Discovering Objects in Images and Videos

David Liu

August 12, 2008

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.

Thesis advisor
Prof. Dr. Tsuhan Chen, Chair
Prof. Dr. Vijayakumar Bhagavatula (Carnegie Mellon University)
Prof. Dr. Martial Hebert (Carnegie Mellon University)
Prof. Dr. Larry Davis (University of Maryland)

Abstract

This thesis presents a novel way of scene analysis in images and videos. Traditional scene analysis using object detection involves a lot of human labor for labeling the images, and also has the difficulty of handling a large number of objects categories. Our approach to scene analysis is unsupervised in nature. Given a video, we want to "discover" the objects of interest. No single labeled image is used to pre-train or initialize the system. Still, the system is able to discover the objects of interest. It works on a wide variety of videos and it can discover objects belonging to a large set of different categories. It works in crowded scenes with distracting background pattern and motion. It works in partial occlusions and total removal. The probabilistic framework consists of an appearance model and a motion model. The appearance model exploits the consistency of object parts in appearance across frames. The motion model exploits the motion continuity across frames. Together, they provide appearance and location estimates of the objects of interest. This framework provides a basis for higher level video content analysis tasks.

Acknowledgments

First, I would like to thank my advisor Tsuhan Chen for his supervision during these years at CMU. Tsuhan provided me with a wealth of knowledge and experiences. Tsuhan brought me in to CMU after we first met in Taiwan, and I am deeply grateful for his support. His enthusiasm to engage students in research problems has served as an inspiration to me. I am also grateful to members of the AMP Lab. The weekly group meetings we had together and the fun time we enjoyed together are all great memories. I would also like to thank my thesis committee members, Prof. Kumar, Prof. Hebert, Prof. Davis, for the discussions and advises you have given to me. Lastly, I would like to thank my family, my parents, my wife for their support.

Contents

List of Figures

List of Tables

Part I

Overview

Chapter 1

Introduction

1.1 Goal

This thesis presents a framework for discovering useful objects from images and videos. A system that could automatically extract objects from images and videos would be of great importance. Applications are countless: surveillance, image and video search, robots with vision capability, computers with visual interfaces, smart vehicles. This chapter serves as an introduction to the thesis.

1.2 Problem definition

Data mining is the process of extracting implicit and useful information from data. In this work, our data consists of images or videos. Let us consider images first. There are two types of input: single input image vs. multiple input images.

The task of making sense out of a single image is called unsupervised image segmentation. One can identify the structure within a single image by analyzing smaller patches and grouping them into larger segments using plausible psychological rules. The outcome? An image segmented into multiple regions by drawing outlines along the boundaries of segments, as if a human would do if asked to do the same task. An image segment does not necessarily correspond to an object, since an object (e.g., a car) can consist of multiple segments (wheels, window, body) that have very distinct visual properties and are naturally segmented into distinct regions.

In this thesis, we are interested in the other scenario: multiple input images. One can still

segment the images one by one separately, but more than that, one can analyze the images jointly. The outcome? One can identify "objects". In the example above, if we have multiple images that consist of cars, we might be able to figure out that the wheels or the windows are smaller parts of a larger entity that repeats over and over again in different images. In this way, the concept of a larger entity that is composed of smaller parts emerges.

A video is a sequence of images. More than that, it is a sampling of a scene usually with a high enough frequency so that things move in a relatively smooth way. Exploiting the temporal continuity for data mining is also of out interest in this thesis.

1.3 Probabilistic Graphical Models

For a computer to reach human vision capabilities is not an easy task. There is too much information in images, yet too few time and space to analyze them. For example, binary images of size 1000×1000 pixels can have $2^{1000000}$ different variations, which is larger than 10^{100000}. If all of these variations contain valuable information, then mining data in such a large space would be infeasible.

Fortunately, images often have inherent structure that could potentially reduce the amount of variations. For example, in natural images, neighboring pixels often share similar properties such as color or texture. Another important structure is that objects are composed of parts, and scenes are composed of objects. Once we recognize some parts in an image, we have better clue about what the other parts in the image might be. Exploiting these sorts of structures for scene understanding and object detection is one of the current trends in computer vision and pattern recognition research. With such a hierarchy at hand, the $2^{1000000}$ variations become more organized, and data mining becomes more reasonable.

If it is indeed true that within the $2^{1000000}$ variations there is a lot of redundant information, then our hope is to represent these $2^{1000000}$ variations by some other factors that are not directly observed, yet carry the structure and explains the large amount of variations that are observed. These factors are called *latent* or *hidden variables*. The relationship between a latent variable (e.g. the concept of a "car") and an observed variable (e.g. the image pixels that constitute a car) can be described by a *probabilistic graphical model*. In general, a probabilistic graphical model describes the mathematical relationship between L-dimensional hidden variables and H-dimensional

observed variables, with $L < H$. Latent variables might represent human interpretations of the real world, or they might simply exist for theoretical convenience. In other words, it is up to the human to interpret the underlying meaning of latent variables. In this thesis, the latent variables can be considered as the hidden (unknown beforehand) object categories we would like to discover.

A nice property of using probabilistic graphical models is that building such a system requires much less human labor in terms of data preparation, as opposed to some state of the art object detection systems based on discriminative models that require intensive human power to prepare training data. This unsupervised nature is especially desirable when the human experts do not have enough data to train the system, or no training data at all! In addition, the avoidance of requiring labeled images poses several other advantages. First, an image may consist of many objects in a complex layout. So far there is no common approach to annotating images at the object level. Second, there are many visual illusions showing that different people may have different understandings of an image, hence the subjectiveness might result in negative impact on image annotations. In summary, it is very expensive and difficult to collect large mount of accurately annotated images for constructing a image understanding system. Considering the abundance of images and videos available on the Internet, probabilistic graphical models provide a promising direction.

A special type of probabilistic graphical models called *topic models* has been used in the statistical text understanding community and later in computer vision. Since images and videos have some distinct properties that text does not have, these properties inspire our extensions to existing topic models. Below we will give an overview of topic models and our extensions.

1.4 Topic models

Topic models have been used in the statistical text understanding community for automatically discovering topics from a collection of documents. In computer vision, documents are analogous to images and words are analogous to visual words, being vector quantized local feature descriptors. An image is considered a mixture of "topics" and each topic is considered a mixture of words. In our work, the foreground object and background clutter are the two *topics*.

First, we find a number of patches to generate the visual words. These patches are determined by running interest point detectors; see [1] for a collection. Features are then extracted from these

patches by Scale Invariant Feature Transform (SIFT) [2], yielding a 128-dimensional local feature descriptor for each patch. In all of our experiments, we *intentionally* discard color information and extract patches and SIFT descriptors from grayscale images in order to make object discovery more challenging. Patches and features extracted from color images [3] can easily be used instead.

The SIFT descriptors are then collected from all images and vector quantized using k-means clustering. The resulting J cluster centers (we use $J = 200$) form the dictionary of visual words, $\{w_1, ..., w_J\}$. Each patch can then be represented by its closest visual word. Patches are now represented by discrete visual words instead of continuous SIFT descriptors. Note that acquisition of visual words does not require any labeled data, which shows the unsupervised nature of this system. This also means they are general enough to be applied to a wide range of different tasks.

Denote the images by $\{d_1, ..., d_N\}$ and define topic variables $z_i(k)$ indicating if the i^{th} patch in image d_k is originated from the foreground object or from the background clutter. $\{z_i(k)\}$ are hidden variables; it is our goal to infer their values. Define the conditional probabilities $P(z|d)$ and $P(w|z)$ for each patch as follows: $P(z = z_{FG}|d = d_i)$ indicates in image d_i how likely a patch originates from the foreground object ; $P(z = z_{BG}|d = d_i)$ is defined likewise. $P(w = w_j|z = z_{FG})$ indicates how likely a patch originated from the foreground object has appearance corresponding to visual word w_j; $P(w = w_j|z = z_{BG})$ is defined likewise.

In the following, we will give an overview of five approaches, **A1** to **A5**, which we will later refer to.

1.4.1 A1: No location or shape info

A particular topic model called Probabilistic Latent Semantic Analysis (PLSA) has recently been applied to object discovery [4][5] and has shown good results. PLSA asserts that the probability of observing a patch in image d originated from topic z with appearance w is given by

$$P(d, w, z) = P(w|z)P(z|d)P(d). \tag{1.1}$$

Using inference methods, one can infer the values of the hidden topic variables based on $P(z|d, w)$ [6]. One important drawback of PLSA is that it is based on the bag-of-words image representation which completely ignores the position of the visual words. In other words, if we randomly shuffle around the patches in the image, PLSA would still infer the same hidden topic for each patch! This is often not desirable because the spatial configuration of patches can give us a clue about their

identity. Approaches that use PLSA in text [6], video [7] or still images [4] would suffer from this inherent drawback.

One point worth mentioning is, PLSA is known for its capability of handling polysemy: if a visual word w is observed in two images d_i and d_j , then the topic associated with that word can differ in d_i and d_j : $\arg\max P(z|d_i, w)$ can be different from $\arg\max P(z|d_j, w)$. In other words, PLSA allows a visual word to have different meanings in different images.

1.4.2 A2: Spatial location

When w is merely an appearance descriptor and contains no spatial information, the model does not care about the spatial ordering of the w's. This is problematic when spatial information is an important cue for recognition. A number of approaches have been employed in the literature to include spatial location information of local descriptors for recognition [8][9][10][11] [12] [13].

We can use a topic model to specify where the object is more likely located. Let r denote the location of a patch. For a patch in image d with appearance w and location r, the joint distribution $P(d, w, r)$ has the form $P(d, w, r, z) = P(d)P(z|d)P(w|z)P(r|z)$. $P(r|z)$ is a spatial distribution that models where a patch with topic z is more likely to occur. This model can be useful in modeling pedestrians for instance less likely to walk in the sky. Including location information more or less solves the ambiguity mentioned in **A1**, that is, the spatial ordering of w's now has an impact on the discovery of topics, even when the topic appearance has large ambiguity (large overlap between the distributions $P(w|z_i)$ and $P(w|z_j)$, $i \neq j$**)**. The spatial distribution $P(r|z)$ uses the same parameters across all documents, hence it is a global location model, and it is not translation nor scale invariant.

1.4.3 A3: Spatial clustering

In Section 2 we will introduce a model that is based on the assumption that an object normally consists of patches that co-exist tightly rather than scattered around loosely. The location and scale of the object is hypothesized through a spatial distribution $P(r|d, z)$, where r denote the location of a patch, and the joint distribution is $P(d, w, r, z) = P(d)P(z|d)P(w|z)P(r|d, z)$. The algorithm for finding the topic and object location and scale can be viewed as performing joint spatial and appearance clustering. Different than **A2**, the parameters for the spatial distribution $P(r|d, z)$ are estimated per image, hence it provides translation and scale invariance. Notice that this approach

only models the spatial clustering behavior of patches but it does not model the relative position or the ordering between patches.

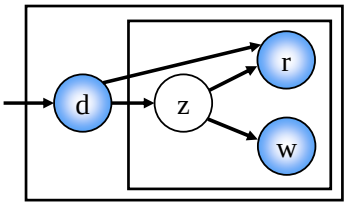


Figure 1.1: The graphical model. The outer plate indicates the graph is replicated for each image, and the inner plate indicates the replication of the graph for each patch. The topic variable z is hidden.

1.4.4 A4: Correspondence

We extend the topic model framework to incorporate information about spatial configuration. Rather than building a shape model as in constellation models [14], pictorial structures [15], or fragment-based models [16], we will exploit the fact that similar objects in different images are more likely to have strong *correspondences* and extend the topic model to include this extra piece of information.

Correspondence-based object recognition has been in the literature for nearly forty years. Even though computing correspondence is computationally expensive, it is still popular, because of its promising performance. Recent work by [17][18][19] [20] use correspondence as a central element in their object recognition framework. However, their model and learning algorithm differ substantially from what is proposed here.

Our use of correspondence is to provide a non-parametric representation of the location of the *consistent* patches. By using correspondence methods that take into account the spatial distortion of a correspondence and allow partial matching, the more matches a patch has, the better chances the patch belongs to a foreground object, as opposed to background clutter. This piece of information is employed by the topic model in the form of a feature r, or *reward*. More precisely, the reward for a patch is high when this patch is repeatedly matched against other patches in other images. On the other hand, patches from random clutter are less likely to find as many matches, which results in lower rewards. This is precisely the notion that objects of interest normally show higher

consistency across images and it is the consistency that tells an object apart from background clutter or other objects.

1.4.5 A5: Temporal consistency

Extending topic models from still images to motion videos requires the integration with a temporal model. We propose a novel spatial-temporal framework that uses topic models for appearance modeling, and the Probabilistic Data Association (PDA) filter for motion modeling. The spatial and temporal models are tightly integrated so that motion ambiguities can be resolved by appearance, and appearance ambiguities can be resolved by motion. With extensive experiments, we show promising results that cannot be achieved by appearance or motion modeling alone.

1.5 Limitations of the system

One important question to ask is, when and where the system does not work. The answer is as follows: if two objects are always stationary with respect to each other (no relative motion), and if they always co-occur in each and in all frames, then the current system cannot separate them.

My understanding of why human beings might still be able to separate them into two distinct objects is due to the availability of additional visual cues that the system is not utilizing yet. For example, one of the two objects could be highly salient in color while the other object has a very dull appearance. In this case, human might focus more on the visually appealing one, and hence implicitly considering them as distinct objects. Another possibility is that, human has prior experience of seeing one object in a situation where the second object was not present. Both cases can be incorporated into our system framework, as we discuss below.

In the first case, additional features such as color saliency can be represented as random variables. By introducing additional conditional probability distributions (such as the ones we used in Markov Random Field image segmentation and in Geometric Consistent Regions), we can automatically learn the correlation between, say, the color saliency and how likely the object is of interest. If this new feature is distinct for these two objects, we have chances to separate them.

In the second case, we simply need to find data where two objects are separated. Once we present the system with images where only one object appears, the system is then able to tell them apart.

The current system is based on a limited number of visual cues, and introducing more visual cues would certainly increase the system's capability. However, as easy as it sounds, the extraction of visual cues itself is a challenging topic.

Part II

Discovering objects in images

Chapter 2

Spatial Clustering

2.1 Semantic Shift algorithm

Here we introduce the Semantic-Shift algorithm that explicitly takes spatial structure into account. Semantic-Shift consists of a modified version of PLSA, which has an extra spatial distribution component, and after every iteration of Expectation-Maximization [21], the probability of each word belonging to a specific topic (i.e., the latent semantic of each word) is being updated. As a result, the location and scale estimates of the foreground object are shifted. Inspired by the mode-seeking 'mean-shift' algorithm [22][23], our algorithm seeks for the semantics or topics, hence named 'semantic-shift'.

2.1.1 Model description

We assume that in each image there is no more than one single foreground object. Experiments on the UIUC car dataset (see Fig.2.2) show that even if there are multiple cars (foreground objects) in one image, Semantic-Shift still can produce good results as long as the following assumptions are satisfied. We make the assumption that the foreground object has no holes and has a convex shape. Both assumptions hold for most objects. The reason we need these assumptions is because we want to model the image area occupied by the object as a Gaussian, which is convex in a 2D image. Since there is no specific reason our model should be confined to a Gaussian except for simplicity, it should be possible to loosen the Gaussian shape assumption so that the model can handle more complicated shapes of objects.

We introduce the conditional probability $P(r|d,z)$. The dependence of position r on image d allows the foreground object to have *different* locations and scales in every image d. In other words, the foreground object is allowed to be at vastly different positions in both learning and testing. The scale can also vary freely over different images. Allowing different locations and scales in different images is desirable, as it provides the basis for scale-invariance and translation-invariance. The dependence of r on z allows the foreground object and the background clutter to have different locations and scales. We model the location distribution of the background clutter as the probability of the complement of that location being foreground. This describes the realistic situation that, at a particular location, the higher the probability of being foreground, the lower the probability of being background.

2.1.2 Location and scale estimation for foreground object

The conditional probability $P(r|d,z)$ is computed for each topic in each image. Denote the topic z_k that corresponds to the foreground object as z_{FG} and call it the foreground topic. Since we want the system to be unsupervised, we need to create an unsupervised rule for deciding which of the two topics, z_1 and z_2, is the foreground topic z_{FG}. We achieve this by assuming that the foreground topic has on average a smaller spatial support than the background topic. We call this step *foreground topic identification*, as described below.

In the literature, finding a robust estimate of location and scale under the univariate model assumption is not new. In our experiments, we simply take the weighted mean and weighted standard deviation as estimates of the location and scale of the foreground object. Specifically, we define the spatial support of a topic z_k in an image d_i as the weighted standard deviation of the positions of all interest points $\{r_1^{d_i}, ..., r_{|d_i|}^{d_i}\}$, where each interest point $r_p^{d_i}$ is given a weight $v_p = P(z_k|d_i, w_j)$, where w_j is the visual word corresponding to point $r_p^{d_i}$. The weighted standard deviation is defined as

$$\hat{\sigma}_{ik} = \sqrt{\frac{\sum_{p=1}^{|d_i|} v_p (r_p^{d_i} - \hat{\mu}_{ik})^2}{\frac{|d_i|-1}{|d_i|} \sum_{p=1}^{|d_i|} v_p}} \tag{2.1}$$

where $\hat{\mu}_{ik}$ is the weighted sample mean of the positions of all interest points. After foreground topic identification, we denote the location and scale estimate of the foreground object in image d_i as $\hat{\mu}_{i,FG}$ and $\hat{\sigma}_{i,FG}$, and the corresponding topic as z_{FG}. We assume the interest points belonging to the foreground topic have a spatial distribution in the form of a Gaussian, $P(r_p^{d_i}|z_{FG}, d_i) \equiv$

$N(r_p^{d_i}|\hat{\mu}_{i,FG},\hat{\sigma}_{i,FG})$. The spatial distribution of the background is then set to the complement of the former distribution, meaning that the more likely the foreground, the less likely the background, and vice versa.

2.1.3 Model fitting

The goal is to maximize the log-likelihood,

$$\mathcal{L} = \sum_i \sum_j \sum_p n(d_i, w_j, r_p^{d_i}) \log P(d_i, w_j, r_p^{d_i}) \tag{2.2}$$

where the joint probability $P(d_i, w_j, r_p^{d_i})$ factorizes as

$$P(d_i, w_j, r_p^{d_i}) = P(d_i)P(z_k|d_i)P(w_j|z_k)P(r_p^{d_i}|z_k, d_i) \tag{2.3}$$

We use a modified version of the EM algorithm where the location and scale of the foreground object are estimated in each iteration. We use E'-step and M'-step to denote the two iteration steps.

In each iteration of Semantic-Shift, the posterior probability $P(z_k|d_i, w_j, r_p^{d_i})$ is updated as in Eq. (2.4). This quantity tells us how likely the visual word w_j at position $r_p^{d_i}$ in image d_i is part of the foreground (or background) object. Using this posterior probability, we can compute the location and scale estimates of the foreground object, as explained in the previous section. This explains Eq. (2.8).

Here is the Semantic-Shift algorithm:

$\mathbf{E'-step}$:

$$P(z_k|d_i, w_j, r_p^{d_i}) \propto P(z_k|d_i)P(w_j|z_k)P(r_p^{d_i}|z_k, d_i) \tag{2.4}$$

$\mathbf{M'-step}$:

$$P(w_j|z_k) \propto \sum_i \sum_p n_{ijp} P(z_k|d_i, w_j, r_p^{d_i}) \tag{2.5}$$

$$P(z_k|d_i) \propto \sum_j \sum_p n_{ijp} P(z_k|d_i, w_j, r_p^{d_i}) \tag{2.6}$$

$$P(d_i) \propto \sum_j \sum_p n_{ijp} \tag{2.7}$$

$P(r_p^{d_i}|z_k, d_i)$ updated according to Section 2.1.2 (2.8)

---

where $n_{ijp} \equiv n(d_i, w_j, r_p^{d_i})$. Note that these equations need normalization to make them probability distributions. Both learning and inference (test stage) use the above iterative procedure to obtain the conditional probabilities, except that during inference (test stage) the factor $P(w_j|z_k)$ is kept fixed and not being updated anymore.

After each iteration, the location and scale estimates of the foreground topic are shifted to a new value, and the Gaussian distribution $P(r_p^{d_i}|z_k, d_i)$ is updated accordingly. Notice that the E'-step depends on the term $P(r_p^{d_i}|z_k, d_i)$, i.e., the "shift" of the location and scale estimates plays a central role in the overall iterative scheme.

It is worth mentioning that, even though the location and scale estimates are found on a per image basis, they are actually tightly coupled with all the system parameters across all images, since the same conditional probability table $P(w|z)$ is used by all images.

2.2 Experiments

The red and green ellipses in Fig. 2.1 and 2.2 represent the inferred most likely topics of each visual word; red indicates that the system labels the particular region as foreground. Comparing PLSA to Semantic-Shift, it can be seen that foreground objects are more precisely located by Semantic-Shift.

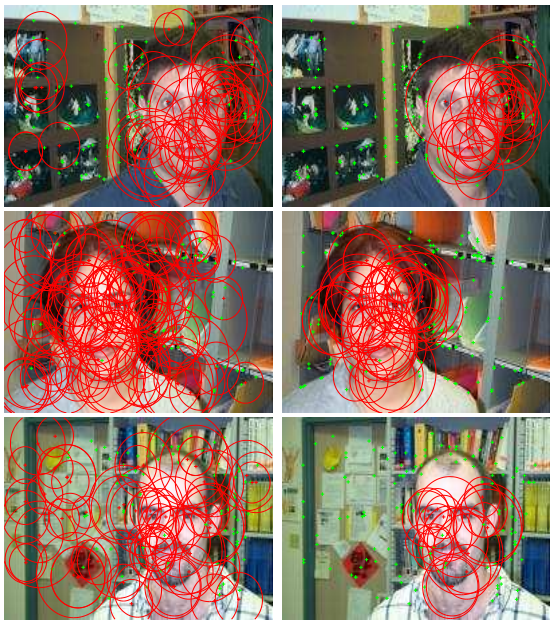


Figure 2.1: Results on test data. Left column: PLSA. Right column: Semantic-Shift.

Figure 2.2: Results on test data. Left column: PLSA. Right column: Semantic-Shift.

Chapter 3

Discovery Integrated with Segmentation

3.1 Introduction

Image segmentation can be categorized into interactive and non-interactive. Interactive systems such as 'Magic Wand' [24] and 'Intelligent Scissors' [25] have practical importance in image editing. These systems start with a user specified region or rough contour and use texture or edge information to achieve segmentation. Recently, there has been improvements on further reducing the amount of required user interaction to achieve comparable segmentation performance. In the GrabCut method [26], only a rough bounding box is needed, which is a significant improvement over previous methods.

Suppose the user wants to segment the same type of object from a *set* of images, instead of a single image. In previous interactive systems, the user must specify the object within each image, which can be time consuming. If the target objects share certain characteristics, these characteristics can be shared across images. Hence it is possible to further reduce the required amount of human interaction.

In this work, we would like to investigate how well we can segment a *set* of images with *zero* mouse clicks. On a high level, this is achieved by the interaction between a method that provides rough bounding boxes of the target objects in each image, and a method that uses the bounding boxes as seeds to achieve foreground-background segmentation. The method that provides bounding boxes will be called 'automatic object discovery', and the method that achieves segmentation is based on the 'GrabCut' method [26].

We present a novel approach to background cutout for image editing. We show how background cutout can be achieved without any user labeling. This is in contrast to current methods, where the user needs to label each image separately. Our method uses automatic object discovery methods to provide location and scale estimates of the object of interest; these estimates then provide seeds for initializing color distributions of a segmentation algorithm. We show that our approach can achieve similar performance as traditional methods that require users to specify for each image a bounding box of the target object.

We will review image segmentation by GrabCut in Section 3.2. We will then detail the automatic object discovery method in Section 3.3, and its interaction with GrabCut in Section 3.4. This is followed by experiments in Section 6.4, and a summary in Section 3.6.

3.2 Image Segmentation by GrabCut

The GrabCut method [26] is based on interactive graph cuts [27], which provides an energy minimization framework for segmenting a single image into foreground (object) and background. Hard constraints are obtained by the user who specifies certain pixels as foreground or background. Soft constraints incorporate both boundary and region information. Minimization is done using a standard minimum cut algorithm. The obtained solution gives the best balance of boundary and region properties among all segmentations satisfying the constraints.

More specifically, two Gaussian mixture models (GMM) are used to model the RGB color of each pixel z_n, one for the foreground and one for the background. A vector $\mathbf{k} = \{k_1, ..., k_N\}$ assigns to each of the N pixels a unique GMM component, one component either from the background or from the foreground model, according to $\alpha_n = 0$ or 1. The energy function $E = U + V$ consists of a node potential $U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n)$ where

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2}\log\det\Sigma(\alpha_n, k_n) + \frac{1}{2}(z_n - \mu(\alpha_n, k_n))^T \Sigma(\alpha_n, k_n)(z_n - \mu(\alpha_n, k_n))$$

so that the parameters are

$$\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = 0, 1, k = 1...K\}$$

and a smoothness potential

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in N} [\alpha_m \neq \alpha_n] \exp -\beta \left\| z_m - z_n \right\|^2$$

where [.] is the indicator function, N is the set of neighboring pixels, and γ and β control the strength of the smoothness term. Once the energy function is defined, the *segmentation mask* (i.e., the foreground-background identity α_n of each pixel) is found through the following steps:

1. User selects a bounding box by mouse clicking. Pixels outside of bounding box are marked as background ($\alpha_n = 0$). Pixels inside the box are marked as α_n unknown.
2. Computer creates an initial image segmentation, where all unknown pixels are tentatively placed in the foreground class and all known background pixels are placed in the background class.
3. Gaussian Mixture Models (GMMs) are created for initial foreground and background classes.
4. Each pixel in the foreground class is assigned to the most likely Gaussian component in the foreground GMM. Similarly, each pixel in the background is assigned to the most likely background Gaussian component.
5. The GMMs are thrown away and new GMMs are learned from the pixel sets created in the previous set.
6. The energy function is minimized to find a new tentative foreground and background classification of pixels (i.e., minimize the energy function E over α_n).
7. Repeat from Step 4 until convergence.

Convergence properties are discussed in more detail in [27][26].

Notice that, without Step 1, the system does not know the color characteristics of the background regions, and hence it cannot determine which regions are foreground and which are background.

3.3 Automatic Object Discovery

As mentioned in the introduction, if multiple images are to be segmented, and if these images contain the same type of object (call them the target objects) that the user wants to segment, then it is possible to analyze the region characteristics that consistently occur across images. It is the

consistency that tells the foreground from the background apart. In this section, we will introduce such a method.

Topic models such as Probabilistic Latent Semantic Analysis (PLSA)[6], were originally used in the text understanding community for unsupervised topic discovery. In computer vision, topic models have been used to discover object classes, or *topics*, from a collection of unlabeled images. As a result, images can be categorized according to the topics they contain. In the context of unsupervised object detection, the object of interest and the background clutter are the two *topics*.

Visual words (or textons) [28] are vector quantized local appearance descriptors from patches. Objects can be represented as collections of visual words. We will discuss in more detail in Section 6.4 on how the visual words are generated. Following the notations used in the text understanding community, $w \in W = \{w_1, ..., w_{|W|}\}$ is the visual word associated with a patch, $z \in Z = \{z_{FG}, z_{BG}\}$ is a hidden variable that represents the *topic* (foreground or background) associated with a patch, and $d \in D = \{d_1, ..., d_{|D|}\}$ is the index of the image associated with a patch.

PLSA assumes the joint distribution of d, w,and z can be written as $P(d, w, z) = P(d)P(z|d)P(w|z)$. PLSA is known for its capability of handling polysemy: if a visual word w is observed in two images d_i and d_j, then the topic associated with that word can differ in d_i and d_j: $\arg\max P(z|d_i, w)$ can be different from $\arg\max P(z|d_j, w)$. In other words, PLSA allows a visual word to have different meanings in different images.

We augment the PLSA model in the following way. We introduce an extra variable r in the graph. This variable is directly associated with the α values in GrabCut (Section 3.2). The idea is to use the segmentation mask produced by GrabCut to guide the automatic object discovery method. This sounds like the opposite direction of what we are seeking for: we wanted to use automatic object discovery to provide seeds for initializing the color distributions in GrabCut. But as we will see later, automatic object discovery and GrabCut are intimately connected, each feeding information to the other.

Figure 3.1 shows our proposed graphical model. The outer plate indicates the graph is replicated for each image, and the inner plate indicates the replication of the graph for each patch. The topic variable z is hidden. The r value for each patch is obtained by looking up the segmentation mask: if this visual word corresponds to a patch that is segmented by GrabCut as foreground, then $r = 1$; otherwise, $r = 0$.

The segmentation mask is correlated with the hidden topic z (foreground or background). This

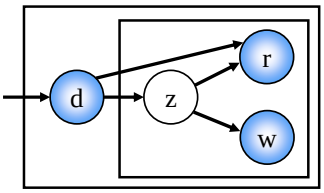


Figure 3.1: The proposed graphical model for automatic object discovery.

correlation is expressed in the graphical model as the link from z to r, or $P(r|z)$. We consider the r value of each patch as an additional feature r that is related to the hidden topic variable z. We learn the parameters using the EM algorithm:

$$\mathrm{E-step}:$$

$$P(z|d,r,w) = k_1 P(z|d)P(w|z)P(r|z) \tag{3.1}$$

$$\mathrm{M-step}:$$

$$P(w|z) = k_2 \sum_{d,r} m(d,w,r)P(z|d,r,w) \tag{3.2}$$

$$P(z|d) = k_3 \sum_{w,r} m(d,w,r)P(z|d,r,w) \tag{3.3}$$

$$P(r|z) = k_4 \sum_{w,d} m(d,w,r)P(z|d,r,w) \tag{3.4}$$

where $k_1, ..., k_4$ are normalization constants, and $m(d,w,r)$ is a co-occurrence matrix that counts the triples (d,w,r).

A typical distribution of $P(r|z)$ is shown in Figure 4.3. From this table we can see how the r value is correlated with z. The foreground topic z_{FG} strongly suggests that the GrabCut segmentation mask is also foreground at the corresponding position, while the ambiguity of the background topic is higher and does not as often correspond to GrabCut's background. The automatic object discovery method is doing inference based on the co-occurrences of the visual words across images (as PLSA does), *and* also based on the segmentation mask returned by GrabCut. The EM algorithm figures out from data how to optimally make judgements from these two "sensors": the GrabCut sensor (which provides $\{r\}$) and the appearance sensor (which provides $\{w\}$).

	r=0	r=1
z_{FG}	0.29	0.96
z_{BG}	0.71	0.04

Figure 3.2: A typical $P(r|z)$.

3.4 Automatic Seeding

In this section, we will use automatic object discovery to assign seeds without any user interaction. Here are the steps:

1. Automatic object discovery determines the topic of each visual word. In the first iteration, we do not know yet which topic corresponds to the foreground object. Use the topic whose positions of visual words has smaller variance as foreground. These visual words are called the foreground visual words.
2. Find a bounding box for each image: The location and scale of the box are the median and four times the standard deviation of the coordinates of the foreground visual words. We use the median as it is more robust to outliers than the mean.
3. Run GrabCut, except that using the computed bounding box instead of using user input. Get segmentation mask $\underline{\alpha}$ for each image.
4. Use segmentation mask to update the r values in automatic object discovery.
5. Repeat from Step 1 until convergence.

   We found this iterative algorithm typically converges in three or four iterations.

Notice that, during automatic object discovery, information is flowing across all images because all images share the same $P(w|z)$ and $P(r|z)$ distributions, whereas during GrabCut, segmentation is done only locally within each image.

3.5 Experiments

We use the Caltech face data set [29] to illustrate the process and results of our method. The method is general and can be applied to other object types as well. We randomly sample twenty images

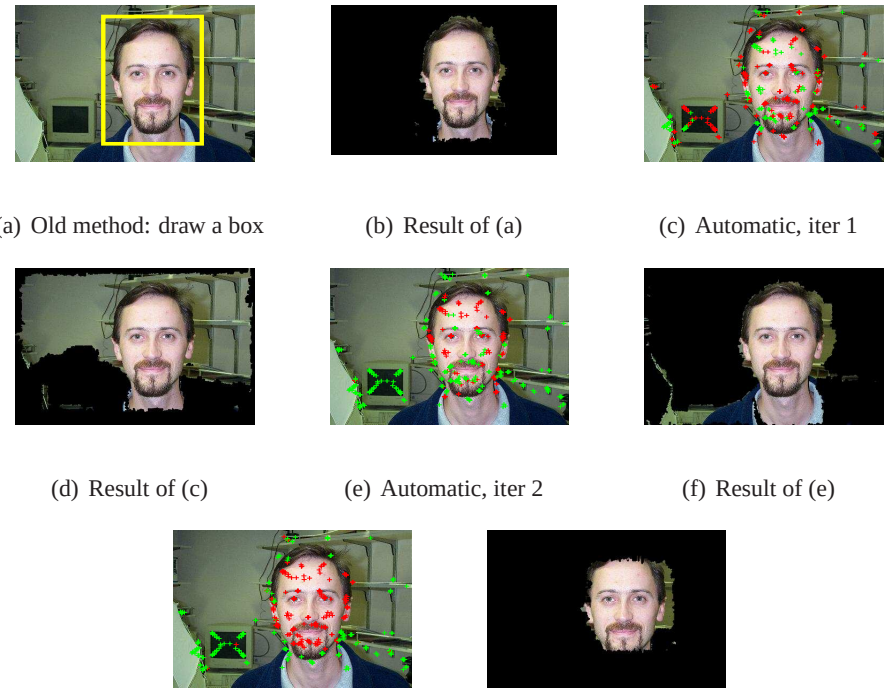
from the Caltech face data set, and resize them to 448×296 pixels.

In GrabCut we use patches found by the Watershed transformation [30] instead of raw pixels as basic units. This speeds up the processing. Using raw pixels can provide better segmentation, while the method stays the same.

