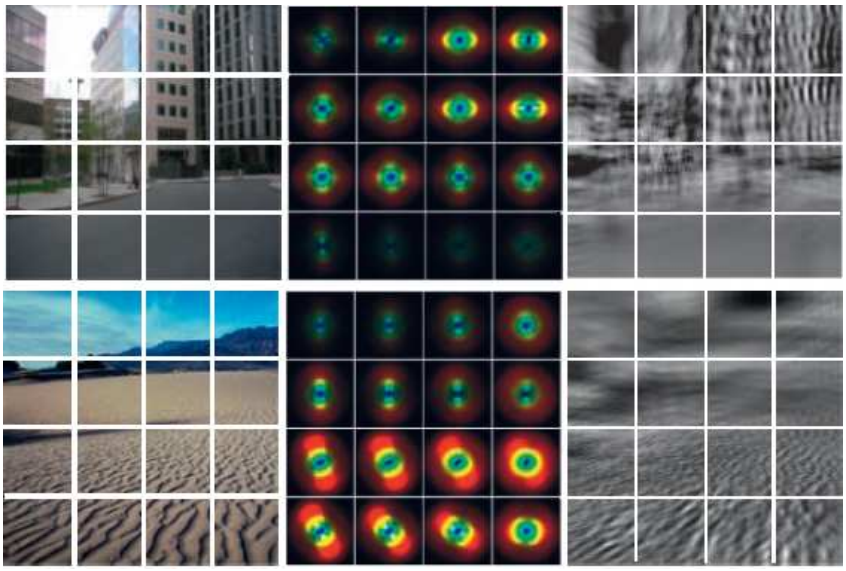


Figure 2.2 This figure illustrates the information encoded by the gist features for three different images. See text for details (Courtesy A. Torralba *et al* [151])

4×4 non-overlapping windows. We use eight orientations ($\beta = 8$), which results in a descriptor with a dimensionality of $4 \times 4 \times 8 \times 4 = 512$.

Figure 2.2 illustrates the amount of information preserved by the gist descriptor. The middle column shows the average of the output magnitude of the multiscale-oriented filters on a polar plot. The average response of each filter is computed locally by splitting the image into 4×4 windows. Each different scale is color coded (red for high spatial frequencies, and blue for the low spatial frequencies), and the intensity is proportional to the energy for each filter output. Right column of Figure 2.2 shows noise images that are coerced to have the same gist features as the target image. The gist descriptor provides a coarse description of the textures present in the image and their spatial organization. It preserves relevant information needed for categorizing scenes into categories (e.g., classifying an image as being a beach scene, a street or a living-room). In addition to recognizing the scene gist can also be used to provide strong contextual priors as we well.

2.1.4.4 Histogram of Oriented Gradients: HOG

Histogram of Oriented Gradients descriptor was first introduced by Dalal and Triggs in [34]. This robust feature descriptor describes local shape and appearance within an image by distribution of gradient orientation. The HOG descriptor, creates a dense image description by using locally contrast normalised $1D$-histograms of oriented gradients. These orientation histograms are computed over small non-overlapping cells (e.g. 8×8 pixels) covering the whole image (or region of interest containing an object etc), thereby creating a dense description. Each of those cells are normalised with respect to

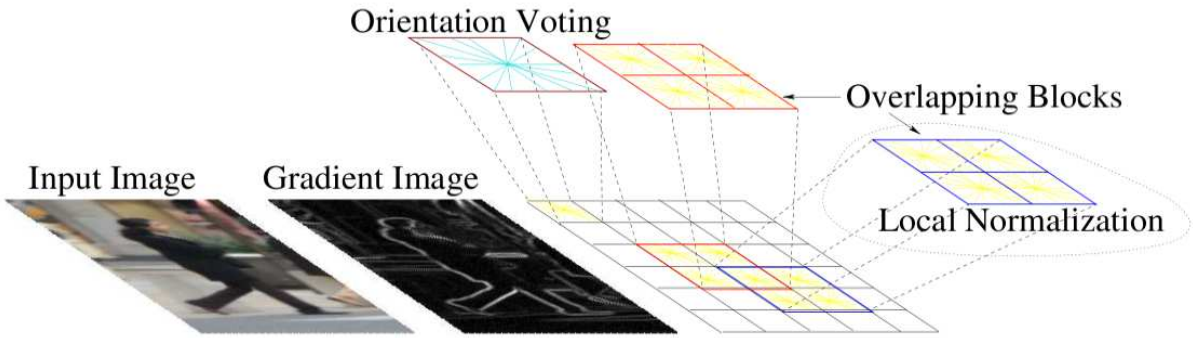


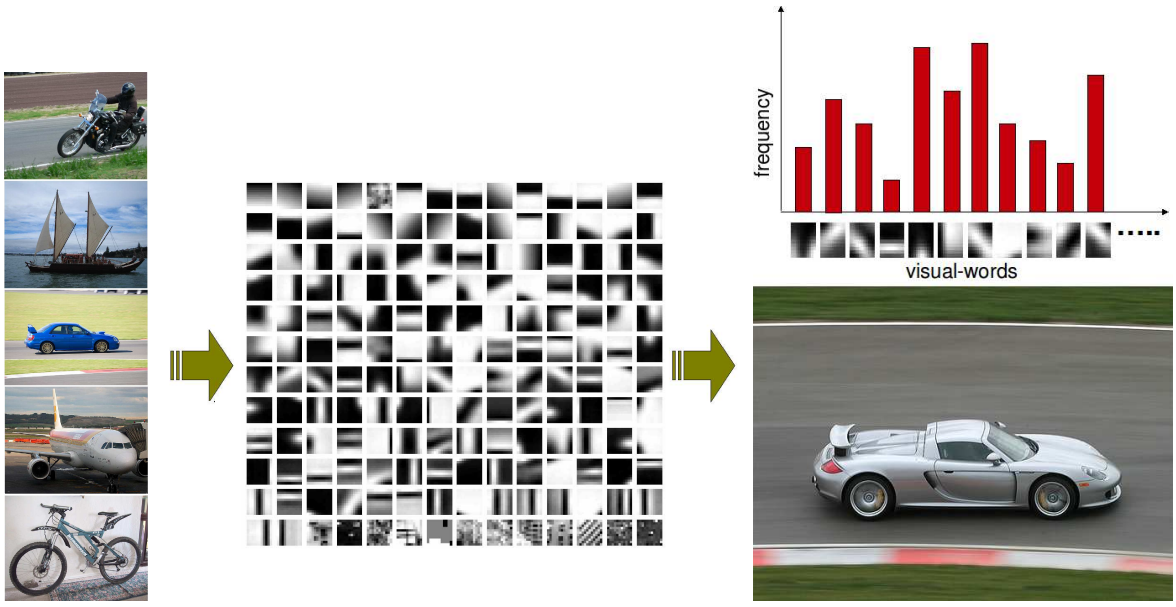
Figure 2.3 HOG feature extraction: The image or ROI (here detector window) is tiled with a grid of overlapping blocks. Each block contains a grid of spatial cells. For each cell, the weighted vote of image gradients in orientation histograms is performed. The block descriptors are locally normalised and collected in one big feature vector. Courtesy [34]

different blocks (larger spatial grid of neighbouring cells) and thus contributes to the HOG descriptor multiple times. Authors found that this local contrast normalization with overlapping descriptor blocks is crucial for good results. They also experimented to study the influence of modifications from this baseline and varying parameters : e.g. radial cells, rectangular cells, different sizes of cells and blocks, fine-scale gradients, fine orientation binning, relatively coarse spatial binning. Figure 2.3 shows the process of computation of HOG descriptor. This descriptor idea can be seen as a dense version of the SIFT descriptor. HOG and its variants have given state of art performance for object detection [48] and image classification [20].

2.1.5 Bag of Words Model for Image Representation

In the last few years, bag of visual words have been commonly used in object recognition, object or texture classification, scene classification, image retrieval and related tasks. It directly relates to the bag of words model (BOW) originally used in text retrieval [12]. It has been introduced into the computer vision community by Sivic and Zisserman [137], who apply it to object retrieval in videos.

The BOW model is usually based on interest points and corresponding feature descriptions. It uses a clustering/vector-quantisation method to quantize the feature descriptors. Eventually each interest point is represented by an ID indexing into a visual-codebook or visual-vocabulary. Visual vocabularies are typically obtained by clustering the feature descriptors in high dimensional vector space. The dataset (or a subset of dataset) is clustered into k representative clusters, where each cluster stands for a visual word. The resulting clusters can be more or less compact, thus representing the variability of similarity for individual feature matches. The value of k depends on the application, ranging from a few hundred or thousand entities for object class recognition applications up to one million for retrieval of specific objects from large databases. For clustering, most often k-Means is used, but other methods are also



(a) Dataset (b) Vocabulary (c) BOW representation

Figure 2.4 Bag of visual words model: (a) Database of images, features extracted are clustered to get (b) Visual vocabulary or collection of visual words, and (c) An example image represented using constructed vocabulary.

used for example hierarchical k-means is used by Nister *et al* in [101]. Size of vocabulary is chosen according to how much variability is desired in the individual visual words. In object class recognition, the individual instances of a class can have large variations, while in retrieval for specific objects very similar features have to be found.

After vocabulary building an image is then modelled as a bag of those so called visual-words. It can thus be described by a vector (or histogram) that stores the distribution of all assigned codebook IDs or visual words. The complete process for encoding an image with a visual vocabulary is summarized in Figure 2.4. Note that this discards the spatial distribution of the image features. In contrast, the image descriptions introduced in the previous sub-sections also carry spatial information, especially the dense ones, e.g. HOG, are often used directly to provide a spatial description of the objects.

In visual recognition, the bag of words model has been employed by one of the most successful methods in the PASCAL 2006 challenge [43]. One of the best performing approaches uses a combination of the methods introduced in Zhang *et al* [179] and Lazebnik *et al* [75]. The system uses the bag of visual-words model on sparse Harris-Laplace and Laplacian feature detectors or dense features on the one hand, and an extension which uses spatial pyramids to represent spatial dependencies on the other hand. This shows that this model compares to other state of the art object recognition methods despite its apparent simplicity and crude neglect of spatial feature relations.

For image and video retrieval based on visual vocabularies, often several additional methods are borrowed from text retrieval [123], e.g. the most frequent and infrequent visual words are removed from the images using a stop-list, or the features are ranked using a tf-idf variant, weighting frequently occurring features lower. Sivic and Zisserman [137] use tf-idf weighting on the visual-word counts produced by interest point detection and SIFT features in order to retrieve frames in videos containing a query object.

We use visual vocabularies through out this work. In chapter 4 we present *vocabulary trie* based on this model for video filtering. Our approach for mining videos in chapter 5 uses visual words representation. For classification and detection in chapter 3 we use state of art features, some of which are again based on this model.

2.2 Vector Quantization (Clustering)

Vector quantisation of image feature descriptors is a common step in the visual recognition community. One reason for the quantisation is the large range of values and their sensitivity to small image perturbations. Thus the quantisation introduces robustness. It involves data clustering or partitioning a data set into groups of more related samples.

The most widely used method employs k-means clustering algorithm [85]. K-means starts with k randomly selected data points, called cluster centres (different data driven initialisation techniques are used as well). The first step assigns each of the remaining data points to the closest cluster centre. The next step recomputes the cluster centres to be the mean of each cluster. These two steps are alternated until convergence. It finds a partitioning of N points from a vector space into $k < N$ groups, where k is typically specified by the user. The objective it tries to achieve is to minimize total intra-cluster variance:

$$V = \sum_{i=1}^{k} \sum_{x_j \in c_i} (x_j - \mu_i)^2$$

where there are k clusters c_i, $i = 1...k$, and μ_i is the mean of all the points $x_j \in c_i$.

While k is the only parameter that needs to be specified for k-Means, its choice is not trivial. In particular since it affects the outcome of the clustering result greatly. A common way to handle this problem is to just try several values for k. However, for large datasets this approach is too time-consuming because k can vary in a wide range. The time-complexity of the k-Means algorithm is $O(Nkld)$ for N datapoints of dimension d, and l iterations. Here l depends on the distribution of the data in the feature space and the initial centers. Many improvements of the standard k-Means algorithm have been suggested [110, 41]. They either use efficient data structures or improve runtime and memory requirements by reducing the number of distance calculations based on some approximation criteria.

Other methods such as agglomerative clustering in [78], which uses normalised grey-scale correlation on 25×25 image patches, or the mean-shift based method described in [66] by Jurie and Triggs

are used as well. Moosmann *et al* [94] introduced more efficient alternative for building codebooks in Extremely Randomized Clustering Forests - ensembles of randomly created clustering trees.

In our work we use k-Means mostly to cluster local visual features into visual vocabularies.

2.3 Random Forests

Random Forests were introduced in the machine learning community by [8, 21] and are based on decision trees [118]. Dietterich and Fisher did related work [38] for constructing ensembles of decision-trees and compared methods based on bagging, boosting, and randomization. Decision tree classifiers have shown problems related to over-fitting and lack of generalization. Random Forests are trained to alleviate such problems by:

- injecting randomness into the training of the trees, and
- combining the output of multiple randomized trees into a single classifier.

A random forest multi-way classifier consists of a number of trees, with each tree grown using some form of randomization. Randomness can be injected at two points during training: in sub-sampling the training data so that each tree is grown using a different subset; and in selecting the node tests. The leaf nodes of each tree are labeled by estimates of the posterior distribution over the classes. Each internal node contains a test that splits the space of data to be classified. A test sample is classified by sending it down every tree and aggregating the reached leaf distributions. Figure 2.5 shows a Random Forest which consists of T trees.

Random Forests have been shown to result in lower test errors than conventional decision-trees [174] and performance comparable to SVMs in multi-class problems [20], while maintaining high computational efficiency.

2.3.1 Random Forest classifier

We employ *binary decision-trees* as building blocks. Each internal node of the tree has a test associated with it which can be in general of the form $\Phi : \mathcal{X} \rightarrow \{\textit{true, false}\}$, where $\mathcal{X}$ is the feature representation of input sample. In specific, this test can be a combination of a node function $\mathcal{F}$ of the form $\mathcal{F} : \mathcal{X} \rightarrow \mathbb{R}$ and a threshold τ. The node test is then defined as:

$$\Phi(\mathcal{X}) = \begin{cases} \mathcal{F}(\mathcal{X}) > \tau & \text{go to right child} \\ else & \text{go to left child} \end{cases} \tag{2.1}$$

Feature representation (i.e. $\mathcal{X}$) can be of general nature, e.g. any feature descriptor like color histogram, HOG, PHOW etc or output of a filter bank. Also the function $\mathcal{F}$ is of a very general nature: it could be a component or difference of two components of a feature descriptor, a linear classifier, the output of another classifier, to give few examples.

2.3.1.1 Training the classifier

Binary decision-trees are constructed by learning the node-tests discriminatively in a top-down manner. Starting from the root, given the labeled training data, the node function $\mathcal{F}$ and threshold τ which maximize the information gain ΔE are found at each node.

$$\Delta E = E(Q) - \sum_j \frac{|Q_j|}{|Q|} E(Q_j)$$

where Q is the set of data points at the current node (to be splitted) and $Q_j \subseteq Q$ are the *left* and *right* subset caused by partitioning the data with $\mathcal{F} > \tau$. $E(Q)$ is the entropy of data points in Q. The algorithm proceeds iteratively with the left and right subsets Q_j at the children nodes until Q_j is empty or a threshold for $E(Q)$ or ΔE is reached. Sometimes when the training data is very unbalanced it is beneficial to normalize it during computation of ΔE, by weighting each training point with its inverse class prior probability. Note that this is different from normalizing the empirical class posteriors in the leaf nodes. There are four ways in which priors can be used for normalization of training data:

- weighting each training point by its inverse class prior probability during computation of ΔE and also normalizing empirical class posteriors in the leaf nodes.
- using class prior probability during computation of ΔE but not at the leaf nodes.
- normalizing empirical class posteriors in the leaf nodes but not using priors during computation of ΔE.
- not using priors both during computation of ΔE and at the leaf nodes.

During training of the tree each node has available only a randomly chosen subset of the entire pool P of possible node functions. Training is achieved by finding for each non-terminal node a node function and a threshold which yields maximum information gain ΔE within such restricted, randomized search space [20, 167]. The "randomization" can be tuned by several parameters like the size and composition of the pool P, number of node-functions and thresholds (τ) tried. Size of pool P and number of thresholds to be tried at each node for each node-function are chosen according to the application. For example, Shotton *et al* [133] uses 500 node-functions and 10 thresholds at each node while [79], [94] do not optimize τ, but pick it randomly.

The training data is sub-sampled (bagging) and each tree is trained using a different random subset. This is done to increase the independence of the trees [21] and reduce training time. As a result of training the empirical class posterior distributions are stored in the leaf nodes, in the form of histogram counts over the class-labels of the training data as shown in Figure 2.5. Since decision-trees in Random Forest are trained independently training can be easily paralleled. It is crucial to randomize enough so that each tree result in independent classifications of the data [21].

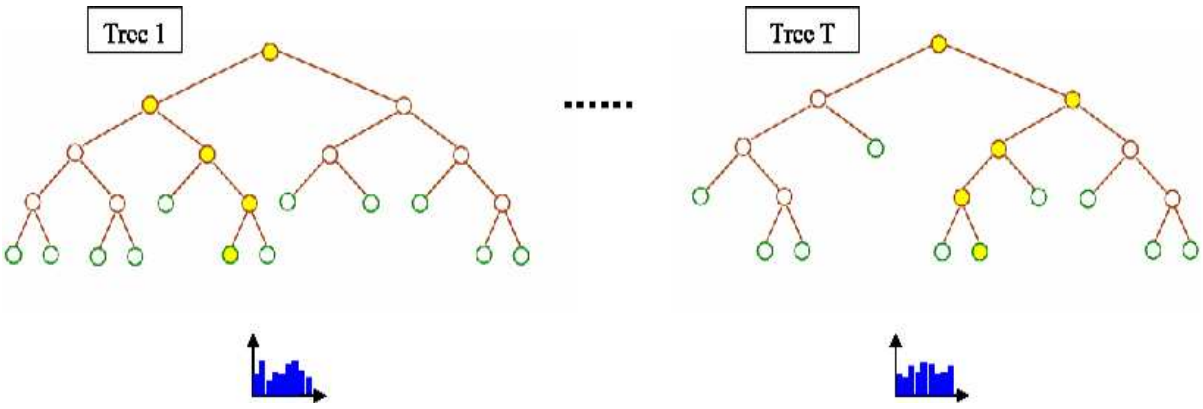


Figure 2.5 Random Forest with T trees, leaf nodes are shown in green. Training samples are traversed from root to leaf nodes and posterior distributions (blue) are computed. A test sample is classified by descending each tree and then aggregating the distributions at each reached leaf. The paths formed while descending are shown in yellow.

2.3.1.2 Random Forest Parameters

The following describes the effect of different parameters on the training of the Random Forest.

- Number of decision-trees in the Random Forest (T): The number of decision trees greatly influences the performance of the Random Forest. Additional independent/randomized decision-trees add further information over the training data, as each tree partitions the feature space into different cells and collects the empirical class posteriors for those cells in the leaf nodes. Our experiments show that the performance increases the more decision-trees are used, but the improvement is small after a certain number of decision trees are used.

- Node test parameters: These are the main parameters to influence the "randomness" of the decision-trees.

  - Number of functions chosen at each node (n_f): If n_f = 1 there would be no optimization of the information gain ΔE and the decision-trees would define a random partitioning of the feature space. The greater n_f the more the partitioning will be driven by discriminating between the object classes, as for each node the one node-function out of n_f functions that maximizes the information gain ΔE is selected. These n_f node-functions are sampled randomly from the initial pool of node-functions P that is created for each decision-tree.
  - Number of thresholds tried for each node-function at each node (n_τ): For each node-function n_τ thresholds are tried over the feature responses of training samples. The thresholds are usually sampled uniformly across the responses. Like n_f, increasing n_τ leads to more discriminative partitioning.

- Decision-tree parameters: These parameters describe properties of the decision-trees only.
  - maximum depth of a tree: Experiments indicate that deeper trees tend to improve performance as stronger trees are built. Thus the depth is mostly determined by computational and memory considerations, but depending on the specifics of the implementation and the number of trees in the forest it can also lead to over-fitting.
  - fraction of training samples used to train a tree: Similar to the maximum depth, when larger fraction of training data is used per tree, stronger trees are built. But too many training sample per tree may cause over-fitting and an ensemble of larger number of weaker trees may have better generalization.
  - information gain or entropy stopping criterion: In order to avoid "over-fitting" of the tree, i.e. partitioning of feature space areas into areas where there is no further partitioning necessary, these two criteria can be used to stop expansion of the nodes before the maximum depth is reached.

2.3.1.3 Classification

Data is classified independently by each decision tree during testing. Each sample is pushed through the tree from the root to a leaf node. At each internal node depending on the evaluation of the node-test a sample is sent to the left or right child. This classification results in the assignment of the empirical class posterior distribution to the test sample. It is often better to not use the empirical class posteriors directly, but to weight them with the class prior probabilities. To combine the multiple class posterior distributions $P_i(c|\mathbf{x})$ with $i \in \{1, ..., T\}$, we use Mixture of Experts method where individual probability distributions are averaged:

$$P(c|\mathbf{x}) = \frac{\sum_i^T P_i(c|\mathbf{x})}{Z}$$

where Z denotes the normalization such that $P_i(c|\mathbf{x})$ is a proper probability distribution. In this method each individual tree has an influence in voting for a specific class. In [15], Biau *et al* show that the voting and averaging classifiers are consistent and also investigate the consistency of Random Forests. It turns out that if the individual decision-trees are consistent the averaging classifier is consistent as well.

2.4 Frequent Pattern Mining

Frequent pattern mining (FPM) has become one of the most actively researched topics in data mining and knowledge discovery in databases. It was catalysed by market basket analysis and the task to mine transactional data. It described the shopping behavior of customers of supermarkets, mail-order companies and online shops, for products that are frequently bought together. For this task, which became generally known as frequent item set mining, a large number of efficient algorithms were developed, which are based on sophisticated data structures and clever processing schemes.

Item set mining was followed by to sequence mining and the extensions are fairly straightforward. But they led to exciting new applications like genome mining and temporal pattern extraction from data describing, for instance, alarms occurring in telecommunication networks. More complex problems of mining frequent trees and general frequent subgraphs have been addressed in recent developments. These have applications in biochemistry, citation network, web mining and program flow analysis. Frequent Pattern Mining comprises of these problems:

- Frequent Itemset Mining (FIM)
- Frequent Sequence Mining (FSM)
- Frequent Tree Mining
- Frequent Graph Mining

In this work we only use frequent itemset mining and its extension frequent sequence mining. Using FSM we deal with sequences of items or itemsets. Frequent itemsets play an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more. Pattern mining has also been used in computer vision by Till Quack *et al* [114, 115, 116] and Nowozin *et al* [102, 103]. We have used FIM and FSM for mining patterns in videos in chapter 5.

In this section we first give the definitions of important terms and concepts. Then we give the general problem statement for FPM followed by specific cases FIM and FSM.

2.4.1 Basic Terms and Notions

Here we summarize the relevant terms and notions for frequent itemsets. We also define association rules before discussing frequent sequence mining.

- Let $I = \{i_1, ..., i_p\}$ be a set of p *items*. This is called the *item base*.
- Any subset A of I with l items, i.e. $A \subseteq I, |A| = l$ is called an *l-itemset*.
- A *transaction* is an itemset $T \subseteq I$ with a transaction identifier $tid(T)$.
- A *transaction database* D is a set of transactions with unique identifiers $D = \{tid(T_1)...tid(T_n)\}$, $tid(T_i) = tid(T_j)\{i, j\} \in I | i = j$. Every transaction is an item set, but some item sets may not appear in D.
- We say that a transaction T supports an itemset A, if $A \subseteq T$.
- A *sequence* α is an ordered list of items or itemsets (term used is *event*). It is denoted as $\alpha = (\alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_q)$

- A sequence with k items ($k = \sum_j |\alpha_j|$) is called a *k-sequence*. For example, ($B \rightarrow AC$) is a *3-sequence*.
- We say α is a sub-sequence of another sequence β, denoted as $\alpha \preceq \beta$, if there exists a one-to-one order-preserving function f that maps events in α to events in β.
- A *sequence database* S is a set of sequences. We say that a sequence from S, α_S, supports a sequence α, if $\alpha \preceq \alpha_S$.

Let $A \subseteq I$ be an itemset and D a transaction database over I. Also let α be a sequence and S a sequence database over I. We can now define relevant concepts:

Definition 2.4.1 ***Support of an itemset:*** *The support (absolute) of an itemset $A \in D$ is*

$$support(A) := |\{T \in D | A \subseteq T\}|$$

and relative support is

$$support(A) := \frac{|\{T \in D | A \subseteq T\}|}{|D|} \in [0, 1]$$

Definition 2.4.2 ***Cover of an itemset:*** *For each itemset we can also find the transactions, which support the itemset. The cover of an itemset $A \in D$ consists of the set of transaction identifiers of transactions in D that support A:*

$$cover(A, D) := tid(T) | (T \in D, A \subseteq T)$$

Definition 2.4.3 ***Frequent itemset:*** *An itemset A is called frequent in database D if $support(A) \geq min\_supp$ where min_supp is a threshold for the minimal support.*

Definition 2.4.4 ***Closed itemset and Maximal itemset:*** *A frequent item set A is called closed if no super-set has the same support. A frequent item set A is called maximal if no super-set is frequent.*
These are two special types of frequent itemsets that are often discriminated in the literature.

After mining frequent itemsets, one is often interested in the statistical dependence between the individual items or subsets that form a set. These dependencies are typically expressed in the form of association rules.

Definition 2.4.5 ***Association rule:*** *An association rule is an expression $A \rightarrow B$ where A and B are itemsets (of any length) and $A \cap B = \phi$.*

The quality or interestingness of a rule is typically expressed in the support-confidence framework, which was introduced in [5].

Definition 2.4.6 ***Support of a rule:*** *The support of an association rule $A \rightarrow B$ is*

$$supp(A \rightarrow B) := supp(A \cup B) = \frac{|\{T \in D | (A \cup B) \subseteq T\}|}{|D|}$$

In other words, the support of a rule is the support of the joined itemsets that make up the rule. The support of a rule measures its statistical significance.

Definition 2.4.7 ***Confidence of a rule:*** *The confidence of an association rule $A \rightarrow B$ is*

$$conf(A \rightarrow B) = \frac{supp(A \cup B)}{supp(A)} = \frac{|\{T \in D | (A \cup B)T\}|}{|\{T \in D | (A \subseteq T\}|}$$

The left-hand side of a rule is called antecedent, the right-hand side is the consequent. The confidence is a measure of the strength of the implication $A \rightarrow B$.

Definition 2.4.8 ***Support of a sequence:*** *The support (absolute) of a itemset $A \in D$ is*

$$support(\alpha) := |\{\alpha_S \in D | A \preceq \alpha_S\}|$$

and relative support is

$$support(\alpha) := \frac{|\{\alpha_S \in D | A \preceq \alpha_S\}|}{|D|} \in [0, 1]$$

As in case of frequent item-set, a sequence is called frequent if its support is greater than threshold for the minimal support. Also frequent sequence is maximal if none of its super-sequences is frequent.

