ABSTRACT

Title of dissertation: RECOGNIZING OBJECTS AND
REASONING ABOUT THEIR INTERACTIONS

Aniruddha Kembhavi
Doctor of Philosophy, 2010

Dissertation directed by: Professor Larry S. Davis
Department of Electrical and Computer Engineering

The task of scene understanding involves recognizing the different objects present in the scene, segmenting the scene into meaningful regions, as well as obtaining a holistic understanding of the activities taking place in the scene. Each of these problems has received considerable interest within the computer vision community. We present contributions to two aspects of visual scene understanding.

First we explore multiple methods of feature selection for the problem of object detection. We demonstrate the use of Principal Component Analysis to detect avifauna in field observation videos. We improve on existing approaches by making robust decisions based on regional features and by a feature selection strategy that chooses different features in different parts of the image. We then demonstrate the use of Partial Least Squares to detect vehicles in aerial and satellite imagery. We propose two new feature sets; Color Probability Maps are used to capture the color statistics of vehicles and their surroundings, and Pairs of Pixels are used to capture captures the structural characteristics of objects. A powerful feature selection analysis based on Partial Least Squares is employed to deal

with the resulting high dimensional feature space (almost 70,000 dimensions). We also propose an Incremental Multiple Kernel Learning (IMKL) scheme to detect vehicles in a traffic surveillance scenario. Obtaining task and scene specific datasets of visual categories is far more tedious than obtaining a generic dataset of the same classes. Our IMKL approach initializes on a generic training database and then tunes itself to the classification task at hand.

Second, we develop a video understanding system for scene elements, such as bus stops, crosswalks, and intersections, that are characterized more by qualitative activities and geometry than by intrinsic appearance. The domain models for scene elements are not learned from a corpus of video, but instead, naturally elicited by humans, and represented as probabilistic logic rules within a Markov Logic Network framework. Human elicited models, however, represent object interactions as they occur in the 3D world rather than describing their appearance projection in some specific 2D image plane. We bridge this gap by recovering qualitative scene geometry to analyze object interactions in the 3D world and then reasoning about scene geometry, occlusions and common sense domain knowledge using a set of meta-rules.

RECOGNIZING OBJECTS AND REASONING ABOUT THEIR INTERACTIONS

by

Aniruddha Kembhavi

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Professor Min Wu
Professor David Jacobs
Professor John J. Benedetto

Acknowledgments

Most Ph.D. students would agree that graduate life is full of drastic ups and downs. While my experiences were not quite as oscillating as the ones shown in Figure 1, I did have my share of highs and lows during my years at the University of Maryland (surprisingly, these were strongly correlated with paper acceptances and rejections). It was the support and encouragement of a large number of people that motivated me through the tougher periods and made my graduate experience a memorable one.

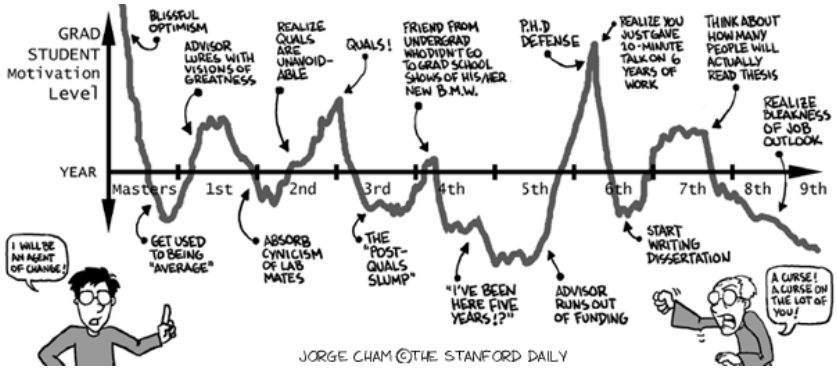


Figure 1: '**P**iled **H**igher and **D**eeper' by Jorge Cham. *www.phdcomics.com*

First and foremost, I thank my advisor Prof. Larry Davis, who was a constant source of strength and support. While his guidance helped define my research directions, he also provided me with a lot of freedom to develop my own ideas. I will forever cherish the countless discussions with him over the past few years. I also thank David Harwood with whom I have worked very closely, ever since the beginning of my graduate studies. Interacting very closely with David has taught me the value of taking a pragmatic approach to solving problems. His strong emphasis on data visualization and his attention to detail have rubbed off on me, and I hope will hold me in good stead throughout my career.

Prof. Rama Chellappa served as my advisor during the early stages of my graduate

studies. I thank him for his advice and encouragement throughout my graduate program. I also thank Prof. Ramani Duraiswami and Prof. David Jacobs for their guidance and interesting discussions regarding my research. They were both very accessible to me, and I often stopped by their offices to gain a different perspective about my research. I also thank Prof. Min Wu and Prof. John Benedetto for generously sparing their time and agreeing to serve on my dissertation committee.

Most importantly, I owe my deepest gratitude to my parents and my wife. They have served as my inspiration throughout this journey. Their love and continued belief in me has been incredible. My wife Pratibha has been the person I have turned to the most. Whenever I would make any progress in my research, I could not wait to tell her about it (and probably bored her with endless technical details). At the same time she was always there to encourage me when things got tough.

My dear friends and colleagues: Vlad, Ryan, Behjat, Abhinav and William(s) deserve a special mention. They provided me with the utmost support throughout these years, especially during the **H**ard **T**imes. In addition, many of the ideas in this thesis resulted out of the long and interesting discussions I had with them, often into the wee hours of the morning. I also thank my colleagues Shiv, Vinay, James, Antonio, Brandyn, Tom, Afshin, Arpit, Zhe, Zhuolin, Son, Sernam, Mohamed, Radu and Balaji for making this a fun as well as creative learning environment. My friends outside of school, Jayant, Rahul and Johanna made these last few years a memorable and enjoyable experience. Thank you so much for showing me that there existed a life outside graduate school.

There are of course many other people that I have not named, who have each been part of this journey. I thank all of you.

Table of Contents

List of Figures

Chapter 1

Introduction

Scene understanding is one of the fundamental objectives of computer vision. This task involves recognizing the different objects present in the scene, segmenting the scene into meaningful regions, as well as obtaining a holistic understanding of the activities taking place in the scene. Each of these problems has received considerable interest within the computer vision community.

We present contributions to two aspects of visual scene understanding. First we explore multiple methods of feature selection for the problem of object detection, which help to improve the efficiency and performance of detectors for single images as well as video. Second, we develop a video understanding system for scene elements, such as bus stops, crosswalks, and intersections, that are characterized more by qualitative geometry and the activities of overlying objects than by intrinsic appearance.

1.1 Feature Selection for Object Detection

With the ability of modern computers to store and process datasets with hundreds of thousands of variables, a fair amount of research over the last few years has focused on the problem of feature selection. Feature/variable selection is a very relevant problem in the computer vision community. Standard video streams often have an image resolution of 640×480 pixels. Given a standard frame-rate of 30 frames/sec, this requires the need

to process over 9.2 million pixels per second. With the ever increasing resolution of cameras, the amount of data that needs to be processed is tremendous. Feature selection can help deal with such large datasets.

There are many advantages of feature selection [34]. Reducing the number of features reduces the effect of the *curse of dimensionality*. It also leads to a speed-up of commonly used classifiers with regards to the training as well as the testing times. Feature selection often reduces the number of noisy variables, leading to an improvement of the generalization capability of the system, which in turn, leads to an improved performance. Furthermore, reducing a large dataset to one with just a few variables, helps tremendously with data visualization and understanding.

Feature selection methods can be broadly classified into two categories: *Feature Ranking* and *Feature Subset Selection*. *Feature Ranking* involves ranking features based on a metric (which often depends on the given application), and then discarding the features that are assigned a low score. This, however, is a suboptimal approach and can lead to the selection of redundant variables. *Feature Subset Selection* attempts to select a subset of variables that have good prediction capabilities as opposed to ranking variables by their individual predictive powers. This can lead to a more optimal selection of variables. However, in the process of removing redundant variables, *Feature Subset Selection* may remove several relevant variables. For a more extensive review of *Feature Ranking* and *Feature Subset Selection* methods, we refer the reader to [34] and [6].

We present a few feature selection strategies for the problem of object detection in several computer vision applications. First, we demonstrate the use of Principal Component Analysis to detect avifauna in field observation videos. Second, we use a Partial

Least Squares selection strategy to detect vehicles and humans in single images. Third, we use an Incremental Multiple Kernel Learning (IMKL) scheme to detect vehicles in a traffic surveillance scenario.

1.1.1 Detecting avifauna:

Feature selection using principal component analysis

Sociobiology seeks to understand the social behaviors of a given species by considering the evolutionary advantages these behaviors may have. To observe these social behaviors in their natural setting, biologists conduct a substantial portion of their research in the field, recording observations on videotapes. While fieldwork is very demanding, videotape analysis is truly exhausting. The corpus of video footage must be viewed in its entirety, during which time copious notes and qualitative observations are taken. Our collaborators add more than 2000 hours of video annually to a growing total of more than 30,000 hours. They desperately need computational video analysis tools.

The approach we have developed addresses the challenges inherent in detecting and tracking animals in their native outdoor habitats. Characteristics of these field observation videos include: poor image quality; drastic illumination changes, some rapid due to varying cloud-cover overhead, others slow and spatial due to shadows cast by the rising sun; targets that are motionless for long stretches of time; and non-stationary background, such as vegetation swaying in the wind and also ground clutter kicked or shifted around by the animals being observed. Conventional computer vision techniques are not yet able to handle all of these challenges simultaneously.

Figure 1.1: Satin Bowerbird (*Ptilonorhynchus violaceus*). A perched male Satin Bowerbird (above right) and two frames taken from overhead courtship videos.

Since our goal is to make the Biologist's video analysis much easier, there are several advantages in our favor. First, the video analysis will take place offline. This enables us to utilize all the information in the video's entire space-time volume. Second, we know a priori how many target objects need to be tracked. Third, domain-specific information about the target's appearance is available in the form of a coarse target model.

Our framework leverages these advantages and overcomes many of these problems. Our main contribution is a staged approach for target detection. We first use spatio-temporal volumes to isolate potential target regions. Our algorithm then combines target-specific information with local scene features to tailor individual models for different parts of the scene. Emphasis is thus given to those features which locally distinguish the target of interest.

We demonstrate our framework on an extensive data set of 24 videos comprising a total of more than 200,000 frames where we achieve 82.89% tracking accuracy. These

videos contain courtships of the Satin Bowerbird (*Ptilonorhynchus violaceus*) and were collected by our collaborators, Jean-François Savard and Gerald Borgia.

Researchers in Prof. Borgia's lab study sexual selection (how various traits and behaviors influence mating success) in various species of the Bowerbird family [19, 68] , generally found in Australia, New Zealand and Southeast Asia. Male Bowerbirds attract mates by constructing a bower, a structure built from sticks and twigs, and decorating the surrounding area. Females visit several bowers before choosing a mating partner and returning to his bower. In part, because both courtship and mating occurs at this known location, Bowerbirds are a particularly good bird in which to study sexual selection. Of particular interest are the adjustments made by the male during courtship in response to the female. Their most recent study [76] evaluates how the male modulates his display (measured as distance from the female) based on the response cues given by a robotic female. An early prototype of our system was very valuable in facilitating the spatial tracking of the courting male, greatly reducing the days of work that would be required for manually tracking so many frames.

1.1.2 Detecting vehicles:

Feature selection using partial least squares

Several commercial earth observation satellites, such as IKONOS, GeoEye and QuickBird, provide publicly available imagery at a ground sampling distance (GSD) of 1 meter. High resolution images of a small number of locations are publicly available via Google Earth at an astonishing GSD of 0.15 meter. We consider the problem of detect-

ing vehicles from such high resolution aerial and satellite imagery; this problem has a number of applications. Images of road networks, along with the distribution of vehicles in different regions, can provide information for urban planning and traffic monitoring. Detecting and tracking vehicles in aerial videos is also an important component in visual surveillance systems. In spite of the increasing availability of high resolution aerial and satellite images, vehicle detection still remains a challenging problem. In urban settings especially, the presence of a large number of rectilinear structures, such as trash bins, electrical units and air conditioning units on the tops of buildings can cause many false alarms. Figure 1.2(a) shows an example of an image patch for which previously published vehicle detectors produce a large number of false alarms.

Our proposed vehicle detector improves upon previous systems [61, 18, 22, 104, 33, 38, 77] by incorporating a much larger and richer feature set than previous approaches. First, a novel set of image descriptors are proposed that capture the color properties of an object and its surrounding, called *Color Probability Maps (CPM)*. Then, the commonly used Histograms of Oriented Gradients (HOG) feature is incorporated to capture the spatial distribution of edge orientations. Finally a very simple yet powerful image descriptor, named *Pairs Of Pixels (PoP)*, is proposed to capture the structural properties of objects. The concatenation of these three classes of features leads to a very high dimensional feature vector (approximately 70,000 elements). In contrast, the number of samples that we have to train our vehicle detector is much smaller, rendering many popular machine learning techniques unusable. Furthermore, our features are extracted from neighboring pixels within a detection window, which greatly increases their multi-collinearity. We take advantage of the nature of our problem by employing a classical statistical regression

(a) (b)

Figure 1.2: (a) Two image patches showing the performance of our system. Both image patches were extracted from a much larger image (5007×7776 pixels) which is displayed in its entirety in (b) and at a higher resolution in Figure 3.22. The top figure (region marked in green) shows the high detection accuracy of our system in the presence of a large number of vehicles. The bottom image (region marked in red) shows the low false alarm rate of our system in a region that has many rectilinear structures. Typical vehicle detection systems often produce false alarms in the presence of such structures. © 2009 Google

analysis technique called Partial Least Squares (PLS). Our PLS analysis extracts a low dimensional subspace within which we can use a simple quadratic discriminant classifier to classify image patches into vehicles and background.