In automatic object discovery the basic units are the visual words, which are created as follows. First, elliptical patches are detected by the Hessian Affine interest point detector [1]. We use a codebook size of 500 for quantizing the SIFT descriptors [2] of these patches into visual words. It is worth mentioning here that the SIFT descriptors, and hence the visual words, carry texture information, while the patches used in GrabCut carry color information. Hence our automatic background cutout method is utilizing information from both types of features.

Figure 3.3 demonstrates results. Figure 3.3(b) is the result of interactive GrabCut, which requires the user to specify a bounding box as shown in Figure 3.3(a). Notice that we use Watershed segmented patches instead of raw pixels for speed up, hence the result does not closely follow the object contour, but raw pixels could certainly be used instead. Figure 3.3(c) to (h) shows the result of our automatic method. The red and green crosses in (c)(e)(g) are the foreground and background visual words. Based on the foreground visual words (the red ones), a bounding box is calculated (not shown). Our method produces the results (d)(f)(h) after the first, second, and third iteration, respectively. The automatic result in Figure 3.3(h) is comparable to the interactive approach in Figure 3.3(b).

One of the reasons the result is not perfect is due to the way we calculate the bounding box. Our experience with GrabCut is that the result is quite sensitive to the preciseness of the bounding box, in terms of how close the box covers the object. If too much background is included, then the segmentation is rough. On the other hand, if the box is too small, then some part of the object will be cutout. For example, comparing Figure 3.3(e) and (g), since the visual words inside the human face are labeled more correctly in (g), the bounding box is tighter in (g) than in (e), resulting in the better results in (h) than in (f). Additional results are shown in Figure 3.4. We currently compute the bounding box using median and variance, but more sophisticated robust statistics might provide better boxes.



(a) Old method: draw a box (b) Result of (a) (c) Automatic, iter 1

(d) Result of (c) (e) Automatic, iter 2 (f) Result of (e)

(g) Automatic, iter 3 (h) Result of (g)

Figure 3.3: See Section 6.4 for details.

3.6 Conclusion

First, we have shown how background cutout can be achieved with zero user labeling. This is in contrast to current methods, where the user needs to label each image separately. We have shown that, by using the estimated foreground-background visual words in the automatic object discovery method, a bounding box can be automatically computed and used to initialize GrabCut. In return, the foreground-background segmentation mask of GrabCut can be used to update the features in automatic object discovery, and hence refining the foreground-background labeling of visual words in the next iteration. Second, our method integrates texture and color information in a novel way: automatic object discovery uses visual words, which captures texture around interest points; GrabCut uses color from patches found by Watershed transformation. Compared with previous automatic object discovery methods that only operate on sparse interest points [31], our method provides finer segmentation.

Figure 3.4: Results.

Chapter 4

Utilizing Correspondence Information

We propose a combination of a topic model approach and correspondence-based approach, and show significant improvements over current topic models that model the location of patches. Our approach is advantageous over existing topic model approaches due to a non-parametric representation of the object's spatial configuration. With this non-parametric representation, we can obtain better estimates of the labels for each patch, thereby achieving significant better localization and categorization results.

4.1 Reward by Correspondence

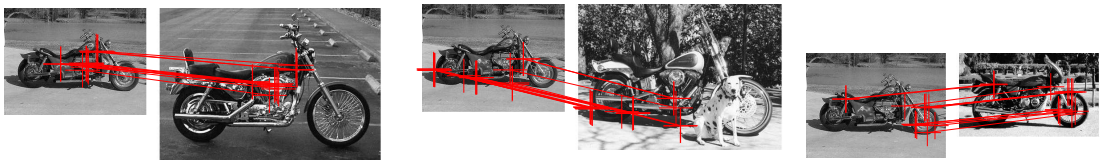


Figure 4.1: An image and three exemplar images. Red lines indicate correspondences found between patches.

In the introductory chapter we mentioned approaches **A2** and **A3**, which make use of spatial information, but do not explicitly consider the spatial configuration of patches coming from the object of interest. Normally the patches from the object are far more consistent than patches coming from background clutter. In the context of unsupervised object detection, it is the consistency of patches across images that tells an object apart from the background clutter: similar objects that

appear repeatedly in the data set will demonstrate a consistent spatial configuration, while the patches from random background clutter lack consistent spatial configuration. Similarly, in the context of image categorization, same object classes share similar spatial configurations, which are distinct for every object class.

Our intent is to extend the topic model framework to incorporate information about spatial configuration. Rather than building a shape model as in constellation models [14], pictorial structures [15], or fragment-based models [16], we will exploit the fact that similar objects in different images are more likely to have strong *correspondences* and extend the topic model to include this extra piece of information.

Correspondence-based object recognition has been in the literature for nearly forty years. Even though computing correspondence is computationally expensive, it is still popular, because of its promising performance. Recent work by [17][18] [19] [20] use correspondence as a central element in their object recognition framework. However, their model and learning algorithm differ substantially from what is proposed here.

Our use of correspondence is to provide a non-parametric representation of the location of the *consistent* patches. By using correspondence methods that take into account the spatial distortion of a correspondence and allow partial matching, the more matches a patch has, the better chances the patch belongs to a foreground object, as opposed to background clutter.

This piece of information is employed by the topic model in the form of an extra feature, or *reward*. More precisely, the reward for a patch is high when this patch is repeatedly matched against other patches in other images. On the other hand, patches from random clutter are less likely to find as many matches, which results in lower rewards. This is precisely the notion that objects of interest normally show higher consistency across images and it is the consistency that tells an object apart from background clutter or other objects. The more good matches a patch gets, the higher its reward, and the resulting reward map, as illustrated in Figure 4.2, is a non-parametric representation of the location of the consistent patches.

It is important to use correspondence methods that respect both appearance and geometric costs. Correspondence methods such as [18] [20] are among these. They measure the cost of a correspondence by observing how similar feature points are to their corresponding feature points, and how much the spatial arrangement of the feature points is changed. These methods also allow outliers to be excluded from the correspondence.

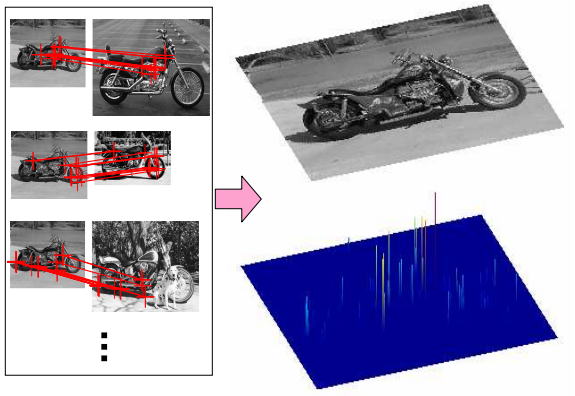


Figure 4.2: Correspondences between patches across images (red lines on the left) provide strong information of the object configuration. This information, shown at the bottom right as a "reward map", is incorporated into a topic model in our approach to enhance object localization and image categorization.

Computing correspondences between all pairs of N images in a data set is expensive, and to avoid the exponential complexity, we generate a list of C exemplars to correspond with each image. By narrowing down from N to C, we decrease the correspondence computation from N^2 times to NC times, which is linear to the number of images in the data set. This is in contrast to systems that need to run correspondence N^2 times for image categorization, such as in [19]. We generate the list of C exemplars by running PLSA and choosing the top ranked images from each topic. The images are ranked according to $P(d|z)$. In the experiments, we use 10 exemplars per topic. See Figure 4.1 for examples where an image is matched to some exemplar images.

4.1.1 Finding Correspondences

We want to find the correspondence between patches across two images, d_i and d_j, that are appearance-wise and geometrically consistent. Suppose there are n patches in d_i. It would be naive to find the single best match based on appearance and would not give a geometrically consistent correspondence. Instead, we use the correspondence algorithm in [20] to find out the appearance and geometric consistent matches.

We first find the k-nearest neighbors for each patch based on appearance, with k large enough so that only appearance-wise very disagreeing matches are excluded. Suppose candidate match a

marries patch p in d_i and patch p' in d_j, written in shorthand as $a = \langle p, p' \rangle$. The appearance affinity of the candidate match a, denoted by $A(a)$, is calculated as the correlation coefficient of the feature descriptors of patch p and patch p'.

Suppose $b = \langle q, q' \rangle$. Then the geometric affinity is defined as:

$$G(a,b) \equiv \max(0, 1 - C_d \cdot \frac{\|(\overrightarrow{pp'} - \overrightarrow{qq'})\|}{\sqrt{\|\overrightarrow{pp'}\|\|\overrightarrow{qq'}\|}}) \tag{4.1}$$

$G(a,b)$ is dimensionless so it does not change when the two vectors $\overrightarrow{pp'}$ and $\overrightarrow{qq'}$ are multiplied by a constant. C_d controls how tolerant this metric is to distortion.

The final affinity matrix M has elements $M(a,b) = G(a,b)A(a)A(b)$, where $a = \langle p, p' \rangle$ and $b = \langle q, q' \rangle$. Pairs of candidate matches will have low affinity $M(a,b)$ if either the geometric affinity or one of the appearance affinities is low. The correspondence algorithm we adopt from [20] figures out the final geometrically and appearance-wise consistent matches based on the dominant eigenvector of the affinity matrix M. Partial matching is achieved by setting the parameter C_d in Equation 4.1 large enough (we use $C_d = 1.5$), so that candidate matches that potentially match by appearance but distort the correspondence too much are excluded from the final result.

4.1.2 The reward map

The geometrically and appearance-wise consistent correspondences that are found in the previous section tell us which patches often co-occur; it also ensures us that, when patches co-occur, they co-occur in a geometrically consistent manner. We count the number of matches each patch has and create a "reward map" (Figure 4.2). In the context of unsupervised object detection, we would expect patches from the object of interest to have more matches, thus the reward value is a good indicator of whether a patch belongs to the object of interest.

Using the reward map to locate the object of interest is often not good enough. As we will explain later in better detail, the reward map is indeed correlated with the topic variables z, but the correlation is not high enough to provide accurate estimation of the topic. In fact, the"quality" of the reward map depends on the number of exemplars ones uses. Unless we use a very large set of exemplars (which would be inefficient, because finding correspondences is at least linear to the number of exemplars), many patches from the foreground object will not have consistent matches in other images. This is because the intra-class variation of similar objects in the real world is often very large, and having appearance-wise and geometrically consistent matches is rare.

Instead of directly using the reward map to locate the object, we use the framework we introduced in the introductory chapter. We consider the reward value of each patch as an additional feature r that is related to the hidden topic variable z. Empirically, we found that using $P(r|z)$ performs better than using $P(r|d,z)$, probably because of fewer parameters and less overfitting. We learn the parameters using the EM algorithm:

$$\mathrm{E-step}:$$

$$P(z|d,r,w) = k_1 P(z|d)P(w|z)P(r|z) \tag{4.2}$$

$$\mathrm{M-step}:$$

$$P(w|z) = k_2 \sum_{d,r} m(d,w,r)P(z|d,r,w) \tag{4.3}$$

$$P(z|d) = k_3 \sum_{w,r} m(d,w,r)P(z|d,r,w) \tag{4.4}$$

$$P(r|z) = k_4 \sum_{w,d} m(d,w,r)P(z|d,r,w) \tag{4.5}$$

where $k_1, ..., k_4$ are normalization constants, and $m(d,w,r)$ is a co-occurrence matrix that keeps the counts of triples (d,w,r) [6].

A typical distribution of $P(r|z)$ is shown in Figure 4.3. Notice that we have quantized the reward values into 4 bins using k-means quantization. It is interesting to see that background topic z_{BG} has almost all its rewards concentrated at the first bin (because patches originated from background clutter often have very few matches), while the foreground topic z_{FG} has rewards distributed more evenly. Still, the first bin of both topics is highly concentrated, which implies that directly using reward values to tell foreground from background has a lot of ambiguity. This is still true if we quantize the rewards into a larger number of bins. Hence, instead of directly using the reward values for object detection, we integrate this correspondence-based information into a topic model. The EM algorithm will figure out from data how to make judgements from these two "sensors": the correspondence sensor (which provides rewards, $\{r\}$) and the appearance sensor (which provides visual words, $\{w\}$). This integration allows us to use a very small number of exemplars. Even so, the performance significantly improves over traditional topic models, **A1** to **A3**, as we will show later.

	r_1	r_2	r_3	r_4
z_{FG}	0.36	0.37	0.20	0.07
z_{BG}	0.97	0.02	0.01	≈0.00

Figure 4.3: A typical $P(r|z)$.

4.1.3 Remarks

By using correspondences, we introduce spatial configuration into topic models. We don't make assumptions and postulate a model for the shape of the object. Neither do we make assumptions as in **A2** and **A3**, about the location and clustering of the object. These are all implicitly taken into account by using the reward map. It will be of future interest to combine A2 and A3 together in this correspondence-based topic model framework.

Here is a summary of the advantages of this framework:

1. The nice property of PLSA (namely, handling polysemy) is inherent in the new method.
2. PLSA can only handle polysemy across documents but not within documents: the same visual word w_j can be assigned to different topics in different images (context dependency) but a visual word within an image will always be assigned the same topic, regardless of its spatial relationship with other patches. In **A2**, **A3**, and our proposed method, the additional feature r allows the topic model to handle polysemy within documents.
3. In situations where finding geometrically consistent matches is difficult (e.g. , when objects have large deformations), methods that purely rely on correspondence would fail. In this case, the reward map would turn out flat or erroneous. However, our method will learn this fact and rely on the appearance information instead of the reward map.
4. Topic models can take advantage of information from background clutter. Pure correspondence-based methods cannot.

4.2 Experiments

In the following experiments, we do not compare with **A2**, since it does not provide translation invariance, which is crucial in our experiments.

4.2.1 Small objects (synthetic data)

The number of patches sampled from the images is the most influential parameter governing recognition performance using bag-of-words representation [32]. Moreover, in topic-model approaches, unless the object of interest occupies a reasonably large proportion of the image, it will not have a sufficient number of detections to compete with patches from the background, meaning that the image is misclassified as background. To show that the proposed approach allows the foreground object to have much fewer patches than in other approaches, we perform the following experiment: the task is to detect dumbbells in cluttered scenes, and we gradually increase the amount of background clutter. In Table 4.1, we see the proposed method allows heavier background clutter than the other topic models.

	Classification			Localization		
	Proposed	A1	A3	Proposed	A1	A3
x1	**100**	83	88	**86**	74	81
x2	**88**	64	69	**78**	62	66
x5	**68**	52	54	**65**	53	52
x10	**62**	53	51	**56**	47	52

Table 4.1: Small objects experiment in Section 4.2.1. By increasing the number of background patches from one times the number of foreground patches to ten times, the performance of the proposed method drops far less drastically than the baseline methods.

4.2.2 Confusing background (synthetic data)

In this experiment, we show that under the presence of changes in location, scale, occlusion and deformation, topic models **A1** and **A3** perform very poorly. The task is to detect dumbbells in cluttered scenes. Synthetic images are generated by embedding nonlinear distortions (using pinch, punch, and perspective transforms with the software Paintshop) of the dumbbell in cluttered backgrounds (Figure 4.4). To confuse the appearance, we *insert patches from the object into the background*. Clearly, if appearance alone were used to classify the patches, there is no way to distinguish object from clutter, because it is the spatial configuration of the patches that tells the object from the clutter apart. This is demonstrated in Table 4.2(a). To demonstrate the effect of occlusion, we removed some patches from the dumbbell. To verify multiple instance detection, two or three dis-

(a)

Classification			Localization		
Proposed	A1	A3	Proposed	A1	A3
100	50	50	**96**	50	50

(b)

	Classification	Localization
	Proposed	Proposed
Size fixed	**100**	**96**
Size varied	**86**	**83**
Distortion	**91**	**86**
Occlusion	**84**	**84**
Combination	**81**	**79**
Multiple	**100**	**92**

Table 4.2: Classification and localization accuracy (%) for Section 4.2.2. (a) A1 and A3 perform poorly because the data is ambiguous in appearance. (b) The proposed method still works well under different kinds of distortions.

torted versions of the dumbbell were embedded randomly in the scene. These variations are shown in Table 4.2(b). Note that the number of objects in the scene is unknown beforehand and the same parameters are used throughout these experiments.

4.2.3 Unsupervised categorization and localization

Patches are detected by the Hessian Affine interest point detector. We use a codebook size of 500 for quantizing the SIFT descriptors into visual words. The SIFT descriptors are then projected from 128 to 30-dimension using Principal Component Analysis.

	Equal Error Rate			Area under ROC		
	Proposed	A1	A3	Proposed	A1	A3
Motorbike	**1.9**	12.5	3.3	**99.8**	91.2	99.5
Airplane	**3.8**	10.2	13.4	**99.1**	95.7	92.6
Face	**2.0**	5.1	2.3	**99.5**	96.0	98.9
Car Rear	**8.6**	18.8	22.3	**92.3**	88.3	88.0

Table 4.3: Image-level classification results (%).

Table 4.3 shows the receiver-operating characteristic (ROC) equal error rates (EER) and the area under ROC curve. Clearly, methods that consider spatial information outperform **A1** (PLSA).

Figure 4.5 has further analysis on the motorbike data set.

Figure 6.2 shows the localization performance. Scores are computed based on the posterior probability $P(z|d,w)$ in **A1**, and based on $P(z|d,w,r)$ in **A3** and in **Proposed**. The proposed method shows significantly better performance in localization. Figure 4.7 shows some localization results. The top 15 highest scoring patches are indicated by the yellow ellipses. Notice that the proposed method has much less false alarms than **A1** (PLSA) and **A3**.

4.3 Conclusion

We have shown how topic models can benefit from finding correspondences and using the "reward map" as an additional feature. We have also shown that traditional topic model approaches can perform poorly when the appearance of the background clutter is extremely confusing. Our method is more robust than traditional topic models when the amount of background clutter is significantly larger than the number of patches from the object of interest, which is a major problem when applying traditional topic models to images.

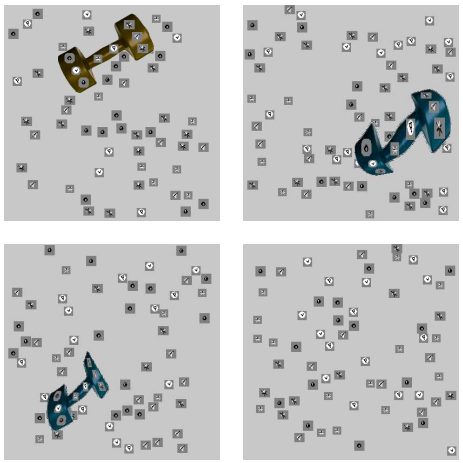


Figure 4.4: Synthetic images for Section 4.2.2. Topic models A1 and A3 cannot distinguish between the object and the background because it is the spatial configuration of the patches that tells the object from the background apart. See performance in Table 4.2.

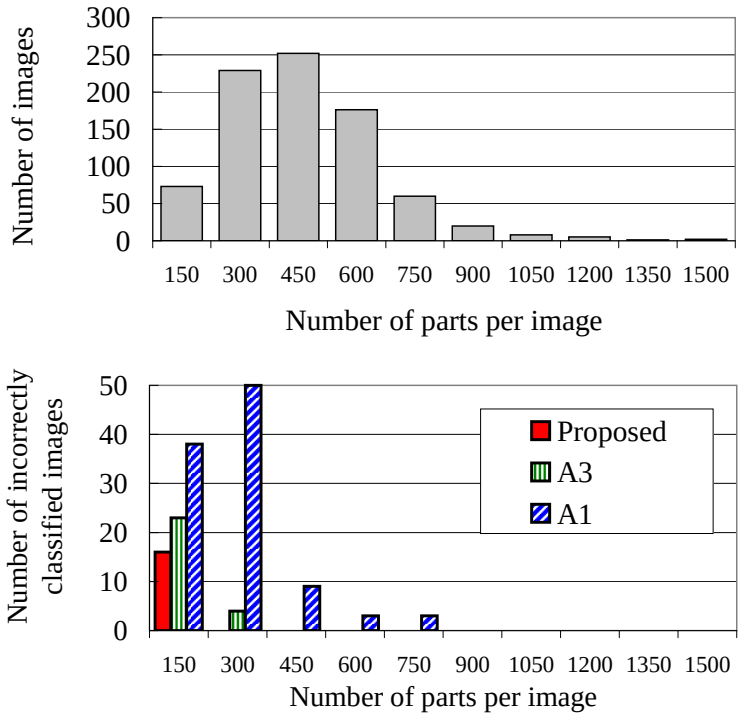


Figure 4.5: Classification of Caltech motorbike versus Caltech background images. The top figure shows that, among the motorbike images, most images have around 300 to 600 patches (foreground plus background). The bottom figure shows the number of wrongly classified motorbike images with respect to the number of patches in the image. Together, we see that the proposed method performs better than **A1** and **A3**. We also see that images with fewer patches are more often classified incorrectly. The proposed method classifies incorrectly 16 out of 826 motorbike images, all of them having less than 150 patches in the image.

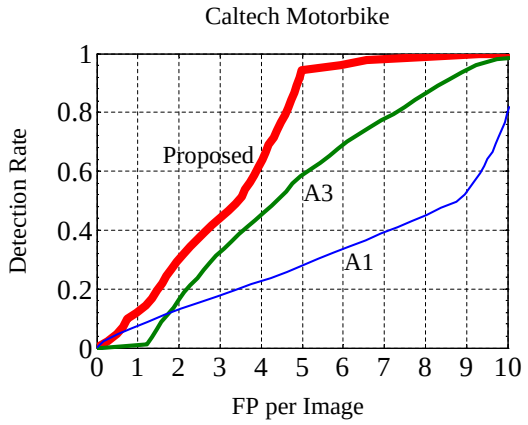
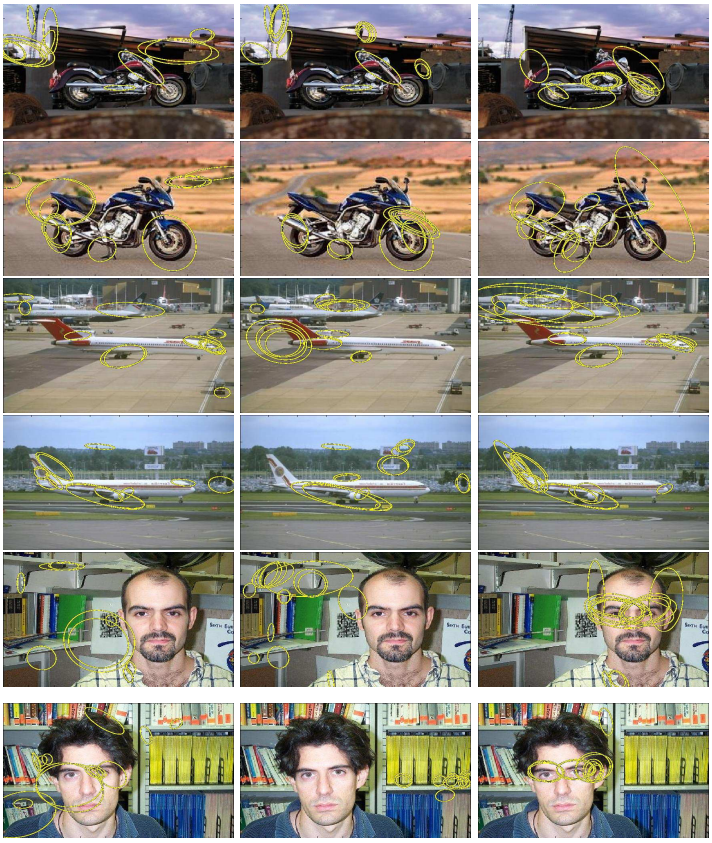


Figure 4.6: Patch-level classification result on Caltech motorbike data set. For each method, its top 20% confident patches are classified as foreground versus background; the closer the posterior probability $P(z|d,r,w)$ (or $P(z|d,w)$ in **A1**) of a patch is to 0 or 1, the higher the confidence of the patch.



(a) **A1** (b) **A3** (c) **Proposed**

Figure 4.7: Localization results.

Chapter 5

Handling Multiple Objects

We propose an iterative method for discovering objects in images. In each iteration, the current estimate of the layout is processed by a sequence of perceptual grouping rules. Perceptual grouping appears to be the basis of visual organization of human. It is concerned with the problem of the formation of wholes from parts. The method does not rely on the mixture of Gaussian model and hence avoids the model selection problem. We use synthetic and real images to demonstrate that the obtained result is better than that obtained by other methods.

5.1 Introduction

In the field of scene understanding, an image can be considered as a set of objects arranged in a spatial layout. A central problem is that of segmenting the image into objects. In order to extract the underlying objects, one can use image segmentation or layout extraction methods. Image segmentation methods partition an image into regions that consist of similar color, texture, or position. This partitioning often operates on a single image. In comparison, image layout extraction partitions an image by objects and uses information from multiple images.

The method we propose has its root in an unsupervised learning method called Probabilistic Latent Semantic Analysis (PLSA)[6]. This model has earlier been used in the text and linguistic domains. PLSA is a generative model, and can be used to interpret how a document is generated. A document is considered as a mixture of “aspects” (“topics”), and each aspect consists of a mixture of words. The power of PLSA originates from the fact that aspects can be learned in an unsupervised manner given a set of document-word pairs. It is hence capable of accumulating information

from multiple documents.

One important drawback of PLSA is that the set of document-word pairs ignores the layout or order of words in a document. PLSA as a generative model is hence generating a document without structure. In other words, if we arbitrarily shuffle the words in the document around, we get the same latent aspects (topics). As a result, the performance of PLSA still leaves room for improvement. Sivic et al. [4] add spatial information into the feature representation and leave the PLSA model unchanged. Liu and Chen [13] extend the PLSA model and explicitly model spatial structure.

However, using a Gaussian or mixture of Gaussians to model the spatial distribution of the objects in each image is not ideal for several reasons. First, the Gaussian distribution decays quickly around the peak and has long tails. Real world objects have uniform nonzero support inside the object boundary and zero support outside of it, and hence the Gaussian distribution does not model the spatial distribution genuinely. Second, since the number of objects varies from image to image, a Gaussian mixture model requires the specification of the number of mixture components per image. In this case, model selection methods such as the Bayesian information criteria (BIC) or the Akaike information criteria (AIC) require not only looping over the possible number of mixtures components, but also over each image, hence being computationally expensive.

We propose an iterative layout extraction method that is based on perceptual grouping [33]. This method does not rely on the mixture of Gaussian model and hence avoids the aforementioned problems. Perceptual grouping appears to be the basis of visual organization of human. It is responsible for the formation of wholes from parts. In psychology, generic principles such as proximity, similarity, closure, simplicity, etc., have been identified for the grouping process. Of special interest is proximity, which states that elements which are close to each other will be grouped together. In each iteration, after we obtain an updated estimate of the layout, we process the layout by a sequence of perceptual grouping rules that agree with human perception.

In the following, we will first describe the probabilistic model. We then describe the perceptual grouping method for refining the posterior map to obtain the spatial distribution. Finally, we present experimental results.

5.2 Probabilistic Model

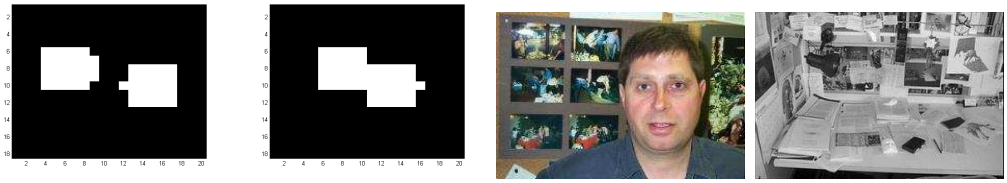
As mentioned in the introduction, if multiple images contain the same type of object, then it is possible to analyze the region characteristics that consistently occur across images and thereby segment the object. It is the consistency that tells the foreground from the background apart. In this section, we will introduce such a method.

Images are represented as collections of visual words (or textons) [28], which are discrete (vector quantized) local appearance descriptors extracted from image patches. We will discuss in more detail in Section 6.4 on how the visual words are generated. Following the notations used in the text understanding community, $w \in W = \{w_1, ..., w_{|W|}\}$ is the visual word associated with a patch, $z \in Z = \{z_{FG}, z_{BG}\}$ is a hidden variable that represents the *topic* (foreground or background) associated with a patch, and $d \in D = \{d_1, ..., d_{|D|}\}$ is the index of the image associated with a patch.

Each image is considered as a mixture of topics: $P(z_k|d_i)$ is the probability of topic z_k occurring in image d_i. Assume there are a predefined number of Z latent topics, $\{z_1, ..., z_Z\}$. Using inference methods, it is then possible to infer the latent variables of the model. In our experiments we will consider the case of $Z = 2$, but extending it to more topics is feasible. Each topic is further considered as a mixture of words: $P(w_j|z_k)$ is the probability of word w_j occurring in topic z_k. We denote W as the total number of (visual) words, $\{w_1, ..., w_W\}$.

We introduce the spatial distribution $P(x_p^{d_i}|d_i, z)$. The dependence of position $x_p^{d_i}$ on image d_i allows the foreground object to have different locations and scales in every image d_i. In other words, the foreground object is allowed to be at vastly different positions in both learning and testing. The scale can also vary freely over different images. Allowing different locations and scales in different images is desirable, as it provides the basis for scale-invariance and translation-invariance. The dependence of $x_p^{d_i}$ on z allows the foreground object and the background clutter to have different locations and scales. The superscript d_i is to indicate that the patch positions can be different for each image, and is omitted in the sequel to simplify the notation.

We define an image-word-position co-occurrence table $n(\ d, w, x)$, with $n(d_i, w_j, x_p^{d_i})$ denoting the number of occurrences of word w_j at position $x_p^{d_i} \in \{x_1^{d_i}, ..., x_{|d_i|}^{d_i}\}$ in image d_i, where $|d_i|$ denotes the number of patches in image d_i. In other words, $n(d_i, w_j, x_p^{d_i}) = 1$ if in image d_i there is a word w_j at position $x_p^{d_i}$, and $n(d_i, w_j, x_p^{d_i}) = 0$ otherwise.



(a) Synthetic (b) Synthetic (c) Face (d) Background

Figure 5.1: Synthetic and real objects used in the experiments.

The joint distribution of document, topics, position, and words is formulated as

$$P(d_i, w_j, z_k, x_p) = P(w_j|z_k)P(z_k|d_i)p(x_p|d_k, z_k)P(d_i). \tag{5.1}$$

5.3 Parameter Learning

The goal is to maximize the log-likelihood,

$$\mathcal{L} = \sum_i \sum_j \sum_p n(d_i, w_j, x_p^{d_i}) \log P(d_i, w_j, x_p^{d_i}) \tag{5.2}$$

This can be achieved by the EM algorithm [21], which alternates between the E-step and the M-step. The E-step updates the posterior distribution as follows:

$$P(z_k|d_i, w_j, x_p) \propto P(z_k|d_i)P(w_j|z_k)P(x_p|z_k, d_i) \tag{5.3}$$

The M-step updates the individual conditional distributions. The update equations of the topic distribution, $P(z_k|d_i)$, and the appearance distribution, $P(w_j|z_k)$, can be derived by the Lagrange multiplier technique as in [6] and they are as follows:

$$P(w_j|z_k) \propto \sum_i \sum_p n_{ijp} P(z_k|d_i, w_j, x_p) \tag{5.4}$$

$$P(z_k|d_i) \propto \sum_j \sum_p n_{ijp} P(z_k|d_i^s, w_j, x_p) \tag{5.5}$$

where $n_{ijp} \equiv n(d_i, w_j, x_p)$. The left hand side of these equations are normalized to become probability distributions.

A standard EM algorithm would attempt to solve the update equation for the spatial distribution as follows:

$$\theta^* = \arg\max_\theta \sum_{d,w,x,z} n(d, w, x) p(z|d, w, x) \log p_\theta(x|d, z) \tag{5.6}$$

where we use $p_\theta(x|d,z)$ to emphasize that the distribution $p(x|d,z)$ is parameterized by θ. In the Semantic-shift algorithm [13], the spatial distribution $p(x|d,z)$ is assumed to be a Gaussian distribution and an update equation can be derived from there. During the EM iterations of the Semantic-shift algorithm, we observed the spatial clustering of a single bump to occur due to the strong prior assumption that one and only one bump exists in the image pixel domain. Since images can contain multiple objects, a natural way to extend the method is to allow multiple bumps to exist in an image. Gaussian mixture models are often used to describe multi-modal data. However, specifying the number of mixture components is nontrivial since each image can have different number of objects, and hence model selection using the BIC or AIC criteria requires not only looping the number of components, but also over all images. This is computationally very expensive.

Motivated by the spatial clustering behavior in the Semantic-shift algorithm, where nearby patches with high posterior probability are grouped together and noisy patches are gradually suppressed, we consider the process of finding the optimal spatial distribution as a perceptual grouping process. In psychology, several generic principles have been identified to account for the human visual grouping process. These principles include proximity, similarity, closure, simplicity, etc.. However, to the best of our knowledge, there is no mathematically model that describes all Gestalt principles under one framework. We therefore propose to use image processing techniques to realize the spatial clustering behavior as that observed in the Semantic-shift algorithm, thereby avoiding making unrealistic assumptions about the spatial distribution such as mixture of Gaussians that don't fit the true layout well.

In this work, we use a non-parametric representation for the spatial distribution. Denote the distribution parameters of $p(x|d,z)$ by θ. Specifically, θ contains the values of $p(x|d,z)$ at every x, for each d and z. This is equivalent to storing a two-dimensional 'posterior map' for each image and topic, which contains the value of the posterior probability for each image patch. Within the EM iterations, we aim at removing small elements and simultaneously group elements that are close to each other. We use simple morphological image processing to achieve this result. We first erode the posterior map, then dilate it. This can be understood as a realization of the proximity principle, which states that elements which are close to each other will be grouped together. The perceptual grouping process does not need to achieve a clean segmentation within the current iteration. Instead, since the perceptual grouping process occurs repeatedly during the EM algorithm iterations, a suboptimal grouping step can result in significant changes in the posterior

estimation in the next iteration. We will illustrate this effect in the experiments.