2.4.2 Frequent Pattern Mining: Problem Statement

In the frequent pattern mining problem, we have a pattern context $\mathcal{PC} = (\mathcal{P}, \mathcal{R})$, for the input data which is a *locally finite poset*. $\mathcal{P}$ is set of all possible patterns in the data and $\mathcal{R}$ is a containment relation. We are given the set of input data $\mathcal{J}$, the pattern context $\mathcal{PC}$, the support function, $suppT : \mathcal{P} \rightarrow N$ and a minimum support threshold, $min\_supp \in N$.

The task is to find the set $F = \{p \in \mathcal{P} : suppT(p) \geq min\_supp\}$ and the support of the patterns in F. Elements of F are frequent and are called *frequent* patterns. Here $\mathcal{P}$ is pattern context (set of possible patterns), $suppT : \mathcal{P} \rightarrow N$ is a support function and min_supp given minimum support threshold. The input data $\mathcal{J}$ is a set of collection of patterns (transactions or sequences). There are many types of patterns: itemsets [5], item sequences, sequences of itemsets [7] etc. All of these are some or other form of collection of $items$ coming from item base, I.

2.4.3 Frequent Itemset Mining

Frequent itemset mining is a case of FPM where patterns are transactions and the containment relation ($\mathcal{R}$) is set inclusion ($\subseteq$) relation. The task is to find all frequent item sets from the transactional database. Finding frequent itemsets is one of the most investigated fields of data mining. It is very popular family of methods to detect the joint occurrence of certain items from a large body of data. The problem was first presented in [5]. The subsequent paper [6] is considered as one of the most important contributions to the subject. Its main algorithm, *Apriori*, not only influenced the association rule mining community, but it affected other data mining fields as well. Association rule and frequent itemset mining became a widely researched area, and hence faster and faster algorithms have been presented. Numerous of them are *Apriori* based algorithms or *Apriori* modifications.

Market basket analysis was the main application considered in the first publications on itemset mining [5], however, since then same kind of problem has been analyzed in various other contexts. This includes web usage mining [31], robust collaborative filtering [124], fraud detection in on-line advertising [91], document analysis [61] or massive recommendation systems for related search queries [81]. In computer vision, frequent itemsets configurations are used to mine videos in [114, 115, 116].

2.4.4 Frequent Sequence Mining

Given a database S of input-sequences and minimum support, the problem of mining sequential patterns is to find all frequent sequences in the database. On this version of FPM data is in the form of sequences and sub-sequence is the containment relation ($\mathcal{R}$). The problem of mining sequential patterns was introduced in [7]. They also presented three algorithms for solving this problem. The *AprioriAll* algorithm was shown to perform better than the other two approaches. In subsequent work [142], the same authors proposed the GSP algorithm that outperformed *AprioriAll* by up to 20 times. Since then improved algorithms kept coming, some of them are FreeSpan [55], PrefixSpan [109], SPADE [176].

Sequence discovery can essentially be thought of as association discovery [125, 142] over a temporal database. While association rules discover only intra-event patterns (called itemsets), we now also have to discover inter-event patterns (sequences). The set of all frequent sequences is a super-set of the set of frequent itemsets. Due to this similarity sequence mining algorithms like *AprioriAll*, GSP, etc., utilize some of the ideas initially proposed for the discovery of association rules. Shortcomings of *Apriori*-like approaches are: (i) potentially huge set of candidate sequences, (ii) multiple scans of databases and (iii) difficulties at mining long sequential patterns. Algorithms like FreeSpan, PrefixSpan avoid the repeated scans setback of *Apriori* which makes them suitable for dealing with very large databases.

Chapter 3

Rapid Object Detection using Random Forests

3.1 Introduction

A long-standing goal of machine vision has been to build a system which is able to recognize many different kinds of objects in a cluttered world. Recent years have seen great progress in the area of object category recognition for natural images. However, in their basic form, many state-of-the-art methods only solve a binary classification problem. They can decide whether an object is present in an image or not, but not where exactly in the image the object is located.

Given an image, task of object detection is to find if the object of interest is present in the image, and if present return the location of it. Some examples of object detection are shown in Figure 3.1. Object localization is an important task for the automatic understanding of images as well, e.g. to separate an object from the background, or to analyze the spatial relations of different objects in an image to each other. Various classifiers have been used for object detection, such as SVMs [34, 49, 86, 108], naive Bayes [128], mixtures of Gaussians [50], boosted decision stumps [148, 165], etc. In addition, many types of image features have been considered, like generic wavelets [128, 165], histogram of gradient orientation (HOG) [34, 49], spatial pyramid histogram of visual words and HOG [20, 84, 137, 161] etc.

The Random Forest is another discriminative classifier that has become very successful in computer vision [8, 20, 21, 79, 133]. The advantages of Random Forest as a classifier are discussed in section 3.6. In this chapter we show that Random Forest can be used for fast and accurate object detection comparable to the state of art. We start with an overview of previous relevant research on object detection followed by theory of Random Forest and its application in computer vision in section 2.3. First in section 3.4, we show strength of Random Forest as a classifier by experiments for image level scene and object classification in high level feature detection task in TRECVID'08 [3]. We explain our detection system by giving details of training, testing, post-processing and features in section 3.5. Then Random Forest is compared with SVM as an object detector on account of accuracy, speed and memory usage. Section 3.7 details our approach of sliding window based RF detector. This Random Forest baseline detector is evaluated on challenging VOC PASCAL dataset [45]. In the following sections we propose enhancements over the baseline detector and present Random Forest as a rapid object detector supported

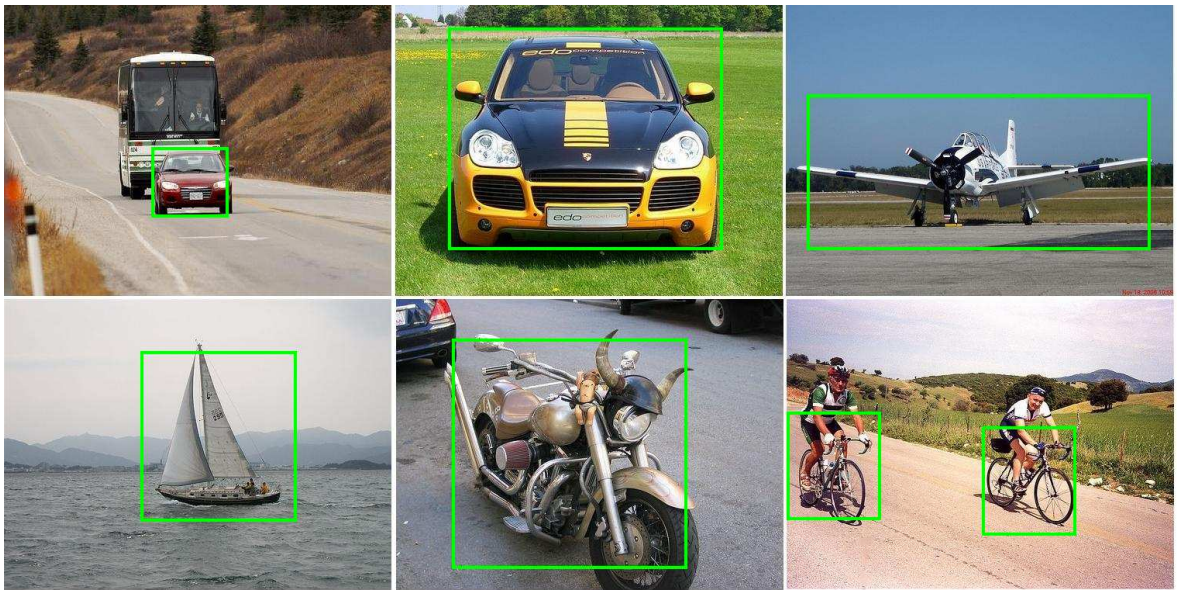


Figure 3.1 Some examples of our object detection results.

by experiments and results. We also show how it performs for object retrieval in high level feature detection task in TRECVID'09.

3.2 Object Detection Literature and Methods

A number of different approaches to object detection have been proposed in the past. In most of them, images are represented using some set of features, and a learning method is then used to identify regions in the feature space that correspond to the object class of interest. There has been considerable variety in the methods based on types of features, classifiers and also based on how the localization is done. Among them localization using sliding window approach is the most popular one.

3.2.1 Sliding window based methods

In this approach detector is based on a binary object/non-object classifier, which scans the image with a detection window at all positions and scales. Feature descriptors extracted from each window are scored by the classifier and multiple overlapping detections are fused to yield the final object detections. Sliding window approaches have established themselves as state of the art. Most successful localization techniques at the recent PASCAL VOC challenges on object category localization relied on this technique.

After the introduction of *Integral Image* in [164] for fast object detection, sliding window based method became the most preferred method. It became possible to quickly compute features for ex-

tremely large number windows by using integral images. The other two major contributions of [164] were: feature selection with *AdaBoost* and combining classifiers in a *cascade*. Their face detection system was most clearly distinguished from previous approaches in its ability to detect faces extremely rapidly. This was followed by great progress in object detection and many methods which used sliding window approach were proposed, [29, 34, 149, 161, 183] to count a few.

The jointboost method for multiclass and multiview object detection was proposed in [149]. It was based on boosted decision stumps, that reduces the computational and sample complexity, by finding common features that can be shared across the classes (and/or views). The detectors for each class are trained jointly, rather than independently. They found that the features selected by joint training are generic edge-like features which generalize better and because of sharing computational cost of multi-class object detection is considerably reduced. Dalal *et al* [34] proposed Histogram of Oriented Gradients (HOG) feature descriptor which gave excellent detection results for human detection. With help of their HOG detector they won the VOC PASCAL2006 challenge for object detection and it became a very popular feature. The Exemplar model of [29] was another very successful method that used sliding window with integral image to efficiently compute the cost function. Vedaldi *et al* [161] used Multiple Kernels with many robust features to achieve the performance exceeding state of the art for number of object classes in VOC2007 and VOC2008 challenges [45, 46].

There are two main drawbacks to sliding-window object detectors: (a) mostly the detectors fail to take into account contextual cues; (b) the detector window has only few fixed aspect ratios making it difficult for articulated objects or objects with large intra-class variation.

3.2.2 Other methods

The appearance of objects of the same class such as cars, person or motorbike in natural images vary greatly due to intra-class differences, changes in illuminations and imaging conditions, and as well as object articulations (person, bicycle). Part-based models provide an elegant framework for representing such object categories and handling above variable conditions, specially object articulations.

One of the earlier successful methods using part-based representation of objects for learning to detect objects was proposed in [4]. Generalized Hough Transform of [77, 78] also uses highly flexible learned representation for object shape. Their method learns the class-specific implicit shape model (ISM), which is essentially a codebook of interest point descriptors typical for a given class. Implicit shape models can integrate information from a large number of parts and thus they demonstrate good generalization. J. Gall and V. Lempitsky [52] proposed an interesting approach for object part detection using a class-specific Hough forest. It is a random forest that classifies image patches as a part of object or not and directly maps the patch to the probabilistic vote about the possible location of the object centroid.

Recently, Felzenszwalb *et al* [48, 49] described an object detection system that is based on mixtures of multiscale deformable part models. To train models with partially labeled data they use a latent SVM

and use object parts as latent variables. Their system is able to represent highly variable object classes and achieves state-of-the-art results in the PASCAL object detection challenges.

Efficient sub-window search based on branch and bound algorithm has been recently proposed by C. Lampert and M. Blaschko [24, 25]. They argued that sliding window has been effective in many situations but suffers from two disadvantages:

- it is computationally inefficient to scan over the entire image and test every possible object location, and
- it is not clear how to optimally train a discriminant function for localization.

They addressed the first issue in [24, 25] by using efficient sub-window search, a branch-and-bound optimization strategy, to efficiently determine the bounding box with the maximum score of the discriminant function. This was further explored in [172], where an efficient method for concurrent object localization and recognition based on a data-dependent multi-class branch-and-bound formalism was proposed. Blaschko *et al* addressed the second issue in [16] by proposing a training strategy that specifically optimizes localization accuracy, resulting in much higher performance than systems that are trained, e.g., using a support vector machine.

3.3 Random Forests in Computer Vision

Random Forest became popular in the computer vision community from the work of Lepetit *et al* [80], Ozuyal *et al* [106]. They proposed methods for fast keypoint recognition using randomized trees. In [106] random forest is employed to recognize the patches surrounding keypoints. [80] used randomized trees as the classification technique and found it both fast enough for real-time performance and robust. Many other papers have applied them to various classification, segmentation and clustering tasks [20, 26, 36, 79, 88, 94, 167, 174, 130, 133].

Textons [87, 159] and visual words [137] have proven powerful discrete image representations for categorization and segmentation. But the whole process of computing descriptors at interest points (sparse or dense), then clustering them by K-means, followed by nearest-neighbor assignment is extremely slow. Even with optimizations such as kd-trees, the triangle inequality [41], or hierarchical clusters [101] it is not fast enough. Moosmann *et al* [94] introduced more efficient alternative in Extremely Randomized Clustering Forests - ensembles of randomly created clustering trees - and showed that these provide more accurate results, much faster training and testing and good resistance to background clutter in several state of the art image classification tasks.

In [133], Shotton *et al* extended [94] and proposed semantic texton forests for both Image Categorization and Segmentation. These are ensembles of decision trees that act directly on image pixels and are efficient and powerful low-level features. Other major work done in image segmentation are by Schroff *et al* [130], Yin *et al* [174]. [130] investigates the use of Random Forests for class based pixel-wise segmentation of images. They show the ability of Random Forests to combine multiple features

which improves the performance when textons, colour, filterbanks, and HOG features are used simultaneously. The benefit of the multi-feature classifier is demonstrated with extensive experimentation on labeled image datasets.

The problem of classifying images and recognizing objects have been explored using Random Forests in [20, 88, 167]. [20] compared the performance of the random forest/ferns classifier with a benchmark multi-way SVM classifier. The performance of random forest/ferns was comparable to multi-way SVM classifier but computationally they were far less expensive.

A method for localizing and recognizing hand poses and objects in real-time is presented in [36]. In their work, the task of simultaneously recognizing object classes, hand gestures and detecting touch events is cast as a single classification problem. [120] addresses human pose recognition from video sequences by formulating it as a classification problem. They propose a pose detection algorithm based on random forests. In [52] a method based on random forests is presented for object detection. They proposed a class-specific Hough forest, which is a random forest that directly maps the image patch appearance to the probabilistic vote about the possible location of the object centroid.

Existing work has shown the power of random forests as classifiers and as a fast means of clustering descriptors. We further explore their suitability to object detection in next sections.

3.3.1 Random ferns classifier

To increase the speed of the random forest Ozuysal *et al* [106] proposed random ferns classifiers. Ferns are non-hierarchical structures where each one consists of a set of binary tests (one at each node). During training there are an ordered set of tests S applied to the whole training data set. This is in contrast to random forests where only the data that falls in a child is taken into account in the test. As in random forests "leaves" store the posterior probabilities. During testing the probability that an image belongs to any one of the classes that have been learned during training is returned. The result of each test and the ordering on the set defines a binary code for accessing the "leaf" node. So, a if fern has N nodes then it will have 2^N leaves. As in random forests, the test image is passed down all the randomized ferns. Each node in the fern provides a result for the binary test which is used to access the leaf which contains the posterior probability. The posteriors are combined over the ferns in the same way as for random forests over trees.

3.4 Random Forect for Classification

In this section we show strength of Random Forest as a classifier experimentally by reporting its performance for image level scene and object classification in high level feature detection task of TRECVID'08 [3]. The TREC Video Retrieval Evaluation (TRECVID) [3, 140] is aimed at promoting content-based analysis and retrieval from digital video via open, metrics-based evaluation. In 2008, they tested systems on five tasks:

- surveillance event detection
- high-level feature extraction
- search (interactive, manually-assisted, and/or fully automatic)
- rushes summarization
- content-based copy detection

As a part of the Oxford-IIIT team we participated in the high-level feature extraction task. The high-level features are semantic categories like Cityscape, Street (scene), hand, boat-ship (object) or some action. In this task test videos, common shot boundary reference for the test videos and the list of high-level feature (HLF) definitions are given. The task is to detect these semantic categories or HLFs in the given reference shot and return the list of at most 2000 shots from the test collection for each HLF. The list is ranked according to the highest possibility of detecting the presence of the HLF.

3.4.1 Dataset, Annotations and Evaluation

TRECVID (2008) provided development (100 hours) and test (100 hours) data of video sequences. The development data was annotated in the collaborative effort [10]. Video shots were annotated into 20 semantic categories: (1) *Classroom*, (2) *Bridge*, (3) *Emergency_Vehicle*, (4) *Dog*, (5) *Kitchen*, (6) *Airplane_flying*, (7) *Two people*, (8) *Bus*, (9) *Driver*, (10) *Cityscape*, (11) *Harbor*, (12) *Telephone*, (13) *Hand*, (14) *Street*, (15) *Demonstration_Or_Protest*, (16) *Mountain*, (17) *Nighttime*, (18) *Boat_Ship*, (19) *Flower*, (20) *Singing*.

TRECVID also provides keyframes for each shot, there are a total of 43616 keyframes in development data and 81274 keyframes in test data. Some examples are shown in Figure 3.2. Note the difficulty of these images, take any category there is lot of intra-class variation in color, texture, shape, depth.

During evaluation (annotation) each feature is assumed to be binary, i.e., it is either present or absent in the given reference shot. TRECVID gives results as *Inferred Average Precision* on test data using the submitted runs. While building the system we experiment on development data (which divided into training and validation) and use *Average Precision* as a metric for evaluation.

3.4.2 Our Approach

For the high-level feature extraction task, we have used a combination of different visual features with Random Forest. We used a reduced subset of MPEG i-frames from each shot, found by clustering i-frames within a shot. The approach was to train the classifiers for all high-level features using publicly available annotations [10]. Then run them on the test set and subsequently rank the test shots by the maximum score over the reduced i-frames.

One versus rest classifiers are trained for all the 20 concepts. For node test we have used two types of node functions: (i) difference of two components and (ii) single component of the feature descriptor. We

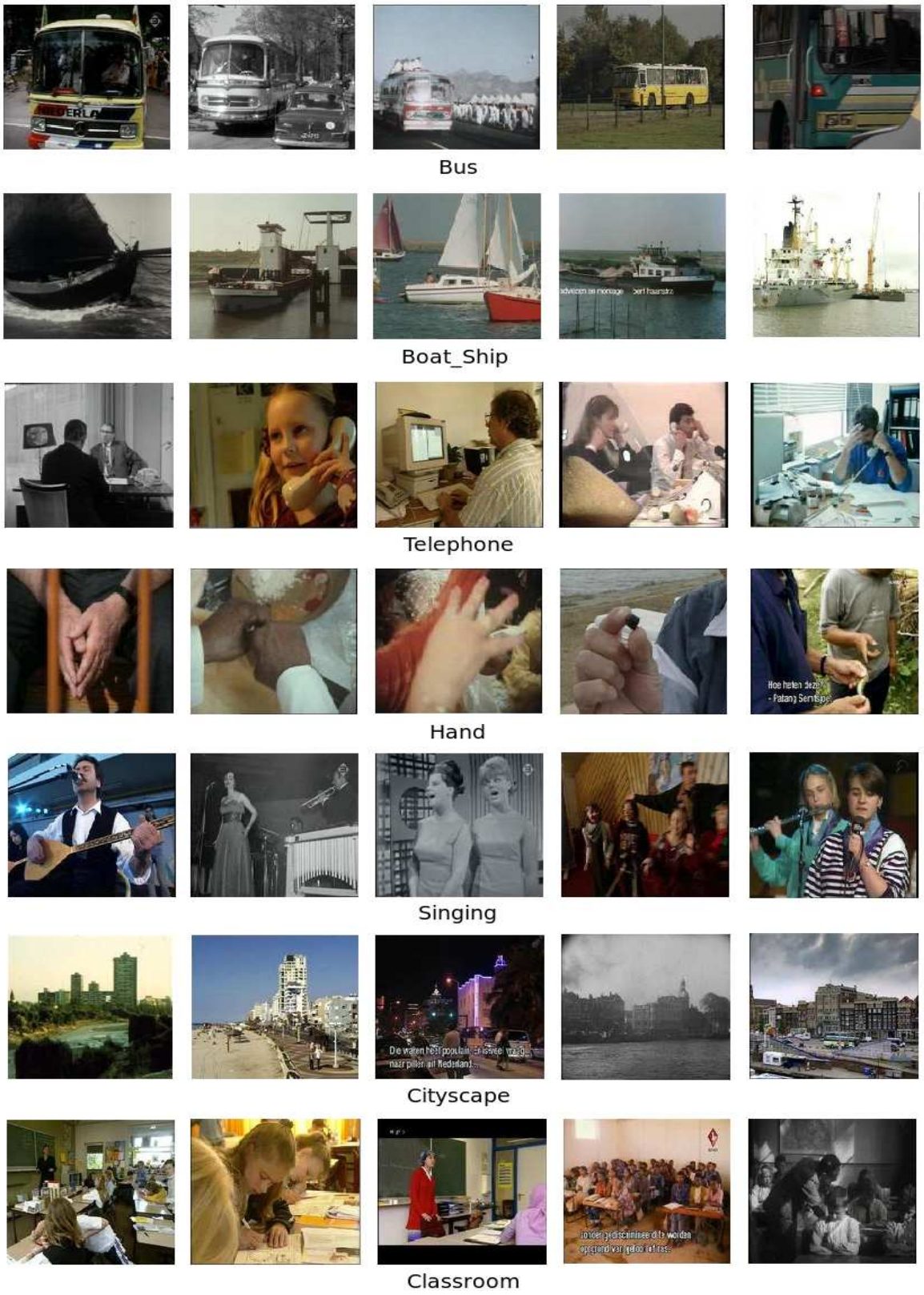


Figure 3.2 Some examples of keyframes from TRECVID dataset

used 100 decision-trees in the Random Forest and maximum allowed depth of a tree was 10. Number of functions chosen at each node (n_f) was kept 100 and number thresholds tried for each node-function at each node (n_τ) was 10. To further inject randomness each tree is trained using 15000 randomly selected samples from the training data. We use a combination of features for example, PHOW + PHOG, PHOW + Color etc to train the classifier. Two appealing features of Random forests which we require here are: (i) probabilistic output and (ii) the seamless handling of a large variety of visual features.

One global feature descriptor is extracted from each image. During testing the feature descriptor of the test image is passed down each random tree until it reaches a leaf node. All the posterior probabilities are then averaged to obtain the classification score of the input image.

3.4.3 Visual representation: Appearance

These descriptors consist of visual words which are computed on a dense grid. Here visual words are vector quantized SIFT descriptors [84] which capture the local spatial distribution of gradients.

Local appearance is captured by the visual words distribution. SIFT descriptors are computed at points on a regular grid with spacing M pixels. We have used gray level representations for each image. At each grid point, the descriptors are computed over circular support patches with radii r. Thus, each point is represented by four SIFT descriptors. These dense features are vector quantized into visual words using K-means clustering. Here, we have used a vocabulary of 300 words. Each images is now represented by a histogram of these visual word occurrences.

We have used $M = 5$, $K = 300$ and radii $r = 10, 15, 20, 25$. To deal with empty patches, we zero all SIFT descriptors with L2 norm below a threshold (200).

In order to capture the spatial layout representation, which is inspired by the pyramid representation of Lazebnik et.al. [75] , an image is tiled into regions at multiple resolutions. A histogram of visual words is then computed for each image sub-region at each resolution level.

Finally, the representation of an appearance descriptor is a concatenation of the histograms of different levels into a single vector which are referred to as Pyramid Histogram of Visual Words (PHOW). Here, we have used three levels at maximum for the pyramid representation. The distance between the two PHOW descriptors reflects the extent to which the images contain similar appearance and the extent to which the appearances correspond in their spatial layout.

3.4.4 Visual representation: Shape

Local shape is represented by a histogram of edge orientations computed for each image sub-region, quantized into K bins. Each bin in the histogram represents the number of edges that have orientations within a certain angular range. This kind of representation is similar to the bag of (visual) words, where each visual word is a quantization on edge orientation.

Initially, edge contours are extracted using the Canny edge detector. The oriented gradients are then computed using a 3×3 Sobel mask without Gaussian smoothing. We have used $K = 8$ bins for an

angular range of [0, 180]. The vote from each contour point depends on its gradient magnitude, and is distributed across neighboring oriented bins according to the difference between the measured and actual bin orientation.

Finally the representation of Shape descriptor consists of concatenation of these histograms in a single vector. This descriptor is referred to as Pyramid Histogram of Oriented Gradients (PHOG). Four pyramid levels were used at maximum for this feature. Each level of PHOG is normalized to sum to unity taking into account all the pyramid levels.

3.4.5 Visual representation: Color

Another feature used is a colour histogram combined with a spatial pyramid over the image to jointly encode global and local information. This is similar to the descriptor proposed in [27], except that we quantize colors more coarsely for all the levels. We used 10, 8, 4 and 2 bins for a cell in levels from 0 to 3 respectively. These are appended to create the final feature vector. We also used feature vector created without decreasing the number of bins with levels.

3.4.6 Experiments for Best Combinations

Classifiers were trained for all the classes using different combinations of:

- features
  - PHOW
  - PHOW + PHOG
  - PHOW + Color
- pyramid levels
  - level 1, 2 for PHOW.
  - level 1, 2 and 3 for PHOG and Color.
- node test
  - single component of feature descriptor
  - difference of two components of feature descriptor
  - single + difference (selected at random)

To combine the features or node tests, one type is randomly selected from the combination. For example, if the combination is: (PHOW + PHOG) + (level 2, level 3) + (single + difference), then randomly one feature from PHOW and PHOG, and one test from single and difference are selected. This makes 6, 18 and 18 possible combinations of pyramid levels and node tests for PHOW, PHOW +

HLF	Feature	Node Test	AP
1 Classroom	PHOW(3) + Color(2)	Single	0.0229
2 Bridge	PHOW(2) + PHOG(2)	Single	0.0391
3 Emergency Vehicle	PHOW(3) + Color(4)	Single	0.0582
4 Dog	PHOW(3) + Color(2)	Difference + Single	0.2095
5 Kitchen	PHOW(2) + Color(4)	Single	0.0571
6 Airplane Flying	PHOW(3) + Color(4)	Difference + Single	0.0482
7 Two People	PHOW(3) + Color(4)	Difference	0.1141
8 Bus	PHOW(3) + Color(3)	Difference + Single	0.0768
9 Driver	PHOW(2) + Color(2)	Difference + Single	0.0775
10 Cityscape	PHOW(3)	Difference	0.2171
11 Harbor	PHOW(2)	Difference + Single	0.1930
12 Telephone	PHOW(2) + Color(3)	Difference + Single	0.0105
13 Street	PHOW(2)	Difference	0.2249
14 Demonstration Or Protest	PHOW(2) + Color(3)	Difference + Single	0.0865
15 Hand	PHOW(3) + Color(4)	Difference + Single	0.1507
16 Mountain	PHOW(2)	Single	0.1309
17 Nighttime	PHOW(3) + Color(2)	Difference	0.2759
18 Boat_Ship	PHOW(2) + PHOG(3)	Difference + Single	0.2764
19 Flower	PHOW(2) + Color(3)	Difference + Single	0.0735
20 Singing	PHOW(2) + Color(4)	Difference + Single	0.0445

Table 3.1 Classification results on Validation set using the best combination of feature, pyramid level and node-test.