Using such a large number of features greatly increases the computational cost of the vehicle detector. A common approach to speed up detectors using a large number of features is to use a boosting algorithm along with a rejection cascade as in [90]. We reduce the computational cost of our system using a dual feature selection approach. First, a recently proposed feature selection method called Ordered Predictors Selection, which combines a number of informative vectors that rank features based on their predictive

performance, is used. This is coupled with multi-stage, multi-resolution image analysis, where a large number of image windows are discarded at an early stage of processing (processing at lower resolutions) and only a small fraction of image patches are analyzed at the highest image resolution (second stage). Our feature selection approach not only increases the speed of our system but also its performance.

We demonstrate our proposed vehicle detector on two datasets. The first consists of color images collected from a satellite and obtained via Google Earth. This is a set of 40 high resolution images over the city of San Francisco. The second dataset is the publicly available Overhead Imagery Research Data Set which consists of a large number of aerial images, annotated with vehicles [85]. We compare our approach to several previously proposed object detection approaches including the Histograms of Oriented Gradients approach of Dalal et al. [20], the Spatial Pyramidal Matching algorithm [52] incorporating SIFT features, the recently proposed Intersection Kernel Support Vector Machines using HOG features [58] and a vehicle detector proposed in [61]. Our solution obtains favorable results as compared to previous approaches.

1.1.3 Detecting vehicles:

Feature selection using multiple kernel learning

The problem of visual category recognition has received considerable interest over the past few years. The most common approach consists of three major components: interest point detection, interest region description and classification. A recent focus has been on improving region descriptors. This has led to a number of powerful descriptors

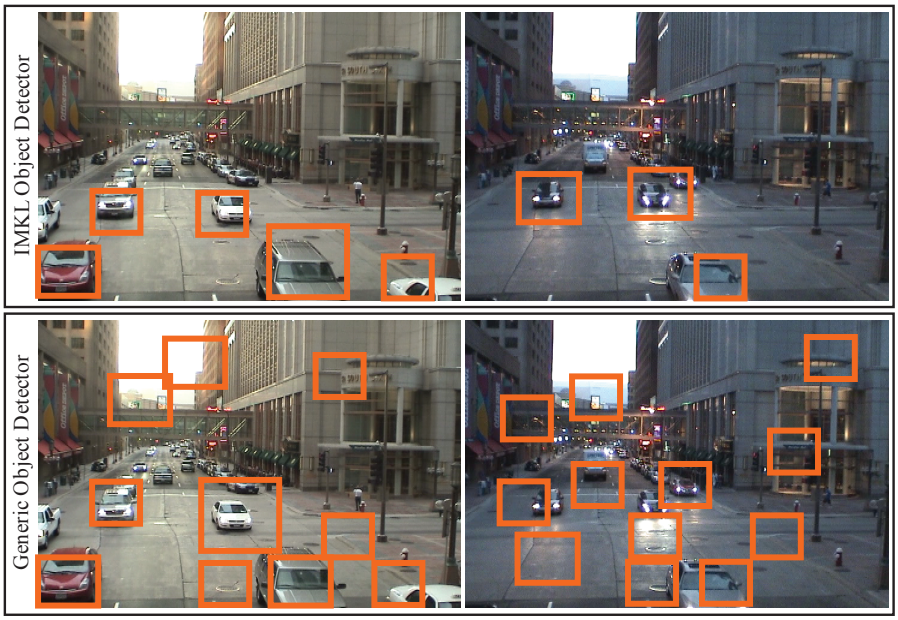


Figure 1.3: Sample result frames showing varying illumination conditions. Our incremental framework (IMKL) tunes itself to the scene by updating itself with images of objects in commonly observed poses and images of the varying background. Thus, it outperforms a static detector built on a generic training set.

being proposed such as Histograms of Oriented Gradients [20], Geometric Blur [5] and Pyramidal Histogram of Visual Words [9]. While each of these descriptors provides good classification accuracies for different object classification tasks, combining information from such multiple sources has been shown to be more reliable [8, 102, 89]. Varma et al. [89] proposed combining multiple descriptors using Multiple Kernel Learning (MKL) and showed impressive results on varied object classification tasks.

Using such a set of powerful descriptors, along with a nonlinear classifier such as

a Support Vector Machine (SVM), can lead to a boost in classification performance. But it is equally important to have a good set of training images, representative of the test images that are expected in the given application. Collecting large number of images and forming a generic training dataset for commonly seen objects is relatively easy using an internet search engine such as Google. Furthermore for many standard objects such as cars, training datasets are already available, such as the UIUC Car Database [1]. However, obtaining a representative training database for a given application is not as straightforward, as it requires a fair amount of manual labor.

Consider a camera at a traffic intersection detecting and classifying vehicles such as shown in Figure 1.3. First, the location of the camera in this scene and typical paths traversed by the vehicles, restricts the observed poses. Second, the camera position restricts the images representing the negative class (in our case, the background images) for this classification task. Third, images corresponding to vehicles as well as background also change over time, due to factors such as illumination changes and shadows cast by the nearby buildings. Obtaining such scene specific examples of the object classes and the background class would clearly benefit the visual classifier, but would require a tedious manual annotation procedure.

Our Incremental Multiple Kernel Learning (IMKL) approach uses an easily obtained generic training database as input, and then tunes itself to the classification task at hand. It simultaneously updates the training examples to tailor them towards the objects in the scene. It also updates the weights that determine the optimal combination of different information sources, while allowing different feature combinations to be chosen for different object classes. Finally, it tunes the classifier to the updated training dataset. As

the scene changes over time, a feedback loop updates our training dataset with detections from all object classes. The incremental procedure is then invoked to update the kernel combination weights as well as the classifier. Our final system is obtained by combining the outputs of this online classifier with the high probability outputs of the original offline classifier trained on the generic training database. This enables us to tune the classifier to the given scene, while reducing the number of misclassifications on rarely seen objects. We can also remove images from our training database over time. This is useful when dealing with gradual illumination changes, for example.

We first describe the MKL formulation of Bach et al. [71], known as SimpleMKL, which we use to obtain a classifier for the initial training database. SimpleMKL carries out this optimization in an SVM framework to simultaneously learn the SVM model parameters as well as kernel combination weights. Our incremental procedure for MKL is an exact online solution that allows us to update the Lagrangian multipliers of the training points, as well as the kernel combination weights, one new point at a time. The central idea is to add a new data point to the solution and update its Lagrangian multiplier while maintaining the Karush-Kuhn-Tucker conditions on all the current data points in the solution. We derive our IMKL procedure in Section 4.2.

We demonstrate our visual categorization framework on the task of vehicle detection and classification. The dataset we use consists of video sequences collected from a camera overlooking a traffic intersection. We initialize our training database with a set of images collected from Google and update it incrementally to improve the classification performance over time. The dataset also shows a significant change in illumination conditions in the scene as day transitions into night. Our system is able to update itself

over time to handle this transition. We compare our algorithm with OPTIMOL [55], an incremental model learning approach, recently proposed for the task of automatic object dataset collection.

1.2 Scene understanding using probabilistic logic models

We build on recent research in appearance-based object recognition and tracking [20, 78, 49, 43], recovery of qualitative scene geometry from images and video [40, 39, 32], and probabilistic relational models for integrating common sense domain models with uncertain image analysis [88], to develop a video understanding system that can identify scene elements (cross walks, bus stops, traffic intersections), characterized more by qualitative geometry and activity than by intrinsic appearance. The domain models we use are naturally specified by humans, and characterize scene elements in terms of geometric relationships (sidewalks are found along roads and are parallel to roads) and activity relationships (people walk on sidewalks, wait and possibly queue for a bus).

These domain models are related to image analysis (appearance, tracking, motion) by representing them as probabilistic logical models (Markov Logic Networks). These logical models describe *what typically happens* in the scene and not *what is visible* in some video of that scene. We bridge this gap using two methods. First, we recover qualitative scene geometry to analyze object interactions in the 3D world rather than the 2D image plane. Second, we utilize a set of meta-rules that capture general rules about scene geometry and occlusion reasoning and fuse them with common sense domain knowledge to detect these scene elements in videos taken from arbitrary viewpoints. This involves

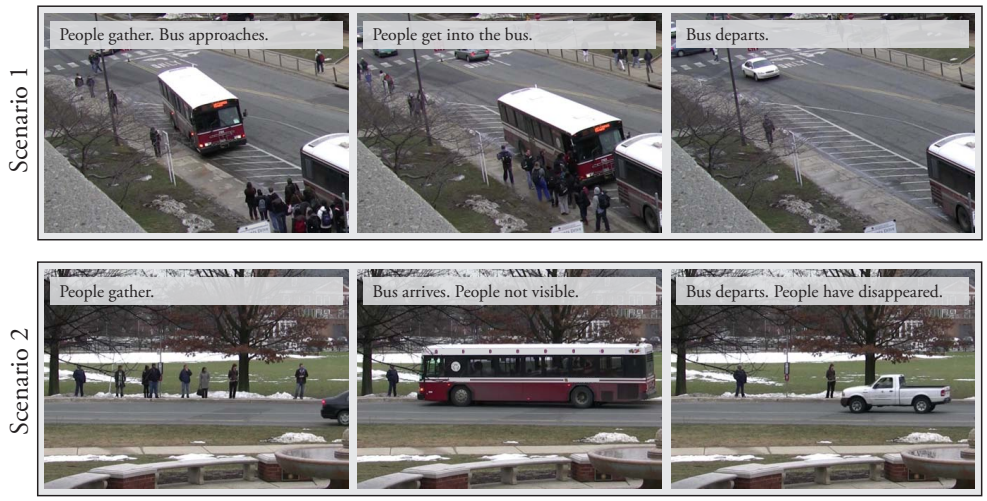


Figure 1.4: Two bus stops observed from different viewpoints. In Scenario 1, all activities associated with a typical bus stop model are observable. In Scenario 2, the bus occludes people departing and entering the bus.

reasoning about unobserved events and inferring their occurrence based on other observations.

As an example, consider a model for a bus-stop. This model might indicate that people wait and queue at a bus stop, a bus stops at the bus stop, the doors to the bus open, people leave the bus through the doors, then the people waiting enter the bus through the doors, the doors close, and finally the bus leaves. From the viewpoint in Scenario 1 (refer to Figure 1.4), all of the activities associated with this bus stop model are observable. Scenario 2 shows a bus stop seen from another viewpoint, in which the bus occludes the people waiting to board, and the bus doors are not visible. In this case, our system reasons about this occlusion, and determines that what we expect to observe are that the people waiting for the bus will be gone when the bus leaves, and that new people will be seen after the bus leaves.

We demonstrate our video understanding framework on a dataset of videos of public

spaces. These video sequences were collected using cameras overlooking scenes from varying viewpoints. Each contains multiple scene elements of interest, such as bus stops, traffic intersections, stop signs, crosswalks, garage entrances, etc. Our system is able to correctly identify a large number of these scene elements described by the human elicited domain models.

Chapter 2

Detecting Avifauna using Principal Components Analysis

2.1 Introduction

Sociobiology seeks to understand the social behaviors of a given species by considering the evolutionary advantages these behaviors may have. Biologists conduct a substantial portion of their research in the field, recording observations on videotapes. While fieldwork is very demanding, videotape analysis is truly exhausting. Our collaborators, Jean-François Savard and Gerald Borgia at the Biology department at the University of Maryland, add more than 2000 hours of video annually to a growing total of more than 30,000 hours. They desperately need computational video analysis tools.

The first step towards achieving the biologist's objectives is to accurately track the animal or animals they are observing. While traditionally done by hand, our goal is to automate the tracking process. A typical method used in computer vision to find and track subjects moving within a scene is background subtraction. A sample of representative work includes algorithms based on Gaussian mixture models (Stauffer and Grimson [81]), non-parametric models (Elgammal et al. [23]), and local binary patterns (Heikkilä and Pietikäinen [35]). Typically, background subtraction algorithms are designed for online and sometimes even real-time analysis. These constraints are unnecessary for our purposes, hence affording the flexibility to use all available temporal information in a video, not just information from the recent past.

Recent work by Parag et al. [67] takes a similar approach to background modeling, selecting distinctive features on a pixel-by-pixel basis. A crucial advantage of our technique, however, is that we not only pick features that are distinctive for a given location in the scene, we choose the features which most effectively differentiate the target object of interest from that part of the scene.

While many effective background subtraction approaches have been and continue to be proposed, to our knowledge, they all encounter difficulties in handling all of the issues of natural outdoor environments such as those in our dataset. The general approach to dealing with background changes such as varying global illumination is to allow the model to evolve, discounting evidence from the more distant past in favor of that just observed. The primary difficulty with this method stems from its inability to simultaneously handle foreground objects that become stationary for some period of time (eg. a sleeping person [87]), instead absorbing them into the background.

Efforts have been made to provide tools in support of field research. HCI researchers have recently built digital tools that allow biologists to integrate various observations and recordings while in the field [98]. In searching for the Ivory-billed Woodpecker, various teams have successfully employed semi-supervised sound analysis software to analyze the large volumes of recordings [29, 37] obtained in the field. However, there remains a need for automated tools capable of analyzing video recordings in natural outdoor environments.