In literature, there are many attempts to simulate the human capability of perceptual grouping, e.g. in [34]. We suggest that methods such as these could be incorporated into the estimation of the spatial distribution as well.

Another alternative is to use a Markov Random Field (MRF) to model the spatial distribution. In that case, we could have associated the hidden states with the hidden variables z. The smoothness prior can roughly achieve the proximity principle. However, finding the solution of a two-dimensional MRF is an NP-hard problem. Hence, even though mathematically the problem is well defined, the approach is not always elegant. On the other hand, using morphological processing or image filtering techniques, we can intuitively incorporate the Gestalt principles into the estimation procedure.

5.4 Experiments

5.4.1 Synthetic images

We created 20 synthetic images of size 20×18, each containing one or two objects. Objects consist of visual words uniformly sampled from 13 visual words. Background consists of visual words uniformly sampled from 10 visual words. In total there are 20 distinct visual words, so the objects and background share 3 visual words. Each pixel corresponds to a visual word, so there are 360 'patches' per image. Examples of the layout of the objects are shown in Fig. 5.1(a)(b).

During each EM iteration, the 'posterior map' is estimated; examples are shown in the first column in Fig. 5.2. The posterior map has many smaller groups of pixels which appear to be noise and larger groups of pixels that appears to be the foreground object. After erosion and dilation, we obtain the result in the second column in Fig. 5.2. Notice that many smaller groups of patches with higher posterior probability are suppressed, and larger groups are clustered. These cleaned up results are the estimated spatial distribution $p(x|d, z)$ and are passed into the E-step in the next EM iteration. The algorithm converges after 20 iterations. The most likely latent aspect of each visual word can be computed by

$$z^* = \arg\max_z P(z|d_i, w_j, x_p) \tag{5.7}$$

Over 99% of patches are correctly identified as foreground vs background. This result cannot be achieved by the Semantic-shift algorithm (results not shown) using a single Gaussian assumption.

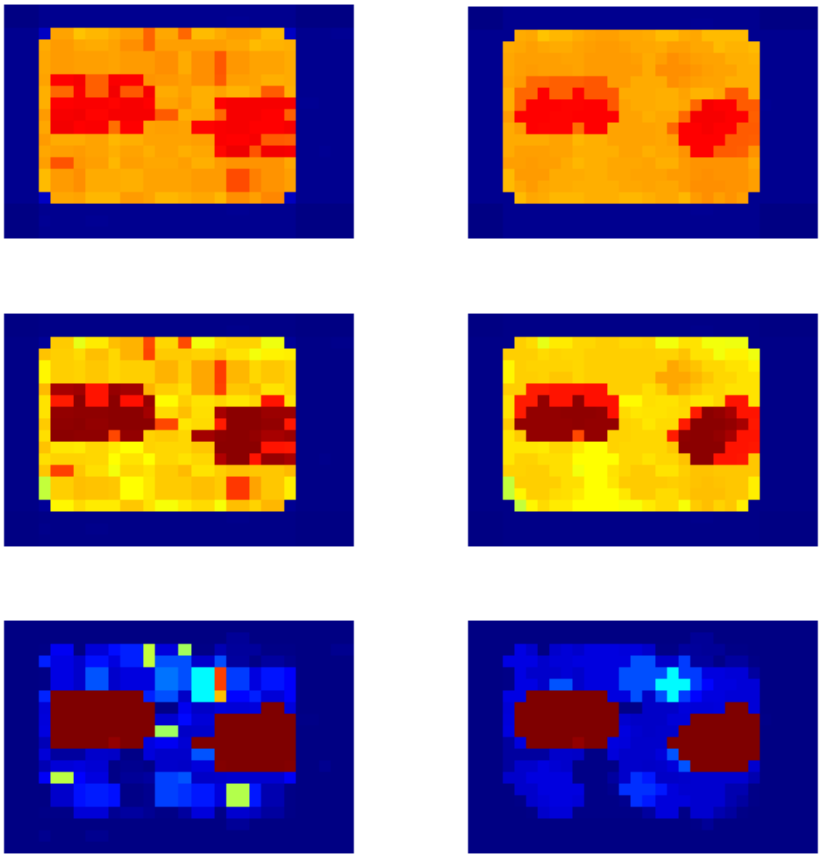


Figure 5.2: From top to bottom: the posterior map (left figure) and the perceptual grouping result (right figure) at iteration 3, 4 and 9. Viewed in color.

Also, as mentioned earlier, model selection becomes an issue if using multiple Gaussian distributions. On the other hand, considering the spatial distribution estimation as a perceptual grouping process, one can easily apply erosion, dilation, edge-preserving smoothing, or other techniques during the spatial distribution estimation.

5.4.2 Real images

We use 50 face images and 50 background images from the Caltech image dataset, and another 15 face images from the CMU face dataset. Some of the face images contain two or more faces. All images are converted to grayscale and resized to 300×200. We use the Harris-Laplace interest

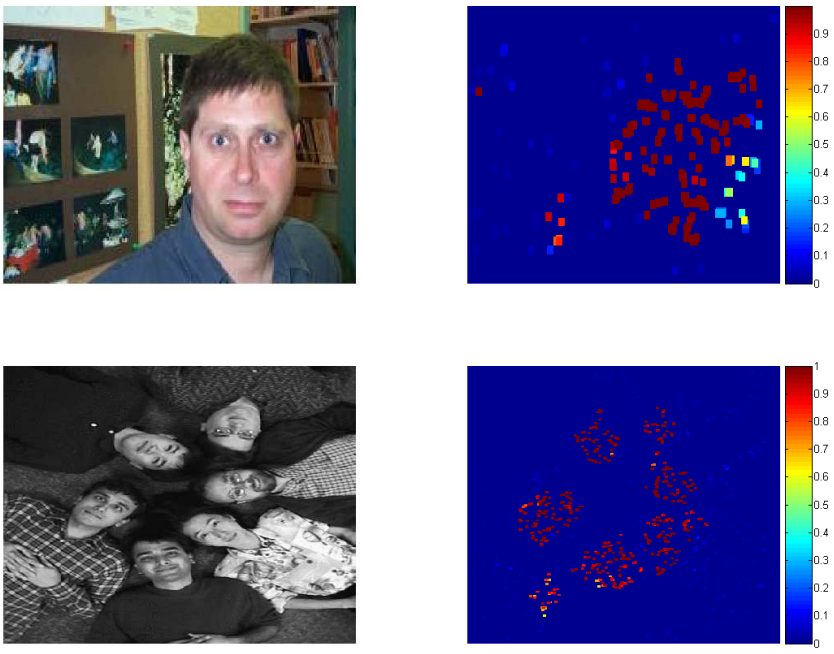


Figure 5.3: Posterior map of some face images at iteration 20. Viewed in color.

point detector to obtain around 200 to 2000 interest regions per image. Since the interest points cover the image only sparsely, we downsample the images to 30×20 before performing image erosion and dilation. The algorithm converges at around 30 iterations.

In Fig. 5.3 we show the 'posterior map' at the final iteration. Due to the non-parametric representation, the spatial distribution can easily accommodate an arbitrary number of foreground objects, such as the multiple faces in the bottom of Fig. 5.3.

Since the image dataset consists of face and non-face images, we can consider a (unsupervised) classification task. Comparing the proposed method with the Semantic-shift algorithm, we obtain the ROC curve shown in Fig. 5.4, where the upper curve is the proposed approach. For fair comparison, since Semantic-shift was originally designed for a single foreground object, we did not use the 15 images that contain more than one face.

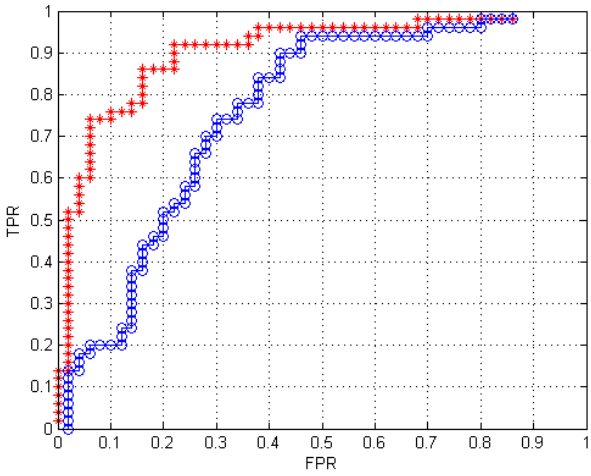


Figure 5.4: ROC curve for face vs non-face classification.

5.5 Conclusion

We proposed an approach for object discovery that incorporates perceptual grouping into the EM algorithm. This method can achieve similar spatial clustering behavior as the Semantic-shift algorithm, and yet it is not confined to the problem of model selection. Also, it allows easy incorporation of other image processing techniques into the loop. The method also achieves better performance on an image classification task.

Part III

Discovering objects in videos

Chapter 6

Utilizing Temporal Information

Here we present a probabilistic framework for discovering objects in video. The video can switch between different shots, the unknown objects can leave or enter the scene at multiple times, and the background can be cluttered. The framework consists of an appearance model and a motion model. The appearance model exploits the consistency of object parts in appearance across frames. We use Maximally Stable Extremal Regions as observations in the model and hence provide robustness to object variations in scale, lighting and viewpoint. The appearance model provides location and scale estimates of the unknown objects through a compact probabilistic representation. The compact representation contains knowledge of the scene at the object level, thus allowing us to augment it with motion information using a motion model. This framework can be applied to a wide range of different videos and object types, and provides a basis for higher level video content analysis tasks. We present applications of video object discovery to video content analysis problems such as video segmentation and threading, and demonstrate superior performance to methods that exploit global image statistics and frequent itemset data mining techniques.

6.1 Introduction

Video object discovery is the task of extracting unknown objects from video. Given a video, we want to ask what is the object of interest in this sequence, without providing the system any examples. This is very different from object detection in the computer vision literature, see for example [35], where the characteristics of the object of interest are learned from labeled data. Object detection not only involves a lot of human labor for labeling the images by putting bounding boxes

on the object of interest, but also has the difficulty of scaling to multiple objects. Since the object of interest in a sequence can be any type of object, it is very difficult to train a comprehensive object detector that covers all types of objects. The state of the art multi-class object detector has a recognition rate only around $55 - 60\%$ for recognizing 101 pre-defined object categories [36] and requires over 3000 human labeled images.

Our approach to object discovery is unsupervised in nature. No labeled images are needed for training the system, and no examples are used for specifying the object of interest. A high level intuition of how this can be achieved is as follows: In a video, if there is a car appearing in multiple frames, we might be able to figure out that the wheels or the windows are smaller parts of a larger entity that repeats over and over again in different images. In this scenario, the concept of a larger entity that is composed of smaller parts emerges. The smaller parts that constitute the larger entity can be generic, and do not need to have semantic meanings such as wheels or windows in this example.

Our approach works well on small objects in low resolution video. The object of interest sometimes has as few as a single feature point out of over fifty background feature points. The system is designed for videos where only a single object of interest will be extracted. We consider this less of a limitation but more of an advantage. In many videos, even though there are multiple objects, there is only one object that is of main interest. Our proposed method is intended to discover this major object of interest.

DISCovering Objects in Video, or DISCOV, involves two processes:

1. At the image level, extracting salient patches that are robust to pose, scale and lighting variations, and are generic enough for dealing with different types of objects. These salient patches serve as candidate parts that constitute larger entities.
2. At the video level, constructing appearance and motion models of larger entities by exploiting their consistency across multiple frames.

6.2 Related work

One approach to video object discovery is to observe the same scene over a long time and build a color distribution model for each pixel [37][38][39]. Unusual objects can then be identified if some pixels observe substantial deviation from their long-term color distribution models. These

kind of background modeling approaches are suitable for video surveillance with a static camera, but if an image sequence is obtained from a moving camera, then a pixel does not correspond to a fixed scene position; unless we can accurately register the image sequence, we cannot build a color distribution for each pixel.

Some methods exploit optical flow to discover objects. Optical flow is the apparent motion between a pair of images. The problem is difficult because of a lack of constraints (the aperture problem) and insufficient sampling near occlusion boundaries [40]. Since optical flow computes local image gradients, it is best suited to successive pairs of frames, not to low frame rates with large motions [40]. Using such short duration flow field, in [41], the optical flow of each frame is clustered, providing initial estimates of object positions in each frame. In [42], frame to frame optical flow fields are concatenated to obtain longer range correspondences, providing information to determine if a motion is consistent in direction over time. This consistency is useful in rejecting distracting motions such as the scintillation of specularities on water, and the oscillation of vegetation in wind. Using such long range optical flow field, however, one must refine the field at each step to avoid drifting, as mentioned by [40].

While optical flow provides a dense but short range motion field, feature tracking using distinctive textured patches provides long range but sparse motion field. In [43], the correspondence of distinctive feature patches are found across successive frames and grouped according to their co-occurrences. Our approach also uses distinctive textured patches, but we do not explicitly compute the correspondences across frames, which can be computationally expensive.

Our work also differs from layer extraction methods [44][45] in which the frames in a video are partitioned into a number of regions, in each of which pixels share the same apparent motion model. In contrast, our approach allows for a very low frame rate, in which case methods relying on image registration (as in [44]) cannot register the background across frames. Our approach does not compute affine transformation between patches as in [45], which would have the same problem at low frame rates.

In [46][47], multiple object detectors of airplanes, buildings, people, etc. provide as input to the data mining algorithm a feature vector describing the presence and absence of each of these objects. However, as mentioned in Section 6.1, the state of the art 101-class object detectors has a recognition rate only around $55 - 60\%$ [36] and requires huge amount of training data, and hence these type of object recognition approaches have inherent difficulties. Some systems build

specialized video object detectors by using labeled data to train an object detector and then track its trajectory or exploit prior knowledge of the color distribution of the target, such as the human skin color distribution [48][49]. Some require track initialization (initial position of the target) and target appearance initialization [50] [51]. These approaches are not intended for object discovery, since they require prior knowledge of the appearance or position of objects.

As mentioned earlier, our approach works well on small objects in low resolution video, where the object of interest sometimes has as few as a single feature point out of over fifty background feature points. This is in contrast to methods that exploit a rich set of textures of the foreground object [52][53] [54].

In [55], a spatial scan statistic is applied to detect clusters in epidemiological and brain imaging data. In [56], the challenge is to find sets of points that conform to a given underlying model from within a dense, noisy set of observations. As in many spatial data mining methods [57], these methods focus on point patterns where the 'density' of the points conveys information. In our data, different appearance features are extracted from different image patches, and hence not only the density but also the identity of each atomic unit plays a role in object discovery.

Recently, topic models [6] have been applied to unsupervised object discovery in images [4][5][58][13] and videos [31][7]. We follow the approach of [31] and present applications including video segmentation and threading. In the image domain, we have an appearance model and a spatial model of patches. In the temporal domain, we use a motion and data association model that is tightly coupled with the appearance and spatial model. This framework yields a principled and efficient object discovery method where appearance is learnt simultaneously with motion in a completely unsupervised manner. The appearance model accounts for appearance variations and background clutter; the motion and data association model accounts for the randomness in the presence/absence of features due to appearance measurement noise. The features we use are simple spatial features demonstrating the generality of our system; more sophisticated spatial-temporal features [59][60] could certainly be used as well.

In Section 6.3, we will introduce the DISCOV framework. We will start from the image representation, which uses generic region detectors and descriptors. We then introduce the appearance and motion models, which provide an unsupervised method for discovering the object of interest in video sequences. In Section 6.4, we present experimental results and also present applications of video object discovery for video segmentation and threading.

6.3 The DISCOV framework

6.3.1 Representation of images

Visual words, or textons, are used as atomic units in our image representation. They were used in various applications, such as photometric stereo [61], object recognition [62], image retrieval [5], etc. Next we will discuss in detail how to generate visual words.

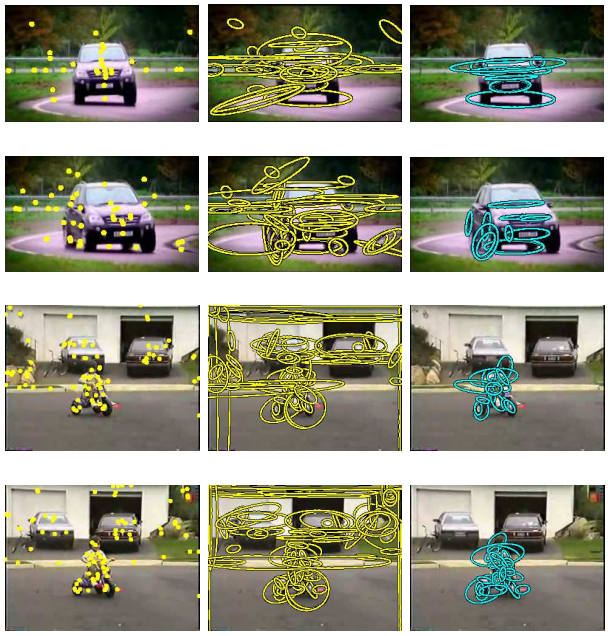


Figure 6.1: Maximally Stable Extremal Regions (MSERs). Left: position of MSERs. Middle: coverage of MSERs. Right: Output of DISCOV, showing the discovered object regions.

First, we find a number of patches to generate the visual words from. These patches are determined by running the Maximally Stable Extremal Regions (MSER) operator [63]. Examples are shown in Figure 6.1. MSERs are the parts of an image where local contrast is high. This operator is general enough to work on a wide range of different scenes and objects and is commonly used in stereo matching, object recognition, image retrieval, etc. as mentioned earlier. Other operators could also be used; see [1] for a collection. Features are then extracted from these MSERs by Scale Invariant Feature Transform (SIFT) [2], yielding a 128 dimensional local feature descriptor for each MSER. Whether or not to use color information is largely application dependent. If the data mining task is to discover all instances of a specific object category, such as all cars in a video, then

color information should not be used because the color can be different across different instances of the same category. On the other hand, if the data mining task is to discover a single object instance, then color information provides good discrimination against other objects in the video. Color information can also be useful when shape and grayscale texture are not discriminative enough. In this work we extract MSERs and SIFT descriptors from grayscale images; patches and features extracted from color images [3] can easily be used instead.

The 128 dimensional SIFT descriptors are collected from all images and vector quantized using k-means clustering [64]. The resulting J cluster centers (we use $J = 50$) form the dictionary of visual words, $\{w_1, ..., w_J\}$. Each MSER can then be represented by its closest visual word. MSERs are now represented by discrete visual words instead of continuous SIFT descriptors. Note that acquisition of visual words does not require any labeled data, which shows the unsupervised nature of this system.

6.3.2 Appearance and spatial modeling

Denote the image frames by $\{d_1, ..., d_N\}$ and define hidden variables $z_i(k)$ indicating if the i^{th} MSER in frame d_k originates from the object of interest or otherwise. We will refer to the MSERs that do not belong to the object of interest as background (bg) clutter. $\{z_i(k)\}$ are hidden variables; it is our goal to infer their values. Define the conditional probabilities $P(z|d)$ and $P(w|z)$ for each MSER as follows: $P(z = z_{obj}|d = d_i)$ indicates in frame d_i how likely a MSER originates from the object of interest; $P(z = z_{bg}|d = d_i)$ is defined likewise. $P(w = w_j|z = z_{obj})$ indicates how likely a MSER originated from the object of interest has appearance corresponding to visual word w_j; $P(w = w_j|z = z_{bg})$ is defined likewise.

Denote the position of the i^{th} MSER in frame d_k as $\mathbf{r}_i(k)$, and its hidden variable as $z_i(k)$, $i = 1, ..., m_k$. The index i are sometimes dropped to avoid cluttering equations. Define an image-word-position co-occurrence table $n(d, w, \mathbf{r})$, with $n(d_i, w_j, \mathbf{r}_i(k))$ denoting the number of occurrences of word w_j at position $\mathbf{r}_i(k)$ in frame d_i, where $|d_i|$ denotes the number of words in frame d_i. In other words, $n(d_i, w_j, \mathbf{r}_i(k)) = 1$ if in frame d_i there is a word w_j at position $\mathbf{r}_i(k)$, and $n(d_i, w_j, \mathbf{r}_i(k)) = 0$ otherwise.

We introduce the spatial distributions $p(\mathbf{r}|d, z_{obj})$ and $p(\mathbf{r}|d, z_{bg})$. They describe how the object of interest and the background clutter are spatially distributed in the image. The dependence of position $\mathbf{r}$ on frame d allows the object to have different locations and scales in every frame.

Allowing different locations and scales in different frames is desirable, as it provides the basis for translation and scale invariance. This concept has been used by the Semantic-Shift algorithm in [13]. However, in the next section we will constrain the object position to follow a motion model. Another important distinction with [13] is that, while in [13] a *foreground topic identification* step is required to correctly identify the hidden variable that corresponds to the object of interest, we found this step unnecessary. The foreground, i.e., the object of interest, can be automatically identified due to the un-symmetric nature of the distributions $p(\mathbf{r}|d, z_{obj})$ and $p(\mathbf{r}|d, z_{bg})$.

Assume the object of interest is located at image coordinate $\hat{\mathbf{r}}$ with horizontal and vertical scale $\hat{\sigma}_h$ and $\hat{\sigma}_v$. These estimates are related to the motion model to be detailed in the next section. The spatial distribution of the object of interest is defined as:

$$p(\mathbf{r}|d, z_{obj}) = k_2 \frac{1}{(\mathbf{r} - \hat{\mathbf{r}})^T \hat{\Sigma}^{-1} (\mathbf{r} - \hat{\mathbf{r}}) + k_1} \tag{6.1}$$

where $\hat{\Sigma}$ is a diagonal matrix with elements $\hat{\sigma}_h^2$ and $\hat{\sigma}_v^2$ which are related to the scale of the object. The values of $\hat{\sigma}_h^2$ and $\hat{\sigma}_v^2$ are unknown and yet to be estimated. Before we detail the parameter estimation procedure in Section 6.3.4, it is worth mentioning that the parameters of the appearance, spatial, and motion model (in Section 6.3.3) are estimated in an iterative manner, and it does not matter which of the models is initialized first. The use of the regularization constant k_1 (we use $k_1 = 1$) avoids numerical issues when $(\mathbf{r} - \hat{\mathbf{r}})^T \hat{\Sigma}^{-1} (\mathbf{r} - \hat{\mathbf{r}})$ approaches zero. The spatial distribution is a probability mass function and the constant k_2 is used to ensure its mass adds up to one. This is achieved by summing up $(\mathbf{r} - \hat{\mathbf{r}})^T \hat{\Sigma}^{-1} (\mathbf{r} - \hat{\mathbf{r}}) + k_1$ over all MSERs $\mathbf{r}_i$ in frame d.

The spatial distribution of the background clutter is simply defined as a uniform distribution. We found empirically our distributions perform better than those in [13], one reason being that their background spatial distribution requires parameter tuning, which is often difficult and data dependent.

Our probabilistic model that combines appearance, location, scale, and motion information is expressed by this joint probability distribution:

$$p(d, w, z, \mathbf{r}) = p(\mathbf{r}|d, z) P(w|z) P(z|d) P(d) \tag{6.2}$$

and it postulates the conditional independence of d and w given z, and hence provides a compact representation of the joint probability. It also provides a basis for efficiently finding the maximum likelihood estimates of the unknown appearance models $P(w|z)$, $P(z|d)$, $\hat{\sigma}_h^2$ and $\hat{\sigma}_v^2$, which we will detail later in Section 6.3.4.

6.3.3 Motion modeling

Motion modeling provides the location and scale estimates $\hat{\mathbf{r}}$, $\hat{\sigma}_h$ and $\hat{\sigma}_v$ used in the spatial distribution in (6.1). Define the state $\mathbf{s}(k)$ as the unknown position and velocity of the object to be discovered, where k is the video frame index. We assume a constant velocity motion model in the plane and the state evolves according to $\mathbf{s}(k+1) = \mathbf{F}\mathbf{s}(k) + \mathbf{v}(k)$, where $\mathbf{F}$ is the state matrix and the process noise sequence $\mathbf{v}(k)$ is white Gaussian with mean zero and constant covariance matrix [65].

Suppose at time k there are a number of m_k observations. Each observation $\mathbf{r}_i(k)$ is the position of an MSER. If an observation $\mathbf{r}_i(k)$ originates from the foreground object, then it can be expressed as $\mathbf{r}_i(k) = \mathbf{H}\mathbf{s}(k) + \mathbf{w}_i(k)$, where $\mathbf{H}$ is the output matrix [65], and the observation noise sequence $\mathbf{w}_i(k)$ is assumed white Gaussian with mean zero and constant covariance matrix. We do not build a motion model for the background clutter.

We want to establish the relationship between the observations and the states. Since we do not know beforehand if an observation is originated from the object of interest or from the background clutter, we have a data association problem [65]. The Probabilistic Data Association (PDA) filter [65] solves the data association problem by assigning each observation an association probability, which specifies by how much the observation deviates from the model's prediction. In the original PDA filter, the association probabilities are calculated based on deviation of observations from the predicted states, where the states consists of only position and velocity, and appearance is not utilized. Here instead, we use the posterior probability $p(z_{obj}|d, w, \mathbf{r})$ as association probability. The posterior probability can be calculated as follows:

$$p(z_{obj}|d, w, \mathbf{r}) = \frac{p(\mathbf{r}|d, z_{obj})P(w|z_{obj})P(z_{obj}|d)}{\sum_z p(\mathbf{r}|d, z)P(w|z)P(z|d)} \tag{6.3}$$

It naturally includes location information (through $p(\mathbf{r}|d, z_{obj})$) and appearance information (through $P(w|z_{obj})$). Then, the state estimate can be written as:

$$\hat{\mathbf{s}}(k|k) = \sum_{i=1}^{m_k} \hat{\mathbf{s}}_i(k|k)p(z_i(k)|d_k, w, \mathbf{r}_i(k)) \tag{6.4}$$

where $\hat{\mathbf{s}}_i(k|k)$ is the updated state estimate conditioned on the event that $\mathbf{r}_i(k)$ is originated from the foreground object. This is given by the Kalman Filter [65] as follows:

$$\hat{\mathbf{s}}_i(k|k) = \hat{\mathbf{s}}(k|k-1) + \mathbf{W}(k)\boldsymbol{\nu}_i(k) \tag{6.5}$$

where $\boldsymbol{\nu}_i(k) = \mathbf{r}_i(k) - \hat{\mathbf{r}}(k|k-1)$ is the innovation, $\hat{\mathbf{r}}(k|k-1)$ is the observation prediction, and $\mathbf{W}(k)$ is the Kalman gain [65]. The state estimation equations are the same as in the PDA filter [65].

We estimate $\hat{\sigma}_h$ and $\hat{\sigma}_v$ as the interquartile range [66] of all MSERs, weighted by their posterior probability. In implementation, we duplicate points in the image space according to the posterior probability, and then compute the interquartile range of these points. The interquartile range provides a more robust scale estimate [66] than the weighted standard deviation used in [13][31].

6.3.4 Maximum likelihood parameter estimation

The distributions $P(w|z)$, $P(z|d)$, and $P(d)$ of the appearance model are estimated using the Expectation-Maximization (EM) algorithm [21], which maximizes the log-likelihood

$$\mathcal{L} = \sum_k \sum_j \sum_i n(d_k, w_j, \mathbf{r}_i(k)) \log p(d_k, w_j, \mathbf{r}_i(k)) \tag{6.6}$$

The EM algorithm consists of two steps: the E-step computes the posterior probabilities for the hidden variables; the M-step maximizes the expected complete data likelihood:

$$\mathbf{E-step}:$$

$$p(z_i(k)|d_k, w_j, \mathbf{r}_i(k))$$

$$= c_1 P(z_i(k)|d_k) P(w_j|z_i(k)) p(\mathbf{r}_i(k)|z_i(k), d_k) \tag{6.7}$$

$$\mathbf{M-step}:$$

$$P(w_j|z_i(k)) = c_2 \sum_k \sum_i n_{kji} p(z_i(k)|d_k, w_j, \mathbf{r}_i(k)) \tag{6.8}$$

$$P(z_i(k)|d_k) = c_3 \sum_j \sum_i n_{kji} p(z_i(k)|d_k, w_j, \mathbf{r}_i(k)) \tag{6.9}$$

$$P(d_k) = c_4 \sum_j \sum_i n_{kji} \tag{6.10}$$

$$p(\mathbf{r}_i(k)|z_i(k), d_k) \text{ updated according to Section 6.3.3} \tag{6.11}$$

---

where $n_{kji} \equiv n(d_k, w_j, \mathbf{r}_i(k))$, and $c_1, ..., c_4$ are normalization constants which have values so that all functions are valid probability mass functions.

We see that the spatial distribution $p(\mathbf{r}_i(k)|z_i(k), d_k)$ is updated within each EM-iteration, which means that the temporal information enters the EM-iteration and influences the appearance estimation.

6.3.5 Initialization

To handle the case where the object may disappear from the scene and re-enter the scene, we re-initialize the motion model when the position of the object is estimated to go out the scene, or when the color histogram of the whole scene changes beyond a certain threshold. This implementation is particularly important for video sequences which are post-edited, so that the camera view changes between the object of interest and other objects.

The distributions $P(w_j|z)$, $P(z|d_k)$, and $P(d_k)$ are all initialized randomly. The spatial distribution parameters are initialized at the center of the frame with scale equal to half the size of the frame. The state estimate $\hat{\mathbf{s}}$ is initialized to the center of the frame and with zero velocity.

6.4 Empirical study

The experiments are conducted on several real-world data sets to validate our framework. Seven video sequences were downloaded from YouTube.com with resolution 320×240 and are sampled at one frame per second. Practical internet video analysis systems are expected to handle such low frame rate videos in order to keep up to speed with the vast amount of available online videos nowadays. We have tried downsampling the original videos into various frame rates and found that one frame per second is good enough to retain the content while providing good computational efficiency. Such low frame rate poses higher difficulty to the system, as object motion could be large and appearance changes could be significant. The duration of these videos range from 67 to 711 seconds, as shown in Table 6.1. In the video segmentation experiment (Sec. 6.4.2), the fraction of frames containing the object of interest ranges from 0.16 to 0.85, hence the videos represent a variety of different shooting styles. The average duration of a shot ranges from 3 to 5 frames in all videos except in BIKE. These videos hence contain a large number of shot transitions, posing difficulty to methods based on motion. In the localization experiments (Sec. 6.4.4) we included two extra videos that contain no shot transitions to demonstrate that our method works equally well in such situation.

In summary, these video sequences pose the following challenges: the object of interest can have wild changes in appearances, including scale, pose, and lighting variations; the background can be highly cluttered and non-stationary; the object can leave and re-enter the scene multiple times, which may occur due to large camera motion or post-editing of the video sequence.

6.4.1 Baseline methods

Here we briefly describe the baseline methods.

Baseline-NM:

This is the Semantic-Shift algorithm in [13]. It is an object discovery method developed for image collections. When we apply it on our video sequences, we treat each video as a collection of images. Motion information is not used, hence we call it Baseline-No Motion.

Baseline-NL:

This is the Probabilistic Latent Semantic Analysis algorithm in [6][4]. Similar to Baseline-NM, it is an object discovery method for still images. Since only appearance information is used but not the location of the image patches, we call it Baseline-No Location.

Baseline-FREQ: Frequent closed itemset mining

In the data mining literature, an itemset refers to a set of items, which in our application refers to a candidate set of regions that could represent an object of interest. A frequent itemset is an itemset that occurs at least a certain number of times, and hence more likely corresponds to an object of interest. A recent data mining algorithm 'CLOSET+' [67] discovers frequent closed itemset, such that for each discovered frequent itemset there exists no superset of equal frequency. This helps in reducing the final number of itemsets to be considered. The CLOSET+ algorithm requires the minimum itemset frequency as an input parameter. Setting the minimum frequency too small will result in too many frequent closed itemsets, and many of them might not correspond to the object of interest. Hence we start from the largest possible minimum frequency, which is equal to the number of frames, and gradually decrease it until M frequent closed itemsets are found. We found $M = 10$ to give the best results.

Baseline-KM1: K-means clustering on image-word co-occurrence matrix

This approach assigns a feature vector to each frame, where the feature vector is the histogram of visual words. In [4] it was reported to perform worse than Baseline-NL. The Euclidean distance is used for computing the distance between feature vectors.

Baseline-KM2: K-means clustering on color histogram

This approach assigns a feature vector to each frame, where the feature vector is the RGB color histogram of all pixels within the frame. We use N regular bins for each color channel and concatenate the three histograms. The Chi-Square distance is used [68]. We found $N = 10$ to give the best results.

6.4.2 Object-oriented video segmentation

Consider a video in which the camera is switching among a number of scenes. For example, in the test drive scene in Fig.6.2, the camera switches between the driver, the frontal view of the car, the side view, and so on. We would like to cluster the frames into semantically meaningful groups. In classical temporal segmentation methods, the similarity between two frames is assessed using global image characteristics. For example, all pixels are used to build a color histogram for each frame, and a distance measure such as the Chi-Square distance is used to measure the similarity between two histograms [68]. K-means clustering or spectral clustering methods can then be employed. This method is suitable for shot boundary detection [68], because when the camera switches between shots, color information provides a good indicator of scene transition.

However, using color information alone cannot provide object-level segmentation. This is because the object of interest often occupies only a small part of the scene, and the global color statistics are often dominated by background clutter. Also, using color alone cannot provide the knowledge of 'what' makes the frames separated into different groups.