PHOG and PHOW + Color, respectively. So, we have 42 possible combinations for each category or HLF.

The advantage of the random forest classifier is the speed of training and testing. Because of fast training and testing it was possible to train 42 classifiers on training data for each concept and find the best combination by testing on validation data. The best combinations for all the HLFs are reported in Table 3.1 with results on validation data. With each feature type the number of pyramid levels used are given in parentheses. For example, PHOW(2) means pyramid level number 0 and 1 are used.

Our method worked well for classes like Dog, Cityscape, Harbor, Street, Hand, Mountain, Nighttime and Boat_Ship. For other concepts like Classroom, Bridge and Telephone in particular results are not good. It is difficult to capture the variations present in these classes. For example take telephone (see Figure 3.2), the definition says: *"any kinds of telephone, but more than just a headset must be visible"*. It can be a cellphone or normal telephone, with person talking or lying on table, with different backgrounds/context. Similarly it is very difficult to represent categories like Classroom, Bridge, Singing, Kitchen Demonstration_Or_Protest etc with just visual features.

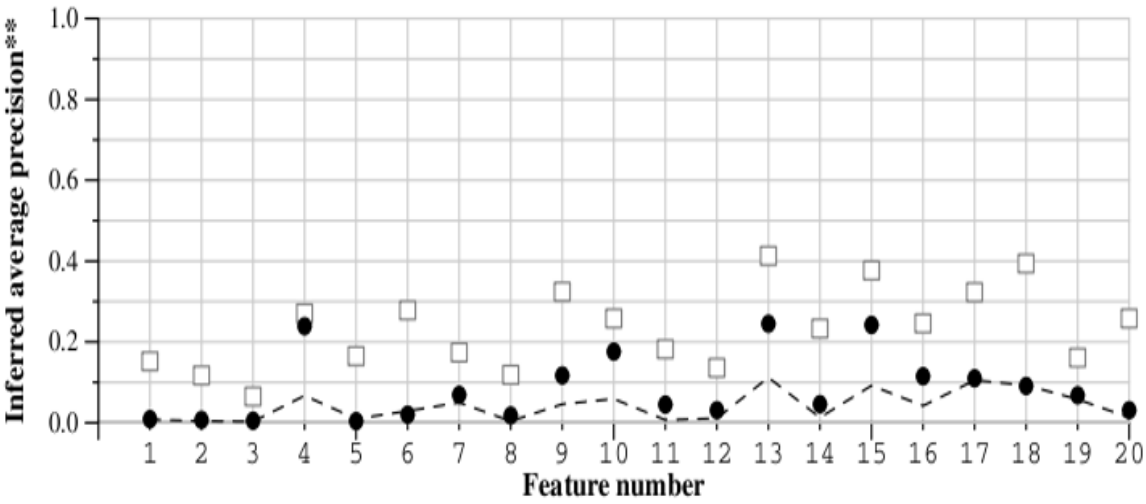


Figure 3.3 Inferred AP for the HLFs: our score (dot), median score (dashes) and best score (box). Inferred AP is estimated using 50% samples.

We observed feature that contributed the most was PHOW. Though Color was helpful for Nighttime, Hand and Flower, but in most of the cases there was only slight improvement in the performance by adding other features.

3.4.7 High-level feature results by TRECVID

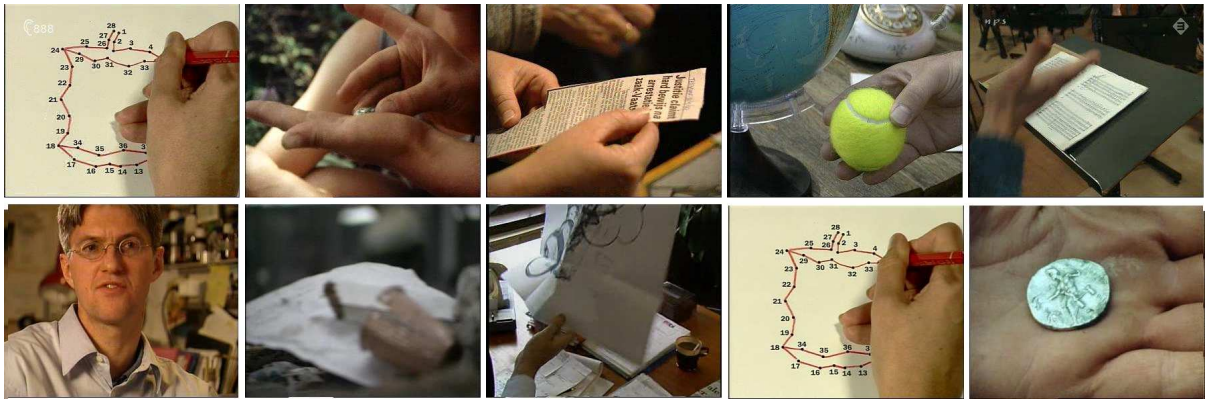
TRECVID uses inferred average precision (inf AP) [173] for evaluating the feature task. One run C_OXVGG 4_4 was submitted using only Random forest approach for all concepts except Two People. The plot in Figure 3.3 gives the inf-AP for scored by our method for all HLFs except feature number 7 (Two People). It compares our score with the best and the median of scores of all the teams by feature. Relative performance of our approach was good for Dog, Cityscape, Street and Hand. Our scores came over the median for all the classes except kitchen and airplane.

Some visual results form test set are shown in Figure 3.4(a) for street and Figure 3.4(b) for hand.

3.5 Object Detection System and Dataset

In this section, we give details of training, testing and post-processing methods we use for object detection. VOC Dataset and detection challenge are also explained.

(a)



(b)

Figure 3.4 Top 10 results (distinct scenes) of (a) Street and (b) Hand.

3.5.1 Training

Each Random Forest classifier is trained to discriminate between candidate regions that do and do not contain an instance of the object of interest. A one-versus-rest classifier is trained for an object category. The trees we train here are binary and are constructed in a top-down manner. For node test we have a node function (difference of two components of the feature vector) and a threshold. Number of decision-trees (T), number of functions per node (n_f), maximum depth are varied from case to case. For training number of positive and negative data samples are required. The ground truth object instances (Region of Interest or ROIs) for a class, plus a number of jittered instances (both from original and flipped training images), are used as positive samples. Regions that do not overlap the target object instances by more than 20% are used as negative samples. The aspect ratio of the detector window is found from the aspect ratios of the ROIs in the training set.

3.5.2 Testing and Retraining

While testing we use a sliding window approach, where a detector window is applied at all positions and scales of an image. The aspect ratio of the detector window is found from the aspect ratios of the ROIs in the training set. Because the number of possible negative samples is exorbitantly large and it is important to find a proper representative sub-set. This is done by bootstrapping or retraining each classifier as follows:

- Train a classifier using positive and negative samples from training data.
- Run the classifier over the training images.
- Compare the detections with the ground truth ROIs, and label them as false positives if the overlap is less than 20%.
- The top false positives are used as hard negative samples and are added to the negative set for retraining.

After one or two rounds of retraining the classifier is ready to be run on test data.

3.5.3 Post-processing

In case of sliding window based detector, strong positive response is obtained even if the detection window is slightly off-center or off-scale on the object. Also when classifiers are trained separately for different poses we have multiple detections for the same object instance. Also a reliable detector would not fire with similar confidence and frequency for non-object windows. So there is need of a post-processing step to merge multiple detections and suppress odd false positives.

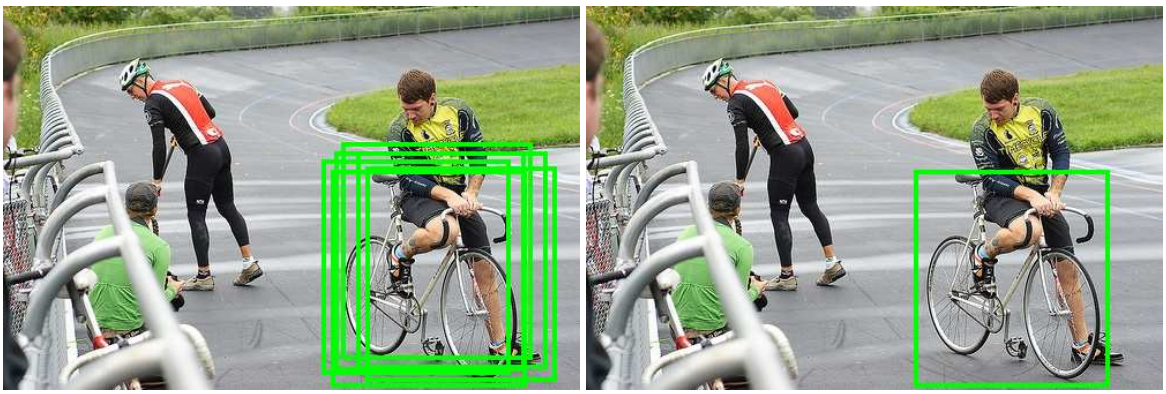


Figure 3.5 Post-processing: On left, a typical result after scanning the binary classifier across the test image at all positions and scales is shown. Results after non-maximum suppression is on right.

Robust non-maximum suppression techniques are proposed in literature. N. Dalal *et al* [33, 34] used a technique which maps each detection to 3-D position in scale-space. Then the mean shift mode detection algorithm is applied to each detection which provides local smoothing of the detections. The result is that strong and overlapping detections (nearby ones in 3-D position-scale space) cluster together. We use a very simple technique for non-maximum suppression given in [161]. The most highly ranked candidate is selected, all other candidates with an overlap greater than 20% are removed and the process is repeated until a safe number (images do not contain more than a few instances of an object) of candidates are selected.

3.5.4 VOC PASCAL dataset and Object Detection Challenge

The PASCAL Visual Object Detection Challenge (VOC) [42] data consists of a few thousand images annotated with bounding boxes for objects of twenty categories (e.g., car, bus, airplane, ...). Along with bounding boxes the views or poses are also provided for each object instance as *frontal*, *rear*, *left*, *right* or *unspecified*. Ground truth also has information if the object is truncated or occluded. There are four sets of images provided: *train*, *val*, *trainval* (union of *train* and *val*) and *test*.

The *detection* challenge is the following: predict the bounding box and label of each object from the target classes in a test image. Each bounding box is output together with a confidence value, and this value is used to generate a precision-recall graph for each class. Detections are considered true or false positives based on their overlap with ground truth bounding boxes. The overlap between a proposed bounding box R and the ground-truth box Q is computed as:

$$\frac{area|Q \cap R|}{area|Q \cup R|}$$

An overlap of 50% or greater is labeled as true positive. Any additional overlapping bounding box (duplicate detections) are rejected as false positives. Performance is then measured by the average precision (AP). Full details of the challenge, including the results of all participants, are given at [42].

3.5.4.1 Features and Object Detector

The descriptor for appearance of the Region of interest (ROIs) is computed using different features. We construct the descriptors in a manner exactly similar to [161] using open source code [160].

Dense words (PHOWGray, PHOWColor) [19]: Rotationally invariant SIFT descriptors are extracted on a regular grid each five pixels, at four scales (10, 15, 20, 25 pixel radii), zeroing the low contrast ones. Descriptors are then quantized in 300 visual words. The color versions stacks SIFT descriptors for each HSV color channels.

Histogram of oriented edges (PHOG180, PHOG360): [34, 19]. The Canny edge detector is used to compute an edge map and the underlying image gradient $\bigtriangledown I(p)$ is used to assign an orientation and a weight to each edge pixel p. The orientation angle is then quantized in eight bins with soft linear assignment and a histogram is computed.

Again we use spatial layout representation of [75] and a three-level pyramid of spatial histograms is computed for each feature. All the histograms/descriptors are l^2 normalized.

3.6 Random Forests Vs Support Vector Machines

Support Vector Machines (SVMs) is one of the leading techniques used for discriminative classification in vision tasks ranging from detection of objects in images like pedestrians [34, 96], cars [108] and others objects [49], multicategory object recognition in Caltech-101 [18, 75], to texture discrimination [179]. Part of the appeal for SVMs is that non-linear decision boundaries can be learned using the so called kernel trick. Though SVMs have faster training speed compared to variants of boosted classifiers, the run time complexity of a non linear SVM classifier is high.

It can be advantageous to use Random Forests over SVMs or boosting because of the following properties:

- their computational efficiency in both training and classification
- independence of the trees allows for easy implementation and parallelism ([132])
- their probabilistic output
- the seamless handling of a large variety of visual features (e.g. colour, texture, shape, depth etc.)
- the inherent feature sharing of a multi-class classifier (see also [149] for feature sharing in boosting).

3.6.1 Support Vector Machines

Support vector machines (SVM) are a widely used tool in the machine learning and computer vision community. They were motivated by results of statistical learning theory and originally developed for pattern recognition they are now described in various books [129, 157] and tutorials, e.g. Burges [23]. The basic idea is to learn a hyperplane in some feature space in order to separate the positive and negative training examples with a maximum margin, thus called maximum margin classifiers. Also see [32] for an early reference. There have been various extensions and improvements over the years. One example is the recent variation [155] to enable the learning of structured output spaces instead of simple two or multi-class classification problems. Bach *et al* [11], Varma and Ray [158] extend SVMs to multi-kernel learners, which combine various base kernels into an optimal domain-specific kernel.

Straightforward classification using kernelized SVMs requires evaluating the kernel for a test vector and each of the support vectors. The complexity of classification for a SVM using a non linear kernel is the number of support vectors $\times$ the complexity of evaluating the kernel function. The later is generally an increasing function of the dimension of the feature vectors. Since the testing is expensive with non-linear kernels, linear kernel SVMs have become popular for real-time applications as they enjoy both

Training Round	Pose	Random Forests		fast IKSVM			
				C=0.1		C=0.05	
		Time (second)	Memory (MB)	Time (second)	Memory (MB)	Time (second)	Memory (MB)
Round 0	F + R	212	122	626	362	804	360
	L + R	258	125	1137	396	1052	394
	U	319	126	1518	482	1714	490
Round 1	F + R	440	235	1740	630	2179	626
	L + R	497	239	2610	670	2978	672
	U	659	241	3654	680	4464	685

Table 3.2 Time and memory requirements while training

faster training and classification speeds, with significantly less memory requirements than non-linear kernels due to the compact representation of the decision function. But this comes at the expense of accuracy ([86]) and linear SVM can not be used on tough datasets like VOC PASCAL.

There has been a fair amount of research on reducing run time complexity of non linear kernels [22, 69, 105, 177]. These approaches are either aimed at reducing the number of support vectors by constructing sparse representations of the classifier, or reducing the dimension of the features using a coarse to fine approach. In [86], an orthogonal approach is proposed for speeding-up intersection kernel SVMs (IKSVMs), a classifier which have been successfully used for detection and recognition [53, 75]. Maji *et al* have shown that one can build (IKSVMs) with runtime complexity of the classifier logarithmic in the number of support vectors as opposed to linear for the standard approach. In the next subsection we compare the performance, training and testing times and memory requirements of fast IKSVM of [86] and Random Forests for object detection. For fast IKSVM, the exact version using binary search is used for the comparison.

3.6.2 Comparison

We experiment on PASCAL VOC 2007 [45] dataset for car detection. See section 3.5 for the details of the dataset and training, testing, retraining and non-maximum suppression methods for object detection. We train using *Trainval* set and test on *Test* set. The training data is divided based on the poses or views of the object instances and train separate classifiers for each set. Some poses are relatively easier to learn than others, thus we can also compare how performance varies with learning complexity.

3.6.2.1 Training

We extract PHOG360 [20, 161] features from the positive (ROIs containing cars) and negative samples taken from training data. Then Random Forest and fast IKSVM are trained (round 0) using these features. In each case we train classifiers separately for Frontal+Rear, Left+Right and unspecified poses. So we have three Random Forests and three IKSVMs trained. We run these six classifiers over the train-

ing data to find hard-negatives for each of them and then retrain (round 1) the classifiers with respective hard-negatives added. There were 175, 222 and 332 positives samples for Frontal+Rear, Left+Right and unspecified poses respectively. From each positive sample 40 jittered and flipped instances were generated for the final positive set. Around 30K negative samples were selected (6 taken from an image at random) from *Trainval* set for each pose in round 0 and 20K high scoring false-positives were added to the negative set in round 1.

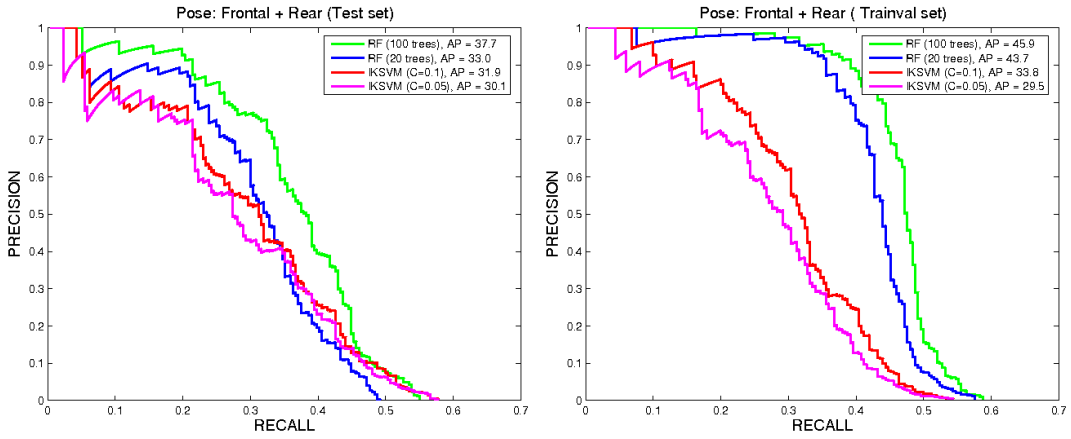
Table 3.2 reports the time and memory requirements for both the classifiers for the two rounds of training. For fast IKSVM we use code provided by [2]. The training time and performance depends significantly on C parameter for SVM. The C parameter was varied and set to 0.005, 0.001, 0.05, 0.01, 0.1, 1, 5, 10, 100 and 1000. The values performing best on *Test* set were found to be 0.1 and 0.05. For Random forest we use 100 decision trees each of maximum depth 15, node-functions per node (n_f) =100, and thresholds per node (n_τ) = 10. Memory requirements for Random Forests is half of that of SVMs as only half of the training samples selected randomly were used to train a decision tree. In case of Frontal + Rear pose in round 0, time taken to train Random Forests is about half or one-third of that of IKSVM (for C=100, 1000). But with increase in number of samples and learning complexity (other poses are more difficult to learn) the training time for IKSVM increases rapidly, while it is more stable in case of Random Forest.

3.6.2.2 Testing

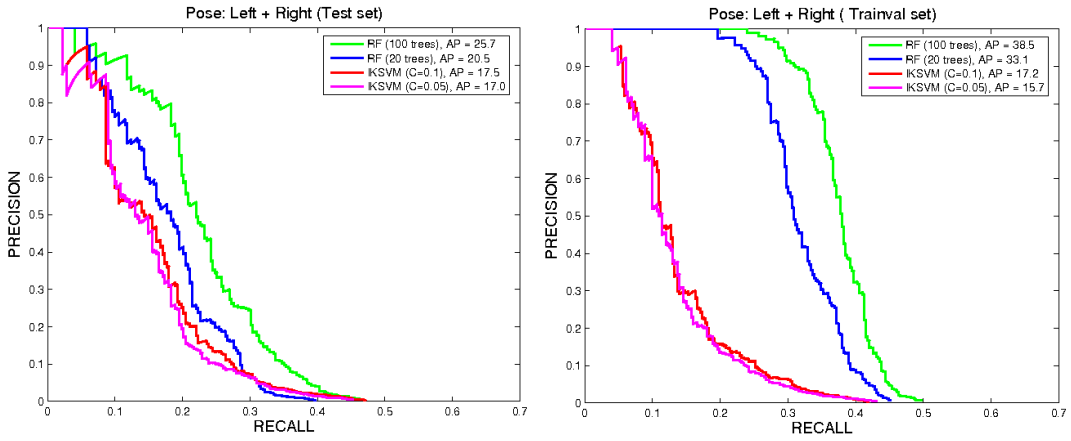
The trained detectors are run over *Test* and *Trainval* set using Sliding window approach, followed by non-maximum suppression. We evaluate the result by usual VOC method: any detection/localization is considered true positive when the area overlap of the detected bounding-box with groundtruth is more than 50%. To each detection a confidence score is assigned by the classifier. Fig 3.7 shows the *precision-recall* plots of the results on test and train set. Table 3.3 compares the results of the detectors.

Random Forest detector obtains better Average Precision score before and after bootstrapping. The difference ranges from 6% to 11% across the poses on test data, for trainval it is even more. Though IKSVM has inferior AP score but it achieves higher recall in some cases. There is slight over-fitting in RF. It's performance relative to fast IKSVM on test set is not as good as that on trainval set. While testing, time taken by fast IKSVM to classify around 50K samples ranges from 2.2 to 4 seconds depending on value of C parameter (or number of support vectors). RF with 100 trees takes approximately 2 seconds for the same task. With only 20 trees also, which 5 times faster, RF performs better than IKSVM (Fig 3.7). All the experiments are done on an AMD Athlon Dual Core Processor (3GHz) with 4 GB RAM.

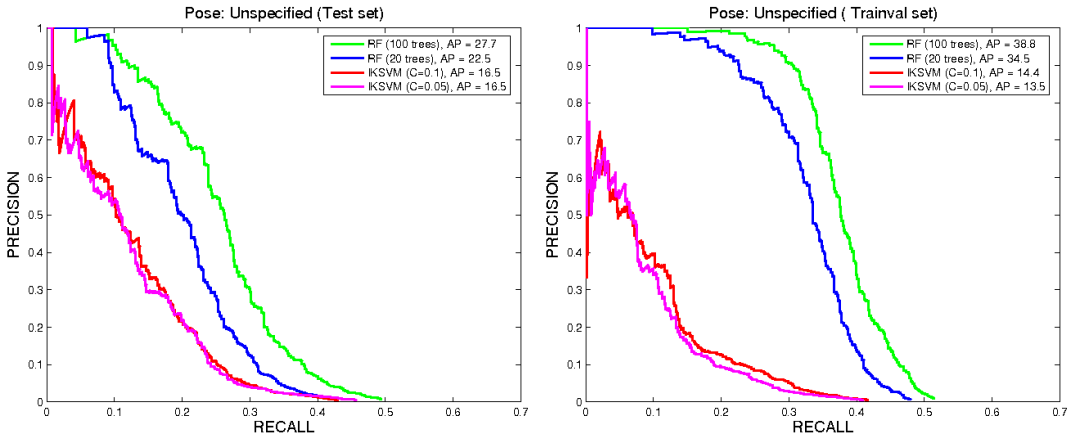
Random Forest shows promising results as an object detector. It achieves better AP and speed than fast IKSVM, specially for left+right and unspecified poses it outperforms IKSVM. We have seen here that with its speed and accuracy it can serve as a robust object detector. In section 3.7 we experiment with Random Forest using multiple features, and increasing the number of trees.



(a)



(b)



(c)

Figure 3.6 VOC 2007 car detection: Performance of Random forest and fast IKSVM compared in the precision-recall plots for test set (left) as well as trainval set (right). Comparison is done separately for (a) Frontal + Rear, (b) Left + Right and (c) Unspecified poses.

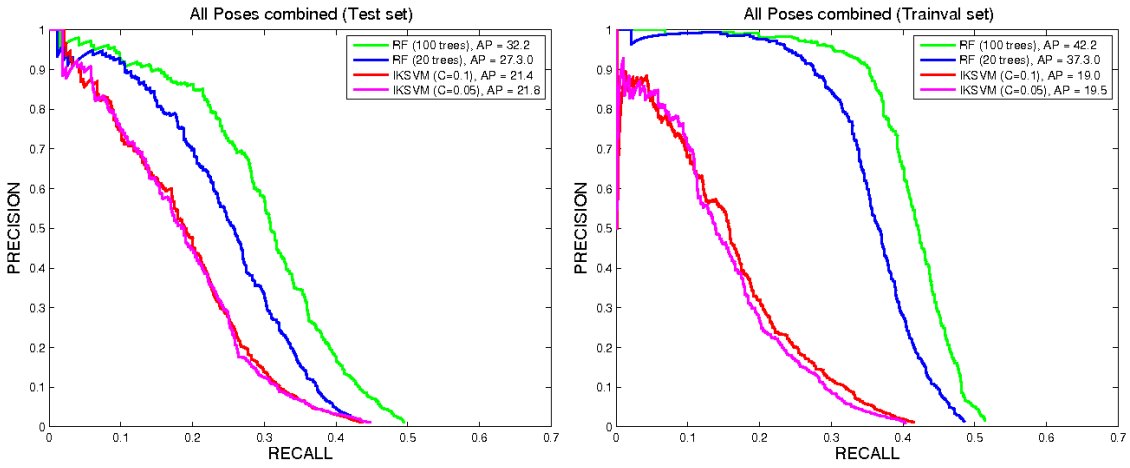


Figure 3.7 VOC 2007 car detection: Performance of Random forest and fast IKSVM compared in the precision-recall plots for all the poses combined.

Poses	Test Set	Random Forest		fast IKSVM			
				C=0.1		C=0.05	
		Avg Prec	Recall	Avg Prec	Recall	Avg Prec	Recall
F + R	Trainval	**45.9**	**58.8**	29.5	54.4	33.8	54.4
	Test	**37.7**	55.1	30.1	57.8	31.9	**57.8**
L + R	Trainval	**38.5**	**49.7**	15.7	43.1	17.2	42.1
	Test	**25.7**	46.9	17.0	45.3	17.5	**47.2**
U	Trainval	**38.8**	**51.5**	13.5	41.0	14.4	41.8
	Test	**27.7**	**49.4**	16.5	45.8	16.5	43.0
All	Trainval	**42.2**	**51.4**	19.5	41.5	19.0	40.4
	Test	**32.3**	**49.5**	21.4	44.9	21.9	43.8

Table 3.3 Testing on VOC 2007 car: Performance comparison

3.7 Random Forest for Object Detection

In this section we present baseline results with Random Forests for object detection. We use the state-of-art features (PHOW, PHOG) [19, 20, 34] and techniques, and employ Random forest as the classifier. Object localization experiments are done on PASCAL VOC2007 for car dataset. We also observe the effect of different parameters of random forest.