We are aware of at least two projects that have previously focused on tracking animals. The Biotracking project at Georgia Tech's Borg Lab has conducted extensive research on multi-target tracking of ants [46] and bees [45, 65] and also tracking larger

animals such as rhesus monkey [47]. The SmartVivarium project at UCSD's Computer Vision Lab has investigated techniques for tracking and behavior analysis of rodents [13, 14]. Their research also includes closely related work on supervised learning of object boundaries [21]. However, in these experiments the animals were observed in captivity, generally under laboratory conditions. While [47] used Stauffer and Grimson's background modeling technique, we have found this method to work very poorly in the Bowerbird courtship videos.

2.2 A Three Stage Approach

Our approach has three major phases: initial pixel classification, pixelwise background model selection and evaluation/final classification. In the first phase, the biologist provides a coarse initial model of the target (a male Bowerbird in our case) that he/she wishes to track throughout the video. This model is used to segment each frame of the video, extracting possible target pixels (in reality some target, some background), ideally leaving behind a set of only background pixels [1]. Here, we use information from all previous and all future frames of the video to take decisions (as opposed to just a few frames from the past). This helps us overcome the problem of the Bowerbird often being stationary for hundreds, even thousands of frames at a time.

A key characteristic of unconstrained outdoor videos is the variation of the background scene, both from video to video as well as from one part of the image to another. Our second phase accounts for this. Here, we use the sets of background and target pixels

[1] We define background pixels to be all those pixels that are not part of the target indicated by the biologist.

and Principal Component Analysis (PCA) on a bag of features, to choose different features at different locations in the image, which can be used to build robust models. Our bag of features includes some that incorporate neighborhood information.

In the third phase, we use non-parametric Kernel Density Estimation (KDE) to build a background model for each individual image location (pixel). We then evaluate this pixel's value over all frames in the video, determining the probability in each frame that the pixel belongs to this model. We explain these three phases in greater detail in the following subsections.

2.2.1 Initial Pixel Classification

Many of the videos in our dataset are affected by drastic changes in global illumination. These are caused by varying levels of cloud cover and by sunlight filtering through the canopy and foliage above. The automatic gain control setting on the camera also produces sudden global changes in the color and brightness of the video. To deal with such global illumination changes, we transform every image from RGB color space into a one dimensional rank-ordered space, equivalent to performing histogram equalization on the grayscale image. The rank feature space assumes that the feature distribution changes very little, instead just shifting due to a change in the overall illumination. It disregards the absolute brightness of a pixel in the scene, rather considering only its value relative to all the pixels in the image. It is invariant to multiplicative and additive global changes and thus is largely unaffected by these effects we have observed.

In order to tune our system to track the target, we require an initialization by the

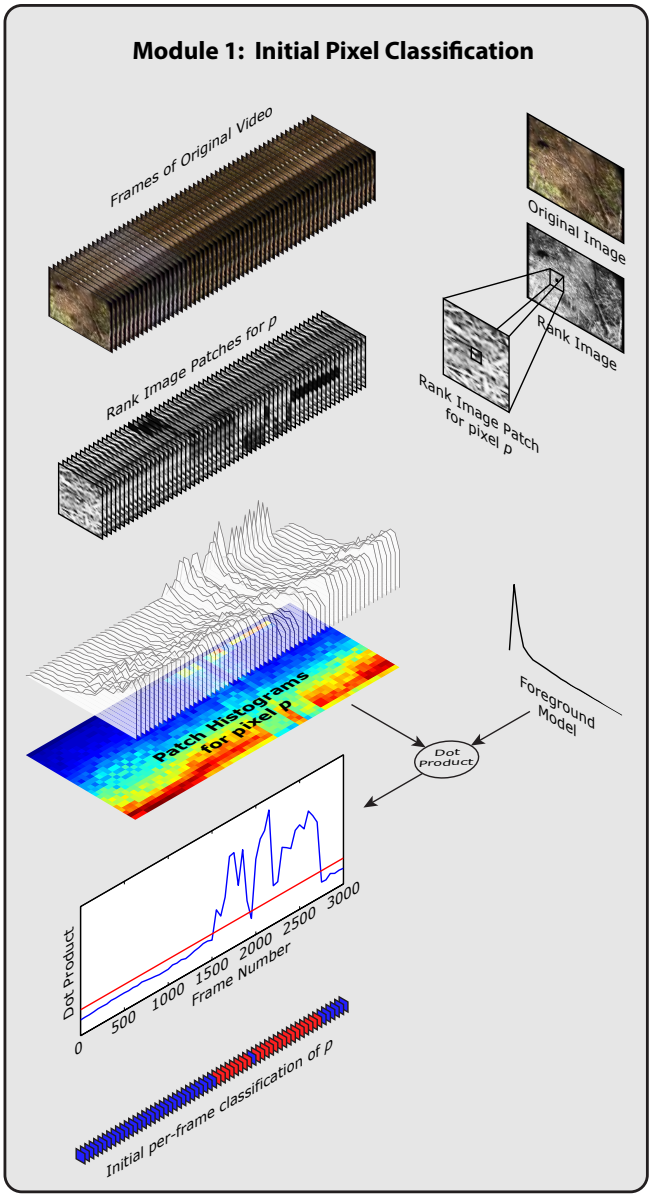


Figure 2.1: Module 1: Initial Pixel Classification.

biologist. Before a video is processed, the biologist analyzes a small number of frames chosen randomly, and for each frame marks out the region enclosing the Bowerbird if it is present in that frame. These pixels are used to build a smoothed histogram which serves as a coarse initial model of the target. This model is used to classify every pixel in the video into one of two sets - “potential” target pixels and “high confidence” background

pixels.

At each image location, the feature that is used for this initial pixel classification is a neighborhood histogram of rank intensity. While most traditional background subtraction approaches have relied on the information contained at a single pixel to build background and target models, we rely more on neighborhood information for the following reasons. First, it reduces the chance of noisy pixels being classified as target pixels. Second, while some background pixels might closely fit the target model, neighboring pixels around it are less likely to simultaneously fit the model as well. Third, our use of regional information allows us to "see through" occluding surfaces such as branches and foliage when the target is passing beneath them.

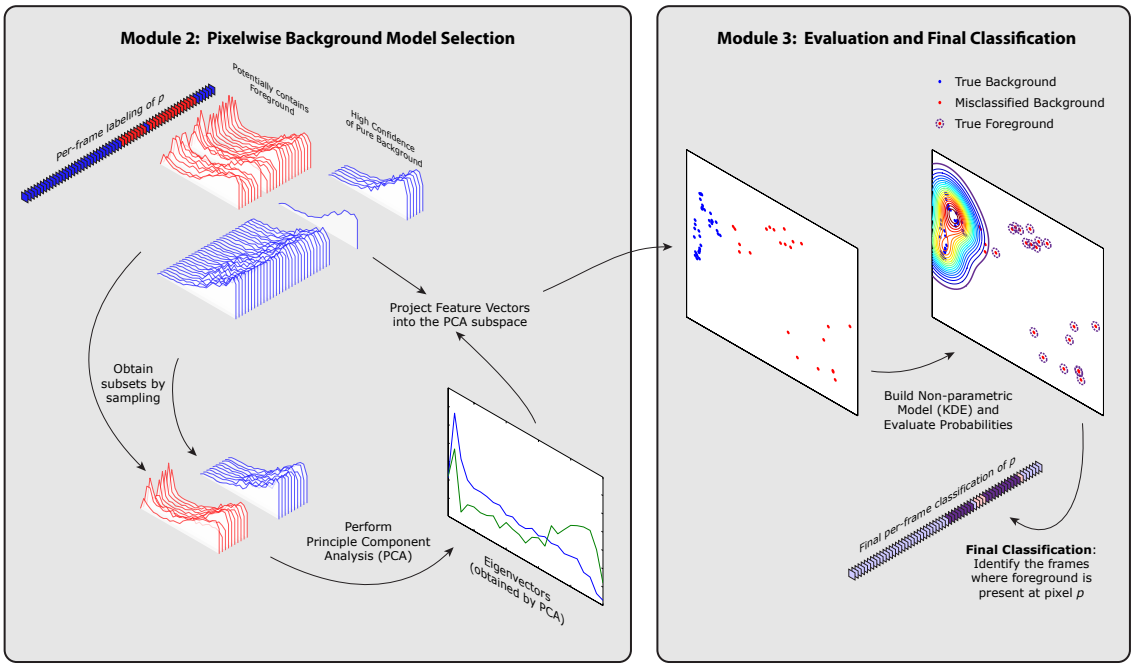


Figure 2.2: Module 2: Pixelwise Background Model Selection. Module 3: Evaluation and Final Classification.

Consider a tube of pixels $p_{ij} = \{p_{ij}^t\}$, where (i, j) denote the spatial location and

$t \in \{1, 2, .., T\}$ denotes the frame number in the video sequence. We calculate a histogram of the neighboring patch at every time step to obtain a sequence of patch histograms as shown in Figure 2.1. Every histogram in this set is projected onto the target model to obtain a 1-D time series as shown in Figure 2.1. A high response at certain times indicates probable presence of the target at those times in the neighborhood of pixel (i, j). This process is repeated for every pixel location (i, j). In summary, we identify pixels whose neighborhoods, at times, change to more closely resemble the target model. Using all patches, both the past and the future frames, has its advantages. In the videos in our dataset, the Bowerbird jumps suddenly from one location to another, and then often waits at a single location for a lengthy period of time, sometimes even thousands of frames. Using a small quantile of the time series to model the response of background patches, we are able to easily identify frames when the bird might have visited the immediate neighborhood.

We take great care not to allow target to be mixed with the background. This hypersensitivity in initial classification reduces the number of false negative target classifications at the cost of marginally increased false positive rates. At each pixel this gives us two sets, F_{ij} and B_{ij}, consisting of the frame numbers that are respectively classified as potential target and high confidence background pixels. In essence, we obtain an *over-background-subtracted* sequence of images. We can now use the reliable set B_{ij} to build more complex and robust background models.

2.2.2 Pixelwise Background Model Selection

Traditional background subtraction techniques rely on a fixed set of features to build their background and foreground models (R,G,B and gray values, gradients, edges and even texture measures). However in outdoor videos, such as the ones in our dataset, the background varies greatly in different parts of the scene as well as across different videos. Furthermore, the additional knowledge we have about the appearance of the target object should play an important role in determining which features would be most effective at different places in the image. For example, sometimes the bird walks over grass-filled regions, where color might be an important cue. At other times, it walks over bright sunlit areas, where a histogram of neighborhood intensities might differentiate it. For highly textured targets, a bank of oriented filters might be appropriate. We utilize information about pixels from both sets, potential target and high confidence background, and choose the most appropriate features for every pixel location from a "bag of features".

Consider pixel p_{ij}^t. At every time step t, we concatenate multiple features to form a joint feature vector f_{ij}^t. These could include any pixel-based or neighborhood-based features. We next determine which elements of the feature vectors are most important for distinguishing target and background pixels at location (i, j) for times $t = \{1, 2, .., T\}$. The set of potential target pixels has a large number of background pixels in it, because of the conservative thresholds we choose for the initial pixel classification. This prevents us from using a standard hard classifier to label the pixels as target and background. Instead, we use PCA to project our feature vectors onto a subspace that maximizes the variance, and KDE to classify them. This probabilistic framework allows us to remove many of

the falsely classified pixels from the potential target set. We only use a small sample of feature vectors from the target set F_{ij} and from the background set B_{ij} to obtain a reduced subspace, as shown in Figure 2.2. Projecting the entire feature set f_{ij} onto this subspace gives us the set r_{ij}, in the reduced space. The reduced dimensionality of r_{ij} helps to drastically reduce the time required to build background models.

2.2.3 Evaluation and Final Classification

For every pixel we build a background model using Kernel Density Estimation on our reduced feature set and evaluate probabilities at all time frames that were initially classified as potential target F_{ij}. Suitably thresholding these probabilities allows us to further break down the set F_{ij} into a set of target pixels and pixels that were misclassified as target by the first module of our system. For $t \in B_{ij}$ (background), $s \in F_{ij}$ (potential target) and kernel K, we obtain:

$$P(r_{ij}^s) = \frac{1}{N\sigma_1..\sigma_d} \sum_{t \in B_{ij}} \prod_{y=1}^{d} K\left(\frac{r_{ij,y}^s - r_{ij,y}^t}{\sigma_y}\right) \tag{2.1}$$

This gives us a target silhouette at every frame of the video sequence, from which we are able to calculate the centroid of the detected region at every time step. We compare these centroid locations to ground truth provided to us by the biologists, and present our results in the following section.

2.3 Computational Considerations

Our implementation of the framework described in Section 2.2 incorporates highly optimized algorithms to facilitate the processing of these large videos. We utilize Integral

Histograms [70] both to generate the patch histograms used in pixel classification and to generate features for background model selection. Further, to optimize the evaluation stage, we build KDEs and determine probabilities using the Improved Fast Gauss Transform (IFGT) [72, 97]. The framework is implemented in MATLAB, with computation- and memory-intensive algorithms such as Integral Histograms and IFGT implemented in C++ and compiled as mex routines. In addition to these algorithmic optimizations, we also employed many workstations[2] (a subset of the *vnode* cluster funded through NSF Infrastructure Grant CNS 04-3313) to process multiple videos in parallel.