Our DISCOV framework provides a natural way for object-oriented clustering, and is also able to point out 'what' is exactly the factor that separates the frames. In Table 6.1, we compare DISCOV to five baseline methods. Each video sequence has a natural object of interest, e.g., the PEPSI and PEUGEOT1 sequences are commercial advertisements where the object of interests are the Pepsi logo and the Peugeot vehicle respectively, and the BENZ and PEUGEOT2 sequences

Sequence	# of Frames	DISCOV	Baseline -NM	Baseline -NL	Baseline -FREQ	Baseline -KM1	Baseline -KM2
BIKE	67 (24)	**96.0%**	88.0%	80.0%	32.0%	60.0%	52.0%
BENZ	711 (232)	**64.0%**	55.9%	57.2%	53.2%	51.8%	50.7%
PEPSI	181 (62)	**73.6%**	67.8%	69.0%	63.2%	69.0%	54.0%
PEUGEOT1	92 (15)	63.1%	**74.3%**	71.4%	51.4%	62.9%	54.3%
PEUGEOT2	273 (231)	**68.9%**	56.1%	57.6%	35.9%	67.0%	52.8%
Weighted Average	—	67.9%	60.5%	61.0%	49.8%	58.5%	51.9%

Table 6.1: Video segmentation performance. Numbers in parenthesis indicate the number of frames containing the object of interest. Detailed in Section 6.4.2.

are test drive videos featuring a car, hence the object of interests in each video are naturally well defined. The BIKE2 and HORSE sequences used later in the localization experiment are not used here because they did not contain transitions from one object to another. The frame rate is one frame per second and the motion of both the object of interest and the background are fast, making it non-trivial to apply optical flow or layer extraction methods for discovering objects. In addition, all sequences frequently transition between different shots. The average duration of a shot ranges from 3 to 5 frames, which is relatively short compared to the video length. This also demonstrates the difficulty of using optical flow based methods. The ground truth data labels the presence or absence of the object of interest in each frame. We evaluate the object discovery performance as a detection problem. The classification rates are shown in Table 6.1.

DISCOV ranks the images according to $P(z_{obj}|d_i)$ for all frames d_i; same for Baseline-NM. This has the interpretation of ranking the images according to how likely it contains the object of interest. Since a ranking is obtained, we report the classification rate at the point where the false alarm rate equals the false reject rate. Baseline-NL, Baseline-KM1 and Baseline-KM2 are clustering methods and do not have knowledge of which cluster corresponds to background clutter and which cluster corresponds to the object of interest. We compute the classification rate for both clusters in turn and report the result with the higher classification rate. Baseline-FREQ assigns different number of frequent closed itemsets to each frame, and we rank the frames according to how many frequent closed itemsets it contains inside.

Baseline-KM1 performs slightly worse than Baseline-NL. This is consistent with the report in [4]. Baseline-NM has similar performance as Baseline-NL. We also see that global color information provides little discriminative ability (Baseline-KM2) and performs the next to the worst. This is due to the large color variations in the background clutter, which dominates over the object of interest. Baseline-FREQ has the lowest classification rate. This shows that the number of frequent closed itemsets in a frame is not a good indicator of the presence of the object of interest. DISCOV outperforms all the others in four out of five experiments. In the PEUGEOT1 sequence, the result of DISCOV is worse than Baseline-NM and Baseline-NL because of the shooting style; the object of interest appears at random locations with fast shot transitions, hence the baseline methods that do not model the motion perform better. Overall, DISCOV has the leading performance in weighted average classification rate, where the weighting comes from the number of frames.

6.4.3 Object-oriented threading

The capability of object-oriented video segmentation suggests an application called "threading", where all occurrences of an object are linked together. Threading is different from keyframe(s) extraction [69]. The aim of keyframe extraction is to obtain a set of frames that covers all aspects of a video sequence, yet these frames need not contain the object of interest. Our aim of object-oriented threading is to obtain a set of frames that includes the object of interest, hence being different from keyframe extraction. Whether threading or keyframe extraction is more useful is application dependent; it is better to understand them as different video summarization techniques. Both methods attempt to cover the temporal domain while threading focuses more on the object of interest.

Our approach to threading is object-oriented. First, we rank the images according to how likely they contain the object of interest (using $P(z_{obj}|d_i)$ as in Section 6.4.2). We put the top twenty frames with the highest values into a candidate set. Since many among these twenty frames are visually similar and hence redundant, we apply k-means clustering (with $k = 5$) using their RGB color histograms as features and pick from each cluster the one with the highest value of $P(z_{obj}|d_i)$, resulting in the five frames as shown in Fig. 6.3. Even though the Pepsi logo appears in only 87 out of 181 frames, each of the five candidate keyframes contains the Pepsi logo. Likewise, the Mercedes-Benz appears in only 278 out of 711 frames. The five candidate keyframes correspond to the 435^{th}, 468^{th} 690^{th}, 694^{th}, and 704^{th} frame in the BENZ video and the 30^{th},42^{nd},55^{th},146^{th},

and 179^{th} frame in the PEPSI video, showing very little temporal redundancy (frames are sampled at one frame per second).

Sequence	Trivial Solution	DISCOV		Baseline-NM		Baseline-NL	
BIKE	69.2%	92.3%	**1.33**	92.3%	**1.33**	88.5%	1.28
HORSE	20.0%	65.0%	**3.25**	50.0%	2.50	50.0%	2.50
BIKE2	50.0%	96.6%	**1.93**	82.8%	1.66	74.1%	1.48
PEUGEOT1	100.0%	81.7%	**0.82**	77.9%	0.78	58.6%	0.59
PEUGEOT2	86.3%	71.7%	**0.83**	57.8%	0.67	59.6%	0.69
BENZ	55.7%	68.3%	**1.23**	65.7%	1.18	59.3%	1.06
PEPSI	14.9%	34.5%	**2.32**	31.2%	2.09	19.5%	1.31

Table 6.2: Localization performance. Detailed in Section 6.4.4.

6.4.4 Object of interest localization

In order to see if the discovered objects truly correspond to the object of interest, here we evaluate the localization performance. The ground truth data provides a bounding box around the object of interest in each frame, and the frames that do not contain the object of interest are not evaluated. Each object discovery algorithm assigns to each MSER an 'object' or 'background' label. Each MSER has a center position. The average position and covariance of all 'object' MSERs provides a rough estimate of the object position, scale and shape. A hit is made if the estimated position and scale matches well with the ground truth bounding box within a certain threshold. The reported numbers shown in percentage are the hit rates averaged over each video sequence. It should be noted that since occasionally some background clutter are assigned an 'object' label, these outliers can move the average position of all 'object' MSERs outside the bounding box, hence showing lower hit rates.

Results are shown in Table 6.2. The trivial solution is a naive algorithm: always return the center of the frame as the position estimate of the object of interest. Since larger objects are more likely covering the center position, the trivial solution provides a sense of the difficulty of each video sequence. The numbers next to the percentages are ratios between the hit rate of the algorithm and the trivial solution. The larger the better. It can be seen that DISCOV clearly

outperforms Baseline-NM and Baseline-NL.

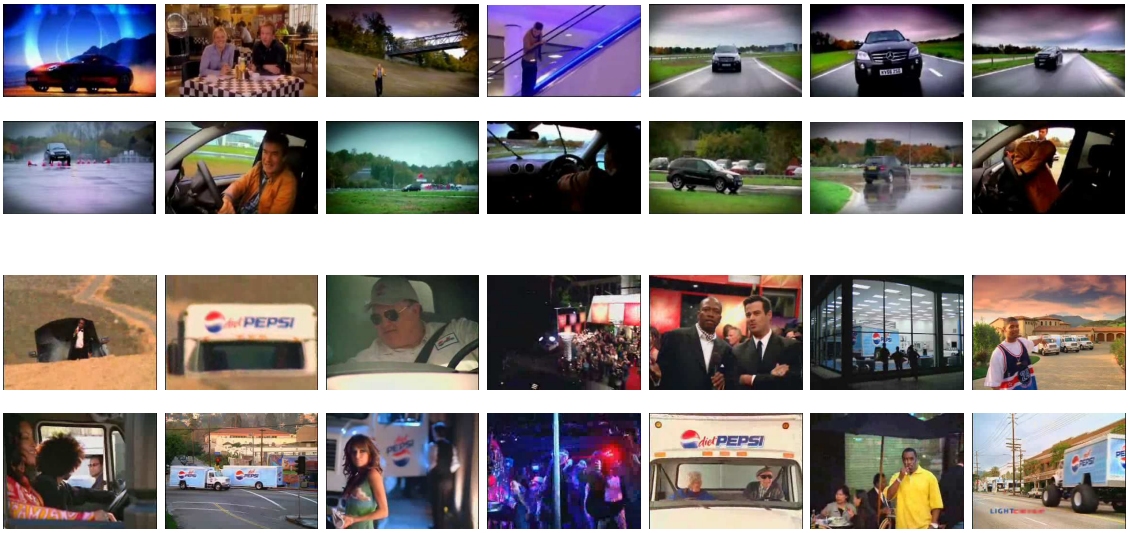


Figure 6.2: Samples of two video sequences from YouTube.com. Top two rows are 14 out of 711 samples of the BENZ sequence. Bottom two rows are 14 out of 181 samples of the PEPSI sequence. Images are displayed from left to right.

6.4.5 Computation speed

The computation time for DISCOV on a 100-frame video sequence is around 30 seconds for MSER extraction and 80 seconds for running the EM algorithm. The EM algorithm is written in MATLAB and not intentionally optimized for speed.

6.5 Handling Multiple Objects

In the previous sections, we considered a model that assumes there is at most one OOI per frame. To deal with multiple objects, we consider a Sequential Monte Carlo framework.

We use a particle filter to address the aforementioned problems of the original DISCOV paper. We call it the unsupervised particle filter because, as explained in the Introduction section, prior work on using particle filtering for tracking requires human intervention [70],[71],[72],[73] or human labeled data [74], or has been using a simplified bootstrap filter [75].

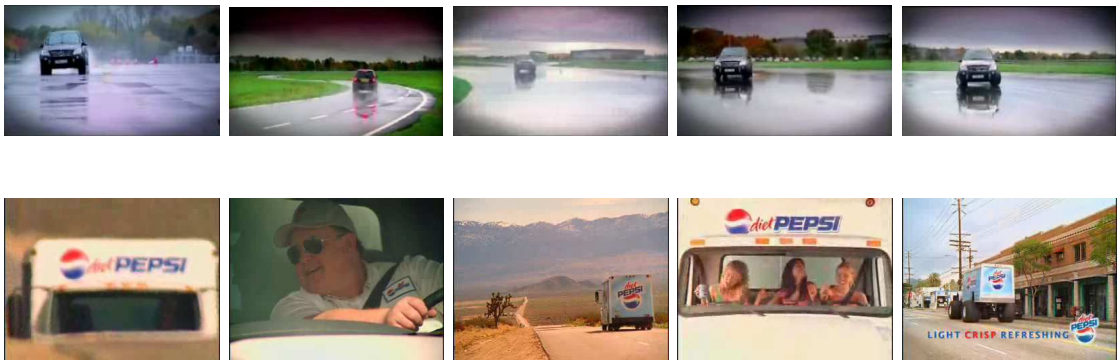


Figure 6.3: Results of object-oriented threading (Section IV-C). Each frame in the top row contains the black Mercedes vehicle; in the bottom row, each frame contains the PEPSI logo. This provides an object-oriented overview of the whole video sequence in a different way than traditional keyframe extraction.

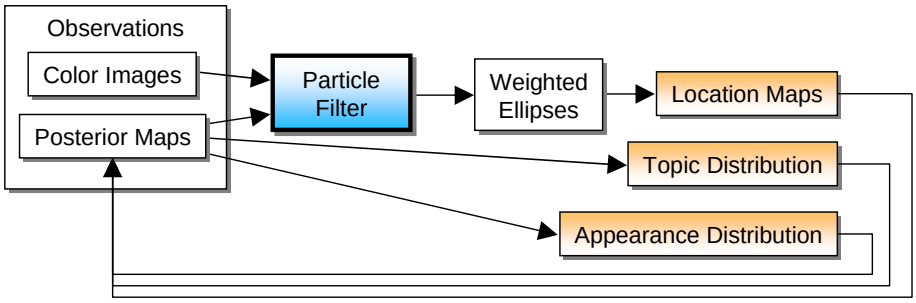


Figure 6.4: Algorithm flowchart. Notice that while the location maps are estimated for each frame, the topic and appearance distributions are shared across all frames.

6.5.1 Overview

Define the 'posterior map', or 'P-Map', as the 'image' that stores the posterior probability $p(z_{FG}|d, w, r)$ of each pixel. The P-Map is updated according to (6.3). Notice that some pixels can be covered by more than one MSER patch. In that case, we assign the pixel the maximum value; some pixels can be covered by none of the MSER patches. In that case, we assign the pixel a close-to-zero posterior probability (we used 10^{-5} in the experiments). Similarly, the 'location map', or L-Map stores the probability $p(r|d, z_{\text{FG}})$ of each pixel.

A region in the P-Map with high values indicates the potential existence of a foreground object at that location. But the P-Map can have spurious regions causing false positives or false negatives.

The purpose of using the particle filter is to 'clean up' the P-Map. The cleaned-up P-Map becomes the L-Map in the next EM iteration. Therefore, the clean-up process is crucial to the performance of video object discovery. The clean-up is based on the following prior knowledge: first, that objects tend to move in a smooth manner; second, that objects tend to be spatially clustered in space.

We use a particle filter for the clean-up process. Notice that the particle filter replaces the role of the PDA filter (a variation of the Kalman filter) in the original DISCOV framework. The advantages of the particle filter over the PDA filter are: (1) it can handle multiple objects, (2) it can handle complex shapes, (3) it is inherently a multiple-hypotheses framework, and (4) it allows us to use a non-linear observation model. While there exists other PDA-like filters such as the JPDA filter [65] that can handle (1)-(3) as well, it is the fourth property that makes particle filters especially suitable for our purpose.

The input to the particle filter includes the P-Map as well as the original color image frames (see Fig. 6.4), collectively denoted as y_k, where k is a frame index. As in the control and tracking literature, we call y the observation to the particle filter.

6.5.2 Importance Sampling

We first briefly review the basics of Importance Sampling.

The expectation of a function f under the probability distribution p is as follows:

$$E_p[f(x)] = \int_{-\infty}^{\infty} f(x)p(x)dx \tag{6.12}$$

Suppose we have N particles, or random samples, $x_1, ...x_N$, that are sampled from p. Then p can be approximated as follows:

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x^{(i)}) \tag{6.13}$$

Substituting the approximate p into $E_p[f(x)]$, we obtain

$$E_p[f(x)] = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \tag{6.14}$$

Since sampling from p is sometimes difficult, we consider another option here. Instead of sampling from p, we sample N random samples, $x_1, ...x_N$, from another distribution q, which is often called the 'proposal distribution'. Also define the particle weight, w, as follows:

$$w(x) = \frac{p(x)}{q(x)} \tag{6.15}$$

It follows that $E_p[f(x)]$ can be computed as follows:

$$E_p[f(x)] = \frac{\int f(x)w(x)q(x)dx}{\int w(x)q(x)dx} = \frac{\frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})w^{(i)}}{\frac{1}{N}\sum_{i=1}^{N} w^{(i)}} \tag{6.16}$$

which further simplifies to

$$E_p[f(x)] = \sum_{i=1}^{N} f(x^{(i)})\frac{w^{(i)}}{\sum_{j=1}^{N} w^{(i)}} = \sum_{i=1}^{N} f(x^{(i)})v^{(i)} \tag{6.17}$$

where $w^{(i)} \triangleq w(x^{(i)})$ and

$$v^{(i)} = \frac{w^{(i)}}{\sum_{j=1}^{N} w^{(i)}} \tag{6.18}$$

Notice that in order to compute $v^{(i)}$ we only need to know $p(x)$ up to a multiplicative constant. This fact is useful when exhaustively evaluating all possible outcomes of $p(x)$ is intractable.

6.5.3 Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods (also called particle filters) is an extension of Importance Sampling. It consists of a two step recursion:

Predict:

$$p(x_k|y_{0:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{0:k-1})dx_{k-1} \tag{6.19}$$

and Update:

$$p(x_k|y_{0:k}) = \frac{p(y_k|x_k)p(x_k|y_{0:k-1})}{\int p(y_k|x_k)p(x_k|y_{0:k-1})dx_k} \tag{6.20}$$

Sequential Importance Resampling (SIR) is a popular SMC method. The proposal distribution is defined through this recursive equation:

$$q(x_{0:k}|y_{0:k}) = q(x_0|y_0)\prod_k q(x_k|x_{0:k-1}, y_{0:k}) \tag{6.21}$$

If we replace $p(x)$ with $p(x_{0:k}|y_{0:k})$ and $q(x)$ with $q(x_{0:k}|y_{0:k})$ in the derivation of Importance Sampling, we obtain:

$$w_k^{(i)} = \frac{p(x_{0:k}^{(i)}|y_{0:k})}{q(x_{0:k}^{(i)}|y_{0:k})} = \frac{1}{q(x_{0:k}^{(i)}|y_{0:k})} \cdot \frac{p(y_{0:k}|x_{0:k}^{(i)})p(x_{0:k}^{(i)})}{p(y_{0:k})} \tag{6.22}$$

If we define

$$w_k^{*(i)} \triangleq \frac{p(y_{0:k}|x_{0:k}^{(i)})p(x_{0:k}^{(i)})}{q(x_{0:k}^{(i)}|y_{0:k})} \tag{6.23}$$

then

$$v_k^{(i)} = \frac{w_k^{(i)}}{\sum_j w_k^{(i)}} = \frac{w_k^{*(i)}}{\sum_j w_k^{*(i)}} \tag{6.24}$$

and

$$\frac{w_k^{*(i)}}{w_{k-1}^{*(i)}} = \frac{\frac{p(y_{0:k}|x_{0:k}^{(i)})p(x_{0:k}^{(i)})}{q(x_{0:k}^{(i)}|y_{0:k})}}{\frac{p(y_{0:k-1}|x_{0:k-1}^{(i)})p(x_{0:k-1}^{(i)})}{q(x_{0:k-1}^{(i)}|y_{0:k-1})}} = \frac{\frac{p(y_{0:k},x_{0:k}^{(i)})}{p(y_{0:k-1},x_{0:k-1}^{(i)})}}{q(x_k^{(i)}|x_{0:k-1}^{(i)},y_{0:k})} \tag{6.25}$$

The $v_k^{(i)}$ are called the normalized particle weights. Based on the conditional independence assumptions shown in the graphical model in Fig. 6.5, we have

$$\frac{p(y_{0:k},x_{0:k}^{(i)})}{p(y_{0:k-1},x_{0:k-1}^{(i)})} = p(y_k|x_k^{(i)},y_{k-1},x_{k-1}^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)}) \tag{6.26}$$

so that the un-normalized particle weights can be computed recursively:

$$w_k^{*(i)} = w_{k-1}^{*(i)} \frac{p(y_k|x_k^{(i)},y_{k-1},x_{k-1}^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{0:k-1}^{(i)},y_{0:k})} \tag{6.27}$$

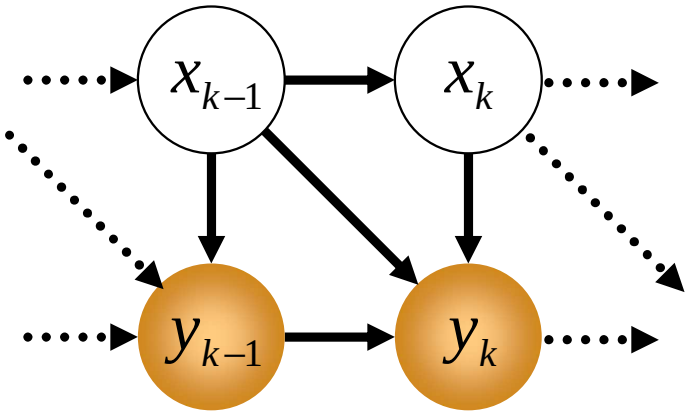


Figure 6.5: Graphical model defining the conditional independence assumptions.

In theory, we would sample from the proposal distribution,$q(x|x_{0:k-1}^{(i)},y_{0:k})$, and obtain samples $\{x_k^{(i)}\}$. Each sample is then weighted by four terms, $p(y_k|x_k^{(i)},y_{k-1},x_{k-1}^{(i)})$, $p(x_k^{(i)}|x_{k-1}^{(i)})$, $q(x_k^{(i)}|x_{0:k-1}^{(i)},y_{0:k})$ and $w_{k-1}^{*(i)}$ before obtaining the un-normalized particle weight $w_k^{*(i)}$. Notice that each of these functions can be subject to an unknown multiplicative factor without affecting the value of the normalized particle weights.

6.5.4 Proposal Distribution

A proper proposal distribution (also called the *importance density* [76]) is essential for keeping the particle filter effective. In practice, we want to use a small amount of particles for run-time efficiency. Hence, distributing the particles effectively in the state space is important. It has been shown [77] that the optimal proposal distribution which minimizes the variance of the particle weight conditional upon $x_{0:k-1}^{(i)}$ and $y_{0:k}$ has the form of $q(x_k|x_{k-1}^{(i)}, y_k)$.

Our approach is to build a proposal distribution from the P-Map, y^{pmap}, which is part of the observation, y_k. Different than all prior work in visual tracking, where the observation consists only of the color image, our observation consists additionally of the P-Map. The P-Map contains information of the position and scale of the objects of interest. In this sense, mode (local maxima) and scale seeking on the P-Map resembles the use of an object detector in [74]. However, mode and scale seeking does not require labeled data or training of an object detector, hence it fits in the unsupervised object discovery framework.

Mode and scale seeking

Our approach of finding the mode and scale is to fit a mixture of Gaussians (MoG) (see [64] for the EM algorithm we are using) to the P-Map, where the control parameter is the number of mixture components, K. In video object discovery of short video clips, such as the Youtube videos used in our experiments, the number of objects of interest in a video are generally less than three. Therefore, by controlling the value of K over a reasonable range of values, we effectively impose a prior knowledge on the number of objects of interest. By maintaining multiple MoGs with different K values, we explicitly explain the possibilities of different numbers of objects of interest in each frame, and implicitly the possibilities of noise in the P-Map, effectively maintaining multiple hypotheses over the number of objects of interest. In the experiments, we use $K = 1, ..., 5$. In order to incorporate the prior knowledge on the number of objects of interest, an algorithm that has direct control over the number of modes is preferred to an algorithm that directly controls the bandwidth of kernels (mixture components), such as the variable bandwidth mean-shift [23],[78],[79],[80],[71].

Finding the optimal number of mixture components is a model selection problem with rich literature [81]. Even though greedy algorithms exist that have running time linear in the number of data points and quadratic in the final number of mixture components [82], in video object discovery,

however, model selection becomes intractable: the number of objects of interest can vary from frame to frame, and hence if we jointly optimize over all frames in a video, the complexity is exponential to the number of frames. Jointly optimizing over all frames is important, because the number of objects of interest is strongly correlated across frames.

Based on the prior knowledge on the number of objects of interest, and taking into account that the P-Maps are noisy, we maintain multiple MoGs with different number of mixture components and collect *all* modes. I.e., instead of trying to determine the "correct" value of K, we make use of multiple MoGs and don't care about determining the optimal number of mixture components. In the experiments, we use diagonal MoGs and $K = 1, ..., 6$. Notice that the K values do not need to be contiguous, nor do they have to start from one. Alternatively, one could use Variational Bayes techniques [83] to estimate the mixture model.

Parts-based representation

The Gaussians collected from the MoG(s) *collectively* represent the potential spatial positions of the objects of interest. Notice that multiple Gaussians may collectively describe a single object of interest. The mixture model is therefore suitable for modeling complicated shapes and articulated objects. This is similar in spirit to the 'multi-ellipsoid' representation in [84], where body parts are collectively modeled by multiple ellipsoids.

To fine tune the scale of each potential object part, the standard deviations in the horizontal and vertical axes are multiplied by a scaling factor for each Gaussian. In practice, we use a set of scaling factors $s = \{0.75, 1, 2\}$, with the hope that one of the scaling factors will approach the true size of the object part.

After the position and scale estimates of each potential object part are obtained, each potential object part is equivalent to an ellipse. We can estimate the velocity of each ellipse as follows. First we obtain the correspondence between ellipses in neighboring frames. In the tracking literature, this is called the data association problem [65]. The nearest-neighbor strategy [65] associates ellipses in the current frame with the closest one in the previous frame. This simple strategy keeps the number of associations at a minimum but is more prone to errors. We adopt a multiple hypotheses strategy [65] where each ellipses is associated with all ellipses in the previous frame. For each association, we compute the estimated velocity vector by subtracting the horizontal and vertical positions of the ellipse in the previous frame from the current frame. Each particle is now

represented by the position, scale, and velocity as follows:

$$x^{(i)} = (pos_h^{(i)}, pos_v^{(i)}, scale_h^{(i)}, scale_v^{(i)}, vel_h^{(i)}, vel_v^{(i)})^T \tag{6.28}$$

where the subscripts h and v denote horizontal and vertical coordinates, and the frame index k being dropped for clarity. Since each ellipse is associated with all ellipses in the previous frame, the number of particles is the squared number of ellipses. Since the first frame does not have a previous frame, particles in the first frame are initialized with zero velocity and properly replicated such that the number of particles is constant in all frames.

In summary, we use multiple MoGs to deterministically estimate the position and scale of potential object parts from the P-Map, y^{pmap}. As a comparison, the MoG (and its approximation by a kernel density function) has been used in the particle filtering literature in fundamentally different ways: to model the particle filter's posterior distribution in [85], or to model both the posterior distribution and the likelihood function in [80] and [71].

Unsupervised proposal distribution

The proposal distribution has the following form:

$$q(x_k|x_{k-1}^{(i)}, y_k) = \frac{1}{N}\sum_{i=1}^{N}\delta(x_k - x_k^{(i)}) \tag{6.29}$$

where N is the number of particles, particles $\{x_k^{(i)}\}_{i=1}^N$ are obtained as described in the previous section, and $\delta(x_k - x_k^{(i)})$ denotes the Dirac-delta mass located at $x_k^{(i)}$.

The proposal distribution has two important features: first, the proposal distribution is a function of the observation (noticing that particles are estimated based on the P-Map); second, and more importantly, the proposal distribution does not rely on human labeled data. While these two features have been individually presented before, they have never been shown together.

The first feature was neglected by the CONDENSATION algorithm [70], which is a popular particle filter algorithm due to its simplicity (for example, [75] and [86]). The proposal distribution is chosen as the dynamic model (also called the prior distribution [76]), $p(x_k|x_{k-1}^{(i)})$, which omits the observation, y_k, from the optimal distribution, $q(x_k|x_{k-1}^{(i)}, y_k)$. Equation (6.27) for computing the particle weights then greatly simplifies, as the nominator and denominator cancel each other out. The simplification comes at a price, because by ignoring the observation, the proposal

distribution could generate few or zero particles around the true state, which would result in poor performance [76].

The second feature was neglected by the Boosted Particle Filter [74], which uses an Adaboost object detector to detect hockey players and thereby defining a proposal distribution. Similarly, the color-based tracker in [86] detects regions of skin color. Since the observations (in their case, the original color image frames) are taken into account in the proposal distribution, the aforementioned problem of CONDENSATION is resolved. However, training an object detector or skin color model requires human labeled data, hence the approach is not suitable for unsupervised learning. On the other hand, while the work in [75] learns the foreground and background models by background subtraction and hence does not require human labeled data, its particle filter is based on CONDENSATION and hence neglects the first feature.

Even though there is a vast literature addressing the two features individually, there is no work that has addressed them simultaneously. We call (6.29) the 'unsupervised' proposal distribution due to the second feature.

6.5.5 Dynamic Model

The dynamic model is a linear equation,

$$x_k = Ax_{k-1}^{(i)} + \epsilon_k \tag{6.30}$$

where ϵ_k is a Gaussian noise vector with zero mean and standard deviation $\sigma = (\sigma_h^{pos}, \sigma_v^{pos}, \sigma_h^{scale}, \sigma_v^{scale}, \sigma_h^{vel}, \sigma_v^{vel})^T$, and

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.31}$$

i.e., assuming constant velocity and constant scale.

For a pair of ellipses in frame $n-1$ and frame n, one can compute the probability

$$\tilde{p}(x_k|x_{k-1}^{(i)}) = \mathcal{N}(x_k - Ax_{k-1}^{(i)}|0, diag(\sigma)) \tag{6.32}$$

where $diag(\sigma)$ is the covariance matrix formed by the diagonal elements taken from the σ vector and zeros elsewhere. We use $\sigma = (1, 1, 10, 10, 0.1, 0.1)$. The scale parameter is given relatively larger freedom to change to accommodate abrupt changes in size in real video datasets.

6.5.6 Observation Model

The likelihood function in (6.27) is defined as

$$p(y_k|x_k^{(i)}, y_{k-1}, x_{k-1}^{(i)}) \propto \exp\{\lambda_P \log L_P + \lambda_C \log L_C\} \tag{6.33}$$

which consists of the P-MAP likelihood function L_P and the color likelihood function L_C. We use $\lambda_P = \lambda_C = 10^{-4}$ in the experiments.

P-Map likelihood

The P-Map (log-)likelihood captures the intuition that the system prefers image regions with higher posterior probability being covered by candidate ellipses, instead of lower ones being covered. A naive implementation would be

$$\log L_P(y^{pmap}, x_i) = \frac{1}{A_i} \sum_j^{A_i} (y_j^{pmap} - 0.5) \tag{6.34}$$

where j is an index over the pixels covered by ellipse x_i, and A_i is the area of x_i. However, for candidate ellipses on the P-Map where the center has the highest posterior value and gradually decreasing values on the sides, this would encourage ellipses that degenerate to a point (that is the mode). A corrected version of the above equation would hence encourage larger ellipses is as follows:

$$\log L_P(y^{pmap}, x_i) = \frac{1}{C_i} (\sum_j^{A_i} (y_j^{pmap} - 0.5) - \sum_k^{B_i} (y_k^{pmap} - 0.5)) \tag{6.35}$$

where k indexes the set of pixels on a band of finite width along the boundary of ellipse x_i, and B_i is the number of such pixels, and $C_i = A_i + B_i$. In the experiments, we noticed that using a boundary with width of 1, 2, or even 3 pixels would significantly improve the performance over 0, which reduces to (6.34). We use a width of two pixel. To better understand (6.35), consider the example shown in Fig. 6.6: suppose the P-Map y^{pmap} has value 1 (shown as white) inside a circular region, and value 0 (shown as gray) outside. Suppose we have a set of two circular-shaped candidate ellipses, the inner one shown with round dotted outline, and the outer one shown with

dashed outline. While (6.34) would assign equal likelihood to these two candidate ellipses, (6.35) would assign a larger likelihood to the outer ellipse, which exactly covers the white area.

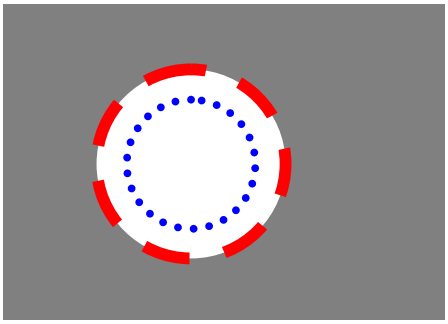


Figure 6.6: Illustration for the definition of P-Map likelihood.

Color likelihood

The color likelihood term is based on the (1) object attraction and (2) background exclusion principles [86],[84]:

$$\log L_C(y_k, x_k^{(i)}, y_{k-1}, x_{k-1}^{(i)}) = \underbrace{B(h_k^{(i)}, h_{k-1}^{(i)})}_{(1)} - \underbrace{B(h_k^{(i)}, \overline{h})}_{(2)} \tag{6.36}$$

where the first term favors the histogram similarity between the i^{th} ellipse in the current frame, $h_k^{(i)}$, and in the previous frame, $h_{k-1}^{(i)}$. The second term favors the difference in an ellipse's appearance from the background. The background color histogram $\overline{h}$ uses image pixels that are not covered by any of the ellipses. The similarity is based on the Bhattacharyya coefficient, $B(h_a, h_b) = \sum_j \sqrt{h_a(j)h_b(j)}$. We use 10 histogram bins for each of the R, G, and B color channels.

6.6 Experiments

6.6.1 Synthetic Data Experiment-1

In this experiment, we want to demonstrate the capability of the particle filter within a single EM iteration, given synthetic observation data. The data consists of 8 frames of P-Maps and color image frames, simulating a single object moving in cluttered background. In the 5^{th} frame, we added structural noise simulating background clutter. The structural noise is represented by high

posterior probability in the P-Map. The goal is to show that the particle filter is robust to the structural clutter.

The data is shown in Fig. 6.17. In the first column of Fig. 6.17, we see that the color of both the object and the background are contaminated by noise. In the second column, the top figure shows the P-Map, and the bottom figure shows the L-Map, which is the output of the particle filter. We can see from the L-Map that despite the color noise and structural clutter, the particle filter tracks the target throughout the video. The computation time is 0.65 seconds per frame using MATLAB on a Intel Core2 Duo 3GHz machine.