The random forest object classifier is learned as per the details given in section 3.5. We divide the training data into disjoint sets based on the views of the object instances and train separate classifiers for each set. This is important because it simplifies learning as there is some alignment among the object instances of same pose or view. Also object instances of same pose generally have similar aspect ratios, this helps during testing. While testing we use one or more representative bounding boxes or templates per classifier as sliding detector window chosen based on the ROIs in the training set. Each

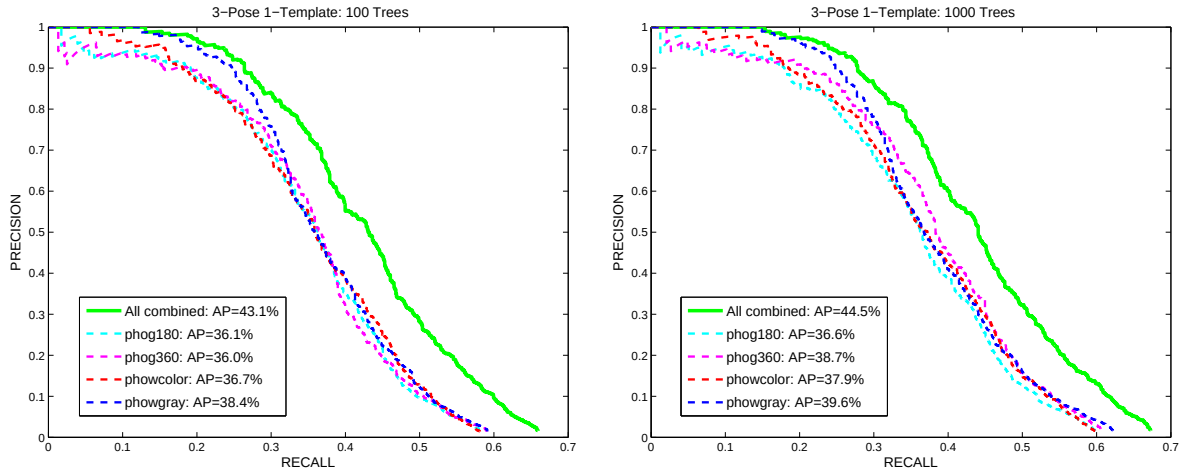


Figure 3.8 3-Pose 1-Template: Performance of different features are compared using 100 tree RF (left) and 1000 tree RF. Large improvement is achieved by combining the features.

template has a fixed aspect-ratio which may slide in scale-space (scale changes) or just in space (scale remains constant). The aim is to cover the variation in aspect-ratio and scale by using a set of templates with a classifier. So, given a test image we run RF detectors trained for different poses and combine the results to get final detection results. Results for different poses are combined by applying non-maximum suppression on ROIs detected by all the pose classifiers. Different types of classifiers based on training (number of poses or classifier per class) and testing (number and type of templates) can be defined as *N-Pose, M-Template*. We analyze performance and efficiency of some of the possible classifiers.

3.7.0.3 3-Pose 1-Template Classifier

The training data is divided into 3 sets based on poses: (1) *frontal+rear*, (2) *left+right* and (3) *unspecified*. Each classifier slides one window of a fixed aspect-ratio found by taking mean of aspect-ratios of the bounding boxes in the associated set. While running classifiers we evaluate around 150K to 180K windows per image with all 3 pose classifiers. We use two values for number of decision trees (T), 100 and 1000, each tree has maximum depth 15. Other parameters are set as: thresholds per node $\tau = 10$, node-functions per node (n_f) is set to 100 for PHOG features and to 300 for PHOW features. Each tree is trained with 50% of the total samples selected at random.

Precision-recall curves in Figure 3.8 compares the performance of different features. Better results are obtained by using PHOW features in both the cases. Combining features yields a significant improvement in Average Precision. Using 1000 trees we have improvement of more than 1% in each case except for PHOG180 and for combination of features AP increases by 1.4%.

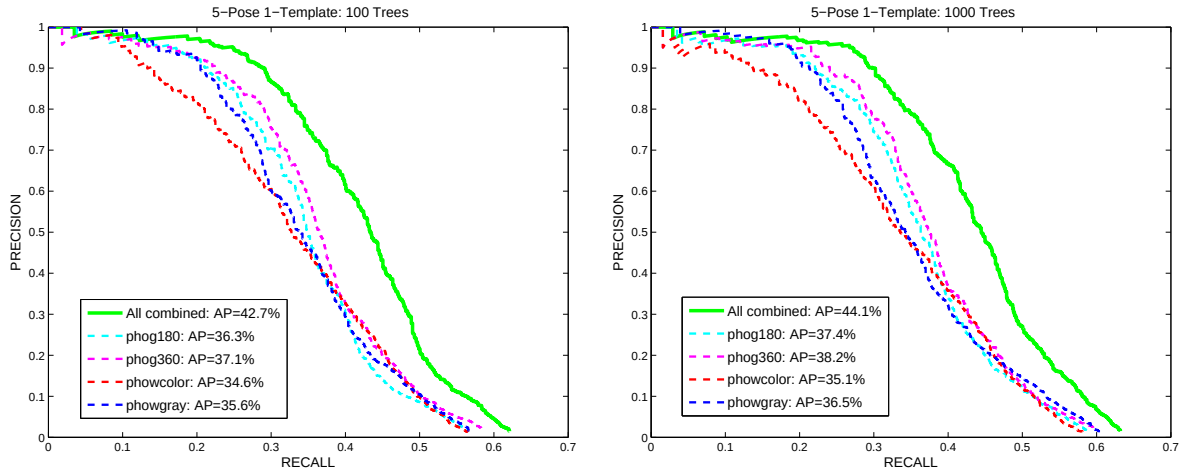


Figure 3.9 5-Pose 1-Template: Performance of different features are compared using 100 tree RF (left) and 1000 tree RF. Large improvement is achieved by combining the features.

3.7.0.4 5-Pose 1-Template Classifier

The training data is divided into 5 sets based on the given 5 poses. So five classifiers are trained each associated with a set and an aspect-ratio chosen from that set. Random forest parameters are same as used for 3-Pose 1-Template classifier. We also run on lesser windows per image per pose classifier so that total number of windows evaluated remains same as that for 3-Pose 1-Template.

Figure 3.9 shows precision-recall plots of different features for 5-Pose 1-Template Classifier. Here performance with PHOG remains similar to what it is obtained in case of 3-Pose 1-Template, but APs with PHOW drop. This is probably because of lesser number of training samples per pose to learn PHOW feature which have higher dimension (6300) than PHOG (336).

Some examples of car detection from VOC2007 *Test* set are shown in the Figure 3.10. The localization results after non-maximum suppression are in green. We also draw groundtruth ROIs in yellow to show the detection overlap and false negatives. High scoring detections of cars of different sizes, orientation and variations are included. Correct detections are drawn in green and false positives in red. Some false negatives are present in last two rows which occur mainly due to truncation, occlusion and very small size. There are instances shown in the last row where truncated cars are also detected.

3.7.0.5 Effect of Random Forest parameters

Here we analyze the effect of random forest parameters on performance and speed for object detection. We basically experiment by varying one of the three: number of node-functions per node (n_f), number of thresholds per node (τ) and number of trees (T), while keeping the other two constant. We

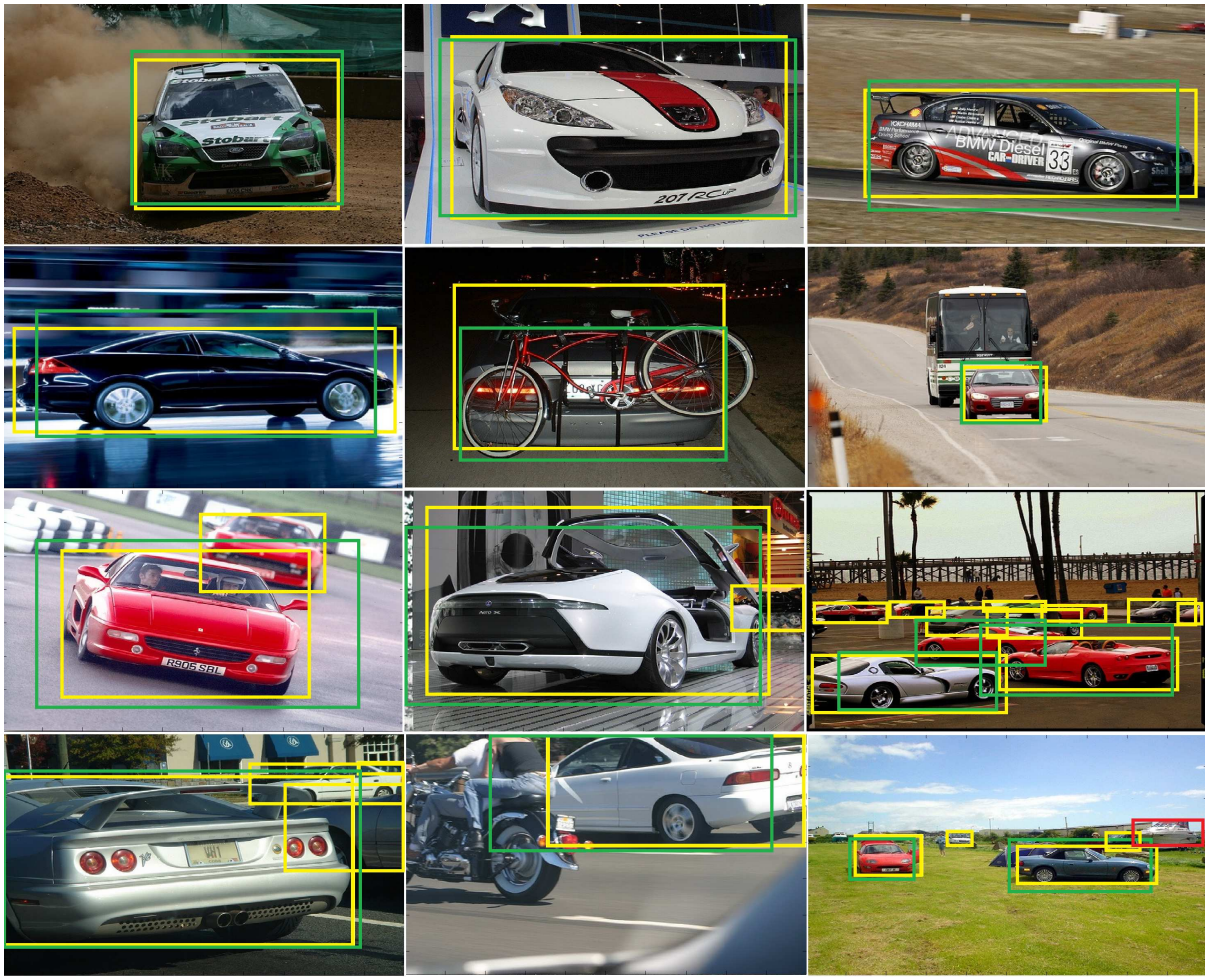


Figure 3.10 Some examples of localization on VOC2007 Test set after non-maximum suppression. Detections are shown by green boxes and groundtruth ROIs are drawn in yellow.

again take car as the target object and use *train* set for training and test on *val* set of VOC 2007. Classifier used for this experiment is 3-Pose 1-Template RF and feature used is PHOG360.

As default we set: $T = 50$, $n_f = 50$ and $\tau = 8$ and vary one at a time. Maximum depth for each tree is kept 10 and 50% randomly training samples are used to train each tree. In each case, classifier is trained 5 times and the mean of *average precisions* is reported. In Figure 3.11(a), τ is varied from 1 to 26 and in Figure 3.11(a) n_f is varied from 1 to 150 with constant parameters kept at their default values. Initially AP increases with number of features and thresholds for both *Train* and *Val* sets but after a point it becomes stable. Variation in τ or n_f doesn't effect the testing time but training time increases linearly with each of them.

Plots in Figure 3.11(c) show the effect of number of trees on the performance. Average Precision increases rapidly for first 20-30 trees and increases considerably till 100-120 trees and then saturates or

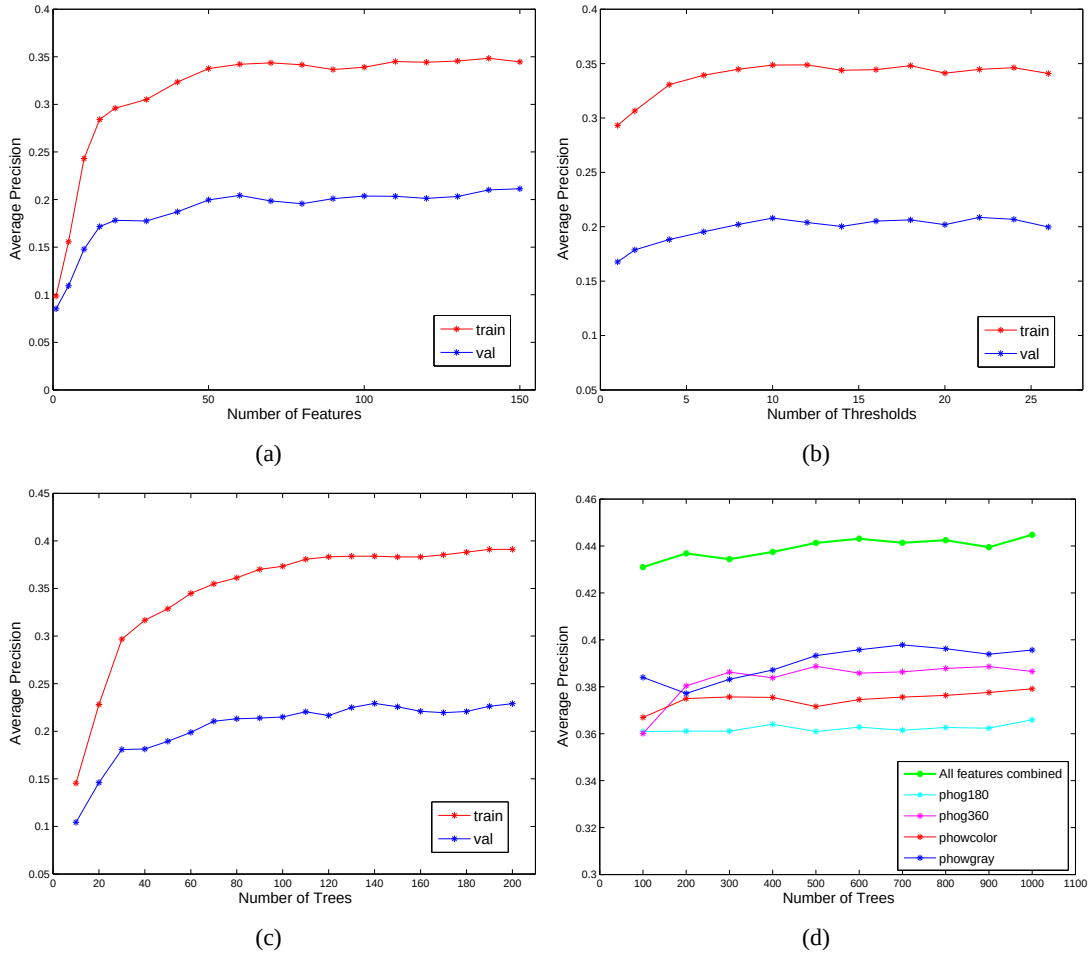


Figure 3.11 Effect of (a) number of thresholds per node (τ), (b) number of node-functions per node (n_f) on average precision, (c) and (d) trees (T) on average precision.

grows slowly. Random Forest do not over-fit when number of trees is increased and improvement can be expected by using very large number of trees. We do one more experiment now on *Test* data, with number of randomized trees increased upto 1000. Figure 3.11(d) shows how *AP* changes for different features and there combination. For every case there is some improvement though not much. For combination of features *AP* improves by 1.4% to 44.5%. The best result for 'car' category among all methods submitted to the VOC 2007 challenge was 43.2%.

We gain 1%-2% by using 1000 trees but this increases complexity by a factor 10. There is a need to improve efficiency, keeping the performance same. We address this issue in the next section by the observation that a small number of trees can also provided decent accuracy in detection.

3.8 Speeding up with cascade structure

The baseline random forest object detector is almost as accurate as state-of-art still the efficiency is not enough for many applications. The main technical obstacle is searching for the best region in the scale, space and aspect-ratio. This also increases training time as multiple rounds of bootstrapping are required. Exhaustive search requires number of operations proportional to the number of regions tested by the classifier, which typically ranges in 1.4×10^4 to 2×10^4.

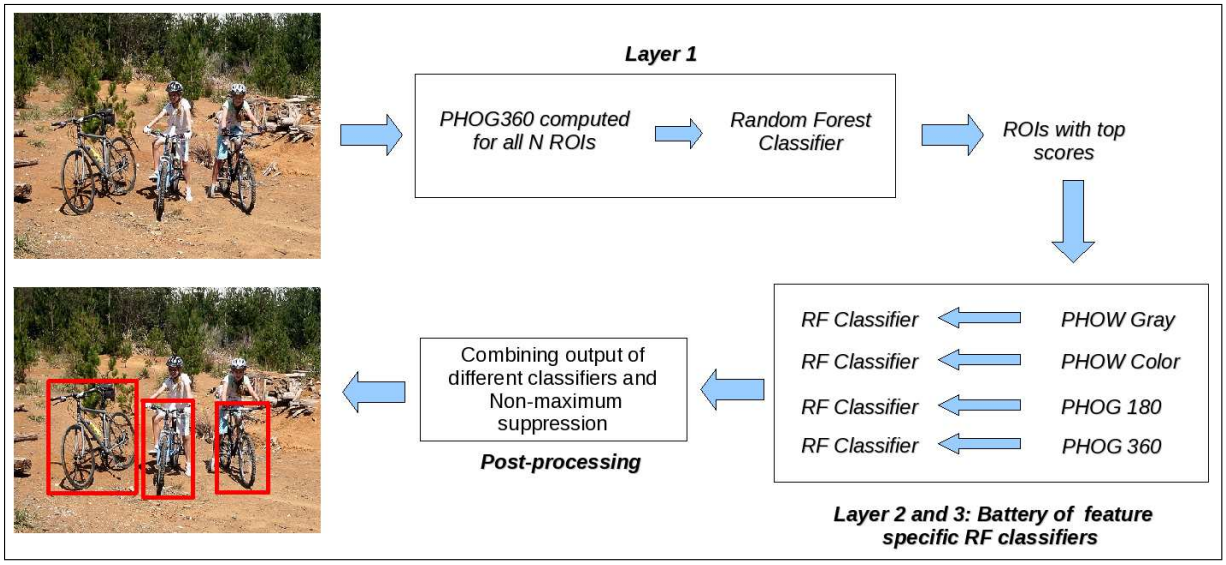


Figure 3.12 Cascade structure of classifiers and features.

We use cascade of increasingly strong classifiers similar to [161, 165] for speed-up. This is natural to random forests and can be done by increasing number of trees used for classification. For example if we have a RF classifier with T trees which is used as n layers. In any layer i only a fraction, fT_i, of total trees are used to classify and based on the detection scores only a top fraction, fR_{i+1}, of total regions are passed to the $i+1^{th}$ layer. In this way in i^{th} layer $fT_i \times T$ trees are used to classify $fR_i \times N$ regions, where N is the total number of regions to evaluate from an image. Speed up obtained by using L layers would be:

$$\frac{1}{\sum_{i=1}^{L} fT_i \times fR_i}$$

In our experiments we use Random Forest in three layers. We observe that by setting $fT_1 = 0.025$, $fT_2 = 0.1$, $fT_3 = 1$ and $fR_i = 0.1^{(i-1)}$ with $T = 1000$, there is no or very less loss in performance. And the speed-up obtained for classification over baseline RF with 1000 trees is around 22 times.

Unlike [161, 165] we build cascade which is also based on complexity of computing feature descriptors. PHOG is used in the first layer which can be computed quickly for all the N number of regions. Based on detection scores only a top $fR_2 \times N$ regions are passed to the second layer. PHOW descrip-

tors are computed only for these top ROIs which saves descriptor computation along with classification time. Computing 6300 dimensional PHOW descriptor using vocabulary of 300 (300 integral images are used) is around 10 times slower than computing PHOG. Now this computation is only done for top f_R fraction of total ROIs visited. Practically, the time required to compute descriptors is reduced by a factor of $\frac{1}{fR_2}$ by using the cascade. Figure 3.12 demonstrates our Cascade structure.

Tables 3.4 and 3.5 summarize the results of our system on the VOC 2007 (car, bicycle and boat) and 2009 (aeroplane, motorbike and boat) datasets. Type of classifier used is 3-Pose 1-Template. The performance with cascade is as good as the baseline system but with significant speed-up. To classify and compute descriptors (all 4 features) for 45K samples it requires approximately 3 seconds and 1.8 seconds respectively. We also compare our result with other systems that entered the official competition in 2007. Our results are better than the team ranked 1 in VOC 2007 for car and boat. For VOC 2009 the results shown are on validation data (training is done on train set) as the groundtruth for the test set is not available.

	car	bicycle	boat
a) baseline	44.5	38.5	9.5
b) cascade	44.3	37.7	9.5
c) voc07 (rank1)	43.2	49.9	9.4

Table 3.4 PASCAL VOC 2007 results (test set): (a) average precision scores of the base system, (b) scores using cascade, (c) top result in VOC07

	aeroplane	motorbike	boat
a) baseline	38.0	26.4	9.9
b) cascade	37.3	26.1	9.7

Table 3.5 PASCAL VOC 2009 results (validation set): (a) average precision scores of the base system, (b) scores using cascade

Some examples of detection are shown from VOC 2007 and 2009 dataset in Figure 3.13. The correct detections are shown in green and ground-truth bounding boxes are shown in yellow. False positives cases are highlighted by red color. Again false negatives occur mainly when object is truncated, occluded or of very small size.

3.9 Extended ROIs

There are lot of variations in shape and appearance of objects, (such as caused by extreme viewpoint changes) that are not well captured by a single template (or aspect ratio). It is common to use multiple templates to encode view or pose variations, for example separate templates for frontal and side views of faces and cars [128]. To interpret the variations in large and difficult datasets like TRECVID or VOC

Figure 3.13 Examples of high-scoring detections on the PASCAL 2007 (top 3 rows) and 2009 (bottom 3 rows) datasets. Last two images in each row illustrate false positives or false negatives for each category.

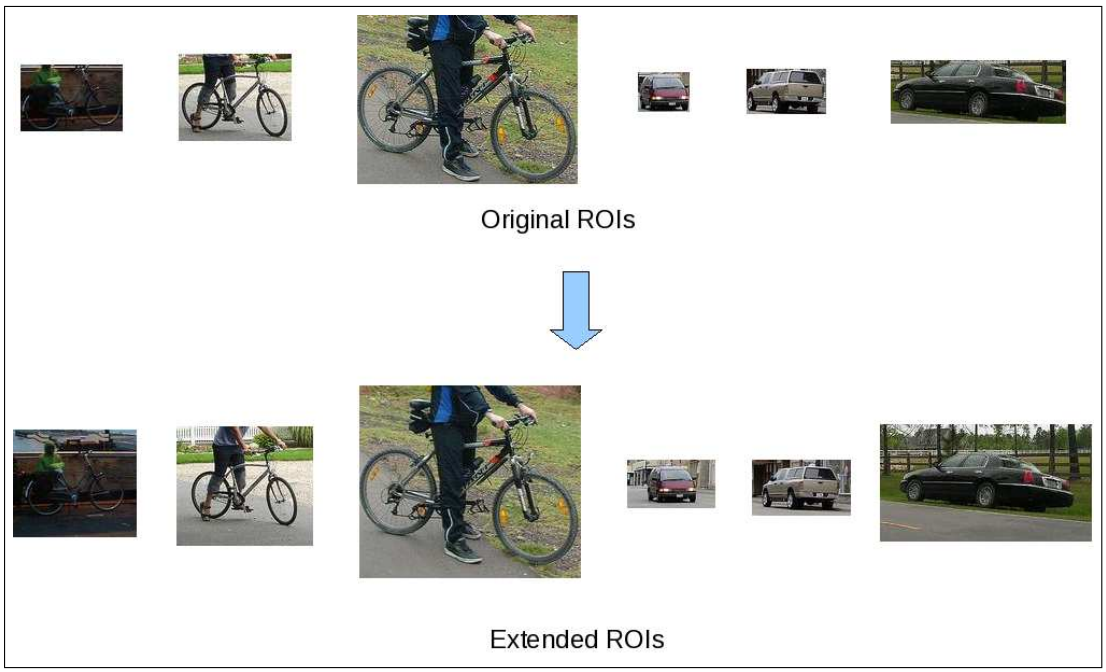


Figure 3.14 Top row shows the examples of original ROIs for classes bicycle and car, and their extended ROIs are shown in the bottom row. Note that all the extended ROIs of same class have same aspect ratio.

it would require many templates. Applying them over such a large dataset for testing/bootstrapping is computationally very expensive. To deal with this we use what we call as *Extended ROIs*.

Extended ROIs are obtained by extending the original ROIs such that its aspect ratio becomes the selected one and its center coincides with the center of original ROI. This allows us to use only one aspect ratio and still cover for large range of aspect ratios while testing. Figure 3.14 shows the original ROIs from and their extended versions. While training Extended ROIs were used as the positive samples. Then all the training ROIs as well as the detector window (while testing) had the same aspect-ratio. The detected windows were also extended ones with object at its center. With this approach we only search in scale and space as the aspect-ratio is constant. We found this very useful for our classification by detection approach for high-level feature extraction task in TRECVID 2009 (see section 3.9.1).

3.9.1 TRECVID 2009

Details about TRECVID tasks, dataset, annotations and evaluation are given in section 3.4. As a part of the Oxford-IIIT team we participated in the high-level feature extraction task of TRECVID again in 2009. Like TRECVID'08 this time also one of our approaches was based on random forests. There were some major changes in our approach for TRECVID'09:

- Classification by detection: In 2008, we used random forest for whole image classification but this time we focused on object categories with sliding-window random forest object detector.
- Removal of noise from TRECVID annotations: We found the collaborative annotations for the TRECVID high level features to be quite noisy. Some shots are wrongly annotated, and others are labeled as 'skip' when they are, in fact, unambiguously positive or negative for the feature. To remove this noise in the annotation, we used a weak classifier trained on the noisy data for each high level feature as follows:
  - Train a classifier using all the +ves and a subset of -ves in TRAIN and VAL sets according to the Collaborative Annotation.
  - Re-rank all the images in the TRAIN+VAL set based on the classifier output.
  - Refine the annotations of the top 5000 ranked images.

  In this manner, we could find many of the wrong annotations with minimal manual effort. This refinement was found to be very effective.
- Use of extra data: Additional data was taken from sources like flicker, google which for under-represented features (like bus) significantly improved performance.
- Bounding-box level annotation: The images containing the target object categories were manually annotated by marking the bounding box of the visible area of the object.

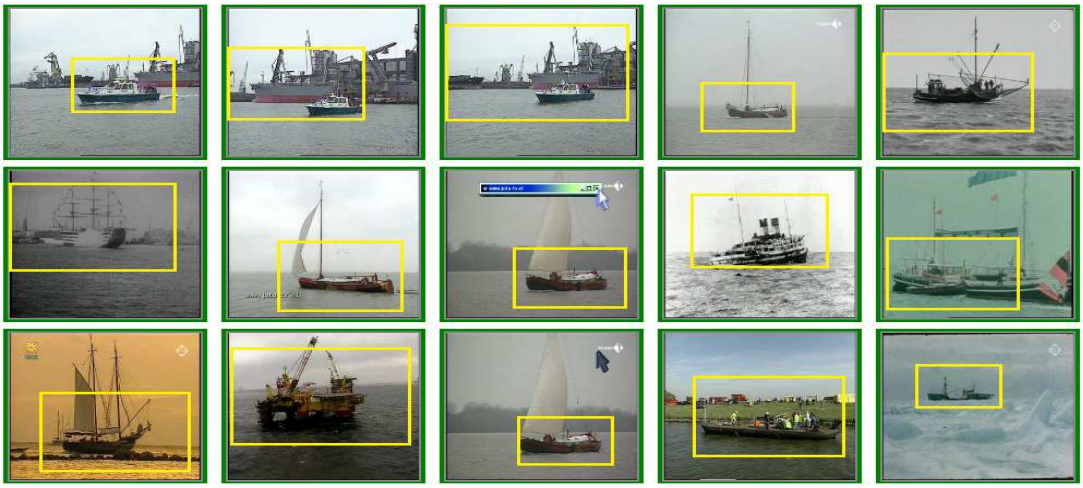
3.9.1.1 Classification by detection

One versus rest random forest classifiers were trained using the ground truth bounding-boxes for the target object. For representing we use only PHOW (explained in) as feature as it performed best in our earlier approach (see 3.4.6).