A key strength of our background modeling approach is the use of a large spatio-temporal window. We consider image statistics, both in a large region around a given pixel and also over a large temporal interval (the entire video). Computing statistics for each image pixel over this large temporal window requires a tremendous amount of data storage. The amount of memory needed to store a single byte per pixel over 10,000 VGA sized frames is 2.86GB. We compute feature vectors per pixel that would require about 100 or more bytes of memory per pixel (25 or more floating-point features). If this entire structure were to be in memory at one time, it would require 100s of GB of memory, rendering this task impossible for even a modern PC. We are further-constrained by the memory limits of a 32-bit version of MATLAB (only about 1.2GB are available for variables).

These considerations led us to implement our processing using data-decomposition as is frequently done in high performance scientific computing (though we process a given video serially on a single machine where a distributed system would run in parallel). We

[2]Workstations have dual 3.0Ghz Intel Xeon processors, 8GB RAM

utilize two kinds of data-structures, *tubes* and *chunks*. Tubes refer to spatial subdivisions of the video (entire space-time volume), such that all frames for a particular subregion of the image fit simultaneously in memory. Chunks are temporal subdivisions, a contiguous set of frames in time that simultaneously fit into memory. These tubes and chunks must be created for not only the original image frames of the video but also for the large data structures that we accumulate during processing. At different stages, our algorithm requires reading in all the data, on a tube-by-tube or a chunk-by-chunk basis.

2.4 Experimental Evaluation

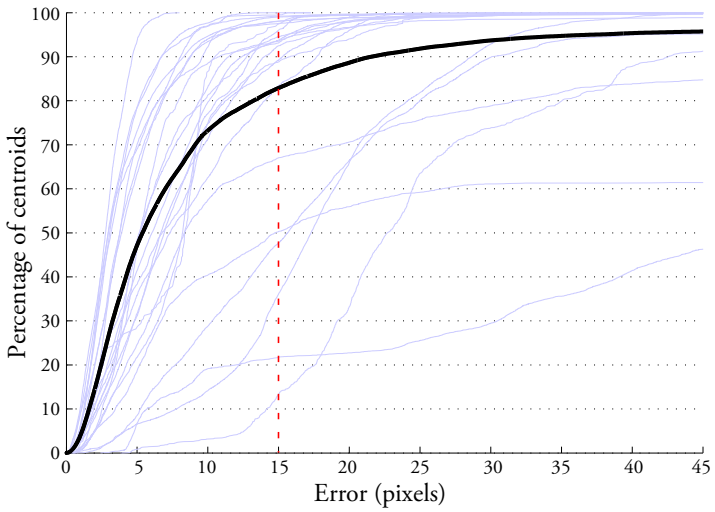


Figure 2.3: Overall Accuracy. Cumulative distribution of accuracy of every video is shown by faint lines. Overall accuracy is shown by the solid line. Red dotted line denotes the accuracy (in terms of centroid detection error) required by the biologists.

Particularly with such a large amount of data, we want to identify metrics that quantitatively assess the quality of our framework. While hand-labeling ground truth target centroids for 200,000 frames is infeasible, we are fortunate to have what we consider a

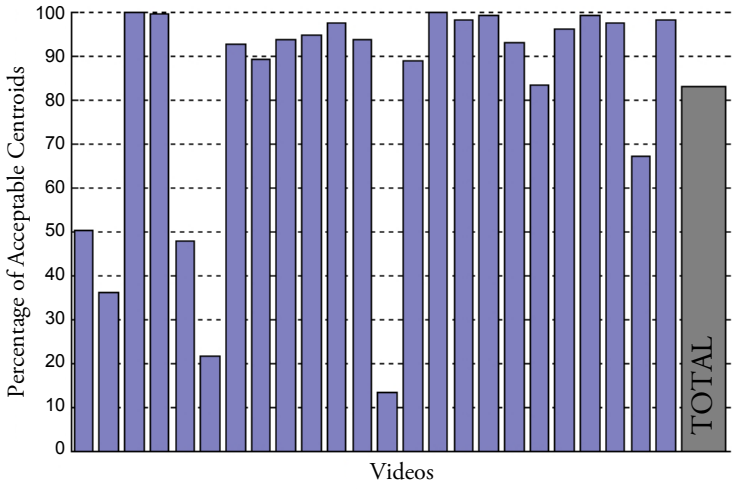


Figure 2.4: Percentage within Tolerance. Per-video percentage of centroids within the biologists specified tolerance.

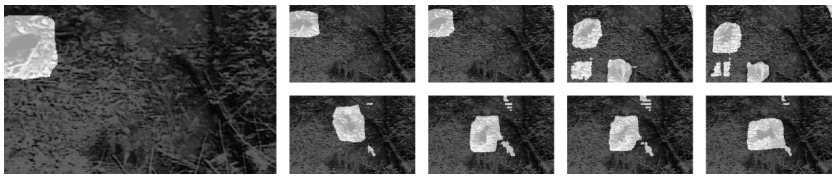
close second. In their study [76], our collaborators used an application implementing a very early prototype of our software. The application included a provision to manually correct the automatic tracking results. On some videos, the results were almost entirely satisfactory, while on a few very difficult videos, manual tracking was required for a fair number of frames. Thus the biologists went through and refined the automatic tracking results such that the centroids were within the acceptable tolerance of 4.5cm in the real world (about 15 pixels in the image).

Given this "ground truth", we seek to evaluate our approach using the following metrics: overall accuracy, per-video percentage within the biologist-specified tolerance, and false-positive and false-negative rates. In Figure 2.3, we present a cumulative distribution of overall accuracy. All videos are superimposed and the required tolerance is shown by the red dotted line. The overall cumulative distribution is shown by the solid line. Figure 2.4 shows the per-video percentage of centroids within the specified tol-

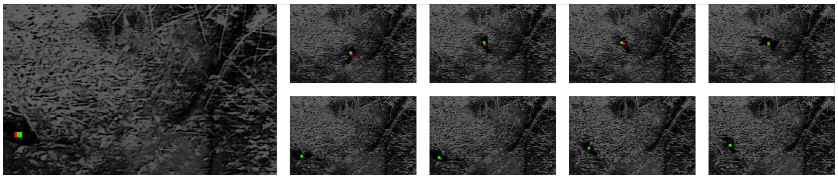
erance. Overall, we are able to track the target within the biologists error tolerance in 82.89% of the frames in our dataset. For most of our videos this number goes beyond 90%. Having to hand label thousands of frames of data for every video in their biological study, biologists often spend days just tracking the object of interest. An accuracy of over 90% represents a very significant reduction in the time required for this process. We obtain an overall false positive detection rate and false negative detection rate of 4.8% and 3.44% respectively. Our false positive detections are primarily caused by moving shadows cast by the overlying trees, and our false negative detections are primarily seen to be caused by severe occlusions by large branches and shrubs in the scene. It is often easier to manually correct false positives as compared to false negatives. The biologist can mark out a sequence of frames when the target is not present in the scene and all false positives within that range can be ignored. Fig.2.4 shows poor results for three of the videos in the dataset. These are caused by severe occlusions by large shrubs in the scene, making it very difficult to locate the target accurately.

Fig.2.5a shows a few frames from one of the videos in the database, sampled approximately every 300 frames. The stark illumination changes from one part of the video to another can be clearly seen. Fig.2.5b and Fig.2.5c show the results of the two modules in our staged approach to target detection. Some of the videos also had a very poor contrast between the target and background pixels, due to the dark shadows cast by the overlying trees, and the dark color of the male bowerbird. Fig.2.6 shows an example frame and detection results from one such video.

(a) Frames from one of the videos in the database showing the male bowerbird to be tracked. Notice the stark illumination and color changes in the sequence.



(b) Initial pixel classification by Module 1 for the above frames. The shaded pixels are classified as potential target pixels. They include a large number of background pixels as well due to the conservative thresholds set in Module 1.



(c) Final results for the above frames. The detected centroid of the target is marked with a green dot, and the ground truth is shown in red.

Figure 2.5: Target detection for a sequence with stark illumination changes.

Figure 2.6: Target detection for a sequence with poor contrast between target and background.

Chapter 3

Vehicle Detection using Partial Least Squares

3.1 Introduction

Several commercial earth observation satellites, such as IKONOS, GeoEye and QuickBird, provide publicly available imagery at a ground sampling distance (GSD) of 1 meter. High resolution images of a small number of locations are publicly available via Google Earth at an astonishing GSD of 0.15 meter. We consider the problem of detecting vehicles from such high resolution aerial and satellite imagery. Object detection systems have typically used image descriptors such as Scale Invariant Feature Transform (SIFT) [57] and Geometric Blur [5], calculated at a number of interest points within the image. These image descriptors are then combined using various aggregating approaches such as Bags-Of-Words [103] and Spatial Pyramids [52] to provide a rich description of the object. However, such approaches have not been extensively used for the problem of vehicle detection, due to the low resolution of traditional aerial images and the need of many commonly used image descriptors for a sufficiently large support region.

Vehicle detection has been previously treated as a template matching problem, and several algorithms have been proposed that construct templates in 2D as well as 3D. Moon et al. [61] propose an approach to optimally detect two dimensional shapes. They derive an optimal one dimensional step edge detector which minimizes the noise power and mean squared error between the input and filter output. This turns out to be the derivative

of the double exponential (DODE) function. The DODE filter is then extended along the shape's boundary contour to obtain a shape detector. The problem of vehicle detection is then equivalent to *detecting parallelograms*. They show impressive results on images of vehicle parking lots. A comprehensive analysis of this vehicle detector under a wide range of operating environments was carried out in [62]. A mathematical analysis was provided to quantify the degradation of the vehicle detector with decreasing illumination and acquisition angle.

Choi et al. [18] use a mean shift based clustering algorithm to extract candidate blobs that exhibit symmetry properties of typical vehicles. Each candidate is then classified using geometric and radiometric characteristics of the blob. Eikvil et al. [22] propose a similar two stage strategy for vehicle detection in satellite images. The first stage consists of segmenting regions into potential vehicles, roads, vegetation, etc. They also leverage multi-spectral information to identify regions of vegetation and Geographical Information System (GIS) data to obtain the road network. The second stage then consists of a region classification algorithm using geometrical properties such as area, moments, etc. Zheng et al. [105] obtain vehicle candidates using a morphological pre-processing stage, which are then classified using a neural network. Such two stage approaches typically suffer from errors obtained in the segmentation stage of the system. Furthermore, geometric properties of blobs are not powerful enough to detect vehicles with high accuracy in urban settings, where the presence of a large number of rectilinear structures cause many false alarms.

Zhao et al. [104] pose the vehicle detection problem as a 3D object recognition problem. They used human knowledge to model the geometry of a typical vehicle. Psy-

chological tests revealed that human subjects most often used cues such as the rectangular shape of the car, layout of windshields and presence of shadows to detect cars. Such cues were then integrated using a Bayesian network. They also made use of camera calibration and illumination information to predict shadow cues. While effective, their algorithm cannot be easily extended to build other object detection systems, due to the large amount of human modeling that is required. Similar 3D models were also used to model vehicle geometries for the purpose of car detection and counting in aerial images by Hinz [38] and Schlosser et al. [77].

Grabner et al. [33] propose an online version of boosting to efficiently train their vehicle detector. Their algorithm avoids building large pre-labeled training datasets by making use an active learning framework. They use three classes of features - Haar wavelets, Histograms of Oriented Gradients and Local Binary Patterns, all of which can be very efficiently calculated using Integral Images and Integral Histograms. Their detection results are further improved by segmenting the image into streets, buildings, trees, etc. and then discarding vehicle detections that are not present on the street segments.

More recently, vehicle analysis has been extended from single images to video sequences. Yue et al. [99] propose a system for vehicle verification in airborne video sequences. The vehicle of interest may leave the field of view for a while or may be obscured. When a new vehicle is observed, verification is needed to confirm whether it was the previously detected vehicle. A homography based view synthesis method is used to generate novel views of the exemplars that are provided during training. This enables the system to be robust to large aspect angle variations of the test sequence. The synthesized novel view and testing object are then compared using a weighted combination of a

rotationally invariant color matcher and a spatial feature matcher.

Our proposed vehicle detector improves upon previous systems by incorporating a much larger and richer feature set than previous approaches. A new feature set called *Color Probability Maps* is used to capture the color statistics of vehicles and their surroundings, along with the *Histograms of Oriented Gradients* feature and a simple yet powerful image descriptor that captures the structural characteristics of objects, named *Pairs of Pixels*. The combination of these features leads to an extremely high dimensional feature set (approximately 70,000 elements). Partial Least Squares is first used to project the data onto a much lower dimensional subspace. Then, a powerful feature selection analysis is employed to improve performance, while vastly reducing the number of features that must be calculated.

We demonstrate our proposed vehicle detector on two datasets. The first consists of color images collected from a satellite and obtained via Google Earth. This is a set of 40 high resolution images over the city of San Francisco. The second dataset is the publicly available Overhead Imagery Research Data Set which consists of a large number of aerial images, annotated with vehicles [85]. We compare our approach to several previously proposed object detection approaches [20, 52, 58, 61] and obtain favorable results.

3.2 Feature Extraction

We use three classes of features in our proposed solution: Color Probability Maps, Histograms of Oriented Gradients and Pairs-of-Pixels (PoP) features.

3.2.1 Color Probability Maps

Vehicles often lie on homogeneously colored backgrounds such as asphalt, cement, dirt roads, etc. Sometimes, the immediate neighborhood of a vehicle might be composed of multiple surfaces (a vehicle parked on the side of the road has an asphalt surface on three sides and a cement sidewalk on the fourth side). Thus, an image patch containing typical vehicle colors towards the center and colors representing typical backgrounds towards the periphery is likely to contain a vehicle. Color Probability Maps capture such color statistics of objects and their immediate environment.