6.6.2 Synthetic Data Experiment-2

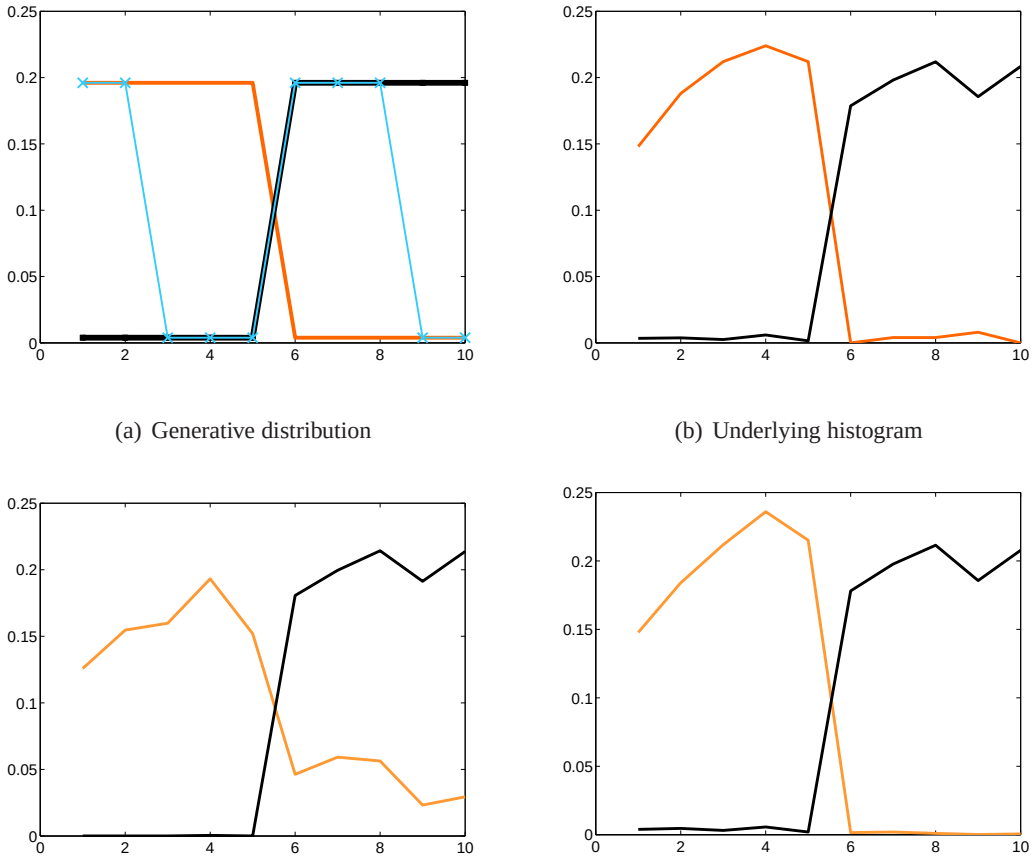
The second experiment has the same setup as the previous one, except that there are two objects instead of one. In the 5^{th} frame, instead of adding structured clutter, we let one of the objects disappear, simulating occlusion, and then re-appear in frame 6. The goal is to show that the particle filter is robust to occlusion.

The data is shown in Fig. 6.18. In the first column, we see that the color is again contaminated by noise. In the bottom figures of the second and third row, we can see that the particle filter can quickly recover from occlusion.

6.6.3 Synthetic Data Experiment-3

The previous two experiments demonstrated the particle filter's utility within a single EM iteration. In this experiment, we run the whole DISCOV framework with 20 EM iterations, and the P-Maps are automatically generated according to the DISCOV framework. There are ten visual words. Two objects are moving in linear motion as in Experiment 2 with the same number of frames. The goal is to illustrate that the particle filter is more suitable for this task than the PDA filter in the original DISCOV framework, because it can handle naturally handle multiple objects.

We sampled 320 visual words for each frame from the generative distribution shown in Fig. 6.7 with a foreground topic distribution $P(z_{FG}|d) = 0.5$, and obtain the underlying histogram, shown in Fig. 6.7. Both the generative distribution and the underlying histogram are hidden, and the goal is to estimate the underlying histogram. From Fig. 6.7 we see that the particle filter-based DISCOV produces appearance distributions that are far more similar to the underlying histogram than the PDA filter-based DISCOV.



(a) Generative distribution

(b) Underlying histogram

(c) SS estimated distribution

(d) DISCOV estimated distribution

Figure 6.7: Visual word distributions.

6.6.4 Real Data Experiments

We collected 94 videos from `www.youtube.com`. This data will be made available online. Videos are converted to images in PNG format at a rate of 2 frames per second. To measure quantitative performance, we collected ground truth data in the following way: three persons without knowledge of our system were asked to draw bounding boxes covering the objects of their interest. Examples are shown in Fig. 6.8. We did not instruct the human labelers an upper or lower bound on the number of objects of interest, hence the number varies for each video, but in all videos the number of objects of interest is less than four.

We first use the mean-squared error (MSE) as performance measure, which measures the discrepancy between the human labeled bounding box and the system generated P-Map for each

Figure 6.8: Sample frames containing one object of interest (top), and multiple objects of interest (middle and bottom). From left to right: original data, system generated output, human labeling.

frame, as illustrated in Fig. 6.9. In Fig. 6.10, on the left, we show the mean-squared error averaged over all frames for each video, then averaged again over all videos; on the right, we show the mean-squared error averaged directly over all frames from all videos. The purpose is to see whether the results are biased by a subset of videos with more frames. We see that the two charts have very similar results.

From the charts in Fig. 6.9 we see that the proposed DISCOV framework with particle filtering performs better than the original DISCOV framework, and human performance is still the best. All results are evaluated against the ground truth labels from person 1. The human performance is mea-

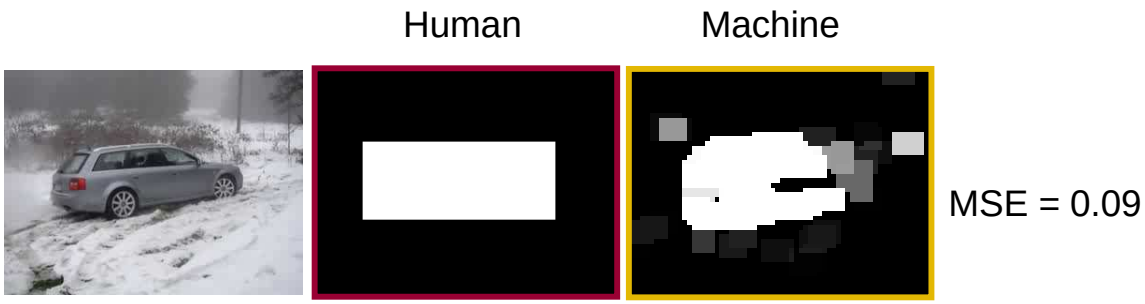


Figure 6.9: Mean-squared error of a single frame.

suring the labeling of person 2 against the labeling of person 1. We also see that the PLSA model [4][6] shown by the 'No Location' bar performs the worst. We did not compare further with other data mining techniques such as the frequent closed itemset mining [67], clustering on the image-visual word co-occurrence matrix [87], or a straightforward clustering on color histograms [87], because the original DISCOV framework in [87] has already demonstrated superior performance to those techniques.

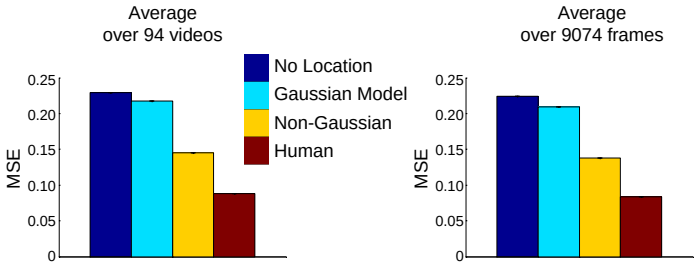


Figure 6.10: Mean-squared error over all videos and all frames.

To gain some understanding of the performance over individual videos, we show in Fig. 6.11 the results of the first ten videos in our dataset. The proposed DISCOV framework performs better than the original DISCOV framework in general, with the exception of video number 6, 8, and 10. The reason can be explained by Fig. 6.9, which shows a frame in video number 6. In this video, both persons labeled the vehicle as the only object of interest throughout the video. In this case, the shape of the vehicle is well approximated by a Gaussian, and hence the original DISCOV framework fits the data extremely well. On the other hand, the particle filter is relatively more prone to data overfitting. However, as seen in video number 8 and 10, where also only a single object of interest exists throughout the video, the performance of the proposed DISCOV framework can approach the original DISCOV framework very closely, and little data overfitting occurs.

In Fig. 6.11 we see an anomaly where video number 1 has better performance achieved by the machine than by human. The reason is that person 1 and person 2 sometimes placed labels differently, as shown in Fig. 6.12, hence the high mean-squared error.

In addition to the mean-squared error, we show in Fig. 6.13 the precision-recall (PR) curve. Since the P-Map consists of probabilities for each pixel, by varying a threshold value on the P-Map we can classify pixels in each frame into those that belong to an object of interest and those that do not. The precision is the number of pixels in all frames that are correctly classified as belonging

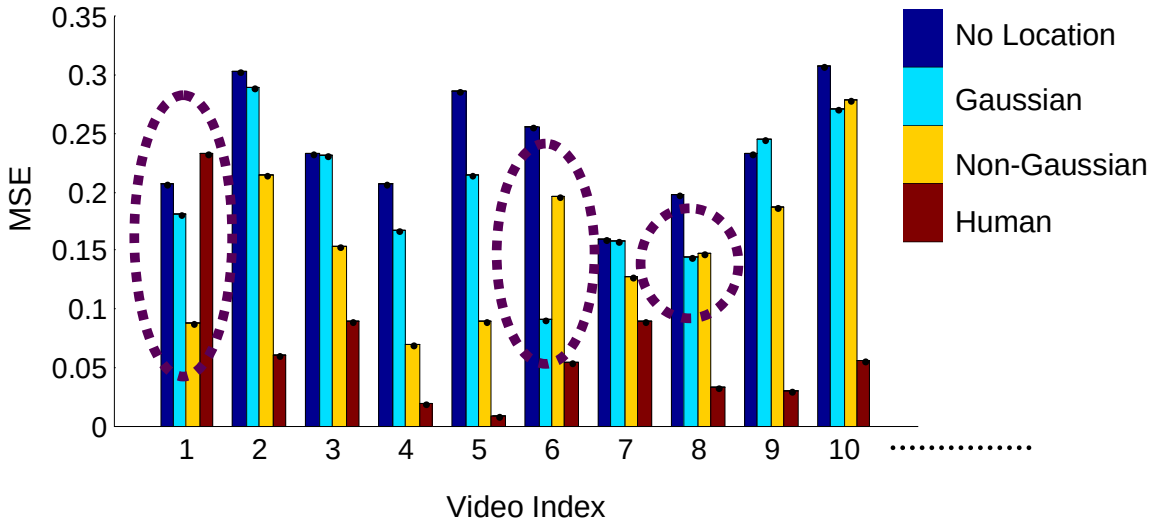


Figure 6.11: Mean-squared error over first ten videos.

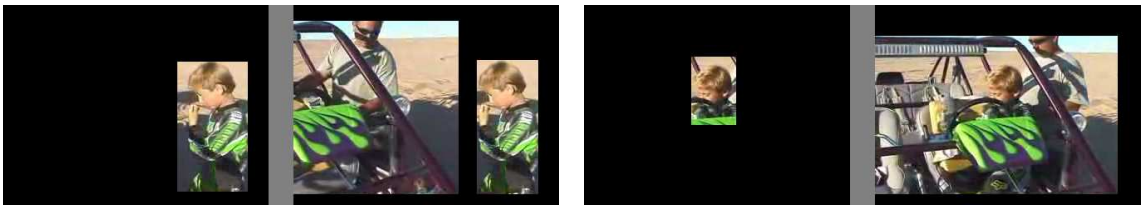


Figure 6.12: Different people have different concepts of object of interest.

to an object of interest divided by the total number of pixels in all frames labeled as belonging to the object of interest. The recall is the number of pixels in all frames that are correctly classified as belonging to an object of interest divided by the total number of pixels in all frames that actually belong to the object of interest. The PR curve is shown in the third column of Fig. 6.13.

Some frames are more suitable for evaluation purpose than the others; let us call them the good frames. One way to select good frames is to select the ones that the human labelers agree with each other on their labeling. We use the F-measure as similarity measure, which is the harmonic mean of precision and recall. The precision and recall values are computed for each frame based on whether each pixel is labeled the same or not. After ranking all frames based on the F-measure, we selected three sets of frames: the first set consists of the top 10% ranked frames, the 2nd subset consists of the top 50% ranked frames, and the 3rd set consists of all frames. Using these three sets of frames, we then evaluated the machine performance. This way of evaluating machine performance takes

into account the consistency of human labelers.

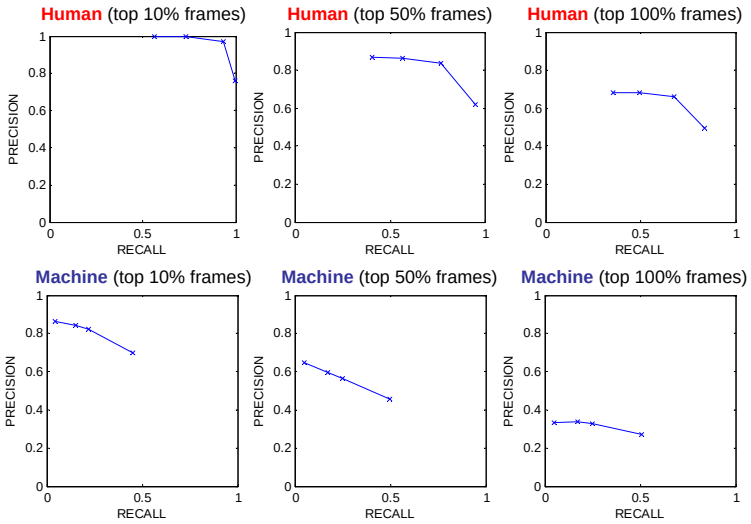


Figure 6.13: Precision-recall curves.

In Fig. 6.14, we show the MSE again, this time using the 'voting' result of the three human labelers. The voting result is obtained as follows: for each pixel, if most human consider it as object of interest, then labeled it as so; otherwise, the pixel is labeled as background. Using the voting result, we again see that the non-Gaussian approach performs the best.

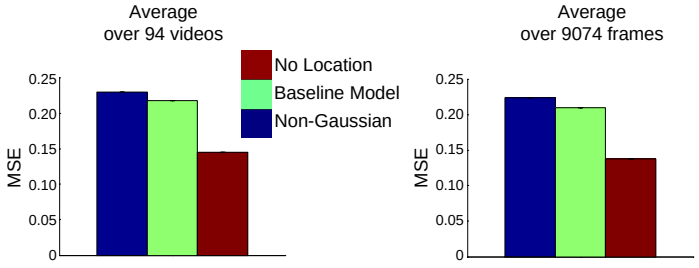


Figure 6.14: Mean-squared error over all videos and all frames.

The EM-algorithm is used both in the construction of the visual word dictionaries and in updating the location map, appearance distribution, and topic distribution (Fig. 6.4). It is well known that the EM-algorithm is sensitive to initialization and has difficulty escaping local extrema [81]. Here we want to see whether this causes different performances in different runs. We ran the whole system repeatedly for 50 times with random parameter initialization and recorded the mean-squared error at each run. In Fig. 6.15 we see that the MSE has little variation over different runs,

demonstrating that the overall system is stable enough to provide consistent results.

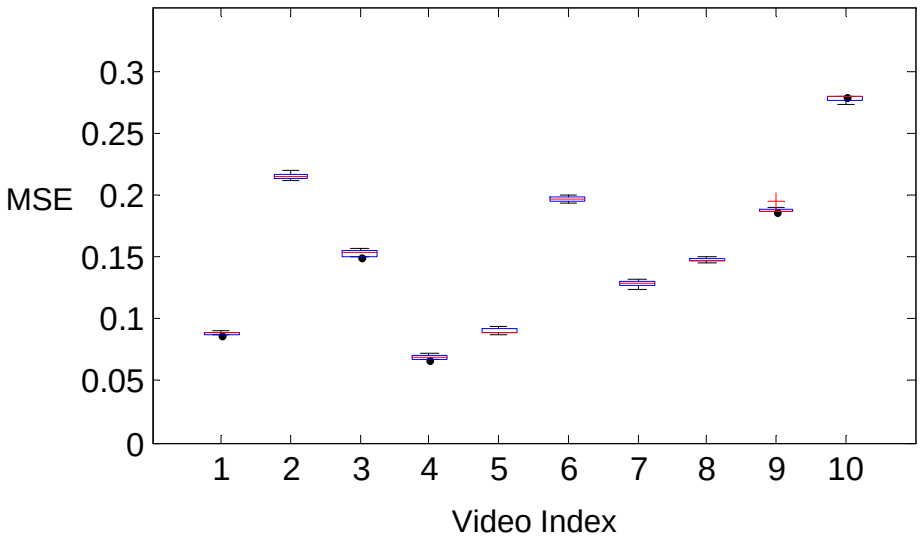


Figure 6.15: System stability/sensitivity to random initializations.

In Fig. 6.16 we study the effect of the visual word dictionary size on the overall performance. We vary the dictionary size from 10 to 5000 and record the average MSE over all videos. The MSE gradually decreases and then increases, showing that the optimal dictionary size is around 100 to 1000. Within this range, the performance is relatively insensitive to the visual word dictionary size.

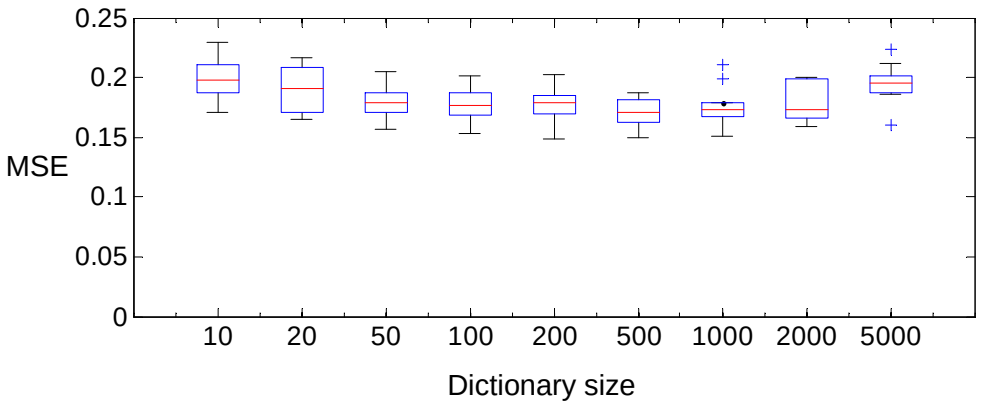


Figure 6.16: Effect of dictionary size on mean-squared error.

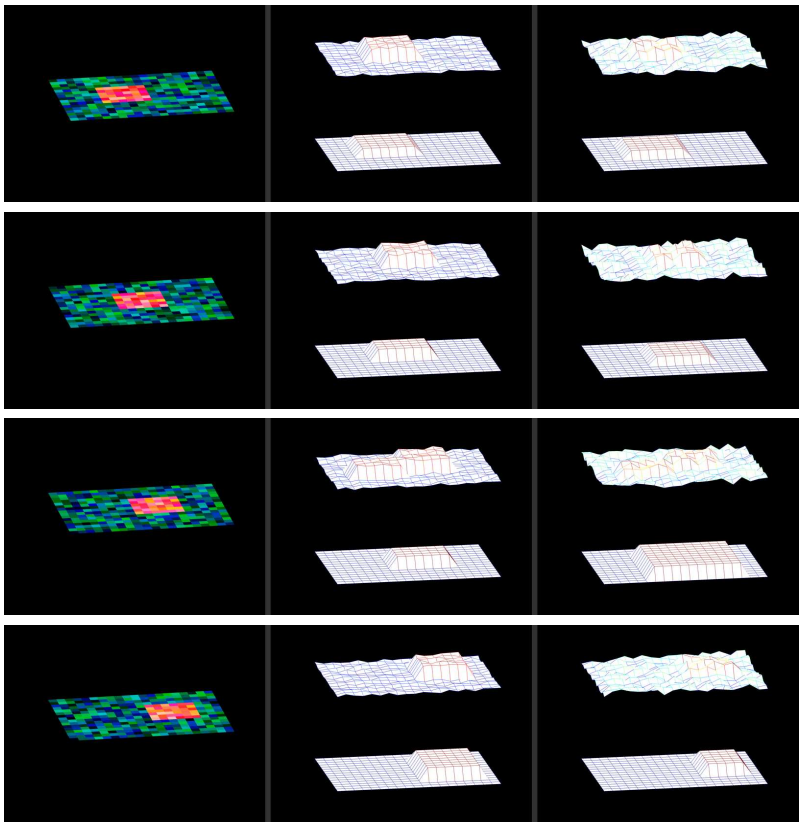


Figure 6.17: Frames 1, 3, 5, 7 shown in rows. Middle and right column: different levels of noise. Top: posterior distribution. Bottom: location distribution after particle filtering. System is robust to noise and clutter (in frame 5).

6.7 Conclusion

The video data mining and 'object-oriented' nature of our approach provides promising new directions for video content analysis. At present, DISCOV only provides a rough position estimate of the object of interest. For keyframe extraction or video segmentation this might suffice, but in some other areas such as high quality editing it might be of interest to obtain a clearer contour segmentation of the image pixels. This might require sophisticated feature detectors in addition to MSERs. We are also investigating applications in spatial data mining tasks where traditionally only the density of feature points were considered, whereas DISCOV is able to handle atomic units with different appearances and thus different identities.

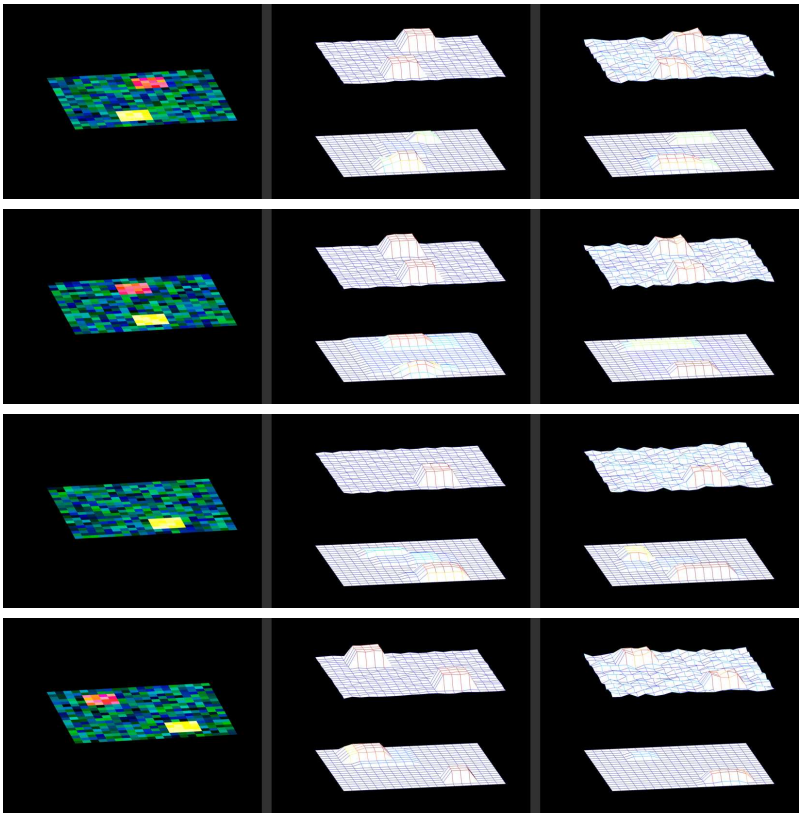


Figure 6.18: Frames 1, 3, 5, 7 shown in rows. Middle and right column: different levels of noise. Top: posterior distribution. Bottom: location distribution after particle filtering. System is robust to noise and occlusion (in frame 5).

Chapter 7

Object-Oriented Retrieval

State-of-the-art video retrieval methods use global image statistics to provide low level descriptors or use object recognizers to provide high level features. Using global image statistics can be hindered by lack of explicitly characterizing the object of interest hence prone to retrieving irrelevant results, while using object recognizers can suffer from having to train a large number of object recognizers for different types of objects.

We present a novel framework for content based video retrieval. We use an unsupervised learning method to automatically discover and locate the object of interest in a video clip. This unsupervised learning algorithm alleviates the need for training a large number of object recognizers. Regional image characteristics are extracted from the object of interest to form a set of descriptors for each video. A novel ensemble-based matching algorithm compares the similarity between two videos based on the set of descriptors each video contains. Videos containing large pose, size, and lighting variations are used to validate our approach.

7.1 Introduction

We present a method for estimating the similarity between two videos based on the *object of interest* each video contains. Our method automatically extracts the object of interest without resorting to any object recognition.

Why do we want to avoid object recognition? Because there is no current algorithm that can handle the large number of objects a human can recognize, not to mention its performance for the objects it is trained to recognize. This poses a problem for video retrieval researchers: should we

wait until the perfect object recognition system is developed so that we can perform video retrieval?

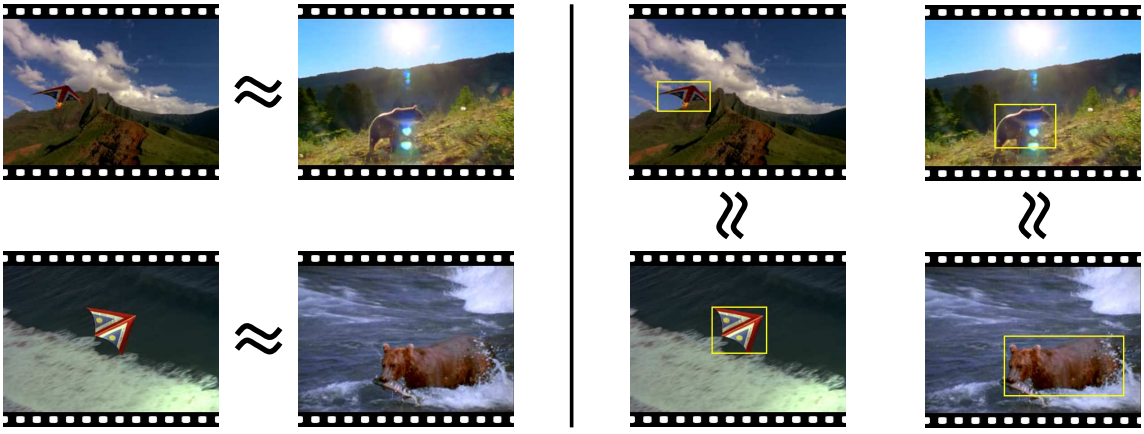
Instead of recognizing the object of interest, which is often the most important factor for retrieving similar videos, researchers instead have been using other less important visual factors such as the global color and texture of each frame. This works well if it is the global property that the user cares about; for example, to distinguish greenish country-side videos from bluish ocean-side videos. However, if it is the object of interest the user cares about, global information can be unreliable because the object of interest often occupies only a small proportion of pixels in a frame and cannot be captured by global image statistics. In Fig.7.1(Left), global image statistics capture the mountain scene and ocean scene and hence relate the four videos horizontally. It cannot find out that the top and bottom videos contain the same type of object.

Our framework discovers the object of interest in each video (hang gliders on the left and bears on the right), thus being able to relate the videos vertically as in Fig.7.1(Right). More precisely, once the object of interest is discovered and located in certain frames, we use a set of local features extracted from the object of interest to represent a video, instead of using global features. Video matching and retrieval involves the ranking of database videos according to their similarity to the query video, where each video can be represented as a set of feature vectors. We propose a novel similarity function that operates on a pair of sets of feature vectors. Distinguished from previous methods, the proposed similarity function also incorporates statistics from the video database. We compare our similarity function with the state of the art and show promising results both in performance and in computation cost.

7.2 Background

Most work in literature on content-based video retrieval relies on global features such as color, texture, or edge descriptors; for example [88]. The use of higher level features to facilitate video retrieval has become popular in TRECVID [89], see example [90][91], where features such as the presence of faces and cars are used. However, reliably extracting these high level features is very difficult even with state-of-the-art object recognizers [92] and, more importantly, it constrains the applicability of video retrieval to scenes containing a very limited number of specific object types.

Our approach of automatically locating the object of interest does not require trained object recognizers. It is general and can handle different types of objects. There are many ways of



Traditional: using global image statistics Proposed: locate the object of interest

Figure 7.1: One frame from each of the four videos is shown. See Sec. 1 for details.

extracting the object of interest. Our approach differs from saliency-based methods [93], since our method focuses on consistency across multiple frames instead of color or shape saliency in a single image. Our work is related to unsupervised learning methods for discovering objects in images [58][5][4] and videos [31]. In [43][94], the correspondence of distinctive feature patches are found across frames and grouped. Our approach also uses distinctive textured patches, but we do not explicitly compute the correspondences across frames, which can be computationally expensive. Our method differs from the 'Video Google' work [51] since we do not rely on a user to manually outline the object of interest to facilitate video retrieval.

Video retrieval involves computing the similarity between videos. Ensemble matching methods [95][96] [97] can be used to compare a set of samples to another set of samples, where here a sample is a feature vector representing the object of interest in a particular frame. However, in video retrieval the problem is not only to match two videos but also to rank the whole database of videos. Hence a better method should take into account the statistics of the whole database while performing video to video matching. In other words, the similarity function should be a function of not only the two videos to be compared, but also the rest of the videos in the database. In Sec.7.3 we present a novel similarity function that has this property. The concept of utilizing database statistics for image retrieval has been exploited in [98], but there it was limited to comparing a feature vector to a feature vector instead of a set of feature vectors to a set of feature vectors. Besides, the method there only applies to binary feature vectors.

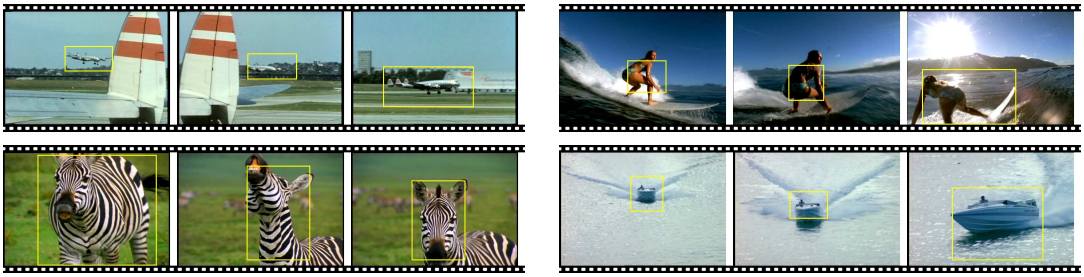


Figure 7.2: Sample frames showing the result of extracting the object of interest. We can handle occlusions, disappearance (top left video), non-rigid motion (top right and bottom left), size variations (bottom right video), and different types of objects. This is achieved without using single-frame color saliency methods or object recognizers. Good frames (according to $P(z_+|d)$) will be used to extract features from.

In Sec.**??** we introduce a framework for localizing the object of interest in a video. Once the object of interest is localized, local features are extracted and compared across videos using the similarity function introduced in Sec.7.3. In Sec.8.5 we show experimental results and conclude with Sec.7.5.

7.2.1 OOI bounding box

The estimated spatial distribution $p(\mathbf{r}|z_+, d)$ tells us the location of the OOI and also provides an estimate of the size of OOI. As shown in Fig.7.2, a bounding box around the OOI is used to specify the region from which we will extract features for video matching. We use a bounding box with size that is twice the standard deviation of the Gaussian distribution $p(\mathbf{r}|z_+, d)$. The choice of using a bounding box versus an elliptical region for feature extraction does not yield significant difference in results.

In the Experiments section we will detail the features we extract from within a bounding box. As mentioned earlier, our framework allows the OOI to disappear or become occluded in some frames. This can be detected by observing the value of $p(z_+|d)$. Frames in which the OOI disappear or being heavily occluded should be excluded from feature extraction. We will detail in the Experiments section how to determine which frames to extract features from.

7.3 Video matching

We use $\{\mathbf{V}\}$ to denote the database of videos and $\mathbf{Q}$ to denote a query video. We represent the i^{th} video $\mathbf{V}^i$ by a set of feature vectors, $\mathbf{V}^i = \left\{\mathbf{v}_1^i, ..., \mathbf{v}_{K_i}^i\right\}$, where K_i is the number of frames we extract features from. Each feature vector $\mathbf{v}_k^i$ has dimension J, i.e., $\mathbf{v}_k^i = \left[v_{k1}^i, ..., v_{kJ}^i\right]^T$. The features we use are histogram features, i.e., the feature values v_{kj}^i are normalized or un-normalized counts of physical properties such as color or texture within an image subregion. Similarly, the query video $\mathbf{Q}$ consists of I feature vectors, $\mathbf{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_I\}$, each vector also with dimension J. Note that the number of feature vectors are generally different for each video.

Next we will discuss how to calculate the similarity between the query video $\mathbf{Q}$ and any database video $\mathbf{V}$. This similarity will be used in a query by example task, where all database videos are ranked according to their similarity to the query video.

7.3.1 Sample-mean matching

A naive way to compute the similarity between $\mathbf{Q}$ and $\mathbf{V}$ is to average the feature vectors within $\mathbf{Q}$ and $\mathbf{V}$ separately and then use any standard similarity measure for vectors. However, information is lost during the averaging process. We call this approach sample-mean matching. A more sophisticated way is to fit the set of feature vectors in $\mathbf{Q}$ and $\mathbf{V}$ separately with a probability distribution, and then measuring the similarity between the two distributions. Yet another method is to perform ensemble matching, as we will detail in the next section.

7.3.2 Ensemble-based matching

Ensemble matching methods [95][96] [97] generally consider the task of obtaining a similarity function which operates on pairs of sets of feature vector, or pairs of *ensembles*. Ensemble matching provides a natural way to calculate the similarity between $\mathbf{Q}$ and $\mathbf{V}$ when a video is considered as an ensemble. The kernel principal angle [96] is the angle between the principal subspaces of two matrices, each matrix composed of feature vectors as columns. We will use it as a baseline method in the experiments.

Proposed method Our method differs in that we take into account the statistics of the database from which the ensembles are drawn. In other words, our proposed similarity function is a function of *three* terms: $\mathbf{Q}$, $\mathbf{V}$, and $\{\mathbf{V}\}$. This idea of taking into account the database statistics is similar

to the work in [98].

Since we use histogram features, we may assume each feature vector within an ensemble follows a multinomial distribution, i.e., $p(\mathbf{v}_k|\boldsymbol{\theta})$ is multinomial with unknown parameters $\boldsymbol{\theta}$. The parameters $\boldsymbol{\theta}$ for different videos can be the same or different; if two videos contain the same object of interest, we assume they share the same parameters $\boldsymbol{\theta}$; if not, we assume they have different parameters. Based on this assumption, we define the similarity function as the ratio

$$R = \frac{p(\mathbf{V}, \mathbf{Q}|\boldsymbol{\theta}_{\mathbf{V},\mathbf{Q}})}{p(\mathbf{V}|\boldsymbol{\theta}_{\mathbf{V}})p(\mathbf{Q}|\boldsymbol{\theta}_{\mathbf{Q}})} \tag{7.1}$$

The numerator can be interpreted as how likely $\mathbf{V}$ and $\mathbf{Q}$ were generated with the same parameters $\boldsymbol{\theta}_{\mathbf{V},\mathbf{Q}}$, i.e., the two videos contain the same object of interest. The denominator says how likely they were generated with different parameters. Hence, the ratio R is a measure of how similar $\mathbf{V}$ and $\mathbf{Q}$ are.