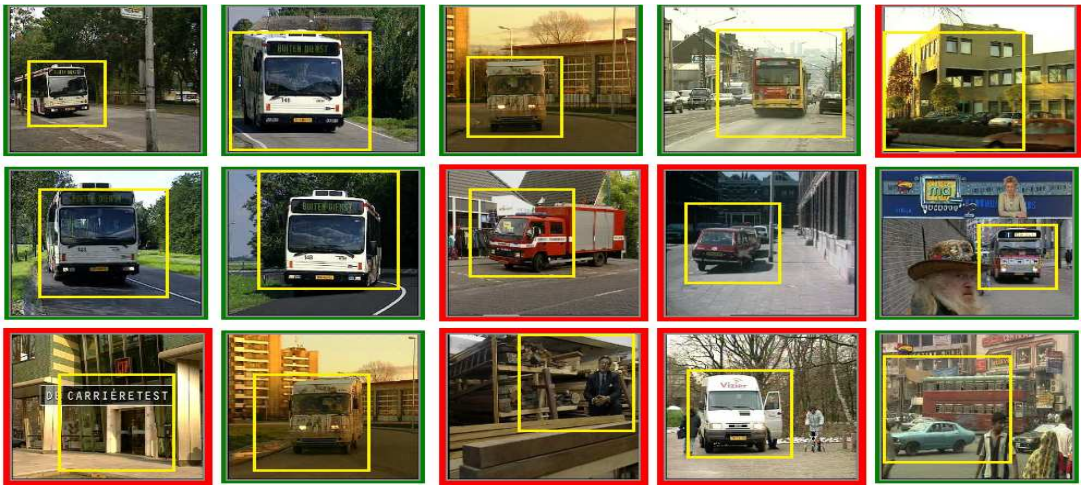
For detection we experiment with our baseline object detector and the one which is trained on *Extended ROIs*. The confidence score of a test image is the maximum of the classification scores of the regions in it. In our baseline object detector we use only one view and one template (1-Pose 1-Template classifier). As the dataset is very large (209990 keyframes) using multiple template is very expensive. The result of this detector for boat_ship class is reported in Table 3.6.

Training set	Test set	Training Round	AP (50 trees)	AP (100 trees)
Train	Train	0	0.1435	0.1833
Train	Validation	0	0.0294	0.0349
Train	Train	1	0.3782	0.4386
Train	Validation	1	0.1940	0.1977

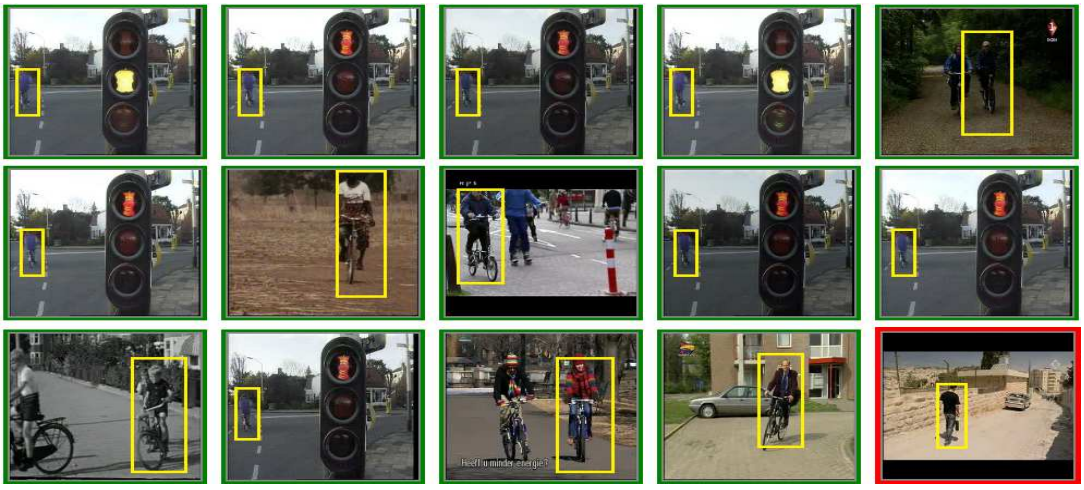
Table 3.6 Classification by detection results for Boat_ship: average precision scores of the base detector before and after bootstrapping



(a) Boat-Ship



(b) Bus



(c) Person-riding-a-bicycle

Figure 3.15 Top 15 retrieved keyframes are shown for (a) Boat-Ship and (b)Bus categories, (c) Keyframes ranked from 71 to 85 are shown for Person-riding-a-bicycle, top 70 are all true positives coming from the same video.

Training set	Test set	Training Round	AP (50 trees)	AP (100 trees)
Train	Train	0	0.5340	0.5525
Train	Validation	0	0.2552	0.3080
Train	Train	1	0.5874	0.5876
Train	Validation	1	0.2787	0.4106

Table 3.7 Classification by detection results for Boat_Ship: average precision scores of the detector trained with *Extended ROIs* before and after bootstrapping

When original ROIs were used average precision after one round of retraining improved from $\mathbf{0.198}$ to $\mathbf{0.411}$. The result of *Extended ROIs* detector for boat_ship class is summarized in Table 3.8. Inspired by this significant improvement we trained classifiers using *Extended ROIs* for 3 other classes: Hand, Person-riding-a-bicycle and Bus. Results for these classes when trained on Train and tested on Validation are summarized in Table 3.8. Considerable improvement can be observed on Validation set when using 100 trees over 50 trees classifier. Our final RF classifiers with Extended ROIs were trained on DEVEL (Train + Validation) set and then are run on the TEST set. Some visual Results shown in Figure 3.15(a) for boat.

Top ranked keyframes for categories: Boat-Ship, Bus and Person-riding-a-bicycle are shown with detected Extended ROIs in Figure 3.15 . For Boat-Ship and Bus top 15 results are displayed, in case of Person-riding-a-bicycle all top 70 ranked results are true positives from the same video and are similar to the first four frames shown for this category in Figure 3.15(c). So, keyframes ranked from 71 to 85 are shown here.

Category	Training Round	AP (50 trees)	AP (100 trees)
Hand	0	0.2074	0.3056
Hand	1	0.2220	0.3736
Person-riding-a-bicycle	0	0.0758	0.1939
Person-riding-a-bicycle	1	0.2824	0.3443
Bus	0	0.0026	0.0141
Bus	1	0.0625	0.1316

Table 3.8 Classification by detection results (*Extended ROIs*): average precision scores of the detector trained on Train set and tested on Validation set.

3.9.2 BBC

Classifiers trained on TRECVID data were run on the video data provided by BBC. This is a collection of 428 videos of TV programmes. The total duration of the videos is 220 hours and in total there are 137921 keyframes. The top retrieved results from BBC data are shown in Figure 3.16 for Hand and Boat_ship categories. The results are very good considering that the source of training data was different and the generalization is excellent.

Figure 3.16 Top 15 results from the BBC video dataset for Boat or Ship and Hand categories.

3.10 Summary

In this chapter, we have shown that Random Forest classifier can be used for fast and accurate classification and object localization. Its computational efficiency in both training and classification makes it a promising choice. We have used a combination of different visual features with random forest for the high-level feature extraction task of TRECVID'08. Random forest is presented as a rapid object detector with results on challenging datasets like VOC PASCAL and TRECVID09. We achieved results comparable to the best in VOC'07. In TRECVID'09, we used sliding window based RF detector for four object categories (Boat-Ship, Person Riding a Bicycle, Bus and Hand). Efficiency with accuracy of random forest was a key factor in running the detector over such a large dataset of about 200 thousand key-frames.

Chapter 4

Online Video Spotting and Processing

4.1 Introduction

In last few years, due to cheap storage, bandwidth and imaging hardware, huge amount of multimedia data is being generated and stored. The world is covered with millions of cameras with each recording a huge amount of video. With this ubiquitous video content there is a need of processing online video sequences for information extraction and data mining. In online processing it is very important requirement to be able to retrieve video clips as and when they arrive. So, sifting through millions of videos to find visual content of interest needs to be automated.

The aim of this chapter is to address online content based processing of continuous stream of videos to detect video segments of interest. Our approach is example-based where visual content to be detected or filtered is charaterized by a set of examples availabe *apriori*. Example-based content-level processing of multimedia, has been popular in video and image retrieval literature [101, 137, 147]. The focus has been on identifying appropriate descriptors [147] and developing scalable systems which enable efficient retrieval from millions of images or video key-frames [101, 137]. There has also been significant interest in characterizing and recognizing activities and semantic concepts from video examples [70]. This class of algorithms, first learn to characterize the events from training data by computing a classifier, and then apply the learned concepts in new situations.

However, many concepts of practical interest are not easy to represent and learn. For example, the concept of violence is a difficult concept to characterize, even for the state-of-the-art machine learning algorithms. One may also come across categories like commercials in video streams which have high within class variance and relatively small inter-class variance. On the other hand, many of these concepts can be described with the help of examples. This allows us to model the problem as *simultaneous spotting* in a video stream. This approach could meet the immediate requirement of processing or filtering the video stream based on the visual content.

In many practical situations, a human is present within the loop of a video processing system. For example, a human operator is often associated with surveillance video processing for initiating actions based on the video content. In such cases, *on-line spotting* of relevant information from a video sequence

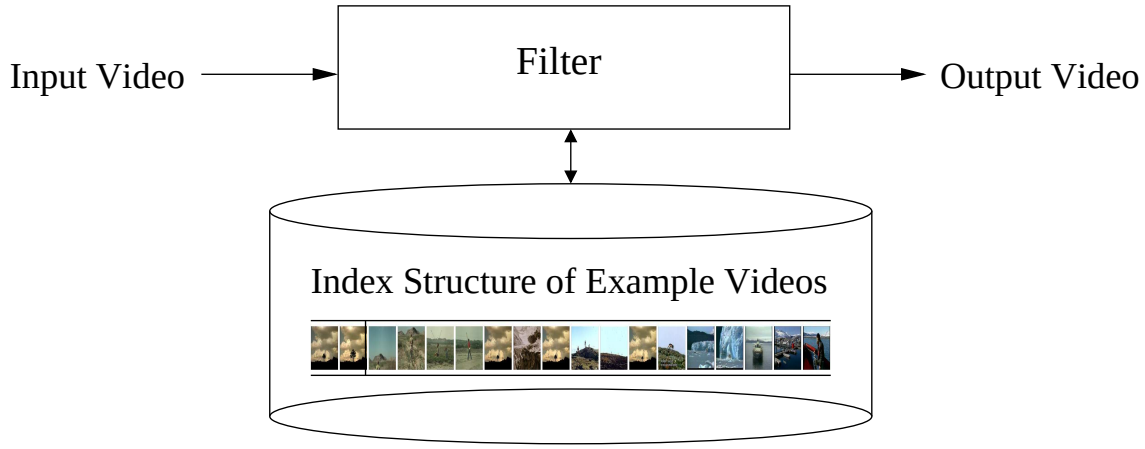


Figure 4.1 Overview of the Example-based Video Processing

can be of immense help. We demonstrate that this is feasible even when a robust recognition of the specific concept is probably impossible.

We approach the problem of video processing in a manner complimentary to that of video retrieval. We begin with a set of examples (traditional "queries") which are indexed in the database. The larger video collection, which needs to be processed, is unseen during the offline indexing phase. The video collection is processed on-line, to identify the concepts represented by the given set of examples. In a way, what we are interested is in spotting rather than retrieving. Traditional retrieval systems focus on scalability to large databases for efficiency in retrieval. Our focus is on enhancing the throughput of the system and making the algorithm capable of simultaneous spotting of multiple examples. Our formulation also effectively utilizes the sequence information of the video stream, rather than treating it as a set of frames.

In this chapter we have presented a survey of existing approach towards video processing in general and our example based method for online processing of videos. The basic idea of example based video processing is presented in Figure 4.1. We present results for commercial removal and content based copy detection (CBCD) as applications.

4.2 Video Processing Approaches

Content-based processing of videos has been proposed by different communities for various research problems. These include:

- Video retrieval
- Filtering
  - content based copy detection

  - advertisement detection
  - action recognition
  - other specific filters
  - searches and mining in videos
- Video summarization and segmentation
- Adding Semantics

4.2.1 Content based video retrieval

Most of the content based image and video retrieval systems identify similar objects to a given query [35]. Both query and database objects are represented with the help of a set of feature descriptors. Earlier approaches used color, texture and shape descriptors computed globally or locally to describe the visual content of the images. This has been successful in retrieving images with concepts which are rather weak, (for example, "images with red flowers" or "scene of a sun-set next to water"). With this initial success, the focus shifted to retrieving specific objects (under widely varying imaging conditions) or object categories. Invariant description of interest points and patches have been the key to the success in these situations.

Image and video retrieval has been successfully attempted for retrieving objects of interest invariant to scale, orientation and illumination [101, 137, 171] in diverse multimedia collections. These methods primarily addressed the scalability issue towards indexing in large databases. The videos are represented by their key-frames, which in turn are described as a bag-of-interest-regions. Features describing regions-of-interest are quantized using K-means or hierarchical K-means, in an offline phase to build a visual-vocabulary for the given data set. The video collection is then indexed against this visual vocabulary. Once indexed, the database can retrieve videos corresponding to "short" queries, such as a (part of) an image or key-frame selected by the user. Another set of works focuses on building efficient indexing schemes for multimedia collections. Successful examples include LSH [63], min-hash [27, 28], pyramid match hashing [54], vocabulary forest [171], etc. Vocabulary tree has been used for efficiently indexing and retrieving large number of images [101]. A hierarchical partitioning of the feature space makes the quantization efficient. Also the retrieval and ranking of documents are simultaneously achieved by traversing the tree.

Focus of most of these approaches has been on indexing large amount of multimedia data to efficiently search within the given collection. However, on-line structures for indexing video streams has received very little attention. One of the related problem which received some interest in recent past is that of adapting the index structure with changes in visual content. In this direction, Yeh *et al.* [171] extended the notion of vocabulary tree to vocabulary forest while making the indexing process applicable to dynamic environments.

4.2.2 Content based video filtering

Content-based filtering of images and videos are attempted in literature for applications like adult content detection [51, 181], removal of commercials [30, 143], event detection [83], copy detection [72] etc. Most of these methods formulate this problem as an object/scene recognition or detection by using an appropriate classifier in the right feature space. For example, the filters aimed at removal of adult content or detection of fire formulate the problem in an appropriate color space [83, 180]. In general, example video frames are used in an offline situation to learn the right model or a classifier. Then the new unseen video frames are classified using the learnt model/classifier. Accept and reject filters used for commercial removal also employ similar techniques. Colombo *et al.* [30] attempt to characterize the commercials with the help of low-level features and classify the video segments *into categories*. With the category of commercials becoming more and more diverse, such classification models in simple feature spaces are found to be insufficient.

Content-based copy detection (CBCD) techniques have received significant attention in recent years [64, 71, 170]. Yan *et al.* [168] performed content based copy detection over streaming videos. Focus of research has been on defining the right set of descriptors which are invariant to the allowable set of transformations [72]. There has also been significant concern about the computational complexity of this class of algorithms because of the practical applications in video sharing systems. In [27, 28] the similar problem of near dublicate detection is addressed. Mining the video content can help in getting important information regarding the internal structure of large video databases [39, 113]. Video mining has been used for automatic video annotation [95], to extract principal objects, characters and scenes in a video by determining their frequency of re-occurrence [138].

Action recognition has been an active research topic and many methods have been proposed. Recent methods for action categorization have used local spatio-temporal features to characterize the video and perform classification over the set of local features [70, 89, 100, 175].

4.2.3 Video summarization and segmentation

Video summarization is the process of creating a presentation of visual information about the structure of video, which should be much shorter than the original video. This abstraction process is similar to extraction of keywords or summaries in text document processing. That is, we need to extract a subset of video data from the original video such as keyframes or highlights as entries for shots, scenes, or stories. The result forms the basis not only for video content representation but also for content-based video browsing. Video summarization techniques have been proposed for years to offer people comprehensive understanding of the whole story in the video [9, 37, 107]. Text have been also used in form of caption and transcript to judge the boundaries of scenes or stories [13, 144]. Most previous works on video summarization target on a single video document.The results are usually redundant due to the lack of inter-video analysis. In [166], an approach is proposed for multi-document video summarization by exploring the redundancy between different videos.

Video segmentation or shot boundary detection esentially invloves examining the information contained in individual video frames and comparing this with other nearby frames to determine if a shot change has taken place. A number of methods have been proposed for frame comparison and for handling gradual transitions to solve this problem [152, 153, 178].

4.2.4 Adding Semantics

It is difficult to map low-level feature (color, texture, shape, motion) description into semantic concepts (such as person-riding-bicycle, cityscape or car-racing scenes). Because of this semantic gap it is difficult to process high-level queries such as "black mercedes". There has been a plethora of interesting research work presented recently that focuses on problem of bridging this semantic gap [57, 62, 141]. Two possible solutions have been proposed to minimize the semantic gap are automatic metadata generation and relevance feedback [59, 182]. Content based semantics can be added by annotation of symantic entities in video. This can give symbolic description of the video in terms of objects or scenes it contains. Low level content based semantics like color, shape, structure and object motion can also be used. Another way is to use structure in the video. It is widely accepted that video documents are hierarchically structured into clips, scenes, shots and frames. Such structure usually reflects the creation process of the videos.

4.3 Vocabulary Trie

For online processing of videos, we would like to retrieve concepts from streaming videos, based on the similarity of a video sub-sequence with one of the given examples. This similarity has to be efficiently computed for each given example, for each incoming frame. This resembles to the concept of *keyword-spotting* popular in speech processing and document image retrieval [119]. Keyword spotting methods locate the possible occurrence of the query word by matching with every possible words in the database. In the case of document retrieval, words are often segmented first and indexed using a set of appropriate features. However such methods are not directly applicable for video data, due to the difficulty of characterizing the visual content corresponding to each concept.

Popular video retrieval systems aim at indexing large quantities of images and videos, and serving a small set of queries while being deployed on the field. Focus has been on the efficiency in retrieval and scalability to large video databases. These formulations typically employs trees [101], hashes [54] or inverted indices [137] for the indexing of the visual data. Our objective is to process large amount of videos with the help of an index structure which is built out of *a relatively small set of example videos.* The indexing scheme that we require should be capable of

1. indexing relatively small number of examples available *apriori*

2. processing of large amount of *unseen* videos

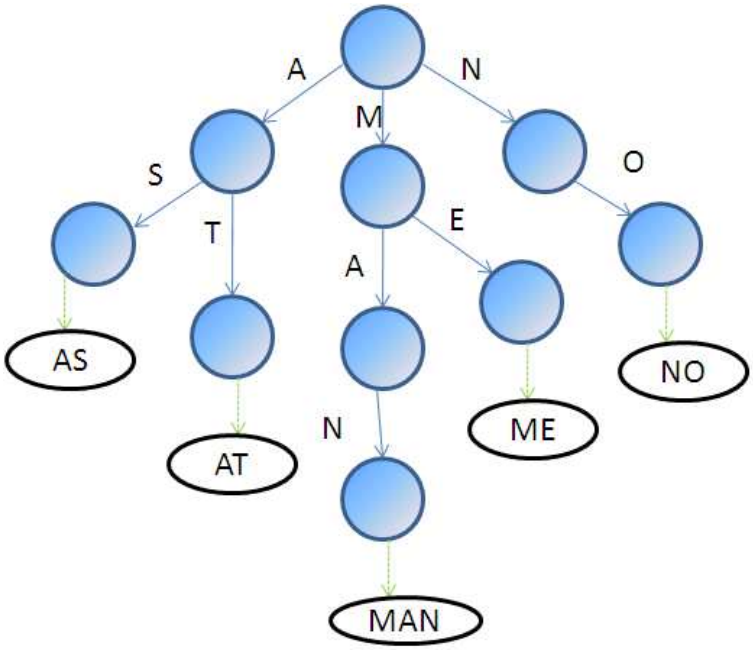


Figure 4.2 Example Trie for set of words

3. avoiding explicit segmentation of video stream for matching with example videos and

4. employing any generic comparison scheme for comparing frames/sequences.

We achieve these objectives with the help of a *Trie* data structure. Tries are ordered tree data structures popular for a number of tasks related to information retrieval [67]. They are useful for matching, based on some similarity measure, for sequences of symbols in a language. A path from the root to a leaf represents a symbol sequence inserted into a trie, during the indexing. The leaf nodes store the identifiers of symbol sequences. An example of trie is shown in Figure 4.2. Tries get constructed from a sequence of alphabets. When trie is used for detection in an on-line setting, the stream of data gets matched/aligned with the sequence of nodes, and any successful termination at the leaf is treated as a valid detection. Trie has been extensively used as an index structure in the area of string matching [131]. It is a suffix tree representation which can be used to find the strings that are exactly or approximately matched to a given query string. Tries offer text searches (exact or approximate) with costs which are independent of the size of the document being searched. Importantly, tries are not sensitive to the curse of dimensionality problems which is a challenge in multimedia computing.

In the following sub-section we present our trie-based architecture, *Vocabulary Trie*, for content-based processing of video streams. Our trie based solution allows simultaneous matching of multiple examples.

4.3.1 Formulation

A video can be represented as a sequence of symbols and indexed into a Trie structure. This is made possible by the quantization of the visual data to produce a finite set of alphabets from a given video sequence [101, 137]. A set of videos to be indexed results in an appropriate vocabulary (words) and define the problem space. Traditional quantization schemes employ K-Means or its variants for the quantization and vocabulary construction. Each frame can be represented as a symbol/alphabet where a symbol can be a scalar or vector or even a set representation based on visual words.

In our case, number and diversity of examples could be significantly smaller than the total amount of video that trie needs to process. In such cases, adding negative examples into the quantization step allows one to control the detection (false positive and false negative) rates. When the examples are diverse enough, influence of the negative examples seems to be negligible. Since the trie is represented in terms of index of clusters, representation is independent of the dimensionality of the feature space, as is the case in any bag of words representation.

There are two basic problems in formulating the on-line video processing problem using Trie: (i) representation of video sequences with the help of discrete symbols (ii) computing similarities of two video frames.

4.3.1.1 Representation and vocabulary trie construction

It is intuitive to use a temporal representation for videos, unlike the popular representation as a set of key-frames [137], which is not suitable for on-line processing of videos. Let us consider a simple representation. A video frame is represented as the average color of the frame and video clip is represented as a sequence of such color descriptors. Such a frame-level representation could be sensitive to the temporal sampling/segmentation process. One could also represent the averaged color over a set of consecutive frames (overlapping or non-overlapping) as another measure for the description. For many practical applications, a simple representation based on color could be quite insufficient. One could also think of representing the video frame(s) with the help of a set of interest points and their representations such as SIFT for matching and detection.

We represent the video at frame level using the features suitable for the given task. The feature space is quantized into K bins using features extracted from a limited set of training data, using a clustering algorithm. Each feature is then indexed to the closest quantized bin, each frame then represented as a set of these quantization indexes. The sequence of the frame features is used in the trie construction and look-up.

The given set of example videos are indexed in a trie. Vocabulary trie construction from example videos, is pictorially shown in Figure 4.3. During the construction phase, the trie is incrementally built from each example. The common prefix sub-sequences are aligned for those examples which have similar frames to begin with.

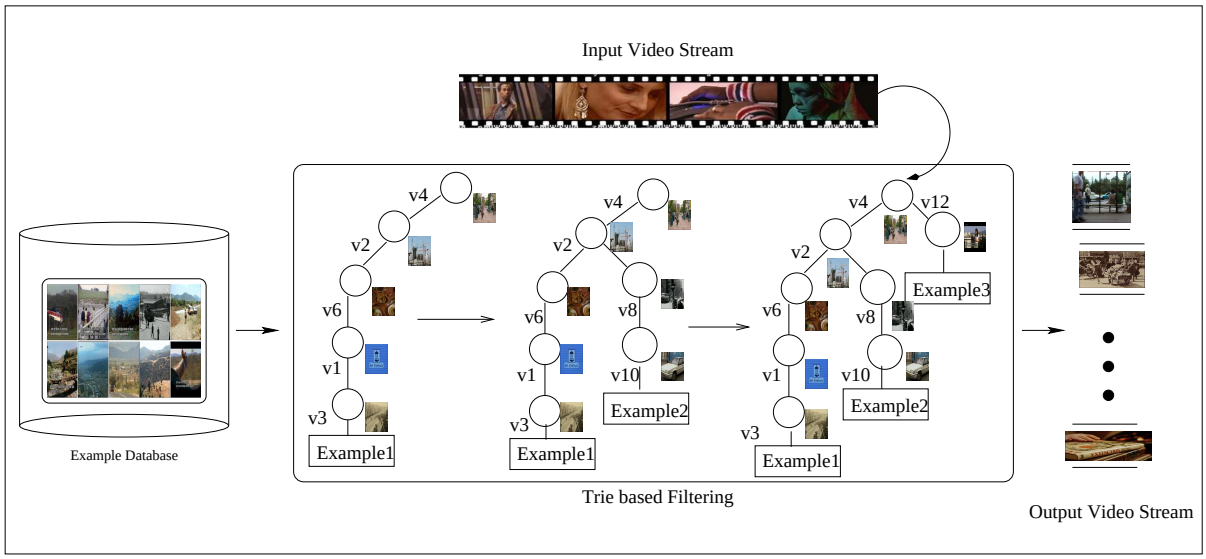


Figure 4.3 Building a Vocabulary trie for video sequences and using it for processing the input video stream.

The trie has a height h and a breadth b. Each frame of an example video occurs at different depths from the node. Hence, the height of the Trie is the length of the longest example video. Each example video constitutes a path from the root to a leaf of the Trie. The leaf is labeled with the concept of the example. Example videos share the nodes corresponding to "similar" frames at the same depth. The total number of leaves in the trie is the number of given examples, N. In the worst case, each example will constitute a distinct path from the root to the leaf. In this case, the storage complexity would be $O(h.N)$ and the time for building the trie would be $O(N^2)$ requiring only the first frame to be compared with the previously built trie. The ideal case is a balanced trie, with equal breadth b at all depths. The storage complexity in the ideal case would be $O(h.b)$ ($b << N$), while the time complexity would be $O(h.b.N)$, since each frame is matched with b nodes at each depth.

Each edge of the vocabulary trie is a symbol. An input sequence of words takes the path along the edge, symbol corresponding to which is most similar to it and the similarity is above a certain threshold. During detection, each frame is checked for a possible match with any of the nodes at $d = 1$. Whenever there is a match, the subsequent frames are matched down the vocabulary trie, and so on. If the sequence of frames from the on-line video terminates in a leaf node, the appropriate concept is said to have been detected.

Algorithm **1** summarizes the trie construction and detection process. During the off-line phase, examples are inserted into the database. During the on-line phase, the trie allows fast processing of the given video sequences.