We begin by identifying colors that are typically present in the background category. First, pixels are sampled from the entire set of background image patches in the training set of images. Each pixel is represented in a 3 dimensional space (r, g, s), where r and g are chromaticity variables representing the fraction of the red and green components respectively: $r = \frac{R}{R+G+B}$ and $g = \frac{G}{R+G+B}$. s represents the brightness component: $s = \frac{R+G+B}{3}$. These pixels (obtained from all the background training images) are clustered to determine the dominant colors present in the background. Each cluster of pixels is used to build a color density model in RGB space using Kernel Density Estimation.

$$\begin{aligned} p^c(r,g,s) &= \frac{1}{N^c_{pts}} \sum_{i=1}^{N^c_{pts}} K_{\sigma_r}(r - r^c_i) K_{\sigma_g}(g - g^c_i) K_{\sigma_s}(s - s^c_i) \\ K_\sigma(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \end{aligned} \tag{3.1}$$

where $p^c(r, g, s)$ refers to the probability of the (r, g, s) triple for the c^{th} cluster. σ_r, σ_g and σ_s represent the channel bandwidths. r^c_i,g^c_i and s^c_i refer to the chromaticity and brightness components of the i^{th} pixel in the c^{th} cluster. N^c_{pts} refers to the number of points in the c^{th} cluster. Given an image, one can obtain a color probability map for every color

model. These probability maps are concatenated to form a feature vector representing the color statistics of the image patch, F_{cmap}. In order to limit the number of probability maps that must be computed for each image patch, only the most discriminating clusters are used. First, clusters that contain a very small number of points are rejected. Then the remaining clusters are ranked based on their discriminative capability. For a given cluster, we generate the corresponding probability map for all positive and negative samples in the training set and calculate the average misclassification error using a 5-fold cross validation procedure. The top N_{cmap} clusters are then chosen.

The Improved Fast Gauss Transform (IFGT) [63] is used for efficient computation of the probabilities. The bandwidths for each channel are estimated independently using a bandwidth selection criterion given in [72]. Given a test image patch, the computation of its Color Probability Maps is further speeded up by *a priori* constructing look-up tables that directly map entries in the $R - G - B$ color space to a probability value. A look-up table is constructed for each of the N_{cmap} color clusters.

3.2.2 Histograms of Oriented Gradients

The second class of features are the Histograms of Oriented Gradients (HOG), which have been used in many object detection algorithms [20]. These features capture the spatial distribution of gradients that are typically observed in image patches that contain vehicles. Since histograms are computed over regions, they are fairly robust to some variability in the location of the parts of the object. Moreover, the HOG descriptor is also invariant to rotations smaller than the size of the histogram orientation bin.

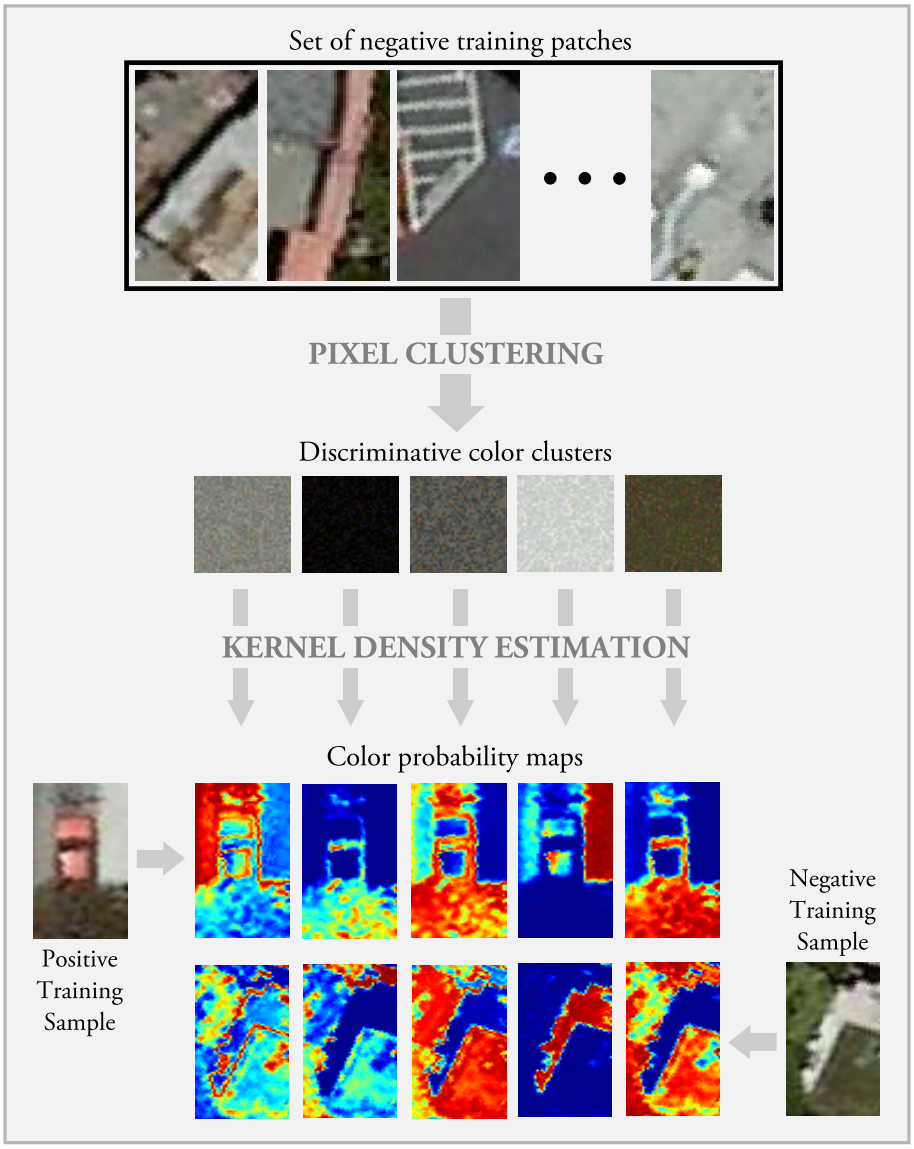


Figure 3.1: **Color Probability Maps.** Pixels extracted from negative training image patches are clustered to obtain models of typical colors observed in the background. Given a new image patch, kernel density estimation is then used to obtain a probability map corresponding to each color cluster.

Each detection window is divided into square cells and a 9-bin HOG feature is calculated for each cell. Grids of 2x2 cells are grouped into a block, resulting in a 36 dimensional feature vector per block. Each block feature vector is normalized to an L2 unit length. Dalal et al. [20] used blocks defined at a single scale. In their approach, for

a detection window of size 64×128 pixels, 105 blocks were used, each having a size of 16×16 pixels. Zhu et al. [106] extended this approach by using blocks at varying scales and varying aspect ratios (1:1, 1:2 and 2:1). We incorporate this multi-scale approach employed in [106].

3.2.3 Pairs Of Pixels

Properties of pixel pairs within an image patch can provide structural information about the object present in that patch. Consider the image patch shown in Figure 3.2. The structure of a car shown in the image can be described using the relationships between the regions highlighted in red, green and blue. The three regions highlighted in red represent the body of the vehicle and typically have the same color. Similarly, regions highlighted in green represent the windows of the vehicle which usually appear dark in color. Regions that are not present on the vehicle might have colors that differ from the color of the vehicle but they might be similar to one another. Such relative color statistics can capture the structure and relationships amongst different parts of a given object. Using relative properties of pixel pairs, as opposed to pixel properties themselves, is robust to illumination changes in the scene, changes in the background, as well as the color of the object itself.

In principle, one can encode many different relative properties of regions, such as the difference between their colors, textural properties, gradient magnitudes, etc. Here, we restrict these regions to be single pixel locations and the relative property to be the Euclidean distance between their color values. The feature vector, F_{pop}, encoding this

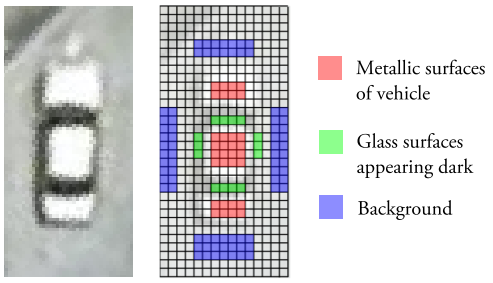


Figure 3.2: The structure of a typical car and its surrounding regions can be described using pairwise relationships between the highlighted regions. Regions that are marked with the same color typically have the same color and texture properties. This information is captured by the PoP feature.

relative property can be obtained by concatenating the distances between the colors of a large number of pixel pairs.

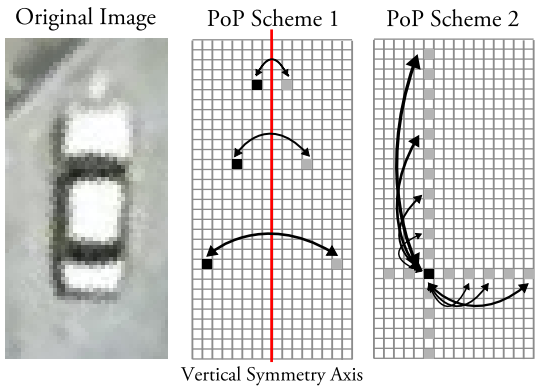


Figure 3.3: POP Schemes. Scheme 1 captures differences between pairs of pixels that lie symmetrically across the central vertical axis of the image patch. Scheme 2 captures differences between pairs of pixels that lie in the same row and column.

If we were to consider all pixel pairs in an image patch, the feature vector would be of length M^2N^2 where M and N are the length and width of an image patch. For a small image patch of size 100 x 100, this would result in a feature vector of length 100 million. In order to restrict this dimensionality, we propose two alternatives. These alternatives are designed to capture a large portion of the structural information, while taking advantage

of the symmetry exhibited by vehicles about a central vertical axis.

Scheme 1: Consider all pairs of pixels that exhibit a symmetry in location about the central vertical axis, as shown in Figure 3.3. This reduces the dimensionality of the feature vector to $M \times \frac{N}{2}$. However, these pixel pairs only capture horizontal differences.

Scheme 2: Consider a pixel p at location (x, y). Consider its differences with all pixels that lie in the same row and column as pixel p at intervals of distance d. This provides an advantage over Scheme 1 by capturing structural properties in both the horizontal and vertical directions while restricting the dimensionality of the resulting feature vector. Using Scheme 2 results in a feature vector of length $M \times N \times (1 - \frac{M+N}{d})$.

We use Scheme 2, since we are able to accommodate the length of the resulting PoP feature vector. The PoP vector is then normalized by dividing it by its L_1 norm.

The three classes of features are finally combined to form the resulting feature vector describing an image patch.

$$F = [F_{cmap} \quad F_{hog} \quad F_{pop}] \tag{3.2}$$

3.3 Partial Least Squares

The combination of the three feature classes results in an extremely high dimensional feature space (approximately $70,000$ dimensions). In contrast, the number of samples in our training dataset is much smaller (about 200 in the positive and 1500 in the

negative class). Furthermore, our features are extracted from neighboring pixels within a detection window, which tremendously increases the correlation between them, rendering traditional Ordinary Least Squares (OLS) regression estimates unreliable. This phenomenon is also known as the multicollinearity of the feature set. The nature of our proposed feature set makes an ideal setting for a statistical technique known as Partial Least Squares (PLS) regression [93].

The PLS method was first developed by Herman Wold in the 1960s and 1970s to address problems in econometric path-modeling [92], and was subsequently adapted in the 1980s to problems in chemometric and spectrometric modeling. In the late 1980's and 1990's, PLS attracted the attention of statisticians [30][36], due to its ability to deal with a small number of examples and a large number of variables.

We present a brief introduction to PLS. For a more detailed analysis, see [12]. Consider a set of p predictor variables, $X_1, X_2, \ldots X_p$, which are used to predict q response variables, $Y_1, Y_2, \ldots Y_q$. Let n equal the number of observation pairs denoted as (x_i, y_i) where $\{i = 1, 2, \ldots, n\}$. The data samples are assumed to be mean-centered. They are concatenated to form the matrices $X_{(n\times p)}$ and $Y_{(n\times q)}$. When $n < p$, classical regression tools cannot be applied since the covariance matrix $X^T X_{(p\times p)}$ is singular.

PLS regression is based on the following latent component decomposition

$$X = TP^T + E \tag{3.3}$$

$$Y = UQ^T + F \tag{3.4}$$

where, T and U give the latent components (known as the *scores* matrices), P and Q provide the coefficients (known as the *loadings* matrices), and E and F are the error

matrices. Note that a decomposition similar to Equation 3.3 is obtained by Principal Components Analysis.

The latent components given by T are obtained by a linear transformation of X as follows,

$$T_{n\times d} = X_{n\times p} W_{p\times d} \tag{3.5}$$

where d is the dimensionality of the latent space. The latent components T, are used for prediction in place of the original data vectors X. There are many variants of the basic PLS algorithm. They can be broadly classified based on their ability to deal with univariate response variables versus multivariate response variables. Multivariate response PLS has two popular implementations. The first variant leads to the NIPALS algorithm, whereas the second variant leads to the SIMPLS algorithm. These two methods differ in the matrix deflation process within the PLS algorithm. In our analysis, we have used the NIPALS algorithm. The NIPALS algorithm is essentially one of many methods that exist for finding the eigenvectors of a matrix. It was originally developed for Principal Components Analysis, but was subsequently used to iteratively extract factors for in a Partial Least Squares Analysis. Algorithm 1 provides a brief outline of the NIPALS algorithm. For more details we refer the reader to [31].