Since the parameters are unknown, we assume a prior distribution over the parameters and integrate them out in the Bayesian fashion. We assume the feature vectors within an ensemble are i.i.d. such that

$$p(\mathbf{V}|\boldsymbol{\theta}) = \prod_k^K p(\mathbf{v}_k|\boldsymbol{\theta}) = \prod_k^K \Big(\frac{(\sum_j^J v_{kj})!}{\prod_j^J (v_{kj}!)} \prod_j^J \theta_j^{v_{kj}} \Big) \tag{7.2}$$

$$p(\mathbf{Q}|\boldsymbol{\theta}) = \prod_i^I p(\mathbf{q}_i|\boldsymbol{\theta}) = \prod_i^I \Big(\frac{(\sum_j^J q_{ij})!}{\prod_j^J (q_{ij}!)} \prod_j^J \theta_j^{q_{ij}} \Big) \tag{7.3}$$

and the parameters $\boldsymbol{\theta}$ follow a Dirichlet distribution,

$$p(\boldsymbol{\theta}) = \frac{\Gamma\Big(\sum_j^J \alpha_j\Big)}{\prod_j^J (\Gamma(\alpha_j))} \prod_j^J \theta_j^{\alpha_j - 1} \tag{7.4}$$

where the hyperparameter $\boldsymbol{\alpha}$ is a vector of dimension J, with α_j being set to the average over the dimension j of all feature vectors within the database, $\{\mathbf{V}\}$. From here we see that the similarity function takes into account the statistics of the whole database instead of only focusing on $\mathbf{V}$ and $\mathbf{Q}$.

It can be shown that the ratio R is as follows:

$$R = c \prod_j^J \left(\frac{\Big(\sum_i^I q_{ij} + \alpha_j\Big)\Big(\sum_k^K v_{kj} + \alpha_j\Big)}{\sum_i^I q_{ij} + \sum_k^K v_{kj} + \alpha_j} \right) \tag{7.5}$$

The constant c is irrelevant to the ranking of videos and can be ignored. The summation over the features of each database video, $\sum_k v_{kj}$, can be computed and stored off-line. This renders the ratio R into the form of R' as follows:

$$R' = \prod_j^J \left(\frac{\left(\sum_i^I q_{ij} + \alpha_j \right) c_j}{\sum_i^I q_{ij} + c_j} \right) \tag{7.6}$$

where c_j is a constant that can be computed and stored off-line. The score R' can hence be computed very efficiently online with complexity linear with respect to J, the dimension of a feature vector. Empirically, the ratio can be computed orders of magnitude faster than the kernel principal angles [96], which we will use as a baseline method. The kernel principal angle method involves heavy Singular Value Decomposition. Even with the simplicity of our method, no precision is sacrificed, as we will show later in the experiments.

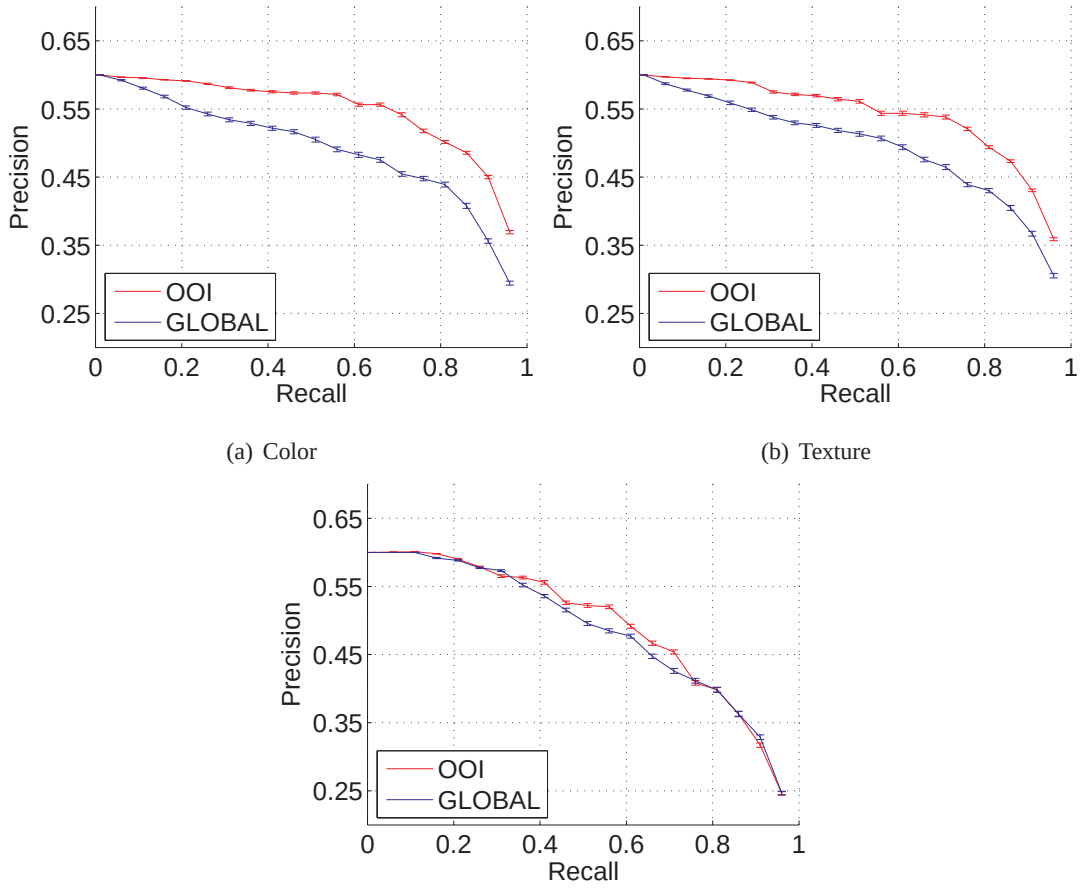
7.4 Experiments

We collected 150 videos from the internet. We sampled each video at two frames per second. The total number of frames is around 20,000. We categorized the videos into 15 categories such as bear, cheetah, giraffe, helicopter, and space shuttle. To evaluate the performance, we randomly select a video as query and use the rest of the videos as database. The database videos are ranked according to similarity to the query video and a recall-precision curve is obtained for each query. This is repeated 100 times.

We have two contributions and want to evaluate them one by one. First, we want to show that extracting features from the object of interest for video retrieval yields better results than using global image statistics. We will call these two approaches OOI and GLOBAL, respectively. Second, we want to show that the proposed similarity function outperforms (1) kernel principal angles and (2) sample-mean matching using Euclidean distance.

7.4.1 OOI versus GLOBAL

We experimented with three different types of features, including (1) color, (2) texture, and (3) MSER+SIFT. The OOI and GLOBAL methods are then compared using the proposed similarity function for retrieval.



(a) Color

(b) Texture

(c) MSER+SIFT

Figure 7.3: Comparing OOI with GLOBAL. OOI consistently performs better. The error bars show the standard error about the mean. See details in Sec.7.4.1.

For color features, we use the CIE-Lab color space and cluster pixels into 36 clusters. In GLOBAL, a frame is represented by a histogram over the entire image. In OOI, a frame is represented by a histogram over pixels inside the bounding box, which is shown by the yellow boxes in Fig.7.2. Results are shown in Fig.7.3(a).

We also experimented with texture features. We use the same filter banks as in [99] in the CIE-Lab color space. The filter responses are clustered into 100 clusters and the same representation as for color features is used. Results are shown in Fig.7.3(b).

For MSER+SIFT features, we first find MSER patches, compute SIFT features and then quantize them, as in Sec. 3.1. The same representation as for color features is used. Results are shown in Fig.7.3(c).

OOI is better than GLOBAL

From Fig.7.3 we see that OOI consistently performed better than GLOBAL, regardless of the choice of features. Fig.7.4 shows some examples. The reason that OOI performs better can be attributed to at least two factors:

1. The GLOBAL method does not have knowledge of the object of interest. Our framework provides the advantage of extracting the most relevant features, that is, features from the object of interest. Since features from background will inevitably distort the similarity measure (unless it *is* the background that we are interested in), extracting features from the whole frame is a disadvantage.

2. The OOI method explicitly models the proportion of patches originated from the object of interest versus the background with $P(z_+|d)$. This information is used to prune away frames that are less likely to contain the object of interest, which is helpful when the object of interest is occluded or disappears due to shooting style or editing. We achieve this by simply retaining the top 50% of frames with larger values of $P(z_+|d)$. In contrast, the GLOBAL method does not possess this information and is hence prone to including more irrelevant background information into features.

Before we conclude this section, we will discuss the features we experimented with.

Since the object of interest is often very small in a frame, and also because of the relatively low resolution of the videos (480×324) and compression artifacts, texture features did not perform significantly better than color features. We expect a combination of texture and color features to

Figure 7.4: Examples where the GLOBAL method performs worse than the OOI method. The **bottom left** pair shows a hang glider in two different videos; since the object of interest is small, global image statistics do not capture the object of interest and matches them closer to ocean related videos and mountainous videos, respectively. The **bottom right** pair shows a windsurfer and a hovercraft; the GLOBAL method considers these two videos similar, while the OOI method is able to differentiate them better.

gain further improvements, but this is not the focus here. Our goal is to demonstrate that OOI is better than GLOBAL.

The visual words based on MSER patches followed by SIFT feature extraction were used in Sec.**??** to locate the object of interest and here again for video retrieval. Fig.7.3 shows that the retrieval performance is worse than using color and texture. We observed that, a feature that is good for locating the object of interest *within* a video is not necessarily good for matching *across* videos. One reason is that different videos within the same category contain the same object class but not necessarily the same object identity. MSER+SIFT features are probably too discriminative for video retrieval tasks so that objects of the same class but with different identities have distinct features, thus having a negative impact on the performance. In the Video Google work [51], similar features were used with good results, but the goal there was to retrieve frames containing the object with the same identity as the one the user manually labels. In that scenario, features that are highly discriminative are desirable.

7.4.2 Comparing similarity functions

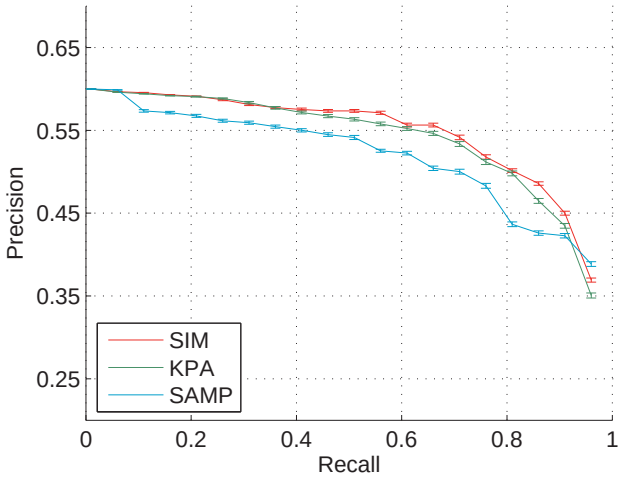


Figure 7.5: The proposed similarity function, SIM, outperforms the state of the art, KPA. It also runs orders of magnitude faster, an important factor for video retrieval applications. The error bars show the standard error about the mean. See details in Sec.7.4.2.

In Fig.7.5 we compare three different similarity functions for video matching using color features and the OOI method. The proposed similarity function (call it SIM) performs the best, fol-

lowed tightly by ensemble matching using kernel principal angles (KPA), and significantly outperforms the naive baseline sample-mean matching using Euclidean distance (call it SAMP). We observed the same result for the other two types of features. It is worth emphasizing that SIM runs orders of magnitudes faster than KPA, because KPA involves Singular Value Decomposition. This offers an important advantage of our proposed similarity function, especially for video matching and retrieval applications.

Computation speed: On a Intel 3.2 GHz Linux machine, processing a 1 min compressed video using color features for video matching takes around 1.5 min for MSER+SIFT extraction, 2 min for extracting the OOI, and 0.5 min for color feature extraction. Video matching using our proposed similarity function is roughly as fast as computing Euclidean distance. The MATLAB code is not optimized for speed yet.

7.5 Conclusion

While object recognition has not yet reached the maturity of handling all types of objects human can recognize, our contribution is to use an unsupervised learning method to extract the object of interest, hence enabling object based retrieval. We demonstrated that using the automatically located object of interest one can perform better video retrieval than using global image statistics.

We presented a new ensemble matching algorithm and compared it with the state of the art. Our method offers several orders of faster computation without loss of precision. The high precision can probably be attributed to the fact that it takes into account the statistics of the database from which ensembles are drawn. It would be of future interest to apply this algorithm to other problems, such as object recognition from videos.

Part IV

Utilizing Information from Human

Chapter 8

Discovery with Frame-Level Labeling

8.1 Introduction

The endless streams of videos on the Internet often contain irrelevant data. Our goal is to cut video clips shorter and retain the frames that are relevant to the user input. We assume the user has an "*object of interest*" (OOI) in mind, which can, for example, be a car, a book, or the scene of a forest. The system will infer which frames contain the OOI. This application can be used, e.g., for shortening surveillance videos or TV programs.

We propose a novel method for removing irrelevant frames from a video given user-provided frame-level labeling for a very small number of frames. We first hypothesize a number of candidate areas which possibly contain the object of interest, and then figure out which area(s) truly contain the object of interest. Our method enjoys several favorable properties. First, compared to approaches where a single descriptor is used to describe a whole frame, each area's feature descriptor has the chance of genuinely describing the object of interest, hence it is less affected by background clutter. Second, by considering the temporal continuity of a video instead of treating the frames as independent, we can hypothesize the location of the candidate areas more accurately. Third, by infusing prior knowledge into the topic-motion model, we can precisely follow the trajectory of the object of interest. This allows us to largely reduce the number of candidate areas and hence reduce the chance of overfitting the data during learning. We demonstrate the effectiveness of the method by comparing it to several other semi-supervised learning approaches on challenging video clips.

We consider the case where the system is provided with very limited information. Specifically, the user will label at least one frame as relevant and another frame as irrelevant. These labels are at the frame-level instead of at the pixel-level. Although pixel-level labeling (such as using a bounding box or segmentation mask to specify the location of the OOI) can provide more information, we intend to explore the possibility of letting the user provide coarser and less tedious labeling.

We formulate the task as a self-training multiple instance learning problem. For each frame, we postulate a number of candidate areas, and use a multiple instance learning algorithm to simultaneously find out whether the OOI exists in the frame, and if it does, where it is located. The reason that we go one step beyond our goal (that is, trying to locate the OOI) is because we are able to exploit the temporal smoothness property of video objects to consolidate their locations. That is to say, objects tend to move in a continuous manner from frame to frame.

We use sporadically labeled frames to train a multiple instance learning algorithm called MILBoost [100]. It was originally applied to a face detection problem. In their work, images are manually labeled by drawing a rectangle around the head of a person. In our system, we only have frame-level labels, i.e., no rectangles are available.

Our semi-supervised framework can be distinguished from prior work in several aspects. Our work does not require pixel-level labeled data. In[101], learning requires both pixel-level labeled data and frame-level labeled data. An object detector is initially trained on the pixel-level labeled data, and the learned model is used to estimate labels for the frame-level labeled data. As illustrated in Fig. 9.1, we "**discover**" the OOI since no bounding box is given, which also distinguishes our work with the video object retrieval work in [51][94], where the OOI is explicitly labeled at the pixel-level.

Image retrieval systems often allow users to provide positive and negative feedback, hence the task of image retrieval can also be cast under the self-training [102] or multiple instance learning [100] framework. Nonetheless, our system exploits temporal information of videos in a novel way, which distinguishes itself from the image retrieval literature. In [103], activities in a video are condensed into a shorter period by simultaneously showing multiple activities. It does not intend to **discover** the frames that contain the user-desired OOI from limited user input.

Our method is based on the bag-of-words representation, which is part-based. Different than other part-based methods such as the one-shot learning framework [104], we leverage motion consistency to improve recognition, while the one-shot learning framework did not utilize that.

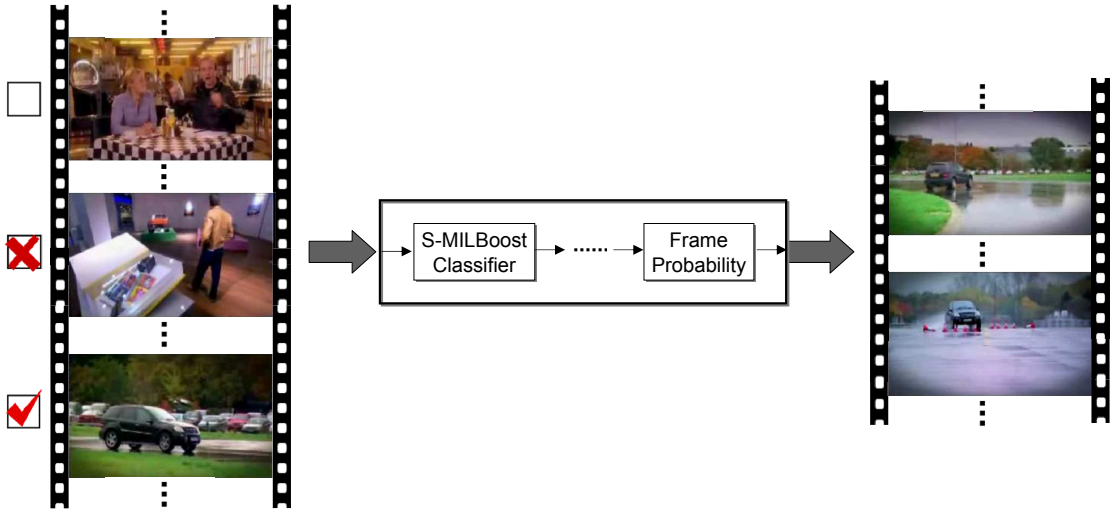


Figure 8.1: Frames are unlabeled (top left), labeled as irrelevant (middle left) or relevant (bottom left). The system will find out what the object of interest is (in this case, the black vehicle) and remove frames that don't contain the vehicle.

Our contribution can hence be summarized as follows: **1)** A novel application that summarizes videos based on the implicitly specified OOI. **2)** A novel system that uses weakly labeled data for object discovery in video. **3)** A novel method that takes advantage of the temporal smoothness property during semi-supervised learning.

In section 8.2 we define the type of user labeling information that is available to the system. In section 8.3 we introduce a baseline method, where features at the frame-level are used for semi-supervised learning. In section 8.4 we explain in detail the proposed method. In section 8.5 we will compare the proposed method with the baseline method and several variants of the proposed method. Finally, we conclude in section 8.6.

8.2 Frame-level labels

The amount of user label information as well as its format has a major impact on system design. The amount of user label information can range from all frames being labeled to none. For those frames being labeled, the labeling can be as detailed as providing bounding boxes for each frame (which we call pixel-level labeling), or as coarse as "this frame does (or does not) contain the OOI" (which we call frame-level labeling).

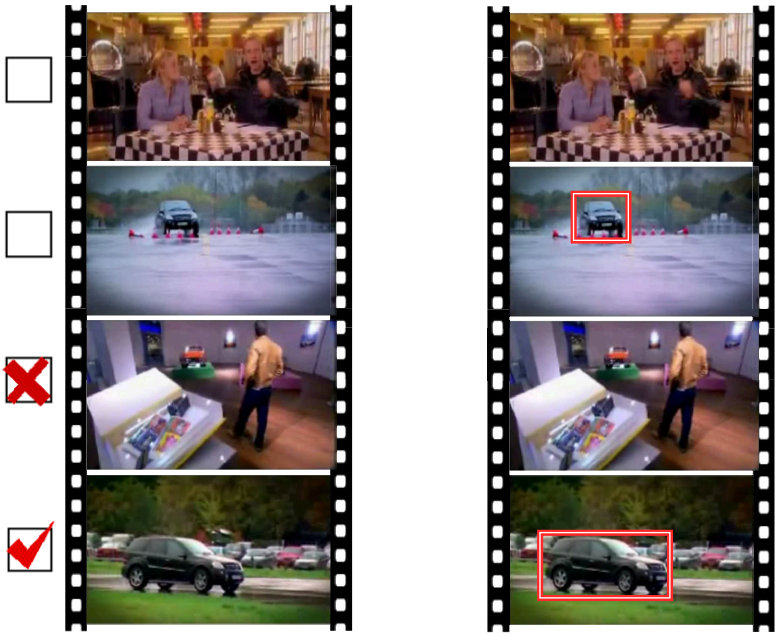


Figure 8.2: (a) Labeling at the frame level assumed in this work. Frames can be unlabeled, or labeled as positive or negative. (b) The bounding box type of labeling provides more explicit information regarding the object of interest, but is also more tedious in the labeling process.

Here we consider the more challenging task of having as input only frame-level labeling; see Fig. 8.2 for a comparison. This kind of 'weak labeling' is very different from traditional object detection; see for example [35], where the characteristics of the OOI are learned from plenty of pixel-level labeled data. This is also different from the recent video retrieval work in [51][94]. Traditional object detection not only involves a lot of human labor for labeling the images by putting bounding boxes on the OOI, but also has the difficulty of scaling to multiple categories of objects. Since the OOI in a sequence can be of any category, it is very difficult to train a comprehensive object detector that covers all types of objects.

8.3 Semi-supervised learning at frame-level

Our first attempt to achieve the goal of VideoCut is to use semi-supervised learning at the frame-level. Each frame is represented as a histogram of *visual words*, or *textons* [105]. To generate

visual words, we use the Maximally Stable Extremal Regions (MSER) operator [63] to find salient patches [1]. MSERs are the parts of an image where local contrast is high. Other operators could also be used; see [1] for a collection. Features are extracted from these MSERs by Scale Invariant Feature Transform (SIFT) [2]. In this work we extract MSERs and SIFT descriptors from grayscale images. Patches and features extracted from color images [106] can also be used instead. The SIFT features from a video are vector quantized using K-Means Clustering. The resulting $J = 50$ cluster centers form the dictionary of visual words, $\{w_1, ..., w_J\}$. Each MSER can then be represented by its closest visual word.

The histograms of the labeled frames along with their labels are fed to the system to train a classifier. The classifier is then applied to the unlabeled frames. Frames with high confidence scores are assigned pseudo-labels. The pseudo-labeled data is combined with the original labeled data and the classifier is trained again. The classifier we use is Discrete AdaBoost [107]. We will use this method as a baseline method in the experiments. This kind of self-training [102] procedure has been used extensively in different domains [108][101] and achieved top results in the NIPS competition [107].

8.4 Semi-supervised learning at sub-frame level

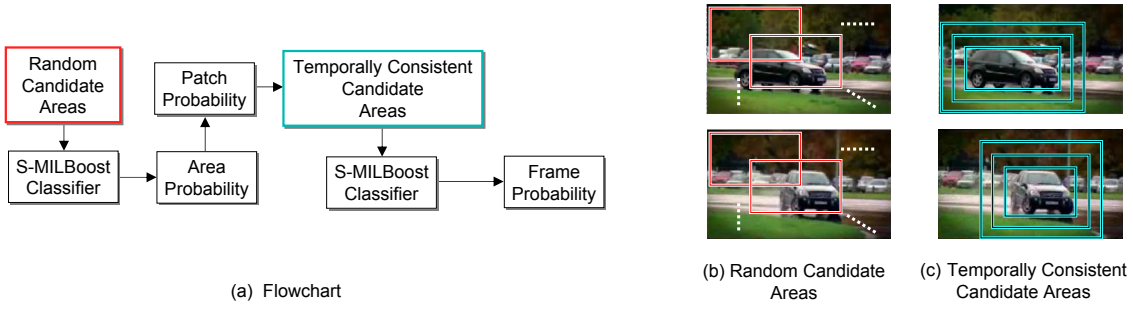


Figure 8.3: Semi-supervised learning at sub-frame level using temporally consistent candidate areas.

There are two issues with the frame-level learning framework in Sec. 8.3.

[1]The word 'region' should not be confused with the 'candidate areas' to be introduced later. Each candidate area contains a set of MSER patches.

1. The OOI can be small and the visual words from the whole frame are usually dominated by background clutter. Hence the full-frame histogram representation is not a truthful representation of the OOI.
2. Objects in video often follow a smooth trajectory, which we call the *temporal smoothness property*. With frame-level learning, the temporal smoothness property cannot be readily exploited.

We address these issues by learning at a sub-frame level. Fig. 8.3(a) shows the proposed system flowchart. In each frame, we propose a number of *Random Candidate Areas* that potentially contain the OOI (illustrated in Fig. 8.3(b)). This will be detailed in section 8.4.1. The candidate areas are passed to a self-training version of MILBoost (S-MILBoost) and assigned an *Area Probability*, a score that tells us how likely this candidate area truly belongs to the OOI. This will be detailed in section 8.4.2. After each candidate area receives a score, we assign each image patch (MSER) a *Patch Probability*, which is defined as the largest *Area Probability* among the candidate areas that cover that image patch. Given the *Patch Probability*, in section 8.4.3 we will explain how to obtain the *Temporally Consistent Candidate Areas*. Basically, this is achieved by fitting a model which simultaneously *discovers* the OOI and *tracks* it across frames. The *Temporally Consistent Candidate Areas* are illustrated in Fig. 8.3(c); using them, we train S-MILBoost once again. As we will show in the experiments, this new S-MILBoost classifier will be more reliable than the previous one trained with the *Random Candidate Areas*. Finally, the S-MILBoost classifier gives us the *Frame Probability*, which tells us how likely each frame contains the OOI. Using the *Frame Probability*, we can determine the irrelevant frames and perform VideoCut.

Notice how the two issues mentioned earlier are resolved by using this proposed flowchart. First, the candidate areas are smaller than the whole frame and hence include less background clutter, which address the first issue mentioned above. Second, the candidate areas in one frame can be temporally correlated with the candidate areas in the next frame by performing 'weak' object tracking (illustrated in Fig. 8.3(c)), which addresses the second issue. We emphasize that this 'weak' tracking is different from traditional object tracking, as we will explain later.

In the experiments section we will compare our proposed flowchart with some other methods, which replace or omit some parts of the modules in Fig. 8.3(a). In the following subsections we will explain the details and merits of each component in Fig. 8.3(a).

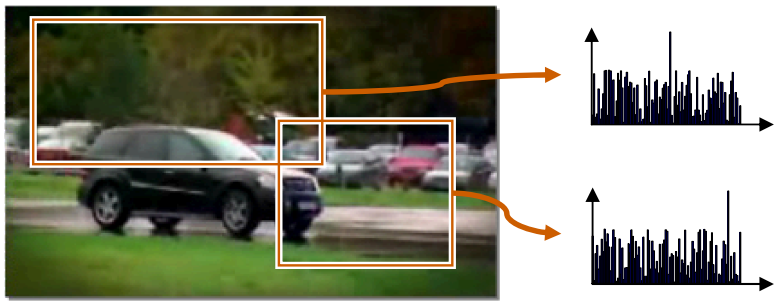


Figure 8.4: Candidate areas, each represented by a histogram over visual words. In the experiments, we use a variety of different densities and spacings of candidate areas.

8.4.1 Random candidate areas

Since the user labeling does not tell us where the OOI is located (neither in the labeled nor in the unlabeled frames), we need to set the candidate areas based on prior knowledge, if any. At the beginning, we use candidate areas with fixed size and uniform spacing and call them the random candidate areas. Each candidate area is represented as a histogram of visual words, as shown in Fig. 8.4. After we have a rough guess (using the techniques in the next two subsections), we will refine the candidate areas by placing them more densely around the estimated location of the OOI. We call these later candidate areas as temporally consistent candidate areas. See Fig. 8.3(b)(c) for illustrations.

8.4.2 Self-training MILBoost

Using a similar self-training procedure as in Sec. 8.3, we first use the labeled frames to train a multiple instance learning [100] classifier. As a result, each candidate area of the labeled frames is assigned an area probability, which is the probability that an area contains the OOI. The classifier is then self-trained with the unlabeled frames and pseudo-labels included. As a result, the area probabilities of candidate areas in unlabeled frames are obtained as well.

Different than in Sec. 8.3, we have multiple histograms per frame, instead of a single one, therefore we use a multiple instance learning classifier, MILBoost [100]. First let us define some notations. We denote the histogram over visual words of a candidate area as $x_{k,a}$, where k indices over frames and a indices over the candidate areas inside frame k. Let $t_k \in \{0, 1\}$ denote the label or pseudo-label of frame k. Each frame has a *frame probability* p_k, and each candidate area has

an *area probability* $p_{k,a}$. The *frame probability* is the probability that a frame contains the OOI, and the *area probability* is the probability that the area contains the OOI. Since a frame is labeled as positive as long as it contains the OOI, it is natural to model the relationship between p_k and $p_{k,a}$ using the Noisy-OR model [109], $p_k = 1 - \prod_{a \in k}(1 - p_{k,a})$. The likelihood is given by $L(C) = \prod_k {p_k}^{t_k}(1 - p_k)^{(1-t_k)}$.

As implied by its name, MILBoost produces a strong classifer $C(x_{k,a})$ in the form of a weighted sum of weak classifiers: $C(x_{k,a}) = \sum_u \lambda_u c_u(x_{k,a})$, $c_u(x_{k,a}) \in \{-1, +1\}$. The strong classifier score $C(x_{k,a})$ translates into the area probability, $p_{k,a}$, by the logistic sigmoid function $p_{k,a} = 1/(1 + exp(-C(x_{k,a})))$. Using the AnyBoost [110] method, the boosting weight $\varpi_{k,a}$ of each candidate area is the derivative of the log-likelihood, easily to be shown as $\frac{t_k - p_k}{p_k} p_{k,a}$. In round u of boosting, one first solves the optimization problem $c_u(.) = \arg\max_{c'(.)} \sum_{k,a} c'(x_{k,a})\varpi_{k,a}$. A line search is then performed to seek for the optimal parameter λ_u, i.e., $\lambda_u = \arg\max_\lambda L(C + \lambda c_u)$.

In summary, S-MILBoost produces a classifier that assigns each frame a frame probability, and each candidate area an area probability. Notice that the S-MILBoost classifier is always used in a learning mode, during which the area and frame probabilities are estimated.

8.4.3 Temporally consistent candidate areas

The accuracy of the frame probabilities depends heavily on the placing of the candidate areas; as an extreme example, if the OOI appears in a frame but none of the candidate areas cover it, then there would be no chance we could have correctly estimated the frame probability. This suggests a refinement of the placing scheme of candidate areas based on extra information. Notice that, we haven't yet exploited the temporal smoothness property of videos.

We would like to use the temporal smoothness property to refine the placing of the candidate areas. The temporal smoothness property is typically exploited through tracking the object. However, tracking requires manual initialization of the object location and size, information which is not available to us.

The topic-motion model [31] simultaneously estimates the appearance and location of the OOI. However, it was used in an unsupervised setting where one has no prior knowledge about the label (object vs. background) of each image patch. In our case, the area probabilities estimated by S-MILBoost provides information that we could use as prior knowledge.

The topic-motion model was designed for the case where at most one OOI appears in each

frame. But this is not a problem for our system, because as long as one of the possibly many OOIs is discovered, the frame probability will be high. In other words, we don't need to identify every OOI to decide if a frame is relevant or irrelevant. Also notice that discovering the OOI is not our ultimate goal.

Denote frame k as d_k, where k indices over all frames. Each patch in d_k is associated with a visual word w, a position $\mathbf{r}$, and a hidden variable $z \in \{z_+, z_-\}$. Define $p(z_+|d_k)$ as the probability of a patch being originated from the OOI in frame k, and likewise $p(z_-|d_k)$ for the background. We define a spatial distribution $p(\mathbf{r}|z_+, d_k)$ that models the location of the patches originated from the OOI. We assume $p(\mathbf{r}|z_+, d_k)$ follows a Gaussian distribution, but other distributions (such as a mixture of Gaussians) could be used as well. Likewise, $p(\mathbf{r}|z_-, d_k)$ models the location of patches originated from background and we assume it follows a uniform distribution. The third distribution is $p(w|z_+)$, which models the appearance of the OOI. It is the normalized histogram over visual words corresponding to patches originated from the OOI. Likewise, $p(w|z_-)$ models the appearance of the background. We assume that the joint distribution of word w, position $\mathbf{r}$, and hidden label z of a patch in frame d_k is modeled as $p(z, \mathbf{r}, w|d_k) \equiv p(z|d_k)p(\mathbf{r}|z, d_k)p(w|z)$.

Define the state $\mathbf{s}(k)$ as the unknown position and velocity of the OOI in frame d_k. We assume a constant velocity motion model and the state evolves according to $\mathbf{s}(k+1) = \mathbf{F}\mathbf{s}(k) + \boldsymbol{\xi}(k)$, where $\mathbf{F}$ is the state matrix and the process noise sequence $\boldsymbol{\xi}(k)$ is white Gaussian. Suppose at time k there are a number of m_k patches. If a patch is originated from the OOI, then its position can be expressed as $\mathbf{r}_i(k) = \mathbf{H}\mathbf{s}(k) + \boldsymbol{\zeta}_i(k)$, where $\mathbf{H}$ is the output matrix and the observation noise sequence $\boldsymbol{\zeta}_i(k)$ is white Gaussian; otherwise, the position is modeled as a uniform spatial distribution. The state estimate can be written as $\hat{\mathbf{s}}(k) = \sum_{i=1}^{m_k} \hat{\mathbf{s}}_i(k)\beta_i(k)$, where $\hat{\mathbf{s}}_i(k) = \hat{\mathbf{s}}(k^-) + \mathbf{W}(k)\boldsymbol{\epsilon}_i(k)$ is the updated state estimate conditioned on the event that $\mathbf{r}_i(k)$ is originated from the OOI, where $\boldsymbol{\epsilon}_i(k) = \mathbf{r}_i(k) - \hat{\mathbf{r}}(k^-)$ is the innovation, $\hat{\mathbf{r}}(k^-)$ is the observation prediction, $\hat{\mathbf{s}}(k^-)$ is the state prediction, and $\mathbf{W}(k)$ is the Kalman Filter gain [65]. The state estimation equations are essentially the same as in the PDA filter [65]. The association probability $\beta_i(k)$ is defined as $\beta_i(k) \propto N(\boldsymbol{\epsilon}_i(k)|0, \boldsymbol{\Upsilon}(k))p(z_i(k)|w_j, \mathbf{r}_i(k), d_k)$, where the first term contains motion information, the second term contains appearance and location information, and $\boldsymbol{\Upsilon}(k)$ is the innovation covariance.