Trie-Construction: In the offline phase, trie is constructed from example video sequences for the given examples: $\mathcal{V}_1, \mathcal{V}_2 \ldots$

- Initialize an empty trie. For the given examples $i = 1, 2, ...$
- Find the longest prefix sequence which is common to the trie and the i^{th} example video. When a mismatch takes place in the sequence, initiate a new path in the trie resulting in termination of the leaf node labeled with this example.

Online-Detection: In the on line phase, video stream is processed for the possible presence of the examples.

- For the given sequence of words, pass through the trie until either we get a leaf node or no path is available.
- If we reach the leaf node, return back success with the detail of the example and the location from where the possible sequence started.

Algorithm 1: Vocabulary Trie

Such a trie can introduce a latency equal to the maximum length among the example videos, in the worst case. Matching in trie is efficient, since only a few set of nodes will be evaluated for most frames. Such a sequential matching, in general, favor's lesser false positives. However, the matching threshold can be varied to control the detection rates, depending on the application.

4.3.1.2 Matching of videos

Exact matching of two words or bag of words for detecting identical content could be relatively straightforward with any reasonably invariant representation. In many practical situations, one is interested in matching which allows partial and inexact matches of two representation of words. When the alphabets are described by a set of interest point descriptors, one could define a matching score based on the cardinality of intersection of the representations in the video stream and in the trie. Such a similarity score was used earlier in [27].

The score/matching performance of a video sequence depends on (i) the length of the sub-sequence which it matches, normalized with respect to the length of possible paths in the trie which has this as a sub-path (ii) the quality/score of match of each of the alphabets/symbols. (iii) Number of tries which generates warnings/detections. Decision to traverse further at any node in the trie will have to depend on the scores of symbol matching done from the root to the current node. Therefore, we keep a threshold on the mean of these scores to make the sequence matching robust to any rare symbol matching failure. We also keep a threshold, F, on the number of frames matched. If number of frames matched is greater than F and atleast half of the length of possible paths in the trie which has same sub-path, then the video is blocked.

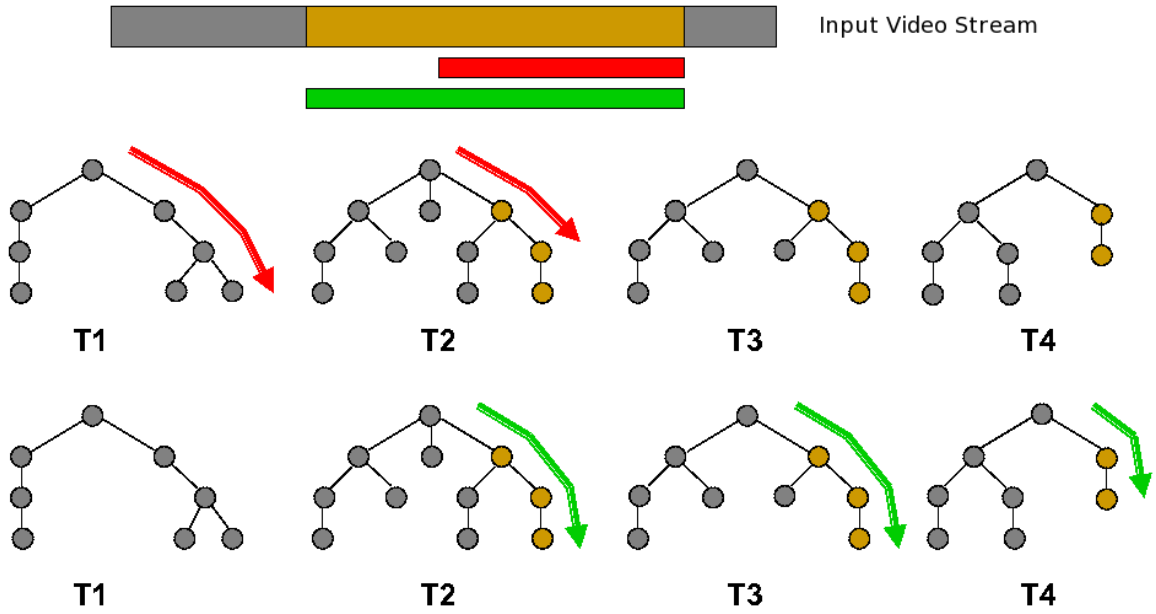


Figure 4.4 Processing a query with forest of tries: The top row shows that a mismatch occurs when we start searching from trie $T1$. The bottom row shows that a copy of sub-sequence of an example can be detected by starting from the next trie.

4.3.2 Forest of Tries

In a generic video filtering situation, there are other practical challenges. Such as when sequence in the query is similar to some sub-sequence of an example, a naive implementation of the video filter could fail.

To deal with this problem we build a forest of N tries numbered from 1 to N, each of maximum depth D. For building the forest of tries any example from the database is first inserted in the trie 1, after inserting D frames we move to next trie and so on. Finally we get $N = \lceil L/D \rceil$ tries, where L is the length of longest example in the database.

While processing the query video stream we initially start from trie 1. If any mismatch happens after starting from the i^{th} trie, then we again start from the $(i+1)^{th}$ trie and continue until a sequence from the query is accepted or we reach the last trie. We move to the first trie when a mismatch occurs in the last trie or a sequence is accepted. By this we ensure that we do not miss any sub-sequence of $length \geq F + D$ in the query (assuming that when correct frames are compared they do match). This is because we can miss a maximum of D initial frames of any example when a forest of depth D is used. D (can vary from 1 to L) acts as a trade-off parameter between performance and time which can be observed in our next experiment.

Figure 4.5 Example frames from the Commercial Videos used

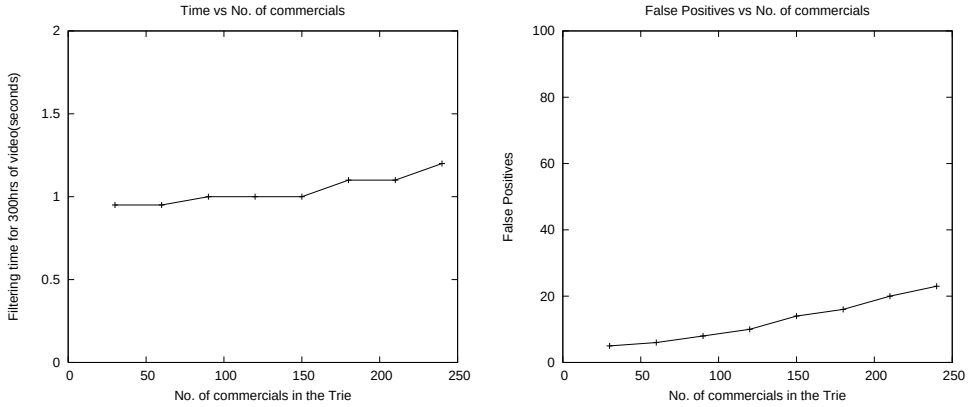
An example of how forest of tries work is shown in Figure 4.4, the brown part of the input video stream is a copy of sub-sequence of an example video in the database (brown colored nodes in the forest). When processing starts from trie $T1$ it leads to mismatch as shown by red path. The actual copy of sub-sequence of an example is found when we search by starting from the trie next to $T1$, i.e., $T2$, as shown by green path. Detecting copy of such sub-sequences of examples is not possible with a single trie.

4.4 Applications

We now demonstrate the application of vocabulary trie on a spectrum of situations. We start by demonstrating the applicability of this method to the detection and removal of a set of *apriori* known commercials from a broadcast video stream. The task is to detect the possible presence of a sequence of video frames which are identical or highly similar to those available in the database. In the second application, we address the problem of detecting copies of videos where a larger set of transformations are possible [72]. Our method allows the detection of copies of multiple videos in a single pass (processing cycle). We then demonstrate the applicability of vocabulary trie in situations where relatively complex concepts of human activity, is spotted in images and videos.

4.4.1 Commercial Removal

Removal of commercials (or a set of example videos) help in segmenting, summarizing, storing and processing of broadcast videos [82, 143]. They are also an integral part of information retrieval systems designed for broadcast videos. Identification of the examples could be done either manually or with the help of audio-visual clues. Given a set of commercials, we index them into a vocabulary trie in the offline phase and use it for detecting the presence of similar video segments from the “test” videos.



(a) (b)

Figure 4.6 Scalability of Trie for detecting commercials in broadcast TV. (a) Time Vs No. of commercials and (b) False positives Vs No. of commercials. One can observe the scalability of the system to large number of examples

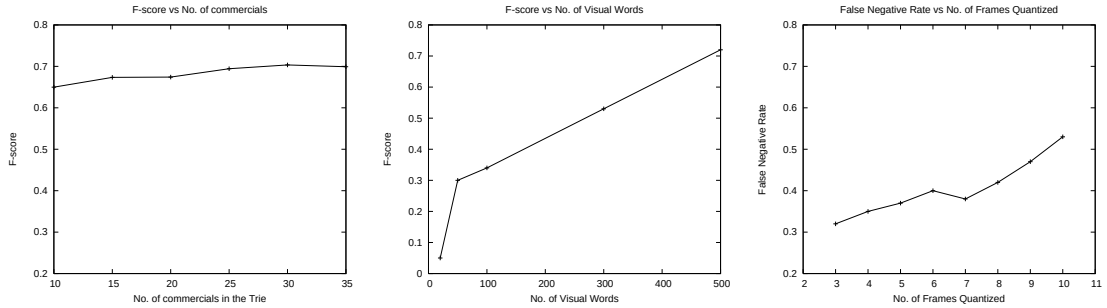
During indexing, we extract color histogram features and build an associated vocabulary by clustering them using K-Means, to 500 clusters. The visual words (or the cluster indices) are then used to construct the trie.

The trie is tested over a video sequence of 300 hours duration (or approximately 300 GB in MPEG) captured from 10 different broadcast news channels. We detect the possible presence of a commercial in this video sequence in about a second (excluding the feature extraction time). The false positive rate of detecting the commercials is about 28%. The false positive rate could be reduced further by using more complex and discriminative features (see the next sub-section). Our method scales to large number of commercials without any significant loss in computational efficiency or the precision as demonstrated in Figure 4.6. The exact time requirement depends on the percentage of commercials in the video sequence. In our case, commercials occupied 16% time of the video duration.

To further evaluate the performance of the vocabulary trie on detection of commercials, we manually ground truth-ed a database of 20Hrs with 250 commercials. In addition to the label, start and end frames were also annotated. Some example frames from the commercial videos used can be seen in the Figure 4.5. The detection performance of the commercials depends on various parameters. We use F-score as evaluation measure, which takes both the precision and the recall into account. It is defined as

$$F = \frac{2 * (precision * recall)}{(precision + recall)}$$

In Figure 4.7(a), we demonstrate the effect of length of commercial on the detection rate. In general, it is observed that longer the duration of the commercial, better the detection rate. For this experiment, we have used the number of clusters (visual words) to be 500. Number of visual words used for rep-



(a) (b) (c)

Figure 4.7 Effect of (a) duration of commercials, (b) number of visual words on F-score and (c) Temporal quantization parameter p on false-negative rate

resentation of the video sequence also affect the detection rates. In Figure 4.7 (b), we demonstrate the effect of number of clusters on the detection rate. With increase in number of clusters, the detection rate also increases.

Many practical situations for video filters require controlling of the false positive/false negative rates depending on the application. As mentioned in the previous section Vocabulary Trie allows flexibility in design, and thereby parameters which can directly affect these rates. We vary the length of the example and query videos by grouping p consecutive frames together and obtain the word corresponding to the mean of their feature descriptors. We demonstrate the variation of false-negative rate with p in Figure 4.7(c). We can observe that false-negative rate increases with temporal quantization parameter p.

Thus, it can be seen that the vocabulary trie allows efficient and scalable spotting of commercials in a video stream with significant amount of flexibility on false positives/false negatives.

4.4.2 Content based copy detection (CBCD)

Content-based copy detection has received significant attention in recent years due to its immediate practical applications [65, 72]. On-line CBCD [168] is becoming an important problem, to filter duplicates in multimedia collections. The vocabulary trie approach is directly applicable to the problem of on line CBCD.

Popular methods for CBCD extract a small number of pertinent features (called signatures or fingerprints) from images or a video stream and then match them with the database according to a dedicated voting function [72]. An important requirement which has come to existence in this problem is the capability to detect (or match) possible copies of multiple video clips with minimal computational overhead. There are two important steps in solving this problem: (i) efficient methods for similarity computation (ii) detection of copies by accumulating the similarity scores. State-of-the-art methods focus on solving the first part efficiently. Our method is also capable of addressing the scalability in number of videos to be matched as demonstrated in the last section.

In the CBCD setting, one needs to allow larger amount of variability for defining duplicates. A copy could be a video clip which is modified in appearance (eg. color, contrast), geometry (eg.re-size, cropping) or re-capturing (eg. perspective effects, overlaid text) etc. [73]. To accommodate these variabilities, we use SIFT [84] and SURF [14] feature descriptors computed over interest points to describe the frames. The visual vocabulary is built using hierarchical K-Means algorithm. In most situations, vocabulary is constructed by quantizing the feature descriptors obtained from example videos. In our case, Trie is supposed to function on similar examples as well as large number of non-example situations. Thus we tried introducing feature descriptors from non-example videos while quantizing. While clustering we weigh the distance from non-example videos by α, a measure of importance. We build the trie with a symbol/alphabet represented frames, which converts a video into a sequence of sets (bags) of visual words. Given two elements of the sequence, A and B, we define the similarity as:

$$Sim(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

In our first experiment on CBCD, we compare the performance of different features and trie parameters for a set of 1000 video clips. Original video clips were obtained from broadcast news channels. Video clips were manipulated by blurring, adding noise, cropping, resizing, gamma correction etc. We use average precision for performance evaluation as used in most of the CBCD tasks [64].

We compare the performance of the above two features and Trie parameters, and present the results in Table 4.1. The input video stream is formed by 100 transformed example videos and videos not present in the database which constitute a total of 46K frames when sub-sampled at the rate of $2fps$. Times reported do not include the time taken for feature extraction. It can be observed that the performance in general improves with Vocabulary size, K. Results are also not much affected by increasing the number of examples, N, to build the trie though the time of processing increases.

4.4.2.1 Experiment on MUSCLE-VCD-2007 database

For our second experiment we use MUSCLE-VCD-2007 database [73]. This database is composed of about one hundred hours of videos spread over 101 different files and it's ST1 query set is composed

	Vocabulary Size (N=210)			Number of Examples (K=10^4)		
Feature	K	Average Precision	Time (secs)	N	Average Precision	Time (secs)
SIFT	9^4	0.7273	59	100	0.7907	28
	10^4	0.7778	62	150	0.7799	44
	11^4	0.8007	64	210	0.7778	62
SURF	9^4	0.7236	40	100	0.7633	20
	10^4	0.7656	42	150	0.7647	30
	11^4	0.7509	42	210	0.7656	42

Table 4.1 Performance of Trie for copy detection

Figure 4.8 Examples of original and transformed video frames of Muscle data-set

of 15 videos of total length of about two and a half hours. Out of these 15 videos, 10 are transformed from some video in the database and rest five are not from the database. Some examples of original and transformed frames from Muscle data-set are shown in Figure 4.8.

We use the ST1 query set as our database and join 101 videos from MUSCLE database to form a 100 hour input video stream. This is according to our objective of filtering large amount of videos with the help of trie which is built out of a relatively small set of example videos. Feature descriptors computed over interest points of frames from 15 videos of the database (sub-sampled at the rate of $0.5fps$) are quantized into 10K visual words and a Trie or a Forest of tries is built as explained above.

Results of this experiment using Trie and Forest of tries are shown in table 4.2. We can see the improvement in the performance by using Forest of tries. Performance improves by decreasing D in case of Forest of Tries at the expense of time. We can observe in the table that it improves for SURF and remains constant for SIFT. The above experiments show how our approach provides an efficient and accurate solution to the problem of CBCD.

	Mean Score		Trie Forest		
Feature	Average Precision	Time (secs)	D	Average Precision	Time (secs)
SIFT	0.8182	19	50	0.9011	86
			100	0.9011	51
			200	0.9011	26
SURF	0.8012	9	50	0.9011	35
			100	0.8182	21
			200	0.8012	11

Table 4.2 Results of Copy Detection on MUSCLE data-set

4.5 Summary

In this chapter, we have addressed a problem of video stream filtering given a set of example videos. Our method is example-based where visual content to be detected or filtered is characterized by a set of examples available apriori. We approach this problem in a manner complimentary to that of video retrieval. The given set of examples (traditional queries) which indexed in the database. The larger video collection, which needs to be processed, is unseen during the off-line indexing phase. We have proposed a trie-based architecture, *Vocabulary Trie*, for content-based processing of video streams. This architecture allows simultaneous spotting (or matching) of example videos in a stream of video frames. We demonstrate the applicability of our architecture for commercial removal and content based copy detection (CBCD).

Chapter 5

Video Mining

5.1 Introduction

Large video repositories are becoming omnipresent. Content based analysis of such collections is challenging. Processing these videos is computationally costly, error prone and difficult to scale up. The necessity of content based access has triggered research in visual recognition with newer data-sets, categories, and computationally efficient methods [114, 137, 139]. Many approaches have been proposed for different problems of video analysis including activity recognition, visual search, movie/sitcoms analysis and visual mining. Most of these methods are supervised and requires labeled examples at some or other level. To make it feasible on large collection of videos, unsupervised and weakly supervised approaches are desired as argued in many of the recent works [76, 169].

In this work, our objective is to mine the videos in order to discover or detect important patterns. We discover characteristic patterns in videos based on frequency of occurrence of scenes, actors and sequence of frames, in an unsupervised setting. With our approach, we are able to detect the representative scene and main characters of movies. Going beyond objects and people, we extend our work to mine frequent video sub-sequences. We define *"video stop-words"* and present a method for detecting them in broadcast news videos. *Video stop-words* are analogous to stop-words in text classification and search. We define *video stop words* based on frequency of occurrence of sub-sequences in videos over the period of time and across different news channels in Section 5.3.3. Detecting them can assist in removing redundancy in videos.

Movies are fascinating data sets with significant visual variation and diversity. In movies, certain scenes, main characters or objects appear more frequently than others. Characteristic patterns in movies could convey a lot about the visual content and major theme of the video. We aim to discover these patterns directly from the video. The pattern of interest could be individuals, scenes etc. In Section 5.3.2, we do automatic labeling of characteristic scenes and main actors in movies. Our approach successfully extracts the characteristic patterns (scenes and people) from our movie database. The characteristic scene discovered from the movie database could vary significantly in visual content. Movie characterization through such mining or otherwise can help a great deal in building movie recommendation

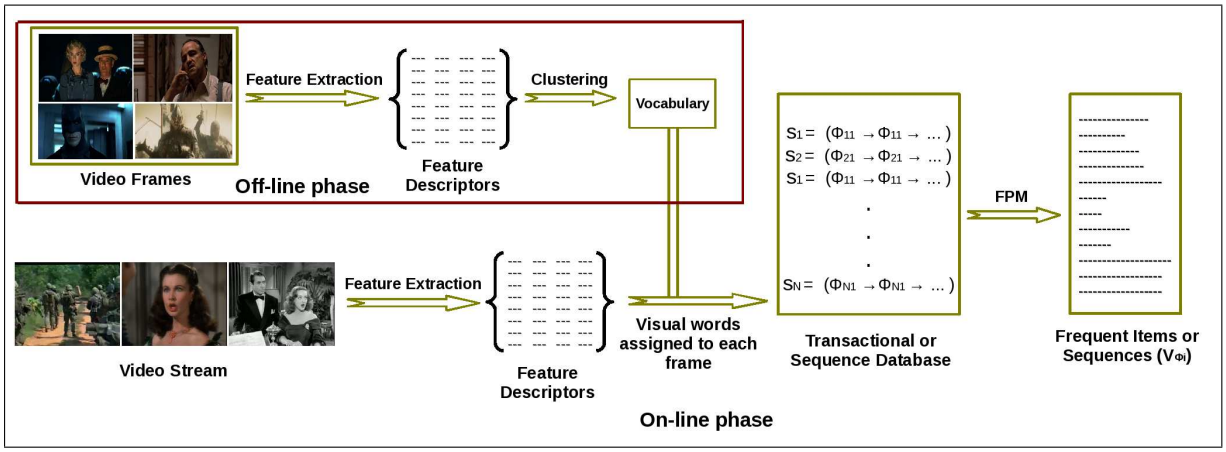


Figure 5.1 Frequent Pattern Mining in Video: Feature descriptors of frames are quantized to build vocabulary in offline phase. During online processing, video is represented as a transactional (or sequence) database, which is mined using Frequent Pattern Mining algorithms.

systems which to date are manual or semi-automatic requiring comprehensive human intervention. Another application is to mine patterns for sociological studies. The techniques are generic and are widely applicable in other category videos.

In broadcast news videos, many events like breaking news and commercials occur repeatedly. Such frequent *items* or *sequence* can be used for automatic characterization and understanding videos. Our goal is to efficiently detect frequently occurring sequence of frames in the news videos. This requires partial or complete matching of frames or sequences of frames of variable lengths from different parts of the videos. It is also desired that the method is robust enough to deal with the situations when sequences are repeated with few extra or fewer frames, but with ordering preserved. The challenge is to detect these sequences very efficiently.

Mining the visual content and thereby characterizing videos, has been attempted in the recent past. Sivic and Zisserman proposed a video mining approach in [139] to obtain principal objects, characters and scenes. Frequently occurring spatial configurations of features were found using clustering algorithms, rather than any frequent itemset mining schemes. Very few works have tried to adapt traditional data mining methods for visual data [114, 154]. The objective has been, often, to find the most frequent spatial configuration of points (eg. a building) from large number of video frames. Mining in visual data has been complemented by the processing of associated text (subtitles) [44] and speech [154] components. Our method relates to visual recognition as well as data mining. In this sense, the closest to our work is that of Quack and Gool in [114]. They use *frequent itemset mining* for finding frequently occurring configurations of features for mining frequently occurring objects and scenes from videos. They also apply frequent itemset mining: (a) on instances of a given object class to assist in object detection [115], and (b) for mining object and events from community photo collection [116]. Nowozin *et*

al. in [102] introduced discriminative subsequence mining to find optimal discriminative subsequence patterns. Rather than focusing on objects or point configurations, our primary interest is in scene characterization, based on a global set of features. We also design the mining scheme to suite large video collection, as required in our case.

We employ two different video mining schemes; that are aimed at detecting frequent and representative patterns. For one of our mining approaches, we use an efficient frequent pattern mining algorithm over a quantized feature space, as in the case of visual bag of words methods. In our second approach we suggest a sequence representation of videos based on Random Forest [21] and propose to mine frequent sequences. This mining approach is also based on clusters by randomized trees.

The remainder of this chapter is organized as follows. We explain our two mining approaches in the next section. Then we evaluate and compare these approaches quantitatively in Section 5.3.1. In Section 5.3.2, we present our results on movies and show results for discovering characteristic scenes and main characters in movies. In Section 5.3.3, we define *video stop-words* and present the method to detect them. We demonstrate the accuracy and efficiency of the proposed approach by experimenting on a broadcast news video data.

5.2 Our Mining Approaches

For mining videos, we represent features in quantized code-books. This has been popular for many recognition, retrieval and classification tasks [19, 111, 137]. Representing video frames using code-books helps in accommodating the uncertainty of the visual description while retaining the essential discriminative information. We employ *Frequent Pattern Mining* (FPM) [5, 7] to extract frequent sequence or items from videos.

In FPM, a set of patterns (transactional database) and minimum support threshold are given. Patterns are some or other form of collection of $items$ such as itemsets [5], item sequences, sequences of itemsets [7]. The task is to find all the frequent patterns whose frequency of occurrence is no less than the minimum support threshold. *Frequent Itemset Mining* and *Frequent sequence mining* are special cases of *Frequent Pattern Mining*. In FIM transactions are set of items and in FSM they are sequence of items or itemsets. We say that a transaction supports an itemset (in case of FIM) or sequence (FSM), if itemset is sub-set or sequence is sub-sequence of the transaction. Transactional database is more popularly known as sequence database in case of FSM. Frequent sequence mining [7] has been successfully applied to several large-scale data mining problems such as market basket analysis or query log analysis [5]. Many algorithms have been proposed in the literature for solving FIM as well as FSM such as APriori [5], PrefixSpan [109], SPADE [176] etc.

In our first approach we use an FPM methods over video frames represented based on vocabulary built by K-means clustering. We then propose sequence representation for frames/images using nodes of randomized trees. We consider this representation using Random Forest for following reasons:

- Ensemble of clustering trees are able to find natural clusters in high dimensional spaces [94].

- Random Forest leads to more efficient clustering and less memory usage than k-means based algorithms.
- The existence of an implicit hierarchy in the trees can take care of partial matching of samples.

For large number of trees this sequence becomes very long and can not be mined efficiently using *PrefixSpan* algorithm. Therefore we propose *Randomized Mining Forest* in Section 5.2.2 to mine frequent patterns from such sequences. We discuss them in detail in the rest of this section.

5.2.1 Visual Frequent Pattern Mining

Our approach of mining videos is illustrated in in Figure 5.1 as the online and offline phases. In offline phase, features are extracted from example frames and quantized by k-means to build vocabulary. During online phase, input data is assigned the visual words. Each frame or shot represented by visual words makes a transaction (or sequence) and thus transactional (or sequence) database is built. Frequent itemsets or frequent sequences are then mined from it using FIM or FSM algorithms.

We now state the problem of mining frequent sequences in video when frames are represented as items or set of items and shot as a sequence. In other similar cases, for example when frame is itself a sequence or transaction this can be modified accordingly. Let $\mathcal{V} = \{w_1, w_2, ..., w_k\}$ be the visual vocabulary of k visual words. A frame, ϕ , is represented as a visual word or unordered set of visual words, $\phi = (w_1, w_2, w_3, ..., w_m)$ and $\phi \subseteq \mathcal{V}$. A sequence is an ordered list of such frames.

A sequence of frames, $\Phi_\alpha = (\phi_{\alpha 1} \rightarrow \phi_{\alpha 2} \rightarrow ... \rightarrow \phi_{\alpha p})$, is said to be sub-sequence of another sequence $\Phi_\beta = (\phi_{\beta 1} \rightarrow \phi_{\beta 2} \rightarrow ... \rightarrow \phi_{\beta q})$, $\Phi_\alpha \preceq \Phi_\beta$, if there exist integers $1 \leq j_1 < j_2 < ... < j_p \leq q$ such that $\phi_{\alpha 1} \subseteq \phi_{\beta j_1}$, $\phi_{\alpha 2} \subseteq \phi_{\beta j_2}$ ..., $\phi_{\alpha p} \subseteq \phi_{\beta j_p}$. A video is represented as a database of shots. Any sequence Φ is valid if $\Phi \preceq s_i$, $i = 1...N$, where s_i is a shot and N is the number of shots in the video. The relative support of a sequence, Φ, in a video or shot database, D, is the ratio of number of sequences containing Φ to the number sequences present in the database.

$$support_S(\Phi) = \frac{|\{(s_i \in D)|(\Phi \preceq s_i)\}|}{|D|} \in [0, 1] \tag{5.1}$$

A frame sequence Φ is called frequent in D if $support_S(\Phi) \geq min\_sup$ where min_sup is a threshold for the relative minimal support.