W_i and T_i represent the i^{th} columns of the matrices W and T respectively. The regression model is given by,

$$Y = XB + F \tag{3.6}$$

where, $B_{p\times q}$ is the matrix of regression coefficients. Algebraic manipulations yield,

$$B = WQ^T = W(T^TT)^{-1}T^TY \tag{3.7}$$

Algorithm 1 NIPALS Algorithm

1: **for** $i = 1$ to d **do**

2: Matrix Projection

$$W_i = X^T Y / \left\| X^T Y \right\|$$

$$T_i = XW_i / \left\| XW_i \right\|$$

3: Matrix Deflation

$$X = X - T_i T_i^T X$$

$$Y = Y - T_i T_i^T Y$$

4: **end for**

A new observation x_{new} thus yields a response value given by,

$$y_{new} = \frac{1}{n} \sum_{i=1}^{n} y_i + B^T (x_{new} - \frac{1}{n} \sum_{i=1}^{n} x_i) \tag{3.8}$$

The data we are interested in, falls into two classes - vehicles and background. We use the PLS regression algorithm as a *class aware* dimensionality reduction tool by setting the class label of a sample in Y to 1 or -1. Thus, for our purpose, $q = 1$. Note that the matrix of regression coefficients B, is now a single vector $(B_{p\times 1})$, with a single coefficient for every feature. In practice, we do not project a new observation onto B. Instead, we project it onto the first k columns of matrix W, and then apply a classifier on that subspace. This method allows us to apply any classifier within this subspace (linear or non-linear) and has been shown to provide improved performance. The number of PLS factors k is obtained using cross validation.

Dimensionality reduction techniques can be broadly classified in two ways: linear vs non-linear methods and supervised vs unsupervised methods. Non-linear methods are

generally computationally intensive and are not very suitable for extremely high dimensional feature spaces, such as ours. For the purposes of classification, supervised methods hold an advantage over unsupervised methods, owing to their use of the class information within the training dataset.

Principal Component Analysis (PCA) is a classical linear unsupervised method. While PCA creates orthogonal latent vectors by maximizing the covariance between the data vectors x_i, PLS (a supervised approach) also considers the class labels. Figure 3.4 demonstrates the advantage of PLS over PCA. A subset of the training dataset (80% of the data points with all 69,552 variables) is provided to both PCA and PLS, and then projected onto the first two factors given by each dimensionality reduction method. The first row shows the training points projected onto the subspaces. The second row shows the test points (the remaining 20% of the data) projected onto the subspaces. The first two factors given by PLS are clearly more discriminating than PCA.

Fisher Discriminant Analysis (FDA) is in this way similar to PLS. It is a linear supervised method. However, FDA suffers from the *small sample size* problem. When the number of features exceeds the number of samples $(n < p)$, as in our case, the covariance estimates are not full rank, which are required to obtain the projection vectors. A number of extensions have been to LDA have been proposed to deal with this problem. Belhumeur et al. [4] first projected points onto a lower dimensional subspace using PCA (which does not suffer from the small sample size problem), and then applied FDA on the reduced subspace. Chen et al. [17] used a modified version of Fisher's criterion and proposed an efficient and stable algorithm to calculate the discriminant subspace. However, FDA has a further limitation, in that it retains only $l - 1$ meaningful latent vectors, where l is the

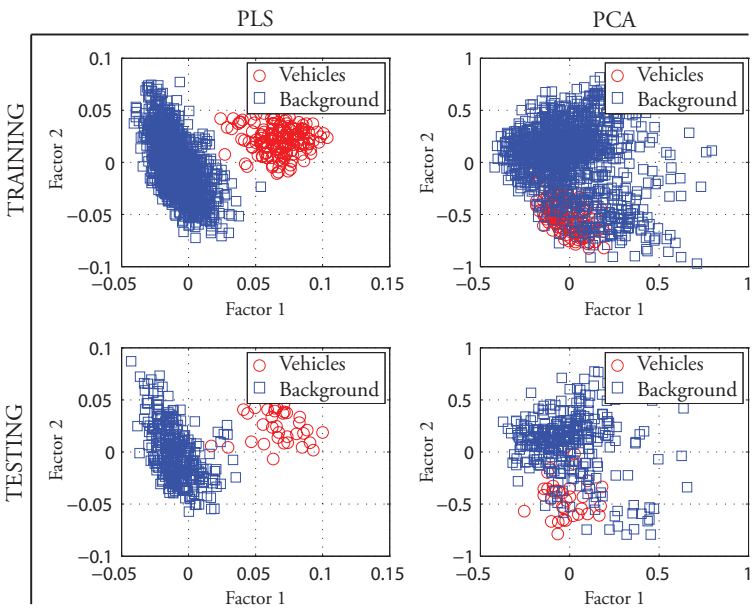


Figure 3.4: Projection of data points from a 69,552 dimensional feature space onto a 2 dimensional subspace. In this illustration 80% of the training dataset is used to obtain the subspaces, and the remaining 20% of the data from each class are used as the testing samples. The left column shows the subspace extracted using PLS. The right column shows the subspace extracted using PCA. Clearly, PLS extracts a subspace that is more discriminating than PCA.

number of classes being considered. For our 2 class problem, a 1 dimensional subspace might not be sufficiently discriminative.

3.3.1 Visualization of PLS factors

It is useful to be able to visualize data points in the reduced PLS latent space. We propose a technique to visualize the separation between the data points in the two classes as the dimensionality of the PLS latent space increases.

Figure 3.5 shows the visualization of data in a 5 dimensional PLS factor space. The top left plot shows the data plotted in the space spanned by the first two PLS factors. One can observe that the two classes are not completely separated in this 2 dimensional

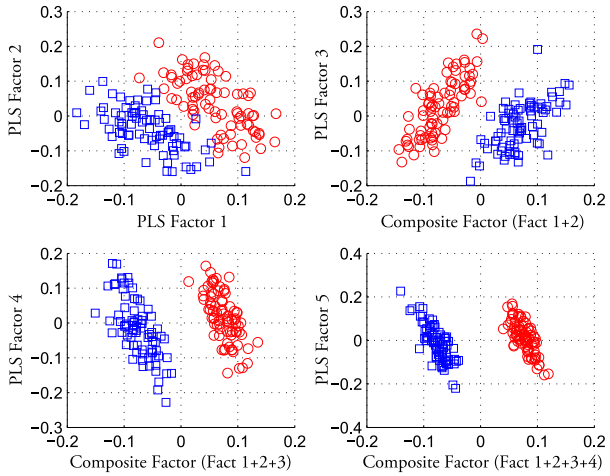


Figure 3.5: Visualization of multiple PLS factors. The effect of the n^{th} PLS factor can be visualized by plotting it against the composite factor obtained by combining the first $n-1$ PLS factors.

space. In order to visualize the effect of the third PLS factor, the data points are plotted in the space spanned by the third factor and the composite PLS dimension obtained from the first two PLS factors. The composite factor is essentially the vector of regression coefficients (B) given in equation 3.7. The data projected on B represents the output of PLS regression obtained using a 2 dimensional latent space. The top right plot shows the data projected on B as against the third PLS factor. In this way, data projected on factor i is plotted against the data projected on the composite factor obtained from the PLS factors 1 to $i-1$.

One can observe the increase in the separation between the two classes as the number of PLS factors is increased. This visualization is a useful tool to observe the structure of the data in the two classes in a high dimensional PLS factor space.

Note that it cannot be used to determine the optimum number of PLS factors for a given problem, which is obtained by a cross validation procedure on the training dataset.

3.4 Feature Selection

The large number of features greatly increases the computational cost of the vehicle detector. Calculating the features requires a substantial amount of time. At the same time, projecting a data point from an p dimensional space down to a d dimensional PLS factor space requires $p \times d$ multiplications and $(p-1) \times d$ additions. A typical image which must be scanned at multiple orientations and scales has hundreds of thousands of windows to be evaluated. Clearly, this would be a very time consuming process. This can be greatly speeded up, without a significant loss in performance, by using an effective feature selection process.

Furthermore, feature selection can often improve the performance of a PLS based classification system. In our set of thousands of features, one can expect many of them to be very noisy and redundant. Feature selection can help discard a large fraction of such variables.

We use two methods to perform feature selection: Ordered Predictive Selection and a Multi-Stage Multi-Resolution Analysis.

3.4.1 Ordered Predictors Selection (OPS)

Variable Importance on Projection (VIP) is a widely used technique for PLS based feature selection. VIP provides a score for every variable, that ranks them according to their predictive power. A cross validation scheme can then be used to select the number of variables required to obtain a desired level of classification accuracy. In general any informative vector which provides a measure of the predictive power of the variables, can

be used for feature selection.

Teofilo et al. [86] used several informative vectors and their combinations to perform feature selection for regression problems. Their method is computationally efficient as compared to other variable selection methods, such as genetic algorithms, and can be completely automated. They used several different datasets in their analysis, some having a multiple number of dependent variables. Their analysis showed a rather surprising result. The number of PLS factors that were obtained by a cross validation procedure for the purpose of feature selection was often higher than the number of PLS factors that were optimal for the purpose of regression.

We perform a similar feature selection analysis, enabling us to reject a large fraction of noisy features. We only deal with a single dependent variable, which is set to the class label ($+1/-1$). The following informative vectors are used in our analysis.

1. Variable Importance on Projection (VIP) - The VIP score for the j^{th} variable is a measure based on the weighted PLS coefficients. The higher the score, the more importance a variable presents. The average VIP score over all variables equals 1. The *thumb rule* used to select variables according to their VIP score is to retain only those variables whose VIP score is greater than 1.

$$\text{VIP}_j = \sqrt{p \sum_{k=1}^{d} B_k^2 W_{jk}^2 / \sum_{k=1}^{d} B_k^2} \tag{3.9}$$

2. Regression Coefficients (B) - The regression coefficients B defined in Equation 3.7 represent the expected change in the response, per unit change in the variable. The absolute value of the regression coefficients are thus used as informative scores.

3. Correlation (CORR) - The correlation informative vector contains the Pearson correlation coefficients between every predictor variable X_i and the response variable Y. A high correlation indicates that the predictor variable is very informative about the PLS classification model.

$$\mathrm{R} = \frac{X^t Y}{n-1} \tag{3.10}$$

Since the Pearson correlation coefficient lies between -1 and $+1$, with 0 indicating an absence of any correlation, the absolute value of the correlation coefficients is used.

4. Covariance Procedures Vector (CVP) - The CVP score was proposed by Reinikainen et al. [73] as a measure of variable importance. The ranking of variables is based on the covariance of the dependent and independent variables which is given by,

$$\mathrm{CVP} = diag(X^t Y Y X) \tag{3.11}$$

Informative Vectors CORR and CVP do not depend on the dimensionality of the PLS factor space. However, VIP and BETA vary with the dimensionality of the factor space. Motivated by the OPS results obtained in [86], a nested cross validation procedure is used to determine the optimum dimensionality of the PLS factor space for classification and the optimum dimensionality of the PLS factor space to calculate the informative vector.

3.4.2 Multi-Stage Multi-Resolution Analysis

The nature of the three feature classes introduces a fair amount of redundancy in the feature set. For instance, the color probability maps capture color statistics of every pixel

Algorithm 2 OPS Cross Validation Procedure

1: **for** $\theta = 1$ to N_{select} **do**
2: Build PLS model using θ factors
3: Calculate informative vector and sort variables by their informative score
4: Choose top K variables based on feature selection criteria
5: **for** $\phi = 1$ to N_{model} **do**
6: Build PLS model using ϕ factors and top K variables
7: Calculate classification accuracy on the validation set
8: **end for**
9: **end for**

within an image patch. Clearly, neighboring pixels are highly correlated and introduce a lot of redundancy. Downsampling an image can help remove this redundancy somewhat, at the cost of performance.

We build 2 PLS models using features computed from training images at different resolutions. The first model is built from training images reduced to a width and height that equal $1/2$ the original image dimensions, vastly reducing the dimensionality of the feature vector. The second model is computed from the original training images. The features that contribute towards each PLS model, undergo the OPS feature selection strategy that was outlined in Section 3.4.1. Thus we obtain two trimmed PLS models. In principle, one may add more downsampling stages to obtain a further speedup, possibly at the cost of accuracy.

Given a testing image, each and every image patch is classified using the first and fastest PLS model. A subset of these are then sent to the second stage which contains the

second PLS model (lowest speed, highest performance).

3.5 Experiments

3.5.1 Google Earth San Francisco Dataset

We test the performance of our vehicle detector on satellite images of the city of San Francisco taken from Google Earth. This dataset consists of 40 satellite images at a resolution of 979×1348 pixels and a color depth of 24 bits per pixel (RGB). Figure 3.6 shows one such image. Each image looks down on an urban scene with multiple cars present in each image. The total number of vehicles present in the dataset is 650. The average size of a vehicle is 48×16 pixels[1].

The first 5 images in the dataset are used for training purposes and the performance of our vehicle detector is tested on the remaining 35 images. 184 positive image patches (vehicles) are extracted from the training images and aligned vertically. Similarly 1500 negative image patches are randomly chosen from the training images. Each image patch has a size of 81×41 pixels. Figure 3.6 shows a few training image patches from the dataset. The test images are scanned using a sliding window approach. The size of the window is fixed to 81×41 pixels, since the training and test images have been captured at the same resolution and all vehicles are observed at a single scale. More generally however, one may need to scan the image at multiple scales. The horizontal and vertical step sizes are set to 5 pixels.