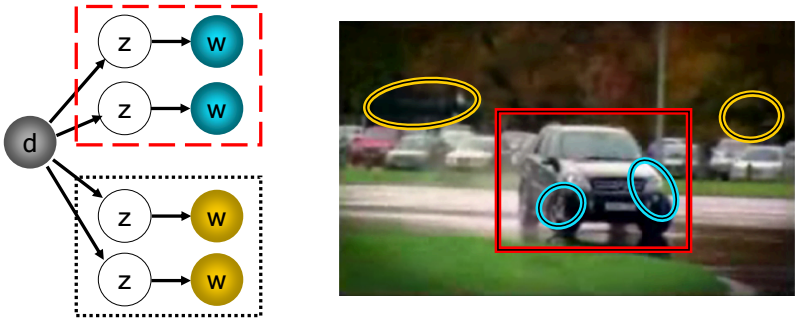


Figure 8.5: Graphical model representation. Dashed lines are not the typical plate representation.

Parameter estimation

The distributions $P(w|z)$, $P(z|d)$, and $P(\mathbf{r}|z,d)$ are estimated using the Expectation-Maximization (EM) algorithm [21], which maximizes the log-likelihood $\mathcal{R} = \sum_k \sum_j \sum_i n_{kji} \log p(d_k, w_j, \mathbf{r}_i(k))$, where $n_{kji} \equiv n(d_k, w_j, \mathbf{r}_i(k))$ is a count of how many times a patch in d_k at position $\mathbf{r}_i(k)$ has appearance w_j. The EM algorithm consists of two steps. The E-step computes the posterior probabilities for the hidden variables:

$$p(z_l|d_k, w_j, \mathbf{r}_i(k)) = \frac{p(z_l|d_k)p(w_j|z_l)p(\mathbf{r}_i(k)|z_l, d_k)}{\sum_R p(z_l|d_k)p(w_j|z_l)p(\mathbf{r}_i(k)|z_l, d_k)} \tag{8.1}$$

The M-step maximizes the expected complete data likelihood. We adopt a Bayesian approach to estimating the probabilities, using *m*-probability-estimation [111]. First, notice that the *area probability*, $p_{k,a}$, computed from S-MILBoost contains prior knowledge about the OOI. This prior knowledge should be incorporated into the detection of temporally consistent candidate areas. This is a significant improvement over the algorithm in [31], which was completely unsupervised.

Noticing that each patch can belong to multiple candidate areas, we define the *patch probability* as the largest *area probability* among the candidate areas that cover an image patch. The *patch probability* is written as $p_{MIL}(z_l|d_k, w_j, \mathbf{r}_i(k))$, with the subscript "MIL" emphasizing that this probability is estimated from the outcome of S-MILBoost. A simplified graphical model is illustrated in Fig. 8.5, where the variable $\mathbf{r}$ is omitted to simplify illustration. Dashed lines indicate groups of image patches having the same value of p_{MIL}. More specifically, dashed lines in red correspond to the red box (candidate area) in the picture, and blue (yellow) nodes in the graphical

model correspond to blue (yellow) ellipses in the picture. We then obtain:

$$p(z_l|d_k) = \frac{\sum_{j,i} n_{kji} p_{MIL}(z_l|d_k, w_j, \mathbf{r}_i(k)) + \sum_{j,i} n_{kji} p(z_l|d_k, w_j, \mathbf{r}_i(k))}{\sum_{l,j,i} n_{kji} p_{MIL}(z_l|d_k, w_j, \mathbf{r}_i(k)) + \sum_{l,j,i} n_{kji} p(z_l|d_k, w_j, \mathbf{r}_i(k))} \tag{8.2}$$

$$p(w_j|z_l) = \frac{\sum_{k,i} n_{kji} p_{MIL}(z_l|d_k, w_j, \mathbf{r}_i(k)) + \sum_{k,i} n_{kji} p(z_l|d_k, w_j, \mathbf{r}_i(k))}{\sum_{j,k,i} n_{kji} p_{MIL}(z_l|d_k, w_j, \mathbf{r}_i(k)) + \sum_{j,k,i} n_{kji} p(z_l|d_k, w_j, \mathbf{r}_i(k))} \tag{8.3}$$

$$p(\mathbf{r}_i(k)|z_+, d_k) = \mathcal{N}(\mathbf{r}_i(k)|\hat{\mathbf{r}}(k), \mathbf{\Sigma}_{d_k}) \tag{8.4}$$

where $z_l \in \{z_+, z_-\}$ is the value taken by $z_i(k)$ and $\hat{\mathbf{r}}(k) = \mathbf{H}\hat{\mathbf{s}}(\mathbf{k})$ is the position estimate. The covariance $\mathbf{\Sigma}_{d_k}$ in the Normal distribution in Eq.(8.4) is the weighted covariance matrix of the observations $\mathbf{r}_i(k)$. The weighted covariance matrix is the covariance matrix with a weighted mass for each data point, with weights equal to the association probabilities $\beta_i(k)$. As a result, if the association probabilities have high uncertainty, the spatial distribution $p(\mathbf{r}|z_+, d)$ will be flatter; if low uncertainty, it will be sharper around the position of the OOI.

Finally, we propose a number of temporally consistent candidate areas that have $\hat{\mathbf{r}}(k)$ as center and with various sizes, as shown in Fig. 8.3(c). We use a 1.2 scale ratio between two areas, with the smallest one equal to the variance specified by $\mathbf{\Sigma}_{d_k}$ in Eq.(8.4), and with no more than 5 areas in total. Using various sizes is to increase system robustness in case of inaccurate size estimates.

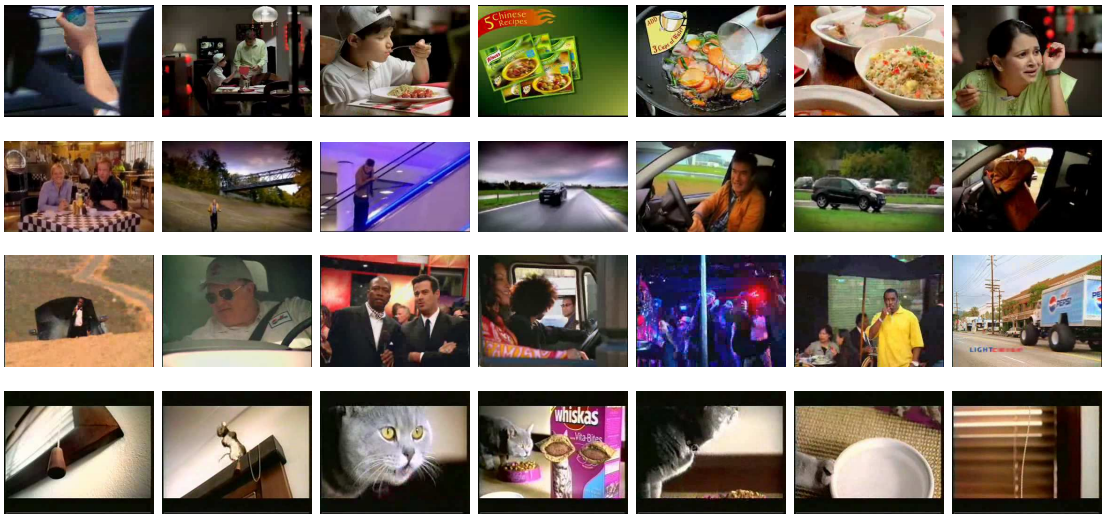


Figure 8.6: Sample frames. Name of video clip, from top to bottom: Knorr, Benz, Pepsi, Whiskas.

8.5 Experiments

We use 15 video clips from YouTube.com and TRECVID [89]. Sample frames are shown in Fig. 8.6. Most of the clips are commercial advertisements with a well defined OOI and range from 20 to 356 seconds in length. We sample each video at two frames per second. In total, there are 3128 frames of size 320×240. The frames have visible compression artifacts.

The video frames are ground-truthed as positive or negative according to whether they contain the OOI; e.g., in a PEPSI commercial, we assume the PEPSI logo is the OOI. Each video clip is run twenty runs, where in each run we randomly select N_p frames from the positive frames and N_n frames from the negative frames as labeled data, where N_p and N_n are one or three. The rest of the frames are treated as unlabeled data. Results are averaged over the twenty runs. Notice that the labeled frames are labeled at the frame-level but not pixel-level.

Table 8.1 shows the average precision (area under precision-recall curve) of different methods. In the following, we will introduce the different comparative methods listed in Table 8.1 while we discuss the results. In general, we have the following observations:

Method 1: Supervised learning using only labeled data is consistently outperformed by the semi-supervised variants. When the number of labeled frames is low, its performance is close to by chance.

Method 2: Semi-supervised learning at frame level performs only marginally better than supervised learning when the number of labeled frames is as low as $(1+, 1-)$, but improves significantly as the number of labeled frames increases.

Method 3: Semi-supervised learning at sub-frame level with random areas consistently outperforms semi-supervised learning at the frame level. This justifies our claim in Sec. 8.4 that frame-level learning can be hindered when background clutter dominates the appearance features. Using sub-frames (candidate areas) helps the learning process to focus on the features originated from the OOI. The candidate areas consist of rectangles of size 160×120 with equal spacing between each other. In addition, a rectangle of size 320×240 covering the whole frame is used in here, in Method 4, and in the proposed method, in order to take care of large objects and inaccurate size estimates. After training S-MILBoost, we did not refine the placing of candidate areas, as we do in Method 4 and in the proposed method.

We experimented with different numbers of rectangles by changing the spacing between them and obtained different performances as shown in Fig. 8.7. There is a sweet spot at the number of

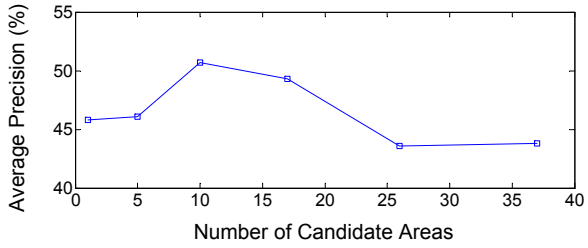


Figure 8.7: Increasing the number of areas does not lead to increase in performance.

10 areas, which shows that the more candidate areas does not necessarily yield better performance. Even though increasing the number of areas will increase the chance that one of the candidate areas faithfully represents the OOI, the chance of overfitting also increases, hence the drop in performance. We also experimented with placing the areas more concentrated around the center of the frame but obtained similar results.

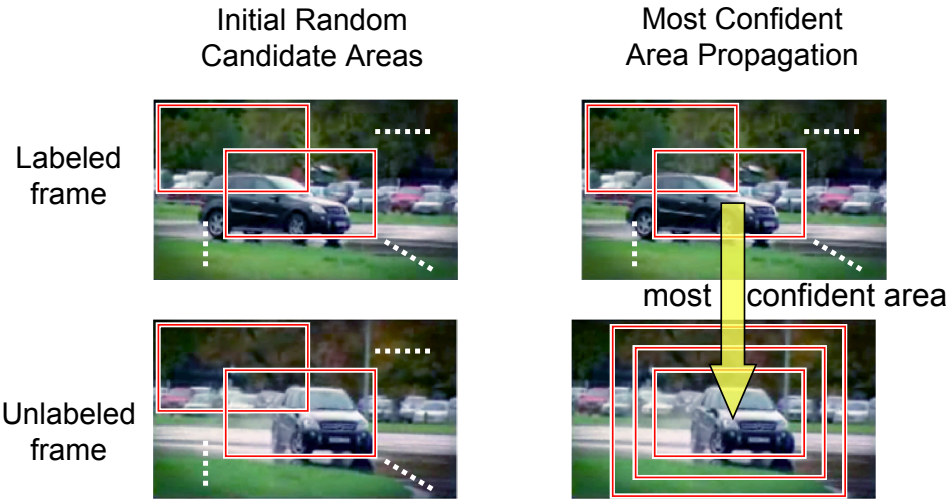


Figure 8.8: Illustration of Method 4.

Method 4: Most confident area propagation: This method is the closest to the proposed method. Instead of using 'weak' tracking, we assume the OOI is stationary within a shot. As illustrated in Fig. 8.8, each unlabeled frame obtains its 'base' candidate area by replicating, from the nearest labeled frame, the size and position of the most confident area. Nearness can be defined as the visual similarity between frames or as the time difference between frames. We found the

latter to work better. The base area is then resized and replicated within the frame using a 1.2 scale ratio between two areas, with the smallest one equal to the size of the base area, and no more than 5 areas in total. Since videos often contain multiple scene transitions or shots, we only allow the replication to happen within a shot and not across shots. If there are no labeled frames within a shot, we place random candidate areas in that shot.

In summary, the proposed method outperforms all the other methods (Table 8.1). Together with Fig. 8.7, this justifies our earlier expectation that properly placed candidate areas are crucial to the performance; using a huge number of candidate areas overfits the data and lowers the performance. The temporally consistent candidate areas reduce the need for a large number of uninformative candidate areas. Finally, in Fig. 8.9, we display some frames that are inferred by the proposed method.

Figure 8.9: Sample frames that are inferred as positive. A yellow box shows the candidate area with highest area probability. Name of video clip, from top to bottom: Knorr, Benz, Pepsi, Whiskas.

8.6 Conclusion

We have presented an approach for removing irrelevant frames in a video by discovering the object of interest. Through extensive experiments, we have shown that this is not easily achieved by directly applying supervised or semi-supervised learning methods in the literature developed for still images.

On a higher level, our method can be considered as a tracking system but without manual track initialization; the system finds out itself what the "best track" is, with the objective of agreeing with the user's labeling on which frames contain the object of interest.

Sequence	Label	By Chance	Method 1 Supervised	Semi-Supervised			
				Method 2 Frame Level	Sub-Frame Level		
					Method 3	Method 4	Proposed
Benz	1+,1-	32.6	26.0	28.9	31.7	29.3	38.7
	3+,3-	32.5	29.1	52.9	54.6	48.8	58.3
Pepsi	1+,1-	34.1	32.6	34.3	41.9	39.1	42.3
	3+,3-	33.7	39.4	53.9	57.7	50.6	63.2
Whiskas	1+,1-	43.7	49.0	54.2	62.6	64.1	65.3
	3+,3-	43.5	53.9	71.2	77.0	78.1	73.8
SkittlesFunny	1+,1-	3.9	2.8	5.2	10.7	10.7	6.5
	3+,3-	2.0	4.1	11.3	21.2	22.5	22.7
CleanClear	1+,1-	21.2	14.4	15.5	45.4	41.8	36.1
	3+,3-	19.4	21.1	41.9	51.4	57.6	62.2
CatFood	1+,1-	39.0	40.5	41.7	62.7	65.1	66.9
	3+,3-	38.2	58.2	76.0	91.4	91.4	91.4
E-Aji	1+,1-	27.1	26.5	27.0	31.4	29.9	36.0
	3+,3-	25.5	23.8	32.6	42.7	34.7	36.2
CaramelNut	1+,1-	25.9	39.3	53.7	67.6	58.9	58.9
	3+,3-	24.1	58.4	67.5	67.6	70.2	70.2
Knorr	1+,1-	20.7	20.4	32.2	44.2	62.1	59.4
	3+,3-	18.5	20.2	48.9	57.2	69.4	67.7
Kellogs	1+,1-	18.4	19.8	20.0	26.4	30.3	30.3
	3+,3-	14.7	18.6	22.1	25.3	36.4	38.0
FlightSimul	1+,1-	10.8	15.4	43.5	42.6	53.5	59.6
	3+,3-	10.5	18.7	50.7	44.4	40.8	62.1
SpaceShuttle	1+,1-	4.8	2.8	2.8	3.5	3.7	4.2
	3+,3-	4.2	3.6	12.7	27.7	27.3	25.1
WeightAero	1+,1-	11.6	8.5	38.1	27.8	33.9	44.9
	3+,3-	11.2	46.9	56.3	40.9	48.5	56.1
WindTunnel	1+,1-	24.1	14.7	15.0	36.1	33.8	35.2
	3+,3-	23.8	41.6	47.2	56.9	56.7	56.1
Horizon	1+,1-	11.2	15.8	18.4	22.6	28.3	34.1
	3+,3-	10.5	18.0	41.3	44.1	48.9	54.6
Average	1+,1-	**21.9**	**21.9**	**28.7**	**37.1**	**39.0**	**41.2**
	3+,3-	**20.8**	**30.4**	**45.8**	**50.7**	**52.1**	**55.8**

Table 8.1: Comparing the average precision (%). The number of labeled frames are one positive (1+) and one negative (1−) in the upper row, and three positives and three negatives in the lower row for each video sequence.

Chapter 9

Integrated Feature Selection and Extraction

9.1 Introduction

In computer vision, the bag-of-visual words image representation has been shown to yield good results. Recent work has shown that modeling the spatial relationship between visual words further improves performance. Previous work extracts higher-order spatial features exhaustively. However, these spatial features are expensive to compute. We propose a novel method that simultaneously performs feature selection and feature extraction. Higher-order spatial features are *progressively* extracted based on selected lower order ones, thereby avoiding exhaustive computation. The method can be based on any additive feature selection algorithm such as boosting. Experimental results show that the method is computationally much more efficient than previous approaches, without sacrificing accuracy.

The traditional pipeline of pattern recognition systems consists of three stages: feature extraction, feature selection, and classification. These stages are normally conducted in independent steps, lacking an integrated approach. The issues are as follows: 1. **Speed**: Feature extraction can be time consuming. Features that require extensive computation should be generated only when needed. 2. **Storage**: Extracting all features before selecting them can be cumbersome when they don't fit into the random access memory.

Many object recognition problems involve a prohibitively large number of features. It is not

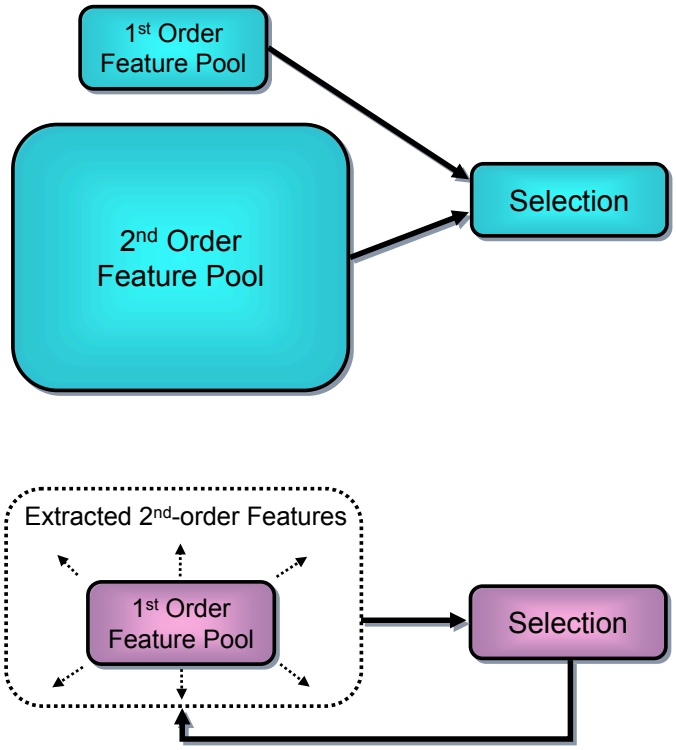


Figure 9.1: The top figure shows the traditional approach where 1^{st} and 2^{nd} order features are extracted before feature selection. Second-order features encode spatial configurations of visual words and are expensive in terms of computation and storage. The proposal is to extract 2^{nd} order features based on previously selected 1^{st} order features and to progressively add them into the feature pool.

uncommon that computing the features is the bottleneck of the whole pipeline. Techniques such as "classifier cascade" [112] reduce the amount of computation for feature extraction in run time (in testing), while the aim here is to improve the feature extraction and selection procedure in training.

In this work, we focus on the *bag-of-local feature descriptors* image representation [113] and its recent extensions [114][115][116]. Local feature descriptors are image statistics extracted from pixel neighborhoods or patches. Recent work of [114][115][116] focused on modeling the spatial relationship between pixels or patches. We call the features originated from local feature descriptors as 1^{st} *order features*, and features that encode spatial relationship between a set of two, three,

or N patches as 2^{nd}, 3^{rd}, or N^{th} order features, respectively. Features with order larger than one are called *higher-order features*. These are analogous to *N-grams* [117] used in statistical language modeling. It is worth mentioning that, by higher-order features, we do **not** mean algebraic expansions (monomials) of lower order ones, such as cross terms ($x_1 x_2$), squares or cubes (x_1^3).

In the recent works of [114][115][116], higher-order features are extracted *exhaustively*. However, these higher-order features are prohibitively expensive to compute: first, their number is combinatorially exploding with the number of pixels or patches; second, extracting them requires expensive nearest neighbor or distance computations in image space [118]. It is the expensive nature of higher-order features that motivates our work.

Instead of *exhaustively* extracting all higher-order features before feature selection begins, we propose to extract them *progressively* during feature selection, as illustrated in Fig. 9.1. We start the feature selection process as early as when the feature pool consists only of 1^{st} order features. Subsequently, features that have been selected are used to create higher-order features. This process dynamically enlarges the feature pool in a greedy fashion so that we don't need to exhaustively compute and store all higher-order features.

A comprehensive review of feature selection methods is given by [119]. Our method can be based on any additive feature selection algorithm such as boosting [120] or CMIM [121][122]. Boosting was originally proposed as a classifier and has also been used as a feature selection method [112] due to its good performance, simplicity in implementation, and ease of extension to multiclass problems [120]. Another popular branch of feature selection methods is based on information-theoretic criteria such as maximization of conditional mutual information [121][122].

9.2 Integrated feature selection and extraction

Each image is represented as a feature vector which dynamically increases in the number of dimensions. Initially, each feature corresponds to a distinct codeword. The feature values are the normalized histogram bin counts of the visual words. These features are the 1^{st} order features, and this is the bag-of-visual words image representation [113]. Visual words, with textons [105] as a special case, have been used in various applications. A dictionary of codewords refers to the clusters of local feature descriptors extracted from pixel neighborhoods or patches, and a visual word refers to an instance of a codeword.

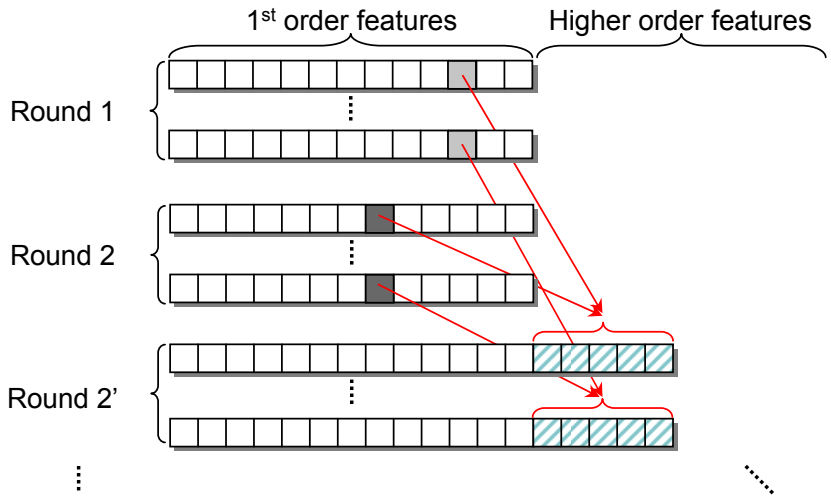


Figure 9.2: The 'feature pool' is dynamically built by alternating between feature selection and feature extraction.

Our method maintains a 'feature pool' which initially consists only of 1^{st} order features. Subsequently, instead of exhaustively building all higher-order features, the process of *feature selection* and *higher-order feature extraction* are run alternately. At each round, *feature selection* picks a feature, and *feature extraction* pairs this feature with each of the previously selected features. The pairing process can be generic, and we will explain the implementation in Sec. 9.3. The pairing process creates new features which are concatenated to the feature vector of each image. In the next round of feature selection, this enlarged 'feature pool' provides the features to be selected from.

In Fig. 9.2, we illustrate this process for the first few rounds. In the first round, *feature selection* picks a feature (the light gray squares) from the 'feature pool' and puts it in a 1^{st} order list (not shown in Fig. 9.2) that holds all previous selected 1^{st} order features. Since the list was empty, we continue to the second round. In the second round, *feature selection* picks a feature (the dark gray squares) from the 'feature pool' and places it in the 1^{st} order list. At the same time, *feature extraction* pairs this newly selected feature with the previously selected feature (the light gray square) and creates new features (the diagonally patterned squares). These 2^{nd} order features are then augmented into the 'feature pool'. In general, we may maintain 1^{st}, ..., L^{th}-order lists instead of only 1^{st} order lists. If a selected feature has order L_1, then it was originated from L_1 codewords, and pairing it with another feature of order L_2 means that we can create new features that originate

from a set of $L_1 + L_2$ codewords.

Algorithm 1: Integrated-Feature-Selection-And-Spatial-Feature-Extraction

1 Sample weights $v_n^{(1)} \leftarrow 1/N$, $n = 1, ..., N$.
2 $k \leftarrow 0$.
3 **for** *m*=1,...,*M* **do**
4 Fit decision stump $y_m(\mathbf{x})$ to training data by minimizing weighted error function $J_m = \sum_{n=1}^{N} v_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$
5 Denote feature index selected by decision stump as $i(m)$
6 **if** $i(m)$ *corresponds to a* 1^{st} *order feature* **then**
7 $k \leftarrow k + 1$
8 $z(k) \leftarrow i(m)$
9 **for** *j=1,...,k-1* **do**
10 **for** *each image* **do**
11 BuildSecondOrderFeatures($z(k), z(j)$)
12 **end**
13 Augment feature pool
14 **end**
15 **end**
16 $\epsilon_m \leftarrow \frac{\sum_{n=1}^{N} v_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} v_n^{(m)}}$ and $\alpha_n \leftarrow \ln \frac{1-\epsilon_m}{\epsilon_m}$
17 $v_n^{(m+1)} \leftarrow v_n^{(m)} \exp\{\alpha_n I(y_m(\mathbf{x}_n))\}$
18 **end**
19 Selected features are $\{\mathbf{x}_{i(1)}, ..., \mathbf{x}_{i(M)}\}$ for any vector $\mathbf{x}$

In Algorithm 1 we detail the procedure of computing features up to the 2^{nd} order. We use Discrete AdaBoost with decision stumps for feature selection as in [112], although other feature selection methods could be used as well. AdaBoost maintains a set of sample weights, $\{v_n\}, n = 1, ..., N$, on the N training images (Line 1). At each round, a decision stump tries to minimize the weighted error rate by picking an optimal feature and threshold (Line 4). The selected feature could be a 1^{st} or 2^{nd} order feature. If it is a 1^{st} order feature, it is placed in the 1^{st}-order list $z(.)$

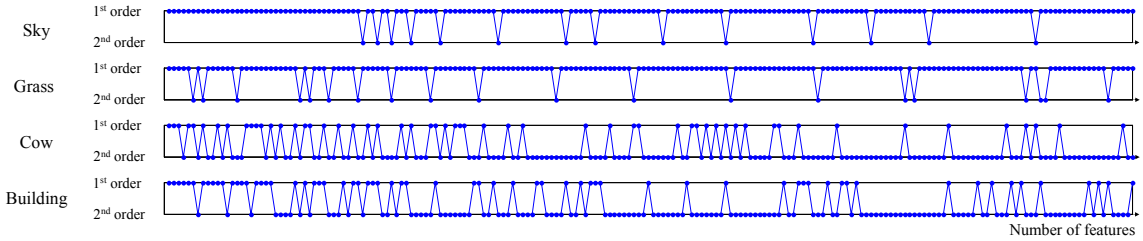


Figure 9.3: The order (1^{st} vs 2^{nd}) of a selected feature in each round.

(Line 8), and then paired with all previous members in the 1^{st}-order list to generate new 2^{nd} order features (Line 11). The new features are augmented into the feature pool (Line 13).

Lines 16 and 17 are standard update rules of AdaBoost. It updates the sample weights in a manner so that the decision stumps can focus on the source of error. This eventually drives the choice of features. Using AdaBoost as a feature selection tool is justified by its taking into account the classification error when selecting features [121]. However, the concept of integrating feature selection and extraction is general, and the feature extraction procedure in lines 6 to 15 can be embedded into other feature selection methods as well.

To show that different object categories result in different temporal behaviors of the integrated feature selection and extraction process, we show in Fig. 9.3 the order of a selected feature at each round of boosting, from rounds 1 to 200. AdaBoost is used in a binary one-vs-rest classification manner. In the first few rounds, 1^{st} order features are being selected and 2^{nd} order features are being built. Structured objects such as 'Cow' and 'Building' soon start to select 2^{nd} order features. At the end, structured objects tend to select more 2^{nd} order features compared to homogeneous objects such as 'Sky'. This agrees with the expectation that sky has less obvious geometrical structure between pairs of 1^{st} order features.

After feature selection and extraction, to make predictions, one can:

1. treat boosting solely as a feature selection tool and use the selected features, $\{\mathbf{x}^{i(1)}, ..., \mathbf{x}^{i(M)}\}$, as input to any classifier; or,

2. proceed as in AdaBoost and use a thresholded weighted sum, $Y(\mathbf{x}) = \text{sign}(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}))$, as the final classifier; or,

3. as we propose, use the set of weighted decision stumps, $\{\alpha_1 y_1(\mathbf{x}), ..., \alpha_M y_M(\mathbf{x})\}$, as features and train a linear SVM.

We will experiment with the last two methods later.

9.3 Second-order spatial features

The algorithm introduced in the previous section is a generic method for integrating the feature selection and feature extraction processes. In this section we provide examples of building 2^{nd} order features, given a pair of 1^{st} order features, (w_a, w_b) (Line 11 in Algorithm 1). In the Experiments section, we will explain how 3^{rd} order features can be built.

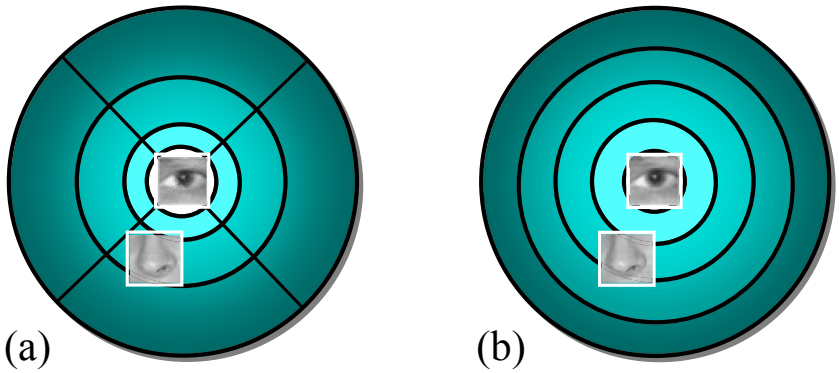


Figure 9.4: Examples of spatial histograms.

Different kinds of spatial histograms can be used for building 2^{nd} order features. In Fig. 9.4(a), we illustrate a spatial histogram with distance approximately in log scale, similar to the shape context histogram [123]. The log scale tolerates larger uncertainties of bin counts in longer ranges. The four directional bins are constructed to describe the semantics 'above', 'below', 'to the left', and 'to the right'. In Fig. 9.4(b), directions are ignored in order to describe how the co-occurrence of (w_a, w_b) varies in distance. In [114], squared regions are used to approximate the circular regions in Fig. 9.4(b) in order to take advantage of the integral histogram method [124]. Of course, squared regions and integral histogram can be used in our work as well.

The goal is to build a descriptor that describes how w_b is spatially distributed relative to w_a. Let us first suppose that there is only a single instance of w_a in an image, but multiple w_b's. Using this instance of w_a as a reference center of the spatial histogram, we count how many instances of w_b fall into each bin. The bin counts form the descriptor. Since there are usually multiple instances of w_a in an image, we build a spatial histogram for each instance of w_a, and then normalize over all spatial histograms; the normalization is done by summing the counts of corresponding bins, and

Figure 9.5: Second-order features. These are best viewed in color.

dividing the counts by the number of instances of w_a. This takes care of the case when multiple instances of an object appear in an image. The whole process is summarized in Algorithm 2.

The spatial histograms yield translation invariant descriptors, since the reference center is always in respect to the center word w_a, and describes the relative position of instances of w_b. The descriptors can also be (quasi-)scale invariant. This can be achieved by determining the normalized distance between instances of w_a and w_b, where the normalization is done by considering the geometric mean of the scale of the two patches. To make the descriptor in Fig. 9.4(a) rotation invariant, we can take into account the dominant orientation of a patch [125]. However, rotation invariance may diminish discriminative power and hurt performance [125] in object categorization.

In Fig. 9.5, red circles indicate words used as reference center. The red-green pairs correspond to a highly discriminative 2^{nd} order feature that has been selected in early rounds of boosting. The images are those that are incorrectly classified when only 1^{st} order features are used for training a classifier. We can see that 2^{nd} order features can detect meaningful patterns in these images. As a result, most of these images are correctly classified by a classifier using both 1^{st} and 2^{nd} order features.

9.4 Experiments

We use three datasets in the experiments: the PASCAL VOC2006 dataset [126], the Caltech-4 plus background dataset used in [127], and the MSRC-v2 15-class dataset used in [114]. We used the same training-testing experiment setups as in these respective references.