We obtain frequent sequences of frames by using PrefixSpan method. It is more efficient than APriori based methods for mining sequential patterns [163] and particularly for lower min_sup values. In case of videos even if a sequence of frames repeats for only a few times it would be considered frequent. Therefore we use PrefixSpan algorithm for our purpose of mining frequent sequences of frames in videos in Section 5.3.3. Corresponding to each frequent sequence, Φ, we have an ordered set of visual words (frequent itemset), $V_\Phi = \{w_1, w_2, ..., w_n\}$ and a set of tuples $M < sid, F >$, where sid is shot-id, F is ordered set of frames in the shot and M is the absolute support of Φ in the video.

When frame is itself a set of items (or transaction) i.e. no sequential information is used (as in Section 5.3.2.1) then it is a problem of FIM. The above formulation can be modified accordingly by

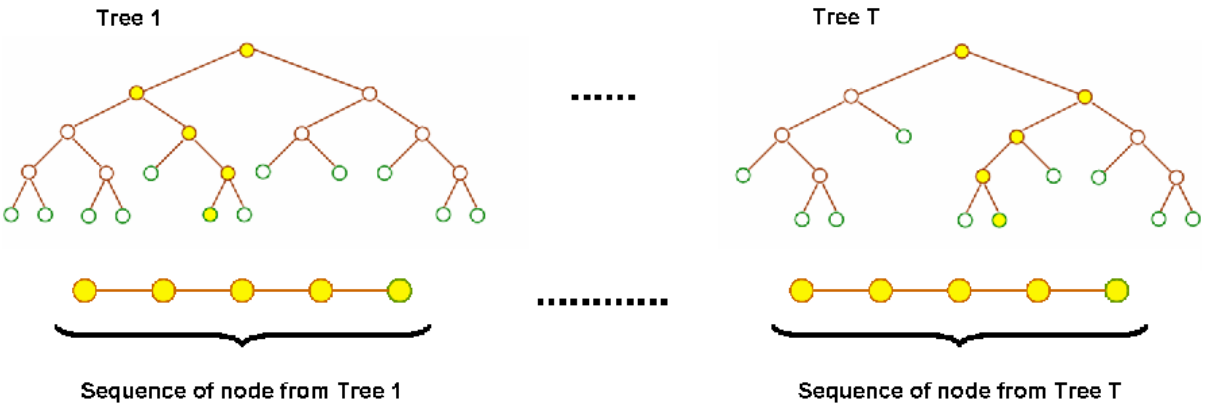


Figure 5.2 Randomized Mining Forest of T trees built without supervision. Each sample while descending updates the counts of the nodes in each tree. The paths traversed by a sample in each tree, shown in yellow, are concatenated and used as a sequence representation of the sample.

representing Φ_α as set of items/itemsets and replacing $\preceq$ by $\subseteq$ in equation 5.1. *APriori* algorithm is used to get frequent itemsets, Φ, and V_Φ and F are orderless.

Now we discuss an alternative and more efficient approach using randomized trees.

5.2.2 Randomized Trees for Mining Videos

Random Forest was introduced in Machine Learning literature by Breiman [21] for classification and regression, and is shown to be comparable with boosting and support vector machines. They have become very popular in the computer vision community. Many papers have applied them to various classification, segmentation and clustering tasks [20, 94, 133]. Moosmann *et al.* [94] proposed an efficient clustering scheme using randomized decision tree. Shotton *et al.* [133] simultaneously exploit both classification and clustering for segmentation and categorization.

We use ensemble of randomized trees for fast clustering and also make use of tree hierarchies in the way similar to [94, 133]. Each tree is built in an unsupervised manner using a randomly selected subset of the training data. At each node a split function that most evenly divides the data is used i.e., each sample is considered to belong to a different class. Entropy at any node N_i with X_i number of sample is given as, $E(N_i) = log(X_i)$.

The tree growing procedure is described as follows:

- At each internal node F node-functions are randomly selected. Here we consider 3 types of node-functions: (a) single feature component, (b) difference of two components and (c) linear combination of few components of the descriptor.
- For each node-function, we need to determine a threshold that best splits (most evenly) the data reached to this node.

- The combination of node-functions and threshold that gives maximum information gain is selected.

When a sample (say a keyframe) is pushed through a tree its path from root to leaf node makes a sequence of nodes. Such sequences of nodes from all the trees are concatenated (Figure 5.2) to represent the frame as an item sequence. Applying FSM on such a sequential database would give us the frequent set of paths across the trees. Similar frames may not reach the same leaf node or may not have long enough common path or prefix sequence in some trees. Using ensemble of T (around 50) trees handles this as similar frames are expected to have enough number of common paths across the trees. With the proposed sequences (using path not just leaf) partial matching can be taken care of when FSM is applied because of tree hierarchies.

Since we use each node as an item, set of all nodes becomes our vocabulary ($\mathcal{V}$). Each frame is represented as a sequence of such nodes, $n \in \mathcal{V}$. Consider a frequent sequence extracted by FSM from this sequential database,

$$seq_{freq(i)} = \{n^i_{11} \rightarrow n^i_{12} \rightarrow ... \rightarrow n^i_{1L_1} \rightarrow n^i_{21} \rightarrow ... \rightarrow n^i_{TL_T}\} \tag{5.2}$$

where n^i_{tj} is j^{th} node in $seq_{freq(i)}$ from t^{th} tree and $n^i_{1L_1}$ is the last node coming from t^{th} tree in $seq_{freq(i)}$. Note that $n^i_{1L_1}$ need not be a leaf node. It is equivalent to represent these frequent sequences by only last nodes:

$$seq_{freq(i)} = \{n^i_{1L_1} \rightarrow n^i_{2L_2} \rightarrow ... \rightarrow n^i_{TL_T}\}$$

So, set of frames supporting frequent sequence $seq_{freq(i)}$ can be given as:

$$F_{Freq}(seq_{freq(i)}) = \{C(n^i_{1L_1}) \cap C(n^i_{2L_2}) \cap ... \cap C(n^i_{TL_T})\} \tag{5.3}$$

where $C(n)$ is the set of frames passing through node n. We extract set of all *maximal frequent sequences*, F_{max}. A frequent sequence is maximal if it is not a subsequence of any other frequent sequence. Support of $seq_{freq(i)}$ according to the definition of FSM would be $suppT(seq_{freq(i)}) = \frac{|F_{Freq}(seq_{freq(i)})|}{|D|}$. Support of all sequences in F_{max} has to be greater than minimum support threshold, $minsupp$. Frames supporting any frequent sequence from F_{max} are frequent or characteristic frames. Set of such frames is given by:

$$\Gamma = \bigcup_{F_{max}} F_{Freq}(seq_{freq(i)}) \tag{5.4}$$

5.2.2.1 Randomized Mining Forest

With T in range of $50 - 100$, the sequences become too long for PrefixSpan algorithm to compute FSMs efficiently. Computational time exponentially increases with number of trees or length of sequence. We suggest Random Forest based solution to find Frequent frames in a given movie. Keyframes (or features) of a video to be mined are passed through the built forest. The paths followed by each frame and number of samples reaching at each internal and leaf node are stored. We call this ensemble of randomized trees with above details of a given video as a *Randomized Mining Forest* (RMF). Figure 5.2

illustrates an example of RMF. The node tests are learned from a sub-set of dataset consisting of several videos. Such a set can be thought to have a number of complex classes (of say scenes). When a frame reaches to a node in RMF it belongs to some hypothetical class with some probability. Therefore, the item-sequence generated by RMF can be seen as a sequence of probabilistic items.

We here suggest a method to mine videos and find frequent frames approximately as given by equation 5.4. In each frequent sequence $seq_{freq(i)}$ we go down the trees after last node ($n^i_{tL_t}$) to some node, $n^i_{tEx_t}$ that has high *depth normalized frequency* (explained below). Thus we get a new sequence $seq^{ext}_{freq(i)} = \{n^i_{1Ex_1} \rightarrow n^i_{2Ex_2} \rightarrow ... \rightarrow n^i_{TEx_T}\}$, where $n^i_{tEx_t} \geq n^i_{tL_t}$. By extending like this or going further down the trees, frames left in the deeper nodes are mutually more similar. Set of these extended sequences are extended maximal frequent sequence. *Depth normalized frequency* of a node n is given as $depF(n) = \frac{|C(n)|}{2^{H-d}}$, where H is height of the tree and d depth of node n. To find n^{ext}, nodes with high $depF$ we start from leaf nodes. In each tree t, M leaf nodes with highest $depF$ are selected at first as a member of set of frequent nodes. Mean *depth normalized frequency* $MdepF$ of the selected nodes is computed. Then we move to the parent of each of the selected nodes and if any of the parent has *depth normalized frequency* greater than $MdepF$ then child is replaced by the parent. This is done iteratively until no parent satisfies the above criteria to get final set of extended frequent nodes $\mathcal{N}$ from all the trees in RMF.

Set of frequent or characteristic frames from the given video can be approximately given as:

$$\Gamma_{Ext} = \bigcup_{n \in \mathcal{N}} C(n) - \{freq(f_j) < \tau | f_j \in \bigcup_{n \in \mathcal{N}} C(n)\} \tag{5.5}$$

where $freq(f_j)$ number of occurrences of frame f_j in $\mathcal{N}$ which should be greater than τ. We define support of a frequent frame $f_j \in \Gamma_{Ext}$ as:

$$suppT(f_j) = \frac{\bigcup_{n \in \mathcal{N}_{f_j}} C(n)}{|\Gamma_{Ext}|} \tag{5.6}$$

where $\mathcal{N}_{f_j}$ are those frequent nodes through which f_j passes.

In summary, we approximate equation 5.4 by equation 5.5. We go further down from a last node ($n^i_{tL_t}$) to some node with high enough $depF$. This node will be traversed by a subset of frames that reached node, $n^i_{tL_t}$. The frames reaching to the extended node are expected to be mutually more similar and also frequent. The idea is that when we take union of set of frames reaching the extended nodes we get a set of nodes approximately similar to that given by equation 5.4. We of course take out those frames that do not occur frequently in set of extended nodes ($\mathcal{N}$) in equation 5.5. We quantitatively evaluate RMF in Section 5.3.1 and apply for mining characteristic scenes from the movies in Section 5.3.2.

5.3 Experiments and Results

In this section, we first quantitatively evaluate the effectiveness of our approaches and compare them on the grounds of accuracy and efficiency. Then we present our experiments on movie and news videos with results for identifying characterisitc scenes and main actors, and *video stop-word* detection.

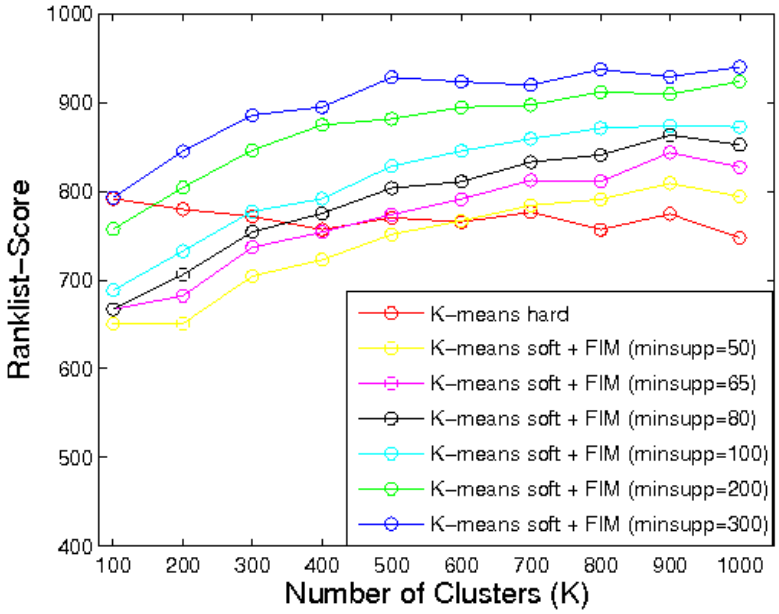


Figure 5.3 Performance of different approaches for ranking.

5.3.1 Quantitative Evaluation of Mining Approaches

The goal of this experiment is to find the frequent object class categories in a given database. Since we do not have the groundtruth for large movie and news datasets, we use VOC 2007 [45] as database for the quantitative evaluation. It has 20 object categories such as person, boat, car etc. The dataset also provides bounding box level ground truth for each object instance. The task is to automatically rank these object instances such that the more frequent categories are higher in the ranked list. For representation of object bounding boxes we use Phog [20] descriptor. Here we analyze and compare our mining approaches experimentally for finding frequent patterns. We also compare them with k-means as a simple baseline.

K-means: In our baseline method we quantize the Phog descriptors by k-means and represent each sample by the cluster ID. Now the samples are ranked based on the size of the cluster they belong to and the ones belonging to larger clusters are ranked higher. Samples belonging to same cluster are ranked based on their distance from the cluster center.

K-means (soft assignment) + FIM: Here each sample is represented by a set of cluster IDs. Set includes the nearest cluster and the clusters having distance not more 1.05 times of the distance from nearest cluster. These sets can be seen as transactions and cluster IDs as items. We apply FIM on such a transactional database to find frequent cluster ID sets. Each sample is assigned to the largest cluster ID set which it supports. When a sample supports more than one sets of same size then it is assigned to that set of clusters which have least mean distance from it.

Randomized Mining Forest (RMF): This is the approach described in Section 5.2.2. Each sample is represented by the sequences of nodes traversed till leaf node in each tree. We build Random forest

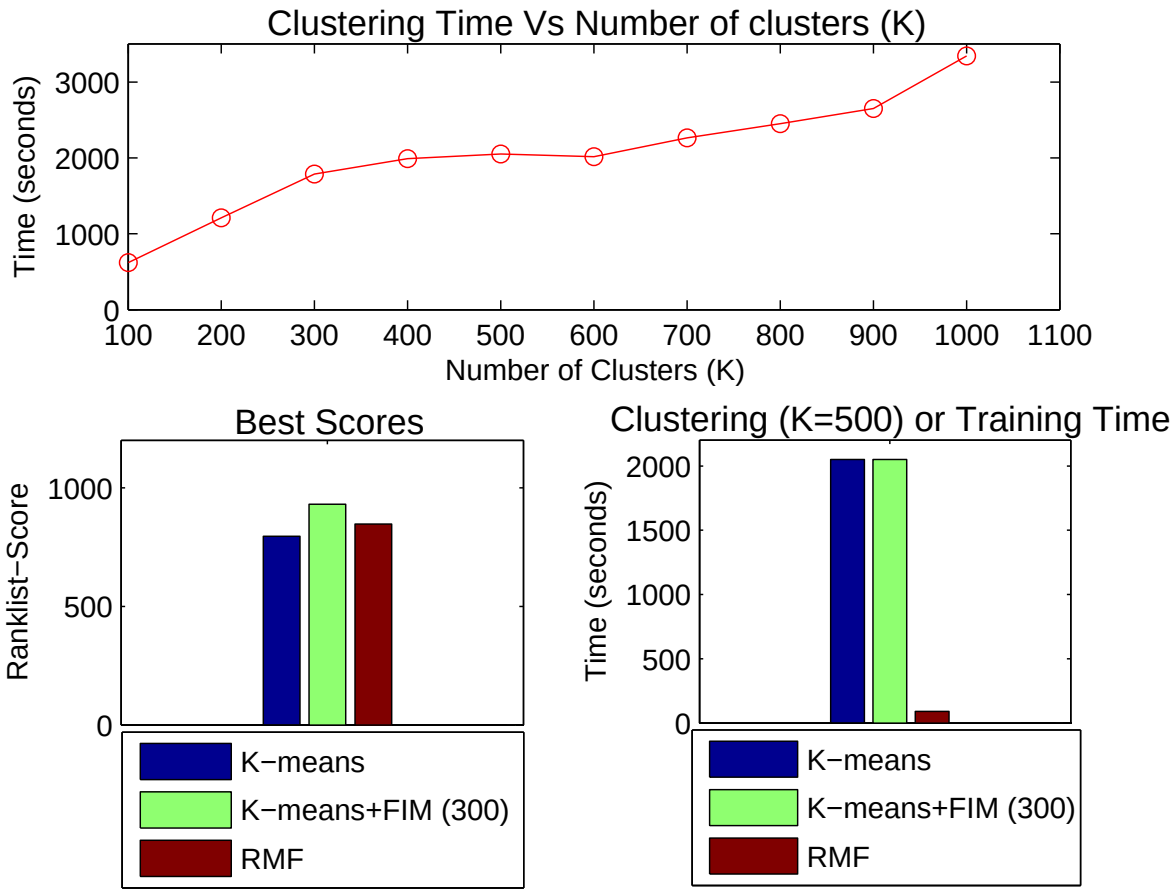


Figure 5.4 Top: Clustering time is too high compared to the time taken to build forests, which takes only about 90 seconds to built 20 trees; Bottom left: Best scores by the baseline and our two methods; and Bottom right: training RMF is about 20 times faster than clustering for k=500, when k-means+FIM reaches its best range of ranking score.

of 20 trees with 50 features, 15 thresholds at each node and maximum depth is set to 20. Therefore, the length of each transaction is about 400. Samples are ranked according to their support given by equation 5.6.

Let class of r^{th} sample in the ranklist (i.e. sample having rank r) is given as $C(r)$. Frequency of an object classes c is given as

$$Freq(c) = \frac{\textit{number of samples of class } c}{N}$$

where N is total number of samples in the dataset. The ranklist-score is computed as,

$$\textit{ranklist-score} = \frac{1}{N}\sum_{r=1}^{N} Freq(C(r)) \times (N - r + 1) \tag{5.7}$$

There are 12839 object instances in VOC 2007 *trainval+test* data (we do not include truncated examples). In addition we randomly take bounding boxes (6642 samples) from background which do not

overlap with any object instance. Frequency of these samples is set to zero ($Freq(c) = 0$). According to equation 5.7 for 19481 samples the expected ranklist-score of a random ranking would be 702.

Figure 5.3 shows the results of applying *K-means* and *K-means+FIM* for ranking these samples. With minimum support greater than 100, *K-means + FIM* method achieves higher *ranklist-score* at almost all values of K. We get better results with larger number of clusters with *K-means+FIM* method. Using *RMF* we get scores ranging in 750 to 847 with different values of M. Figure 5.4 compares the baseline and our two methods for efficieny and performance. Both our methods perform better than the baseline. Best score by *K-means+FIM* is higher than that of *RMF*. But it requires quantization into large number of clusters, which takes significantly more time than that for training RMF. The advantage of *RMF* comes in efficiency as building random forest is much faster (approx. 4.5 seconds per tree). The speed is critical while processing on large datasets as we do in the next section.

5.3.2 Movie Characterization

Figure 5.5 Some examples from the dataset

The dataset for the experiments in these section includes 81 Oscar winning and nominated best movies over the last 60 years ranging from 1950 to 2008. These movies form a good mixture and subset of the huge number of movies available. The genres, directors and other movie details of the dataset were taken from Internet Movie Database (IMDB, [1]). Figure 5.5 shows some example keyframes from the database.

5.3.2.1 Characteristic Scenes of the Movie

In this experiment we apply our method to identify characteristic scenes of the given movie. We extract GIST [104] features from the key-frames to encode the global information. About 50 thousand keyframes are selected at random from the dataset of 81 movies for feature extraction. Using the extracted features the feature space is quantized into 1000 bins by K-means clustering algorithm. In movies, frames belonging to same shot are generally very similar so we do not represent frame as a transaction. Here we represent a shot as a transaction of frames (items) and do frequent itemset mining to find frequent keyframes. Support of a visual word is computed as number of shots it occurs in divided by total numbers of shots in the movie. Certain scenes such as people speaking, road, room etc. are

Figure 5.6 Some examples of characteristic scenes retrieved from movies Braveheart, Lord of The Rings: The Return of The King, Sixth Sense and Chicago (from top to bottom).

very common in the movies. So support computed from a movie is taken as *term-frequency (TF)* and the *inverse document frequency (IDF)* is computed by applying FIM on all the movies together. So the *TF/IDF* support of any visual word (W) in movie m from dataset M is given by

$$support(W) = \frac{support_m(W)}{support_M(W)}$$

Another experiment is done to detect characteristic scenes of the movie but with each frame represented as a sequence of items as explained in Section 5.2.2. The items are the IDs of the nodes traversed by the frame when pushed down the RMF. Each frame is represented as a sequence of items or nodes. We used ensemble of 100 randomized trees, number of features and thresholds tried to create node-test at each node are 100 and 15 respectively.

Figure 5.6 shows some examples of characteristic scenes retrieved from our movie database. First two rows show some examples of results by our first approach using FIM. Corresponding to each visual word we have many frames and shots. The figure shows six keyframes for a movie each representing one of the top 6 words from that particular movie. Last two rows in Figure 5.6 shows some examples of characteristic scenes retrieved by RMF.

Braveheart is an action, drama movie which has many war scenes in it, this can be seen in the retrieved keyframes. Similarly for the *Lord of The Rings: The Return of The King* which is again an action, adventure, fantasy movie. Genre for *Sixth Sense* is a drama, mystery, thriller and *Chicago* is a musical, drama and crime movie. The characteristic scenes retrieved by our approach also suggests the same. We conducted the experiment for all 81 movies and got relevant keyframes as characteristic

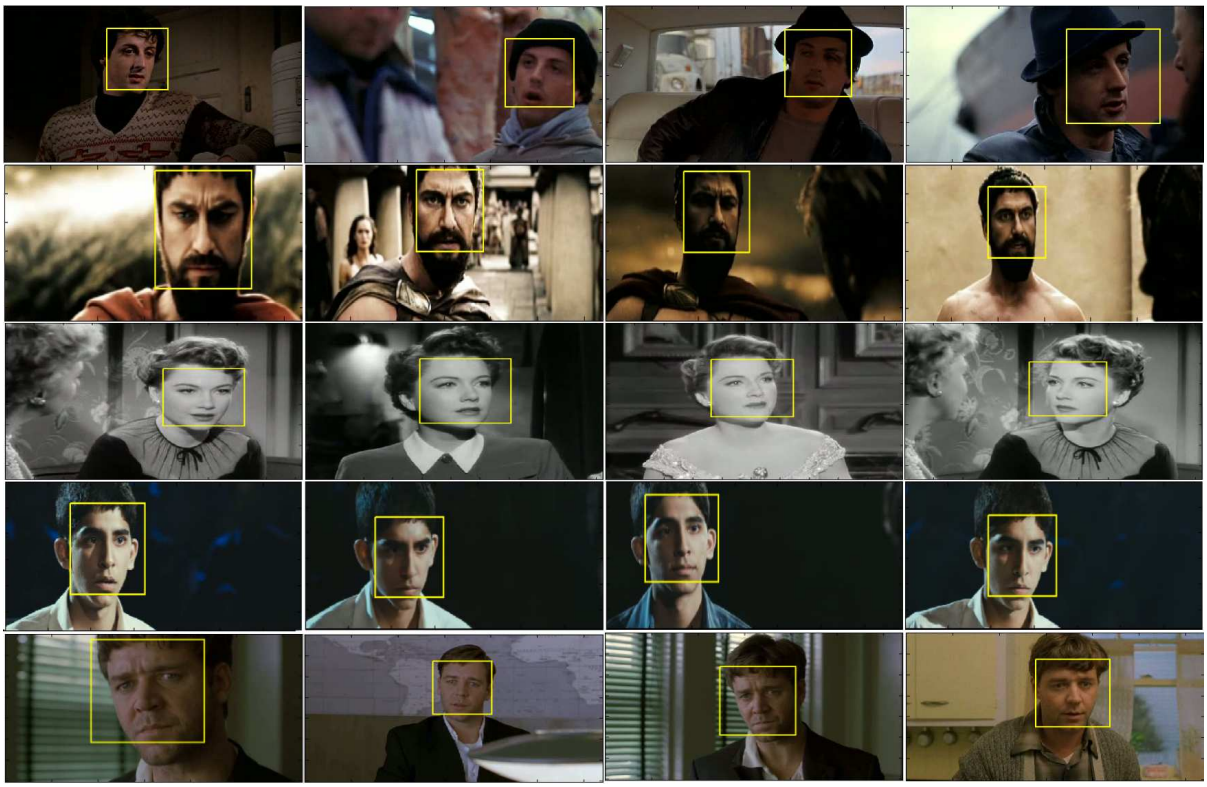


Figure 5.7 Main character discovered from the movies *Rocky1*, *300*, *All About Eve*, *Slumdog Millionaire* and *A Beautiful Mind*.

scenes. In films without much action, adventure or music mostly main characters are visible in the characteristic frames.

5.3.2.2 Identifying main characters in the Movie

The characteristic scenes can be used to predict the genre of the movie, in movie recommendation systems etc. Here we use these keyframes to predict the main character of the movie. Everingham *et al.* [44] have investigated the problem of automatically naming the characters in TV or film material. They do this by aligning subtitles and transcripts, and complement these cues by visually detecting which character in the video corresponds to the speaker.

We only use the key frames corresponding to most frequent visual words to find the main characters of the movie. Face detection and facial feature localization is done on these characteristic key frames. We start from the visual word with highest support and detect no more than 100 faces. Only faces larger than 100×100 are considered. The face descriptors are extracted from detected faces using Oxford VGG face processing code [44]. These face descriptors are clustered into 8-15 clusters by k-means. Then each of the cluster is pruned by removing all the faces which are at a distance greater than D from its cluster center. D is computed as the mean of the distances of faces from their cluster centers. Clusters

having faces only from nearby shots are rejected as the main character should be present at many points throughout the movie. It is desired to have smaller clusters with many members. We compute the cluster density as a summation of inverse of distances of all cluster members from the center. Cluster density for cluster c is given as: $\delta(C) = \sum_{i \in C} \frac{1}{d_i}$.

Only the most dense clusters are returned as the set of instances of main characters. Figure 5.7 shows the keyframes from the most dense clusters for movies *Rocky1*, *300*, *All About Eve*, *Slumdog Millionaire* and *A Beautiful Mind*. This works well as the characteristic scene mostly include many instances of main character's close-up face.

5.3.3 Video Stop Word Detection

Stop words in a language are words that many search engines do not stop for when searching for texts and titles on the web. These are common words, such as "the", "are", "is" etc. Similarly in text classification, elimination of stop words potentially increases the accuracy, reduces the feature space and helps to speed up the computation [134].

In videos too, certain sequence of frames repeatedly occur, motivating us to address the similar problems in videos. In videos, repetition could be either absolute or approximate. Examples are commercials in TV programs and news or routine events in surveillance videos. It is desired to detect/remove such redundancy for many applications like video summarization, and search.

We define *video stop word* as the frequently occurring *sequence of frames* that are not informative. For example advertisement can be a *video stop word*. The frequency of occurrence of a sequence in a video gives us the *TF* part based on which we select frequent sequences or potential *video stop-words*. We use *IDF* to classify it as *video stop-word*. To compute *IDF* for any sequence, we use frequency with which it occurs in news videos across different channels. Higher the *IDF* measure more is the probability that the frequent sequence is a commercial. This works fine as commercials occur frequently in all the channels and consistently for large intervals.

A typical news video would mostly contain important news or commercials in its set of frequent sequences. Therefore when detected *video stop-words* are removed from the set of frequent sequences and we get the *informative content* of the video. This is shown in Figure 5.8, based on *IDF* measure frequent sequences extracted from video are classified as *video stop-word* or informative part of video.