[1]The highest resolution for current commercial satellite imagery is 0.5 meters, where as the images used in the San Francisco dataset correspond to a resolution of 0.1 meters. We have used images captured using the software: Google Earth. Images provided by Google Earth beyond this resolution are obtained by digitally enlarging the images.

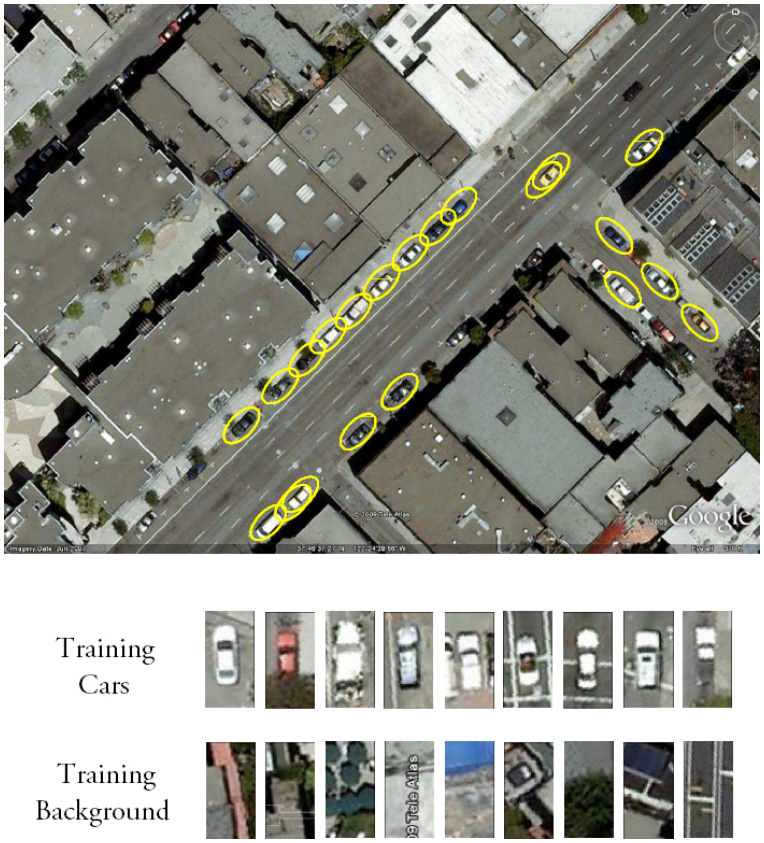


Figure 3.6: Example images from the Google Earth San Francisco dataset. Top image shows a test image from the dataset, looking down on an urban scene, with overlaid detections using our vehicle detector. Bottom image shows a few training images patches from both classes. © 2009 Google

3.5.2 Feature Extraction and Evaluation

Each image patch has a size of 81×41 pixels. 6 color clusters are used to build our color probability maps. This results in a feature vector of length 19,926. The HOG feature is calculated for 661 blocks ranging from a size of 12×12 pixels to a size of 80×40 pixels. Each blocks yields a 36 dimensional feature vector resulting in a total length of 23,796. The PoP feature is calculated using images reduced to size 41×21, which gives a feature of length 25,830. Thus, the length of the resulting feature vector obtained by combining the three feature classes is 69,552.

All the parameter selection in our system is performed by using 3 iterations of a 5-fold cross validation scheme. The analysis presented in Figures 3.7 to 3.14 was carried out using this cross validation procedure on the Google Earth San Francisco training dataset.

Figure 3.7 shows the mean classification error obtained using each of the three feature classes separately as well as their combination. As the number of PLS factors is increased, the classification error reduces. Beyond a certain value however, addition of PLS factors does not yield an improved performance. This saturation point is generally regarded as the optimum number of PLS factors. In some cases, performance decreases as more PLS factors are introduced. The PoP feature class outperforms the other two feature classes, but the best performance is obtained when all three feature classes are combined. The number of factors chosen, when the entire set of features is used, is 5.

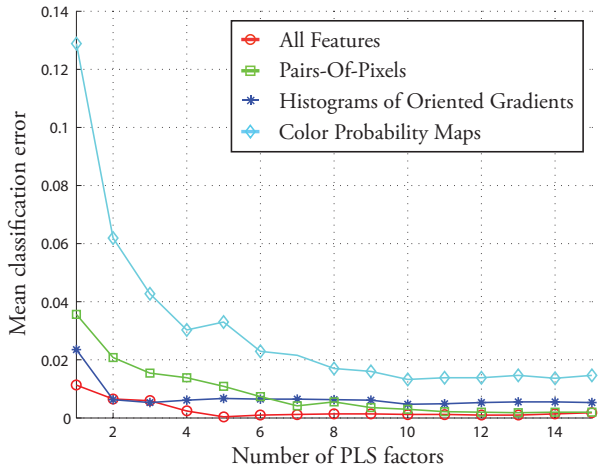


Figure 3.7: Mean classification error obtained after 3 iterations of 5-fold cross validation on the Google Earth San Francisco training dataset. The error plot is shown for each class of features individually, as well as their combination.

Figure 3.8 shows the components of the first 5 PLS latent vectors ($W's$) that correspond to the color probability maps and the PoP features. The latent vectors correspond-

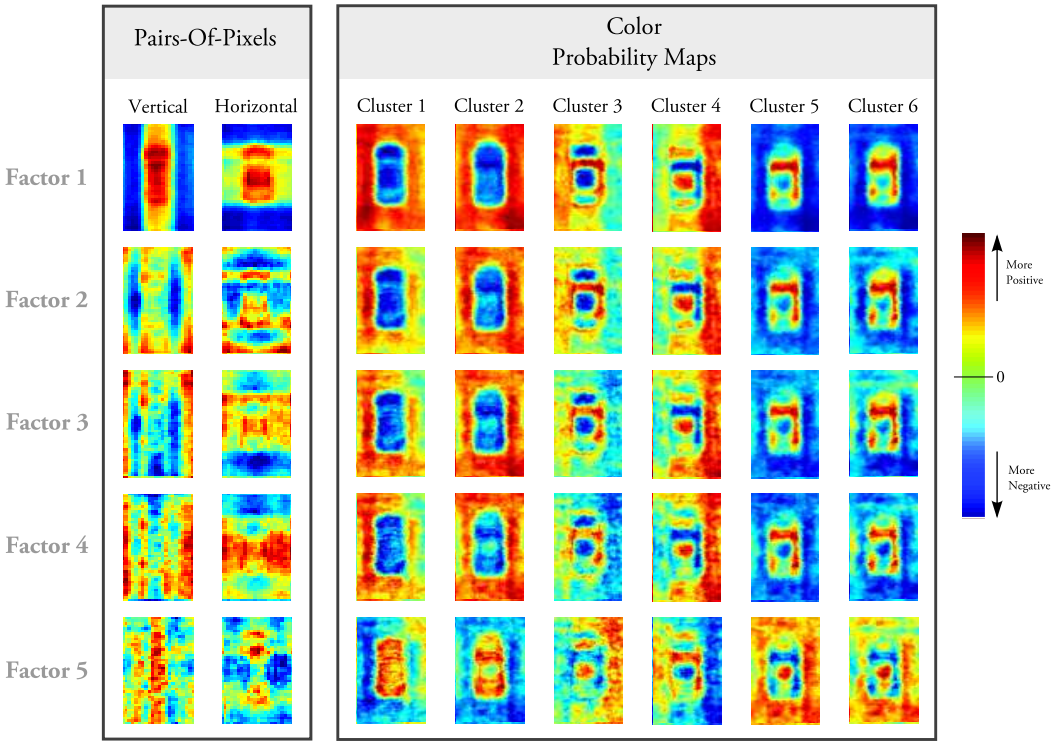


Figure 3.8: The first 5 PLS latent vectors. The components of the latent vectors corresponding to the Color Probability Maps and Pairs of Pixels feature classes are displayed. The images shown for the color probability maps directly correspond to the latent vectors. The images displayed for the PoP feature correspond to accumulator matrices. *(best viewed in color)*

ing to the color probability maps are directly displayed as shown in the figure, since each coefficient corresponds to a single pixel location in the image patch. But each coefficient of the latent vector from the PoP features corresponds to two pixel locations in the image (a pair of pixels), and thus cannot be displayed directly. Instead, the images displayed in the figure for the PoP feature class correspond to accumulator matrices. We initialize the accumulator matrix as a matrix of zeros and increment a location with the corresponding coefficient in the latent vector, when that location forms one half of the corresponding pixel pair. Furthermore, the images corresponding to the PoP feature class are split into the vertical and horizontal components. Each coefficient in the HOG feature vector corre-

sponds to a bin of a histogram and captures information about a neighborhood of pixels. These neighborhoods also have varying sizes (multiple block sizes), and thus cannot be easily visualized as the other two feature classes.

A very positive (dark red) or very negative (dark blue) color indicates a high degree of importance afforded to that pixel location. It is noticeable that the feature classes complement each other well, by extracting information from different parts of the image patch. The first (and most important) factor for the color probability maps shows that information extracted from the neighborhood of the object is given a high weight. This indicates that these features are able to capture the immediate context within which vehicles are typically observed. The PoP feature is seen to primarily focus on pixels that lie on top of the vehicle, in order to capture its regular structure.

3.5.3 Feature Selection: OPS

Figure 3.9 shows the result of the commonly used feature selection criterion using VIP. The VIP informative vector has been calculated using 5 PLS factors (which was the optimal number when using the entire set of features - refer to Figure 3.7). Using the $VIP > 1$ thumb rule reduces the number of features to 21,322 which represents about 30% of the total feature set. The performance does not drop significantly; instead a comparable performance is obtained using a smaller number of PLS factors. Results using varying values of the VIP cut-off score are shown.

Figure 3.10 shows the results of feature selection using the four informative vectors. The VIP and B informative vectors have been calculated using 5 PLS factors. The

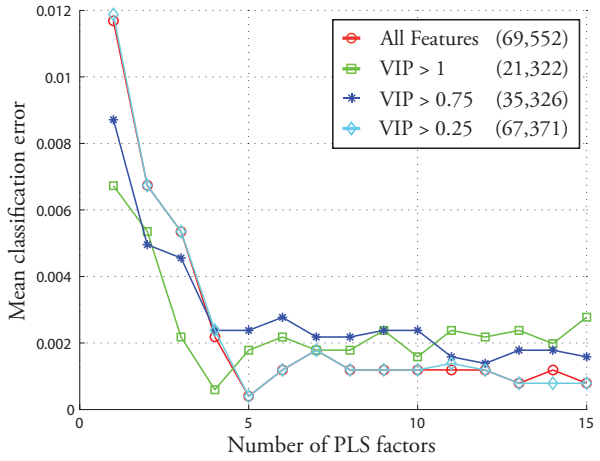


Figure 3.9: Basic feature selection using VIP. The error plot is shown for the original set of features and for a subset of features using the VIP criterion. VIP > 1 is the *thumb rule* usually applied in PLS analysis.

Correlation and Covariance Procedures Vector are calculated independent of the number of PLS factors. As a fair comparison, we use the same number of features with each informative vector. This number, 21,322, is the number of features retained using the $VIP > 1$ rule. The best performing informative vector is the regression vector B.

Figure 3.11 shows the results of feature selection using the B informative vector when the number of PLS factors (θ) used to compute this vector are varied. As was observed in [86], the optimum number of PLS factors used to calculate the informative vector (θ) is not the same as the optimum number of PLS factors used to build the model (ϕ), when all features are used (ϕ was determined to be 5 in our case). The performance is shown to increase with θ and it saturates beyond 11. The number of features selected was, once again, set to the number obtained by the $VIP > 1$ rule. The mean error after cross validation is seen to go down to 0 when B is calculated from 11 PLS factors and 21,322 features ($\approx 30\%$ of the total set) are retained.

Figure 3.12 shows the classification errors obtained using different informative vec-

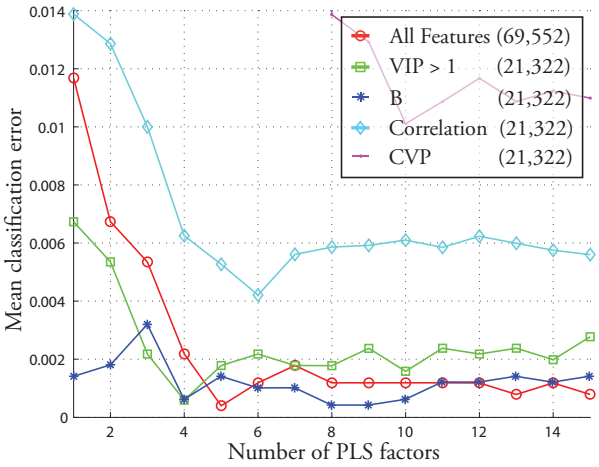


Figure 3.10: Basic feature selection using all four informative vectors. The regression vector, B, when used as an informative vector, outperforms the typically used VIP informative vector. The CVP vector performs very poorly on our dataset.

tors and their combinations. 8 matrices are displayed in the figure in a 2×4 grid. Each column in the grid of matrices corresponds to the results obtained for a particular informative vector. Each row in the grid of matrices corresponds to the results obtained for particular set of input features. We first consider only the first row in this grid of matrices, where all 69,552 features are provided as input to the feature selection module. We discuss the second row in Section 3.5.4. Each matrix in the 2×4 grid is color coded to represent the classification errors obtained by the cross validation procedure. Each matrix column represents a different value of θ. In general, as θ increases, the classification error reduces and then saturates. Each matrix row represents a different number of features retained after the feature selection process. As this number reduces, the error increases. Since the Correlation and CVP informative vectors do not perform very well, they are not displayed in this image.