Algorithm 2: BuildSecondOrderFeatures

Goal: create feature descriptor given a word pair
Input: codeword pair (w_a, w_b)
Output: a vector of bin counts
Suppose there are N_a instances of w_a, and N_b instances of w_b in the image
Initialize N_a spatial histograms, using each instance of w_a as a reference center
for i=1,...,N_a **do**
    Count the number of instances of w_b falling in each bin
end
Sum up corresponding bins over the N_a spatial histograms
Divide bin counts by N_a

For each dataset we use different local feature descriptors to show the generality of our approach. For the PASCAL dataset, we adopt the popular choice of finding a set of salient image regions using the Harris-Laplace interest point detectors [126]. Another scheme is to abandon the use of interest point detectors [128] and sample image patches uniformly from the image. We adopt this approach for the Caltech-4 dataset. Each region or patch is then converted into a 128-D SIFT [129] descriptor. For the MSRC dataset, we follow the common approach [114] of computing dense filter-bank (3 Gaussians, 4 Laplacian of Gaussians, 4 first order derivatives of Gaussians) responses for each pixel.

The local feature descriptors are then collected from the training images and vector quantized using K-means clustering. The resulting cluster centers form the dictionary of codewords, $\{w_1, ..., w_J\}$. We use $J = 100$ for the MSRC dataset, and $J = 1000$ for the other two datasets; these are common choices for these datasets. Each local feature descriptor is then assigned to the closest codeword and forms a visual word.

For the MSRC dataset, we used the spatial histogram in Fig. 9.4(b), in order to facilitate comparison with the recent work of [114]. We followed the specs in [114] with 15 distance bins of equal spacing, the outermost bin with a radius of 80 pixels, and no scale normalization being performed. For the Caltech and PASCAL datasets, we used the spatial histogram in Fig. 9.4(a), where the scale is normalized according to the patch size or interest point size as explained earlier, and the outermost bin has a radius equal to 15 times the normalized patch size. The scale invariance

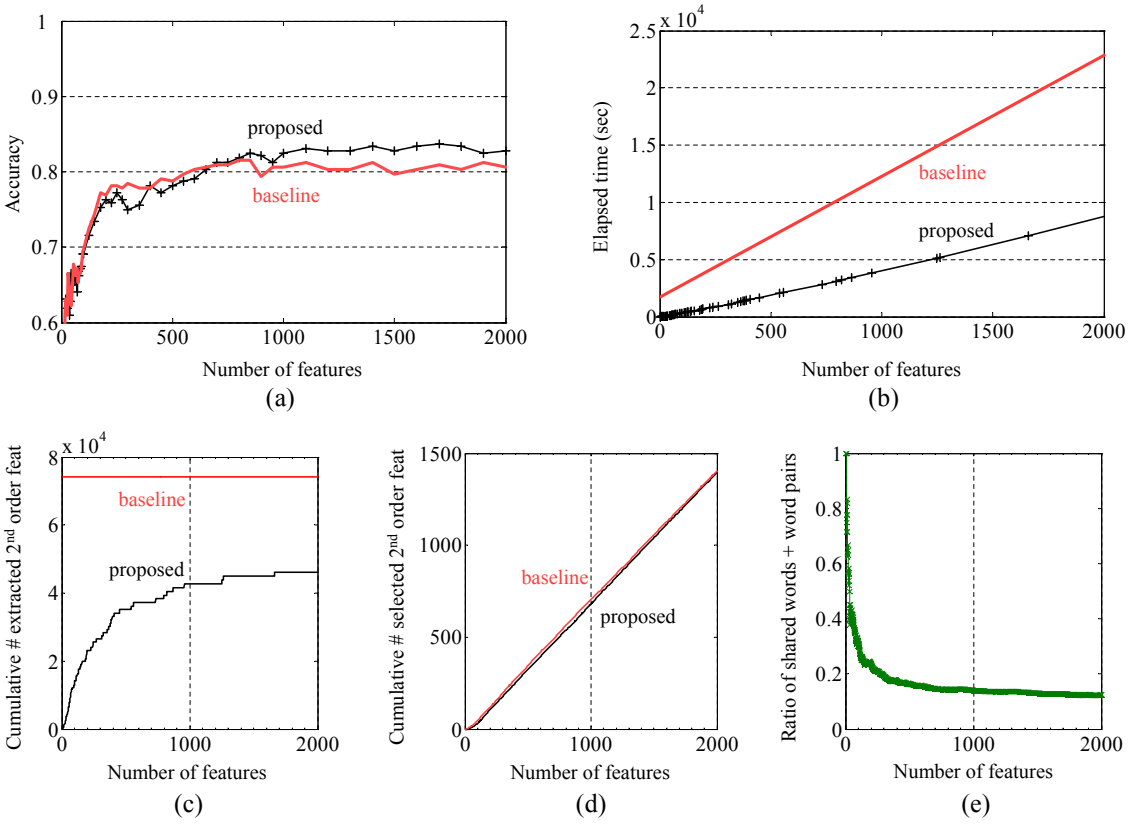


Figure 9.6: Integrated vs separated: After around 800 rounds of boosting, the proposed method outperforms baseline both in (a) testing accuracy and (b) required training time.

can be observed in Fig. 9.5 from the different distances between red-green word pairs.

9.4.1 Integrated vs Separated

Here we present the **main result**. In Fig. 9.6 we show the experiment on the 15-class MSRC dataset. We use a multiclass version of AdaBoost [120] for feature selection, and linear SVM for classification as explained in Sec. 9.2. In Fig. 9.6(a), we see that the accuracy settles down after about 800 rounds of boosting. Accuracy is calculated as the mean over the diagonal elements of the 15-class confusion matrix. In Fig. 9.6(b), we see the integrated feature selection and extraction scheme requires only about 33% of training time compared to the *canonical* approach where feature extraction and selection are two *separate* processes.

Surprisingly, we can see in Fig. 9.6(a) that, in addition to being more efficient, the proposed

scheme also achieves better accuracy in spite of its greedy nature. This can be explained by the fact that 2^{nd} order features are sparser than 1^{st} order features and hence statistically less reliable; the integrated scheme starts with the pool of first order features and gradually adds in 2^{nd} order features, hence it spends more quality time with more reliable 1^{st} order features.

In Fig. 9.6(c)-(e) we examine some temporal behaviors of the two methods. In Fig. 9.6(c), we show the cumulative number of 2^{nd} order features being extracted at each round of feature selection. While the canonical procedure extracts all features before selection starts, the proposed scheme aggressively extracts 2^{nd} order features in earlier rounds and then slows down. This logarithmic type of curve signifies the coupling between the feature extraction and the feature selection processes; if they weren't coupled, features would have been extracted at a constant (linear) speed instead of a logarithmic.

In Fig. 9.6(c), we also noticed that at 800 rounds of boosting, only about half of all possible 2^{nd} order features were extracted. This implies less computation in terms of feature extraction, as well as more efficient feature selection, as the feature pool is much smaller.

In Fig. 9.6(d), it appears that the canonical approach selects 2^{nd} order features at roughly the same pace as the integrated scheme, both selecting on average 0.7 second-order features per round of boosting. But in fact, as shown in Fig. 9.6(e), the overlap between the selected features of the two methods is small; at 800 rounds of boosting, the share ratio is only 0.14. The share ratio is the intersection of the shared visual words and visual word pairs of the two methods divided by the union. This means that the two methods have very different temporal behaviors.

9.4.2 Importance of feature selection

Here we compare with the recent work of [114], where feature selection is not performed, but first and second-order features are quantized separately into dictionaries of codewords. A histogram of these codewords is used as a feature vector. In Table 9.1, all three methods use the nearest neighbor classifier as in [114] for fair comparison [1]. We see that our method yields state-of-the-art performance, compared to the quantized (Method 2) and non-quantized (Method 1) versions. In addition, since the 2^{nd} order features need not be exhaustively computed and also no vector quantization on 2^{nd} order features is required, our method is also much faster than the method in

[1] We re-implemented the work of [114], because they used an untypical quantization scheme to generate 1^{st} order codewords, and results are not comparable; also, their spatial histogram is square-shaped.

[114].

	Proposed	Method 1	Method 2 [15]
Feature selection	√	×	×
Quantization	×	×	√
Accuracy	75.9%	71.3%	74.1%

Table 9.1: Importance of feature selection.

9.4.3 Linear SVM on weighted decision stumps

As explained in Sec. 9.2, we propose to concatenate the weighted output of all weak classifiers, $\{\alpha_1 y_1(\mathbf{x}), ..., \alpha_M y_M(\mathbf{x})\}$, from AdaBoost as a feature vector and then run a linear SVM. Results are shown in Table 9.2. The superior result over AdaBoost comes from a re-weighting of the terms $\{\alpha_1 y_1(\mathbf{x}), ..., \alpha_M y_M(\mathbf{x})\}$.

	PASCAL (EER)	MSRC (1-accuracy)
AdaBoost classifier (1st order feat)	13.4%	24.1%
AdaBoost classifier (1st & 2nd order)	12.1%	21.2%
Linear SVM on weighted decision stumps	10.9%	16.9%

Table 9.2: Performance on the PASCAL car-vs-rest and MSRC 15-class datasets.

The best results [126] reported on the PASCAL VOC2006 and VOC2007 datasets employ the Spatial Pyramid [130] technique on top of the bag of words representation. The Spatial Pyramid technique is orthogonal to the proposed method and combining them is expected to yield even better results.

9.4.4 Increasing the order

In Fig. 9.7, we experiment on the MSRC dataset and see that the classification accuracy obtained from using a feature pool of 1^{st} and 2^{nd} order features is higher than using 1^{st} order features alone. Including 3^{rd} order features does not improve accuracy. We generated 3^{rd} order features by

counting the number of times three codewords (w_a, w_b, w_c) fall within a radius of 30 pixels, i.e., the spatial histogram has only one bin. Third order features are generated every time a 1^{st} order feature is selected (which corresponds to w_a) and paired with each of the previously selected 2^{nd} order features (recall that a 2^{nd} order feature comes from a word pair, (w_b, w_c)), or vice versa. The reason for reducing the number of bins to one is to account for the data sparseness of higher-order features, which we will discuss later.

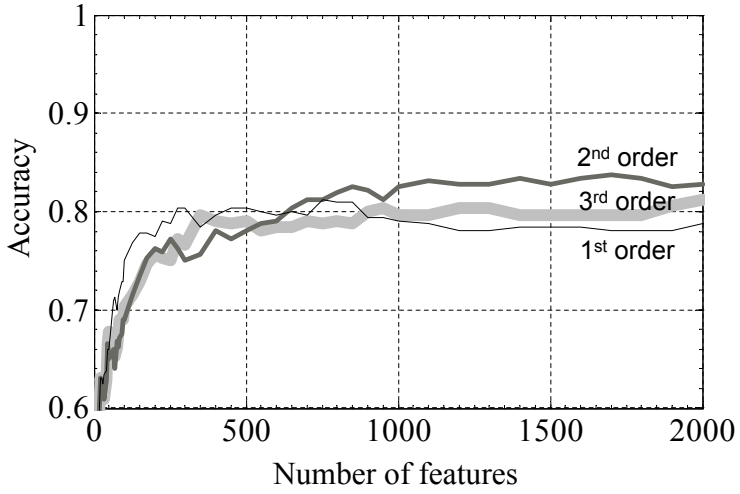
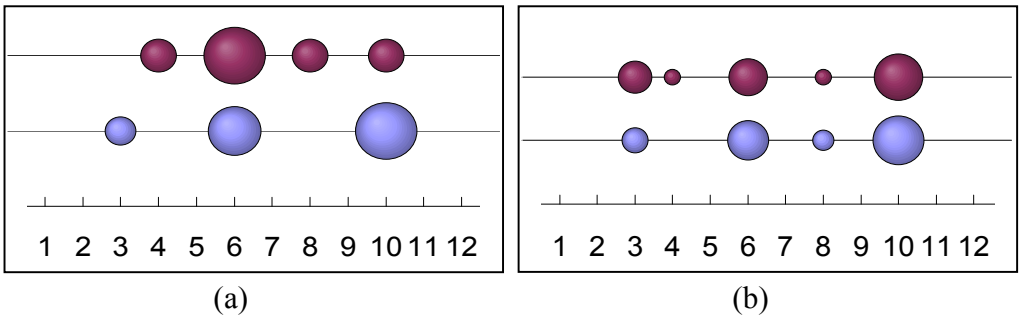


Figure 9.7: Accuracy and feature complexity.

9.4.5 Robustness of co-occurrence counts

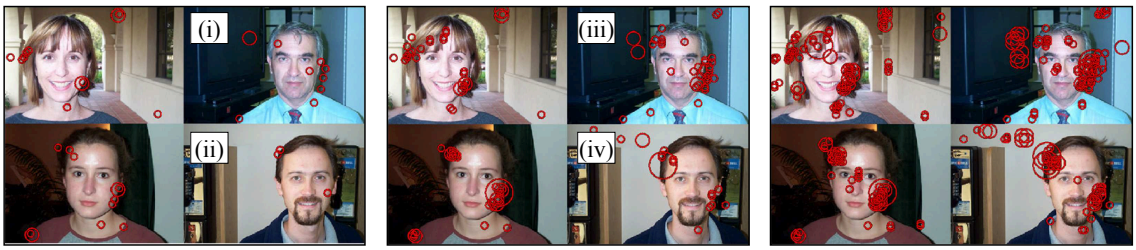
Instead of assigning a local feature descriptor to a single codeword, one can assign it to the top-N closest codewords. In Table 9.3, we vary the parameter c_1 from one to four and ten, which is the number of codewords each image patch is assigned to. In three out of four categories, the performance of the bag of words representation (using 1^{st} order features only) degrades as c_1 increases from one to four or ten, which manifests the popular practice of assigning a descriptor to a single codeword.

Yet, the top-N technique can help avoid the data-sparseness problem of 2^{nd} order features. We define the parameter c_2 as the number of visual words each image patch is assigned to when constructing 2^{nd} order features. Notice that c_1 and c_2 can have different values. In Fig. 9.8 we show the benefit of increasing c_2 from one to ten when constructing spatial features. In Fig. 9.8(a),



(a) (b)

Figure 9.8: Effect of parameter c_2 on the spatial histogram bin counts. (a) Using $c_2 = 1$. (b) Using $c_2 = 10$.



(a) Top 1 (b) Top 4 (c) Top 10

Figure 9.9: Effect of increasing the number of visual words a patch is assigned to.

two normalized spatial histograms with twelve spatial bins are collected from two different face images. The size of the bubbles indicates normalized bin counts. Recall that spatial histograms collect spatial co-occurrence of word pairs; in this case the specific word pair corresponds to a person's nose and eye from real data. Ideally the two histograms would be nearly identical, but image variations and clustering artifacts prevent it from being so. In Fig. 9.8(b), using the top-N technique, the two histograms become more similar to each other. The reason that 2^{nd} order features benefit more from this technique than 1^{st} order ones is due to the sparsity of co-occurrence of a word pair. The chance of *co-occurrence* between a pair of visual words within a specific spatial bin is at the order of approximately $1/(J^2 \times 12)$, where J is the size of the dictionary of codewords. Compared to the order of $1/J$ for the histogram of visual words, slight image variations and clustering artifacts can result in larger disturbances in the spatial feature bin counts than in the visual word bin counts. The top-N technique increases the bin counts (before normalization) and reduces the sensitivity to variations. In Fig. 9.9 we see the population of a particular codeword

getting denser as c_2 increases. In Fig. 9.9(i)(ii), this codeword rarely appears 'correctly' on the chin of the face. Increasing c_2 increases its occurrence on the chin, but also increases its occurrence at other locations, so increasing c_2 indefinitely would lead to performance degrading. Overall, this suggests that using a small value of c_1 but a moderate value of c_2 should give the best result. Indeed, using AdaBoost as classifier, we found that ($c_1 = 1, c_2 = 10$) gives state-of-the-art performance, as shown in Table 9.3.

Class \ (c_1,c_2)		(1,1)	(4,4)	(10,10)	(1,10)
Face	1st order feat	4.15	3.23	5.53	4.15
	1st and 2nd order feat	1.84	1.84	0.92	0.92
Motorbike	1st order feat	1.50	2.00	2.75	1.50
	1st and 2nd order feat	1.50	1.25	1.00	1.00
Airplane	1st order feat	2.75	4.00	4.00	2.75
	1st and 2nd order feat	2.25	2.50	2.00	1.75
Car	1st order feat	1.00	1.50	2.25	1.00
	1st and 2nd order feat	0.50	0.75	1.00	0.50

Table 9.3: Equal error rates (%) for the Caltech-4 dataset. By integrating feature selection and extraction, state-of-the-art results are obtained.

9.5 Conclusion

We have presented an approach for integrating the process of feature selection and feature extraction. The integrated approach is three times faster than the canonical procedures of feature selection followed by feature extraction. In addition, the integrated approach can achieve comparable or even better accuracy than the exhaustive approach, in spite of its greedy nature.

Our approach is generic and can be used with other feature selection methods. It can also be applied to all kinds of spatial histograms. In this work, we considered non-parametric histograms (with spatial bins), but parametric ones could be used as well, where the parameters (e.g., the mean

and covariance of point clouds) could be used as features.

Finally, we presented detailed experiments on three different object categorization datasets which have been widely studied. These datasets cover a wide range of variations on object category (20 in total), object scale (most noticeably in the PASCAL dataset) and pose. For each dataset, we used different state-of-the-art local feature descriptors. These experiments demonstrate that our approach applies to a wide range of conditions.

Part V

Conclusion

Chapter 10

Conclusions and Future Work

In this work, we introduced a probabilistic framework for object discovery in images and video. The framework presents the following features:

1. Fusing appearance, location, geometry, global and local motion
2. Handling multiple instances
3. Extensions to semi-supervised learning
4. Applications in retrieval, segmentation, categorization

In the future, there are several interesting directions to continue on. There are many other visual cues that could be integrated into our framework. One interesting direction is to utilize the occlusion boundary cues. The likelihood function in the particle filter could make use of the occlusion boundary cue by assigning a lower likelihood to ellipses which contain an occlusion boundary. This will effectively encourage the system to discover objects that obey the occlusion boundaries.

One motivation of doing unsupervised object discovery is because state-of-the-art object detectors are still very limited in the number of object categories they can handle. However, some of the object detectors are already running very reliably, such as face detectors. As more and more object detectors become reliable, we could use them to extract visual words that are informative and semantically meaningful. Our framework can then be used on a mixed representation of lower and higher level visual words.

Currently the model is based on visual words, which are cluster centers. However, clustering introduces quantization artifacts. This contributes to the semantic ambiguity and consequently

makes discovery more difficult, because the discovery is then responsible not only for discovery, but also for disambiguating the meaning of visual words. One solution to this problem is to combine the clustering and discovery into one optimization problem. This is doable, because k-means clustering is in fact a special case of mixture of Gaussians, which can be represented as a graphical model. If we combine the mixture of Gaussians graphical model into the current graphical model, then we can do joint optimization.

Bibliography

[1] *http://www.robots.ox.ac.uk/∼vgg/research/affine/*.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. J. Computer Vision*, vol. 60, pp. 91–110, 2004.

[3] R. Unnikrishnan and M. Hebert, "Extracting scale and illuminant invariant regions through color," in *British Machine Vision Conference*, September 2006.

[4] J. Sivic, B. Russell, A.A. Efros, A. Zisserman, and W.T. Freeman, "Discovering objects and their location in images," in *IEEE Intl. Conf. on Computer Vision*, 2005.

[5] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, T. Tuytelaars, and L. Van Gool, "Modeling scenes with local descriptors and latent aspects," in *IEEE Intl. Conf. on Computer Vision*, 2005.

[6] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, pp. 177–196, 2001.

[7] J. C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," in *British Machine Vision Conference*, 2006.

[8] M. Marszalek and C. Schmid, "Spatial weighting for bag-of-features," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[9] A. Agarwal and B. Triggs, "Hyperfeatures multilevel local coding for visual recognition," in *European Conf. on Computer Vision*, 2006.

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

[11] N. Loeff, H. Arora, A. Sorokin, and D. Forsyth, "Efficient unsupervised learning for localization and detection in object categories," in *Proc. NIPS*, 2005.

[12] S. Savarese, J. Winn, and A. Criminisi, "Discriminative object class models of appearance and shape by correlatons," *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[13] D. Liu and T. Chen, "Semantic-shift for unsupervised object detection," in *IEEE CVPR Workshop on Beyond Patches*, 2006.

[14] M. Weber, M. Welling, and P. Perona, "Unsupervised learning of models for recognition," in *Proc. European Conf. Computer Vision*, 2000.

[15] P. Felzenszwalb and D. Huttenlocher, "Pictorial structures for object recognition," *Intl. Journal of Computer Vision*, vol. 61(1), pp. 55–79, 2005.

[16] S. Ullman, E. Sali, and M. Vidal-Naquet, "A fragment-based approach to object representation and classification," in *4th Intl. Workshop on Visual Form*, 2001.

[17] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 509–522, 2002.

[18] A.C. Berg, T.L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[19] K. Grauman and T. Darrell, "Unsupervised learning of categories from sets of partially matching image features," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[20] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *IEEE Intl. Conf. on Computer Vision*, 2005.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[22] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790–799, 1995.

[23] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 603–619, 2002.

[24] *Adobe Photoshop 7, Adobe Systems Incorp. 2002*.

[25] E. Mortensen and W. Barrett, "Intelligent scissors for image composition," in *Proc. ACM Siggraph*, 1995.

[26] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut - interactive foreground extraction using iterated graph cuts," in *Proc. ACM Siggraph*, 2004.

[27] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images," in *IEEE Intl. Conf. Computer Vision*, 2001.

[28] J. Malik, S. Belongie, J. Shi, and T. Leung, "Textons, contours and regions: Cue integration in image segmentation," in *IEEE Intl Conf. Computer Vision*, 1999.

[29] M. Weber, *Caltech face dataset, http://www.vision.caltech.edu/archive.html*.

[30] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. PAMI*, vol. 13 (6), pp. 583–598, 1991.

[31] D. Liu and T. Chen, "A topic-motion model for unsupervised video object discovery," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[32] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," *Proc. European Conf. Computer Vision*, vol. 4, pp. 490–503, 2006.

[33] D. Chang and K. V. Nesbitt, "Identifying commonly-used gestalt principles as a design framework for multi-sensory displays," *IEEE Intl. Conf. Systems, Man, and Cybernetics*, 2006.

[34] A. Hoogs, R. Collins, R. Kaucic, and J. Mundy, "A common set of perceptual observables for grouping, figure-ground discrimination, and texture classification," *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 25 (4), pp. 458–474, 2003.

[35] H. Schneiderman and T. Kanade, "Object detection using the statistics of parts," *Intl. J. Computer Vision*, vol. 56, pp. 151–177, 2004.

[36] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *IEEE Intl. Conf. on Computer Vision*, 2005, pp. 1458–1465.

[37] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[38] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using non-parametric kernel density estimation for visual surveillance," in *Pro-*

ceedings of the IEEE, vol. 90, pp. 1151-1163, 2002.

[39] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2004.

[40] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[41] Z. Pan and C.-W. Ngo, "Moving-object detection, association, and selection in home videos," *IEEE Trans. Multimedia*, vol. Vol. 9, No. 2, pp. 268–279, 2007.

[42] L. Wixson, "Detecting salient motion by accumulating directionally-consistent flow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 774–780, 2000.

[43] M. Leordeanu and R. Collins, "Unsupervised learning of object features from video sequences," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[44] M. Irani and P. Anandan, "A unified approach to moving object detection in 2d and 3d scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. Vol. 20 No. 6, pp. 577–589, 1998.

[45] Q. Ke and T. Kanade, "Robust subspace clustering by combined use of knnd metric and svd algorithm," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2004.

[46] L. Xie and S.-F. Chang, "Pattern mining in visual concept streams," in *IEEE Intl. Conf. Multimedia and Expo*, 2006.

[47] S. Ebadollahi, L. Xie, S.-F. Chang, and J. R. Smith, "Visual event detection using multi-dimensional concept dynamics," in *IEEE Intl. Conf. Multimedia and Expo*, 2006.

[48] B. Gunsel, A. Ferman, and A. Tekalp, "Temporal video segmentation using unsupervised clustering and semantic object tracking," *Journal of Electronic Imaging*, vol. 7(3), pp. 592–604, 1998.

[49] A. Doulamis, K. Ntalianis, N. Doulamis, and S. Kollias, "An efficient fully-unsupervised video object segmentation scheme using an adaptive neural network classifier architecture," *IEEE Trans. on Neural Networks*, vol. 14 (3), pp. 616–630, 2003.

[50] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2000.

[51] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in

videos," in *IEEE Intl. Conf. Computer Vision*, 2003.

[52] J. Winn and N. Jojic, "Locus: Learning object classes with unsupervised segmentation," in *IEEE Intl. Conf. Computer Vision*, 2005.

[53] J. Sivic and A. Zisserman, "Video data mining using configurations of viewpoint invariant regions," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2004.

[54] D. Ramanan, D. A. Forsyth, and K. Barnard, "Building models of animals from video," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28(8), pp. 1319–1334, 2006.

[55] D.l Neill and A. Moore, "Detecting significant multidimensional spatial clusters," in *Advances in Neural Information Processing Systems*, 2005.

[56] J. Kubica, J. Masiero, A. Moore, R. Jedicke, and A. Connolly, "Variable kd-tree algorithms for spatial pattern search and discovery," in *Advances in Neural Information Processing Systems*, 2005.

[57] N. A. C. Cressie, *Statistics for Spatial Data*, Wiley, 1993.

[58] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from Google's image search," in *IEEE Intl. Conf. on Computer Vision*, 2005.

[59] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *ICCV workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[60] I. Laptev and T. Lindeberg, "Space-time interest points," in *IEEE Intl. Conf. on Computer Vision*, 2003.

[61] P. Tan, S. Lin, and L. Quan, "Resolution-enhanced photometric stereo," in *Proc. European Conf. Computer Vision*, 2006.

[62] T. Leung, "Texton correlation for recognition," in *Proceedings European Conference on Computer Vision*, 2004.

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

[64] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification, 2nd Ed.*, Wiley, 2000.

[65] Y. Bar-Shalom and T.E. Fortmann, *Tracking and Data Association*, Academic Press, 1988.

[66] Peter. J. Huber, *Robust Statistics*, Wiley, 1981.

[67] J. Wang, J. Han, and J. Pei, "Closet+: Searching for the best strategies for mining frequent closed itemsets," in *Int. Conf. Knowledge Discovery and Data Mining*, 2003.

[68] W. Zhao, J. Wang, D. Bhat, K. Sakiewicz, and N. Nandhakumar, "Improving color based video shot detection," in *IEEE Intl. Conf. Multimedia Computing and Systems*, 1999.

[69] Y.F. Ma and H.J. Zhang, "Video snapshot: A bird view of video sequence," in *International Multimedia Modelling Conference*, 2005.

[70] M. Isard and A. Blake, "CONDENSATION: Conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[71] B. Han, Y. Zhu, D. Comaniciu, and L.S. Davis, "Kernel-based bayesian filtering for object tracking," *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 227–234, 2005.

[72] C. Yang, R. Duraiswami, and Davis. L.S., "Fast multiple object tracking via a hierarchical particle filter," *IEEE International Conference on Computer Vision*, vol. 1, pp. 212–219, 2005.

[73] L. Yuan, A. Haizhou, T. Yamashita, L. Shihong, and M. Kawade, "Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 1–8, 2007.

[74] K. Okuma, A. Taleghani, N. Freitas, J.J. Little, and D.G. Lowe, "A boosted particle filter: Multitarget detection and tracking," *European Conference on Computer Vision*, pp. 28–39, 2004.

[75] M. Isard and J. MacCormick, "BraMBLe: A bayesian multiple-blob tracker," *IEEE International Conference on Computer Vision*, vol. 2, pp. 34–41, 2001.

[76] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussianbayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2002.

[77] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[78] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with

applications in pattern recognition," *IEEE Transactions in Information Theory*, vol. 21, pp. 32–40, 1975.

[79] H. Bohyung, D. Comaniciu, Y. Zhu, and L.S. Davis, "Sequential kernel density approximation and its application to real-time visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1186–1197, 2008.

[80] H. Bohyung, D. Comaniciu, Y. Zhu, and L.S. Davis, "Incremental density approximation and kernel-based bayesian filtering for object tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 638–644, 2004.

[81] M.A.F. Figueiredo and A.K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 381–396, 2002.

[82] J.J. Verbeek, N. Vlassis, and B. Kroese, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, pp. 469–485, 2003.

[83] H. Attias, "Inferring parameters and structure of latent variable models by variational bayes," *Proc. Conf. on Uncertainty in Articial Intelligence*, vol. 1, pp. 21–30, 1999.

[84] T. Zhao, R. Nevatia, and B. Wu, "Segmentation and tracking of multiple humans in crowded environments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1198–1211, 2008.

[85] J. Kotecha and P. Djuric, "Gaussian sum particle filtering," *IEEE Transactions on Signal Process*, vol. 51, pp. 2602–2612, 2003.

[86] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," *Proceedings European Conf. Computer Vision*, vol. 1, pp. 661–675, 2002.

[87] D. Liu and T. Chen, "DISCOV: A framework for discovering objects in video," *IEEE Transactions on Multimedia*, vol. 10, no. 2, pp. 200–208, 2008.

[88] M. Bertini, A. Del Bimbo, and W. Nunziati, "Video clip matching using MPEG-7 descriptors and edit distance," in *Intl. Conf. on Image and Video Retrieval*, 2006.

[89] *http://www-nlpir.nist.gov/projects/trecvid/*.

[90] S.-Y. Neo, J. Zhao, M.-Y. Kan, and T.-S. Chua, "Video retrieval using high level features: Exploiting query matching and confidence-based weighting," in *Intl. Conf. on Image and Video Retrieval*, 2006.

[91] J. Luo, A. Savakis, S. Etz, and A. Singhal, "On the application of Bayes networks to semantic understanding of consumer photographs," in *Intl. Conf. on Image Processing*, 2000.

[92] H. Zhang, A. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[93] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1254–1259, 1998.

[94] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Object level grouping for video shots," *Intl. Journal of Computer Vision*, vol. 67, pp. 189–210, 2006.

[95] R. Kondor and T. Jebara, "A kernel between sets of vectors," in *Intl. Conf. Machine Learning*, 2003.

[96] L. Wolf and A. Shashua, "Learning over sets using kernel principal angles," *Journal of Machine Learning Research*, pp. 913–931, 2003.

[97] S.K. Zhou and R. Chellappa, "From sample similarity to ensemble similarity: probabilistic distance measures in reproducing kernel hilbert space," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, pp. 917–929, 2006.

[98] K.A. Heller and Z. Ghahramani, "A simple Bayesian framework for content-based image retrieval," in *IEEE Conf. Computer vision and pattern recognition*, 2006.

[99] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *IEEE Intl. Conf. Computer Vision*, 2005.

[100] P. Viola, J.C. Platt, and C. Zhang, "Multiple instance boosting for object detection," *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2005.

[101] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," *IEEE Workshop on Applications of Computer Vision*, 2005.

[102] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," *Intl. Conf. Information and Knowledge Management*, 2000.

[103] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg, "Webcam synopsis: Peeking around the world," *IEEE Intl. Conf. Computer Vision*, 2007.

[104] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. PAMI*, vol. 28 (4), pp. 594–611, 2006.

[105] B. Julesz, "Textons, the elements of texture perception and their interactions," *Nature*, vol. 290, pp. 91–97, 1981.

[106] J. van de Weijer and C. Schmid, "Coloring local feature extraction," *Proc. European Conf. Computer Vision*, 2006.

[107] K.P. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," *Intl. Conf. Knowledge Discovery and Data Mining*, 2002.

[108] Y. Li, H. Li, C. Guan, and Z. Chin, "A self-training semi-supervised support vector machine algorithm and its applications in brain computer interface," *IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 2007.

[109] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Inc, 1988.

[110] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," *Proc. Advances in Neural Information Processing Systems (NIPS)*, 1999.

[111] B. Cestnik, "Estimating probabilities: A crucial task in machine learning," *Proc. European Conf. Artificial Intelligence*, pp. 147–149, 1990.

[112] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *CVPR*, 2001.

[113] G. Csurka, C.R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *ECCV Workshop Statistical Learning*, 2004.

[114] S. Savarese, J. Winn, and A. Criminisi, "Discriminative object class models of appearance and shape by correlatons," *CVPR*, 2006.

[115] X. Lan, C. L. Zitnick, and R. Szeliski, "Local bi-gram model for object recognition," Tech. Rep., MSR-TR-2007-54, Microsoft Research, 2007.

[116] L. Yang, P. Meer, and D.J. Foran, "Multiple class segmentation using a unified framework over mean-shift patches," *CVPR*, 2007.

[117] P.F. Brown, V.J. Della Pietra, P.V. de Souza, J.C. Lai, and R.L. Mercer, "Class-based n-gram models of natural language," *Comp. Linguistics*, vol. 18(4), pp. 467–479, 1992.

[118] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry, Chap. 5*, Springer-Verlag, second edition, 2000.

[119] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *JMLR*, vol. 3, pp. 1157–1182, 2003.

[120] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Submitted*, 2005.

[121] F. Fleuret, "Fast binary feature selection with conditional mutual information," *JMLR*, vol. 5, pp. 1531–1555, 2004.

[122] M. Vidal-Naquet and S. Ullman, "Object recognition with informative features and linear classification," *ICCV*, 2003.

[123] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *PAMI*, vol. 24, pp. 509–522, 2002.

[124] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," *CVPR*, 2005.

[125] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *IJCV*, vol. 73 (2), pp. 213–238, 2007.

[126] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool, "PASCAL VOC2006 Results," http://www.pascal-network.org/challenges/VOC/voc2006.

[127] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," *CVPR*, 2003.

[128] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proc. European Conf. Computer Vision*, 2006.

[129] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.

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