Thus the *video stop-word* detection results can be used to: (a) block undesirable content and (b) summarize video with only important content. The latter can be done by computing the *information measure* of each shot, S_{sid} as:

$$\Gamma_{S_{sid}} = \sum_{F \in F_{freq}} sign(F) * support(F) * |F| \tag{5.8}$$

$$sign(F) = \begin{cases} -1 & \text{if } F \in stopword \\ 1 & \text{otherwise} \end{cases}$$

where F_{freq} is a set of frequent sequences in shot S_{sid} and F is a frequent sequence in F_{freq}. Therefore, more informative shots can be kept in the summary by keeping a threshold on Γ_{sid}.

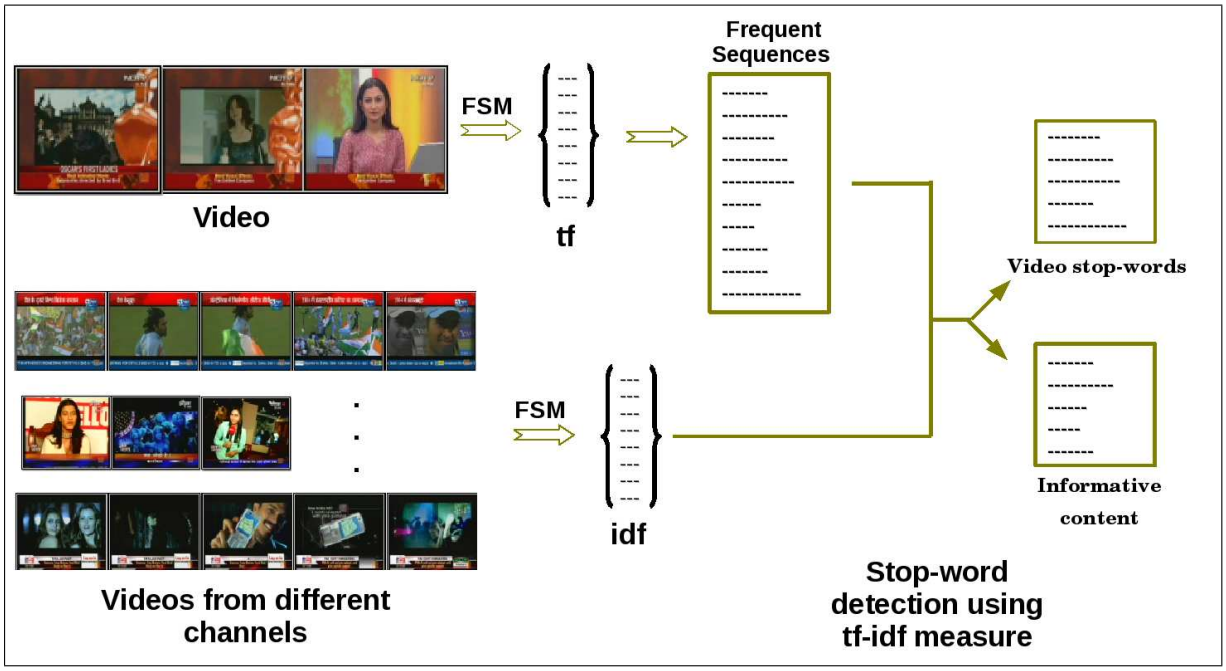


Figure 5.8 Stop word detection from video using Frequent Sequence Mining

5.3.3.1 Experiments

Experiments are done on news videos obtained from eight broadcast news channels. For estimating *IDF* measure, videos from these channels are mined over five days (100 hours) and extracted frequent sequences with their frequency of occurrence (*IDF*) are stored. While testing these *IDF* values are used to separate commercials from the rest of the frequent sequences obtained in the set of test videos. For testing 1000 news video clips of total duration of 10 hours are used. News videos have lot of overlaid text in its lower one-third part, which affects the feature descriptor and similar frames are assigned different visual word or item. To handle this we only extract features from the upper two-third part of the image.

We partition news video clips into shots and pick four frames per second within each shot. The ground truthing is done at the frame sequence level, each manually annotated frequent sequence is marked as *video stop-word* or *informative content*. This resulted in 91 frequent sequences out of which 68 were marked as visual stop-word and remaining 23 as informative content.

We use precision and recall to evaluate the detection performance. Precision and recall for *video stop-word* detection are computed as follows:

$$Precision = \frac{\#\ true\ visual\ stopwords\ detected}{\#\ total\ visual\ stopwords\ detected}$$

$$Recall = \frac{\#\ true\ visual\ stopwords\ detected}{\#\ total\ true\ visual\ stopwords}$$

Vocabulary size	Frequent sequence		Video stop-word		Informative Content	
	Precision	Recall	Precision	Recall	Precision	Recall
100	0.63	0.57	0.67	0.59	0.55	0.52
200	0.81	0.68	0.87	0.71	0.67	0.61
500	0.87	0.78	0.93	0.79	0.71	0.74
1000	0.97	0.91	0.98	0.94	0.90	0.83
2000	0.98	0.90	0.98	0.93	0.95	0.83

Table 5.1 Precisions and recalls for frequent sequence, *video stop-word* and informative content detection with different vocabulary sizes.

Sometimes due to disturbance in the telecast, few frames get transformed. A detection is considered to be true if the overlap between detected sequence F_{Det} and annotated sequence F_{GT} is more than 90%. Overlap is given by $\frac{|F_{Det} \cap F_{GT}|}{|F_{Det} \cup F_{GT}|}$. Figure 5.9 shows some examples of detected *video stop-words*.

For the experiments we set $min\_sup = 0.005$. The performances for detecting frequent sequence, *video stop-word* and informative content in terms of precision and recall are reported in Table 5.1. The precisions are high as it is difficult to get false frequent sequences when large enough vocabulary is chosen. With vocabulary size of 1000 and above, recall and precisions are high for all three cases. Precision always increases with vocabulary size. However recall starts decreasing with too much quantization. This is because with more number of clusters a slight variation in a frame can assign it to a different cluster, which may lead to false negatives. Some of the examples of detected commercials as *video stop-word* is shown in Figure 5.9.

Also the method is very efficient. In the online phase features are extracted at 150 fps and visual words are assigned at 250 fps with vocabulary size 1000. For mining $76,000$ sequences of average length 100 it takes about 40 seconds. This shows that the proposed method is scalable and can be effectively used for real-time on-line applications.

5.4 Summary

We have presented an approach to discover characteristic patterns in videos in an unsupervised fashion. Our method is based on finding frequently occurring patterns. One of our approaches employs frequent pattern mining for efficient characterization. We also propose to use randomized trees to represent frame as sequence of nodes and mine frequent sequences from a database of long sequences. To evaluate the proposed methods we compare them with each other and with a simple baseline. The approach is validated by experiments over a large movie dataset to discover characteristic scenes and main actors in the video. We also define *video stop-words* and detect them using frequent sequence mining. Stop words are identified using *TF-IDF* type measure for frame sequences. Traditional methods from data mining have been successfully used in computer vision and such techniques can result in fast, efficient algorithms for large scale video processing.

Figure 5.9 Some examples of *video stop-word* detection

Chapter 6

Conclusions

6.1 Summary

In this dissertation we have explored different aspects of visual processing in images and videos. Our objective is to achieve efficiency along with accuracy. The following summarizes the key contributions made:

- A state-of-art object detection and classification framework based on Random Forests is developed and evaluated. We have combined different types of feature descriptors based on bag of visual words and gradient orientations. We have carefully selected all the major components of such a framework, investigating features for visual representation, spatial grids, bootstrapping, post-processing, different parameters of random forest and sliding window detector. The evaluation was performed using multiple datasets with many object and scene categories. We have used a combination of different visual features with random forest for the high-level feature extraction task of TRECVID'08. On Pascal VOC'07 challenge our method achieves better performance than by the team ranked 1 did in the competition. We have shown that Random Forest classifier can be used for fast and accurate classification and object localization. Running the detector over large dataset of about 200 thousand key-frames in TRECVID'09 was only possible due to the fast training and testing of random forest. We also proposed Extended ROIs for classification by detection, which allows us to use only one aspect ratio and still cover for large range of aspect ratios while testing.

- We have proposed an architecture, *Vocabulary Trie*, for online content based processing of continuous stream of videos to detect video segments of interest. It is based on trie and bag of words model to simultaneously match multiple video segments in the database with the large input video stream. We focus on enhancing the throughput of the system and for the same we need the algorithm capable of simultaneous spotting of multiple examples. Our formulation also effectively utilizes the sequence information of the video stream, rather than treating it as a set of frames. To handle generic video filtering situation and address other practical challenges we propose Forest

of Tries. Our approach is generic and applicable for many applications that need matching video sequence. We have demonstrated this by doing content based copy detection (CBCD) experiments on MUSCLE VCD 2007 and broadcast news database.

- We have presented an approach for mining characteristic patterns in videos in an unsupervised fashion. Mining is done based on frequently occurring patterns in the video, some possible patterns can be scenes, characters or sequences of frames. In one of our approaches we apply frequent pattern mining algorithm to visual data. Our second approach uses randomized trees to represent frames or images as a sequence of items and finds the frequent ones. We could discover characteristic patterns and main actors of movie using frequent itemset mining. We also defined and detected *video-stop words* in broadcast news videos.

6.2 Future Work

The following perspectives for extension of the work presented in this thesis seem worth investigating:

- In our detection results on VOC datasets we observed many false negatives or missed detection due to truncation and occlusion. Also when the object is very small or with lot of articulation detections are missed. There has been efforts [48, 162] towards finding a way to handle such difficult cases. But still state of art is far from what is desired. Our next step is to work on dealing with the cases of truncation and occlusion.

  Given the fast training and testing of random forests it would be interesting to apply them on videos (with other tracking algorithm) for tracking objects by detection. Continuous improvement of classifier by online learning of trees can be a promising.

- Processing of on-line video sequences for information extraction and data mining has many significant applications in video scale-invariant retrieval. We are working towards designing appropriate processing (indexing, matching, ranking) architectures for information retrieval tasks from broadcast and other similar on-line video streams. One of the challenges in obtaining real-time solutions to the on matching in large line processing tasks is the computational efforts required for feature extraction and matching. Our proposed architecture, *Vocabulary Trie*, is highly parallelizable and a GPU based implementation can speed up the solution significantly.

- Frequent pattern discovery in visual data has potential for many extensions and applications. For example in content based retrieval, it might be worthwhile to use mining to learn structural patterns of features for a given query on-line.

Related Publications

- James Philbin, Manuel Marin-Jimenez, Siddharth Srinivasan and Andrew Zisserman
  Mihir Jain, Sreekanth Vempati, Pramod Sankar and C. V. Jawahar,
  In Proceedings of the TREC Video Retrieval (TRECVID) Workshop organized by NIST, Gaithersburg, USA, 2008.

- Mihir Jain, Sreekanth Vempati, Chandrika Pulla and C. V. Jawahar,
  In Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR), Santorini, Greece, 2009.

- Sreekanth Vempati, Mihir Jain, Omkar Parkhi and C. V. Jawahar
  Andrea Vedaldi, Marcin Marszalek and Andrew Zisserman
  In Proceedings of the TREC Video Retrieval (TRECVID) Workshop organized by NIST, Gaithersburg, USA, 2009.

Bibliography

[1] Internet Movie Database, http://www.imdb.com.

[2] Iksvm implementation. http://www.cs.berkeley.edu/ smaji/projects/fiksvm/.

[3] Trecvid. http://trecvid.nist.gov/.

[4] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 2004.

[5] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD*, 1993.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *International Conference on Very Large Databases*, 1994.

[7] R. Agrawal and R. Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, 1995.

[8] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation, 9(7)*, 1997.

[9] A. Aner, T. Lijun, and J. R. Kender. A method and browser for cross-referenced video summaries. In *International Conference on Multimedia and Expo*, 2002.

[10] S. Ayache and G. Qunot. Video corpus annotation using active learning. In *European Conference on Information Retrieval*, 2008.

[11] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality and the smo algorithm. In *International Conference on Machine Learning*, 2004.

[12] Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. *ACM Press*, 1999.

[13] A. Bagga, J. Hu, J. Zhong, , and G. Ramesh. Multi-source combined-media video tracking for summarization. In *International Conference on Pattern Recognition*, 2002.

[14] H. Bay, T. Tutelaars, and L. V. Gool. SURF: Speeded up robust features. In *Europen Conference of Computer Vision*, 2006.

[15] G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 2008.

[16] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *Europen Conference of Computer Vision*, 2008.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022, 2003.

[18] A. Bosch, A. Zisserman, and A. X. Munoz. Representing shape with a spatial pyramid kernel. In *ACM International Conference on Image and Video Retrieval*, 2007.

[19] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *Europen Conference of Computer Vision*, 2006.

[20] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Internation Conference on Computer Vision*, 2007.

[21] L. Breiman. Random forests. *ML Journal, 45(1)*, 2001.

[22] C. J. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, 1996.

[23] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 1998.

[24] M. B. B. C. H. Lampert and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition*, 2008.

[25] M. B. B. C. H. Lampert and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 2009.

[26] H. A. Chipman, E. L. George, and R. E. McCulloch. Bayesian ensemble learning. In *Neural Information Processing Systems (NIPS)*, 2006.

[27] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *ACM International Conference on Image and Video Retrieval*, 2007.

[28] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conference*, 2008.

[29] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Computer Vision and Pattern Recognition*, 2007.

[30] C. Colombo, A. D. Bimbo, and P. Pala. Retrieval of commercials by video semantics. In *Computer Vision and Pattern Recognition*, 1998.

[31] R. Cooley and J. S. B. Mobasher. Web mining: Information and pattern discovery on the world wide web. In *ICTAI*, 1993.

[32] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

[33] N. Dalal. *Finding People in Images and Videos*. PhD thesis, INRIA, Rhone-Alpes, France, 2006.

[34] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*. IEEE Computer Society, 2005.

[35] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2), 2008.

[36] T. Deselaers, A. Criminisi, J. Winn, and A. Agarwal. Incorporating on-demand stereo for real-time recognition. In *Computer Vision and Pattern Recognition*, 2007.

[37] M. Detyniecki and C. Marsala. Video rushes summarization by adaptive acceleration and stacking of shots. In *Int. Workshop on TRECVID Video Summarization*, 2007.

[38] T. G. Dietterich and D. Fisher. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 2000.

[39] I. Dohring and R. Lienhart. Mining tv broadcasts for recurring video sequences. In *ACM International Conference on Image and Video Retrieval*, 2009.

[40] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *International Conference on Computer Vision*, 2003.

[41] C. Elkan. Using the triangle inequality to accelerate k-means. In *International Conference on Machine Learning*, 2003.

[42] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[43] M. Everingham, L. V. Gool, C. K. I. Williams, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2006 (VOC2006). http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006.

[44] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy - automatic naming of characters in tv video. In *British Machine Vision Conference*, 2006.

[45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[46] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[47] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition*, 2005.

[48] P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 2009.

[49] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition*, 2008.

[50] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition*, 2003.

[51] M. M. Fleck, D. A. Forsyth, and C. Bregler. Finding naked people. In *Europen Conference of Computer Vision*, 1996.

[52] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Computer Vision and Pattern Recognition*, 2009.

[53] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *International Conference on Computer Vision*, 2005.

[54] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *Computer Vision and Pattern Recognition*, 2007.

[55] J. Han, J. Pei, B. Mortazavi-asl, Q. Chen, U. Dayal, and M. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *International Conference on Data Engineering*, 2000.

[56] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[57] A. Hauptmann, R. Yan, W.-H. Lin, M. Christel, and H. Wactlar. Can high-level concepts fill the semantic gap in video retrieval? a case study with broadcast news. *IEEE Transactions on Multimedia*, 2007.

[58] T. Hofmann. Probabilistic latent semantic indexing. In *ACM Conference on Research and Development in Information Retrieval*, 2003.

[59] C.-H. Hoi and M. R. Lyu. A novel log-based relevance feedback technique in content-based image retrieval. In *ACM Multimedia*, 2004.

[60] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *International Conference on Computer Vision*, 2005.

[61] J. D. Holt and S. M. Chun. Efficient mining of association rules in text databases. In *ACM International Conference on Information and Knowledge Management*, 1999.

[62] A. Hoogs, J. Rittscher, G. Stein, and J. Schmiederer. Video content annotation using visual analysis and a large semantic knowledgebase. In *Computer Vision and Pattern Recognition*, 2003.

[63] A. Joly and O. Buisson. A posteriori multi-probe locality sensitive hashing. In *ACM Multimedia*, 2008.

[64] A. Joly, O. Buisson, and C. Frlicot. Content-based copy detection using distortion-based probabilistic similarity search. *IEEE Transactions on Multimedia*, 2007.

[65] A. Joly, C. Frelicot, and O. Buisson. Robust content-based video copy identification in a large reference database. In *ACM International Conference on Image and Video Retrieval*, 2003.

[66] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *International Conference on Computer Vision*, 2005.

[67] S. Karthik and C.V.Jawahar. Efficient region based indexing and retrieval for images with elastic bucket tries. In *International Conference on Pattern Recognition*, 2006.

[68] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition*, 2004.

[69] S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 2006.

[70] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition*, 2008.

[71] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. In *ACM Multimedia*, 2006.

[72] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. Video copy detection: a comparative study. In *ACM International Conference on Image and Video Retrieval*, July 2007.

[73] J. Law-To, A. Joly, and N. Boujemaa. Muscle-vcd-2007: a live benchmark for video copy detection, 2007. http://www-rocq.inria.fr/imedia/civr-bench/.

[74] S. Lazebnik and C. S. J. Ponce. Semi-local affine parts for object recognition. In *British Machine Vision Conference*, 2004.

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

[76] Y. J. Lee and K. Grauman. Object-graphs for context-aware category discovery. In *Computer Vision and Pattern Recognition*, 2010.

[77] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 2008.

[78] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *British Machine Vision Conference*, 2003.

[79] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[80] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition*, 2005.

[81] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang. Pfp: Parallel fp-growth for query recommendation. In *ACM Recommendation Systems*, 2008.

[82] K. Lienhart. On the detection and recognition of television commercials. In *ICMCS*, 1997.

[83] C.-B. Liu and N. Ahuja. Vision based fire detection. In *International Conference on Pattern Recognition*, 2004.

[84] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[85] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[86] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition*, 2008.

[87] J. Malik, S. Belongie, T. K. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 2001.

[88] R. Mar'ee, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Computer Vision and Pattern Recognition*, 2005.

[89] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Computer Vision and Pattern Recognition*, 2009.

[90] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, 2002.

[91] A. Metwally, D. Agrawal, and A. E. Abbadi. Using association rules for fraud detection in web advertising networks. In *International Conference on Very Large Databases*, 2005.

[92] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, pages 63–86, 2004.

[93] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, pages 43–72, 2005.

[94] F. Moosman, B. Triggs, and F. Jurie. Fast discriminative visual codebook using randomized clustering forests. In *Neural Information Processing Systems (NIPS)*, 2006.

[95] E. Moxley, T. Mei, X.-S. Hua, W.-Y. Ma, and B. Manjunath. Auto video annotation through search and mining. In *International Conference on Multimedia and Expo*, 2008.

[96] S. Munder and D. M. Gavrila. An experimantal study on pedestrian classification. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 2006.

[97] H. Murase and S. Nayar. Learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, pages 5–24, 1995.

[98] H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 1995.

[99] K. Murphy, A. B. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *Neural Information Processing Systems (NIPS)*, 2003.

[100] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 2008.

[101] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, 2006.

[102] S. Nowozin, G. BakIr, and K. Tsuda. Discriminative subsequence mining for action classification. In *International Conference on Computer Vision*, 2007.

[103] S. Nowozin, K. Tsuda, T. Uno, T. Kudo, and G. BakIr. Weighted substructure mining for image analysis. In *Computer Vision and Pattern Recognition*, 2007.

[104] A. Oliva and A. B. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[105] E. E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. *Advances in kernel methods: support vector learning*, 1999.

[106] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition*, 2007.

[107] J. Y. Pan, H. Yang, and C. Faloutsos. Mmss:multi-modal story-oriented video summarization. In *International Conference on Data Mining*, 2004.

[108] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 2000.

[109] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *International Conference on Data Engineering*, 2001.

[110] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference on Machine Learning*, 2000.

[111] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, 2007.

[112] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, pages 637–646, 1998.

[113] S. Poullot, M. Crucianu, and O. Buisson. Scalable mining of large video databases using copy detection. In *ACM Multimedia*, 2008.

[114] T. Quack, V. Ferrari, and L. J. V. Gool. Video mining with frequent itemset configurations. In *ACM International Conference on Image and Video Retrieval*, pages 360–369, 2006.

[115] T. Quack, V. Ferrari, B. Leibe, and L. V. Gool. Efficient mining of frequent and distinctive feature configurations. In *International Conference on Computer Vision*, 2007.

[116] T. Quack, B. Leibe, and L. V. Gool. World-scale mining of objects and events from community photo collections. In *ACM International Conference on Image and Video Retrieval*, 2008.

[117] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. V. Gool. Modeling scenes with local descriptors and latent aspects. In *International Conference on Computer Vision*, 2005.

[118] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[119] T. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, pages 139–152, 2007.

[120] G. Rogez, J. Rihan, S. Ramalingam, C. Orritea, and P. H. S. Torr. Randomized trees for human pose detection. In *Computer Vision and Pattern Recognition*, 2008.

[121] C. Rothwell. Hierarchical object description using invariants. *LNCS*, pages 397–414, 1993.

[122] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Computer Vision and Pattern Recognition*, 2006.

[123] G. Salton and M. J. McGill. Introduction to modern information retrieval. *McGraw-Hill*, 1986.

[124] J. J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In *ACM conference on Recommender systems*, 2007.

[125] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *International Conference on Very Large Data Bases*, 1995.

[126] B. Schiele and J. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, pages 31–50, 2000.

[127] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, pages 530–535, 1997.

[128] H. Schneiderman and T. Kanade. A statistical model for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition*, 2000.

[129] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[130] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *British Machine Vision Conference*, 2008.

[131] H. Shang and T. Merrettai. Tries for approximate string matching. *IEEE Transactions On Knowledge And Data Engineering*, 1996.

[132] T. Sharp. Implementing decision trees and forests on a gpu. In *Europen Conference of Computer Vision*, 2008.

[133] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition*, 2008.

[134] C. Silva and B. Ribeiro. The importance of stop word removal on recall values in text categorization. *Neural Networks*, pages 320–324, 2003.

[135] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. In *International Conference on Computer Vision*, 2005.

[136] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *International Conference on Computer Vision*, 2005.

[137] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, 2003.

[138] J. Sivic and A. Zisserman. Efficient visual content retrieval and mining in videos. In *Pacific-Rim Conference on Multimedia*, 2004.

[139] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. In *Computer Vision and Pattern Recognition*, pages 488–495, 2004.

[140] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *ACM International Workshop on Multimedia Information Retrieval*, New York, NY, USA, 2006. ACM Press.

[141] C. G. Snoek, B. Huurnink, L. Hollink, M. de Rijke, G. Schreiber, and M. Worring. Adding semantics to detectors for video retrieval. *IEEE Transactions on Multimedia*, 2007.

[142] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*, 1996.

[143] Sung-Hwan-Lee, Won-Young-Yoo, and Young-Suk-Yoon. Real-time monitoring system for tv commercials using video features. In *ICEC*, 2006.

[144] C. M. Taskiran, Z. Pizlo, A. Amir, D. Ponceleon, , and E. J. Delp. Automated video program summarization using speech transcripts. *IEEE Transactions on Multimedia*, 2006.

[145] G. Taubin and D. Cooper. Object recognition based on moment (or algebraic) invariants. *Geometric Invariance in Computer Vision*, pages 375–397, 1992.

[146] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *Conference on Computer Vision and Pattern Recognition*, 2008.

[147] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *Computer Vision and Pattern Recognition*, 2008.

[148] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition*, 2004.

[149] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 2007.

[150] A. B. Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):153–167, 2003.

[151] A. B. Torralba, K. Murphy, and W. Freeman. Using the forest to see the trees: exploiting context for visual object detection and localization. *Communications of the ACM*, 53(3):107–114, 2010.

[152] B. T. Truong, C. Dorai, and S. Venkatesh. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *ACM Multimedia*, 2000.

[153] T.-H. Tsai and Y. C. Chen. A robust shot change detection method for content-based retrieval. In *Int. Symp. Circuits and Systems*, 2005.

[154] V. S. Tseng, J.-H. Su, J.-H. Huang, and C.-J. Chen. Semantic video annotation by mining association patterns from visual and speech features. In *PAKDD*, pages 1035–1041, 2008.

[155] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004.

[156] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, pages 177–280, 2008.

[157] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1999.

[158] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *International Conference on Computer Vision*, 2007.

[159] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 2005.

[160] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[161] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision*, 2009.

[162] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial occulsion. In *Neural Information Processing Systems (NIPS)*, 2009.

[163] S. Vijayalakshmi, V. Mohan, and S. S. Raja. Mining constraint-based multidimensional frequent sequential pattern in web logs. *European Journal of Scientific Research*, 36(3):480–490, 2009.

[164] P. Viola and M. Jones. Rapid object detection using boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, 2001.

[165] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2001.

[166] F. Wang and B. Merialdo. Multi-document video summarization. In *International Conference on Multimedia and Expo*, 2009.

[167] J. Winn and A. Criminisi. Object class recognition at a glance. In *Computer Vision and Pattern Recognition*, 2006.

[168] Y. Yan, B. C. Ooi, and A. Zhou. Continuous content-based copy detection over streaming videos. In *International Conference on Data Engineering*, 2008.

[169] B. Yao and L. Fei-Fei. Grouplet: a structured image representation for recognizing human and object interactions. In *Computer Vision and Pattern Recognition*, 2010.

[170] M.-C. Yeh and K.-T. Cheng. Video copy detection by fast sequence matching. In *ACM International Conference on Image and Video Retrieval*, 2009.

[171] T. Yeh, J. Lee, and T. Darrell. Adaptive vocabulary forests for dynamic indexing and category learning. In *International Conference on Computer Vision*, 2007.

[172] T. Yeh, J. J. Lee, and T. Darrell. Fast concurrent object localization and recognition. In *Computer Vision and Pattern Recognition*, 2009.

[173] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *ACM International Conference on Information and Knowledge Management*, 2006.

[174] P. Yin, A. Criminisi, J.Winn, and I. Essa. Tree-based classifiers for bilayer video segmentation. In *Computer Vision and Pattern Recognition*, 2007.

[175] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *Computer Vision and Pattern Recognition*, 2009.

[176] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, pages 31–60, 2001.

[177] G. J. Z. W. Zhang and D. Samaras. Real-time accurate object detection using multiple resolutions. In *International Conference on Computer Vision*, 2007.

[178] H. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Syst.*, pages 10–28, 1993.

[179] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 2007.

[180] H. Zheng and M. Daoudi. Blocking adult images based on statistical skin detection. *Electronic Letters on Computer Vision and Image Analysis, Vol4 No2*, December 2004.

[181] H. Zheng, H. Liu, and M. Daoudi. Blocking objectionable images: adult images and harmful symbols. In *International Conference on Multimedia and Expo*, 2004.

[182] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. In *Multimedia Systems*, 2003.

[183] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition*. IEEE Computer Society, 2006.