B is the most consistent and best performing informative vector. It also performs

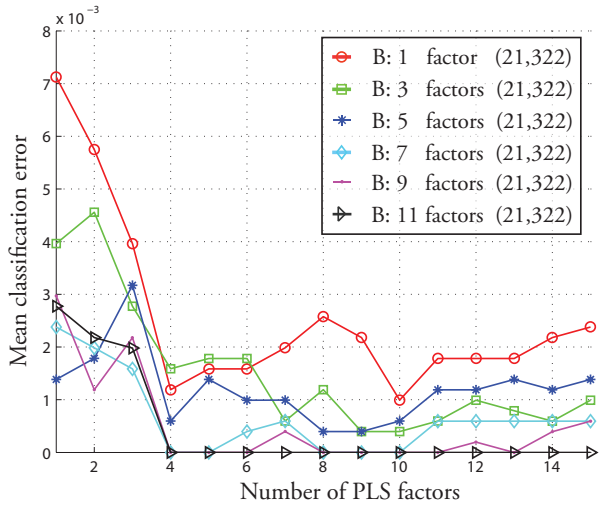


Figure 3.11: Feature selection using the B informative vector. The number of PLS factors used to calculate B (θ), is varied. The error is seen to reduce as θ is increased and it saturates beyond 11.

very well, when just 2000 features of the total 69,552 features are retained. This corresponds to less than 3% of the total number of features. The informative vectors are usually combined by simply multiplying their corresponding bins. The combination of VIP and B outperforms other combinations, and is the only one displayed in this figure. We also introduce a new scheme to combine the B and VIP informative vectors called *VIP-then-B*. First the $VIP > 1$ rule is employed to select a set of features. Then a new PLS model is built using these chosen features and the informative vector B is calculated. This smaller set of features is then ranked using B and a subset of these is finally selected. *VIP-then-B* is seen to perform very well, especially when a small sets of features are chosen. We finally choose the following feature selection scheme: *B* with $\theta = 9$ and 2000 features selected (over all three feature classes). Cross validation determined the number of PLS factors in the model (ϕ) to be 6. The cell corresponding to this parameter choice is marked with the white cross.

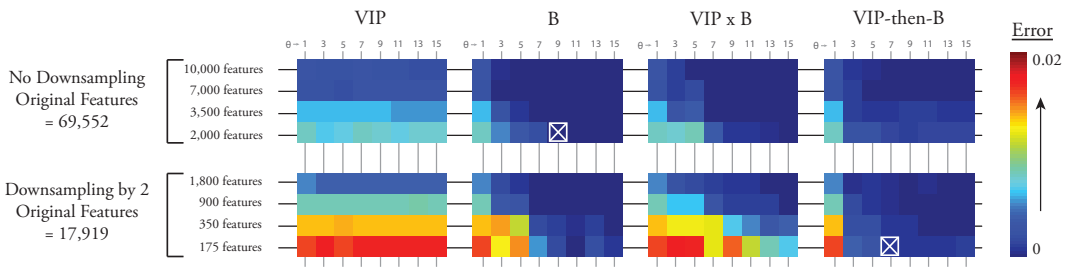


Figure 3.12: Results of the OPS feature selection approach on the original training set and the downsampled set. The errors represent the mean misclassification error after cross validation. (Refer to the text for more details).

3.5.4 Feature Selection: Downsampling

The original image patches have a size of 81×41 pixels. The total number of features computed for these original image patches equals 69,552. These image patches are downsampled by a factor of 2 (41×21 pixels). This reduces the total number of features to 17,919 (5,166 color, 9,288 HOG and 3,465 PoP features).

Figure 3.13 shows the result of downsampling without any feature selection. As is seen, performance does not degrade too much for a downsampling factor of 2. This reduced set of features is used in the first stage of our 2 stage PLS model. This enables us to efficiently reject a large number of background image patches and only pass on a small set of candidate vehicles to the next stage.

For each level of downsampling, a thorough OPS-PLS analysis is carried out using cross validation on the training dataset. These results are displayed in the second row of the 2×4 grid of matrices displayed in Figure 3.12. This determines the informative vector chosen for feature selection and the number of parameters associated with it. For a downsampling factor of 2, the VIP-then-B informative vector combination provides the best performance. Remarkably, the feature selection strategy retains only 175 features

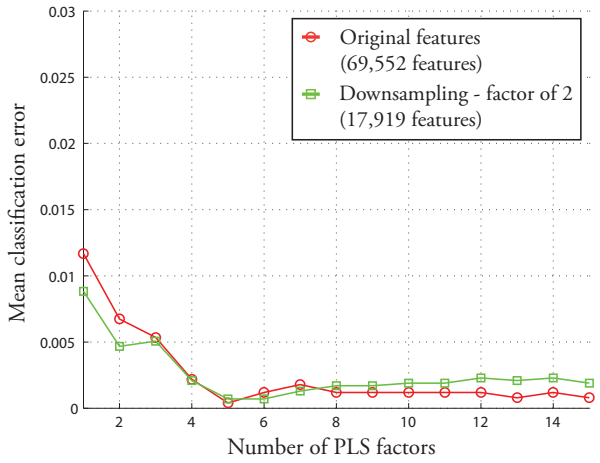


Figure 3.13: Mean classification error obtained using downsampling. No feature selection is used. The performance decreases very slightly as images are reduced to a smaller size.

in the fast first stage (almost 0.25% of the original number of features 69,552). This enables a fast rejection strategy and improves the performance of the system. Figure 3.14 shows the performance at a downsampling factor of 2 when the popular $VIP > 1$ feature selection strategy is used, as well as when the best feature selection criteria is used.

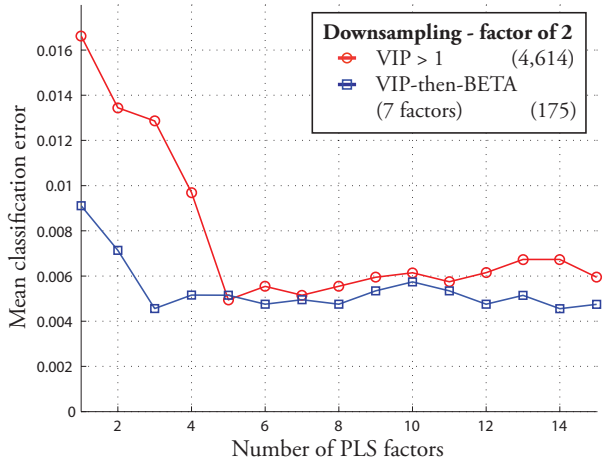


Figure 3.14: Mean misclassification error at downsampling factor of 2. The error rates drop after the feature selection process. The informative vector combination VIP-then-B outperforms $VIP > 1$.

Thus we obtain a 2 stage PLS model, whose parameters are given in Table 3.1.

Table 3.1: The 3 stage vehicle detector

	STAGE 2	STAGE 3
Image size	41×21 pixels	81×41 pixels
Original features	17,919	69,552
Feature selection	VIP-then-B ($\theta = 7, \phi = 7$)	B ($\theta = 9, \phi = 6$)
Retained features	175	2,000

3.5.5 Performance: Google Earth San Francisco dataset

We test the performance of our system on the test images of the Google Earth San Francisco dataset. All test images are fully ground truthed to show the presence of vehicles along with their orientation. Vehicle detections that overlap the ground truth locations by an area equal to 33% of the size of the bounding box, are considered true detections. As per the evaluation criteria commonly used in the PASCAL VOC challenge [24], if multiple detections overlap with a ground truthed location, only one of them is considered a true positive detection. The remaining detections are considered to be false alarms. Since the vehicles in the entire dataset are roughly the same size, we only scan the images at a single scale. If the size of the vehicles was unknown, the images would have to be scanned at multiple scales. Since the orientation of the vehicles is unknown, the image must be scanned at multiple rotations. In order to decrease the number of image patches that must be scanned, we employ a coarse to fine rotation strategy. First the image is rotated in increments of 30° and scanned completely. Candidate windows are selected and are then finely rotated in increments of 5° in a neighborhood of 25° from the original detection[2]. In an urban setting, roads typically form a rectangular grid. Determining this grid orientation can help to initialize the scan angles and further speed up the system.

We compare our system to a number of other approaches. The first approach is a traditional object detection approach[3]. SIFT features are calculated on a dense grid of points in the training images. The SIFT descriptors from the training set undergo vec-

[2]The identical scanning stratgy is used for the proposed vehicle detector and the approaches we compare to.

[3]Code obtained from http://www.cs.unc.edu/ lazebnik/

tor quantization to form visual words. An image patch can then be described using a histogram of the visual words present in the patch. In order to enforce some spatial constraints on the location of these features, we use Spatial Pyramidal Matching [52]. This involves repeatedly subdividing the image and computing histograms of SIFT features at increasingly finer resolutions. The distance measure used is histogram intersection and the classifier used is the Support Vector Machine.

The second approach we compare to, is the vehicle detector proposed by Moon et al. [61]. They derive an optimal one dimensional step edge detector to be the derivative of the double exponential (DODE) function. This is extended along the shape's boundary contour to obtain the shape detector[4]. This results in detecting shapes in the image that resemble parallelograms. Their vehicle detector does not require a training phase.

The third approach we compare to is the popularly used HOG based approach used to detect objects, proposed by Dalal et al. [20]. HOG features are calculated for a large number of blocks within an image window and concatenated together to form the feature vector. Blocks of only a single size are used in this approach. The feature vectors thus obtained are used to train a linear Support Vector Machine (SVM). We used libLinear [25][5], an efficient linear SVM, for this purpose. Dalal et al. note in their work that a kernel SVM can provide improved performance at the cost of a significant reduction in the efficiency of the system. This is because the standard approach to evaluate the kernel for a test vector involves a comparison with each of the support vectors.

Recently, Maji et al. [58] proposed a method to significantly improve the efficiency

[4]Code obtained from the authors

[5]Package available at http://www.csie.ntu.edu.tw/ cjlin/liblinear/

of kernel SVMs for a class of kernels such as the histogram intersection kernel and the chi squared kernel. Their approximate SVM has constant runtime and space requirements, independent of the number of support vectors, as opposed to the typical linear dependency. Furthermore, the loss in classification accuracy using this approximation is negligible. As a fourth comparison, we applied this improved kernel SVM[6] known as the approximate intersection kernel support vector machine (approx IKSVM) to the HOG features proposed by Dalal and Triggs. While [58] demonstrated that the IKSVM evaluates nearly as fast as a linear classifier, it did not address the problem of efficiently training such a classifier. Subsequently Maji et al. [59] proposed very efficient training algorithms for additive classifiers in a max-margin framework.

Figure 3.15 shows the Precision-Recall curves for the vehicle detectors. The dataset we test on, contains images in an urban setting. The presence of a large number of rectilinear structures in such images leads to false alarms with all approaches. Objects present on top of buildings, such as air conditioning units are seen to cause errors. The DODE approach is able to correctly find many vehicles but also produces a very large number of false alarms on rectangular structures. The HOG features when applied to a linear SVM outperform the SIFT based approach. The use of the histogram intersection kernel greatly improves performance over the linear kernel. Our proposed solution outperforms all the other approaches.

Figure 3.16 compares our proposed approach to directly applying an SVM to our proposed set of 69,552 features. The PLS based approach comfortably outperforms the approach using a linear SVM. We used the LibLinear package for this purpose. Using

[6]Code obtained from http://www.cs.berkeley.edu/ smaji/projects/fiksvm/

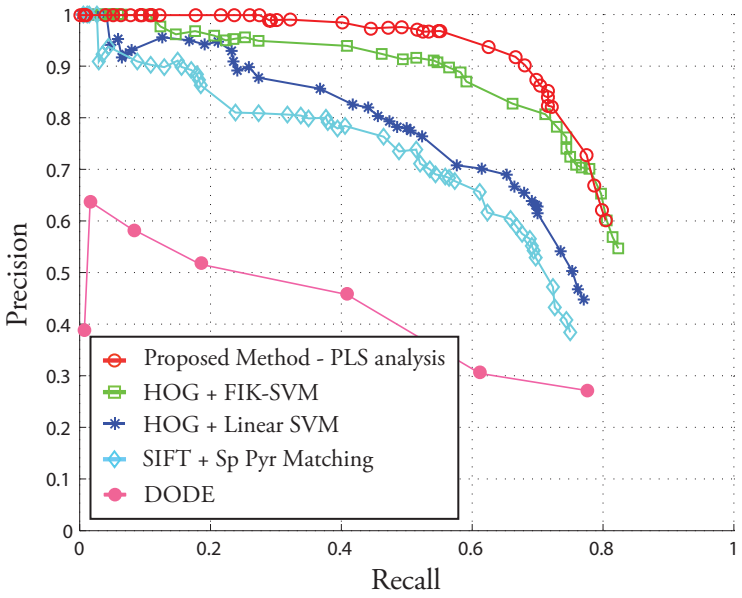


Figure 3.15: Performance of the five vehicle detectors on the Google Earth San Francisco dataset. Our vehicle detector outperforms the other ones.

the approximate and fast IKSVM method improves results over using a linear kernel. However, applying the IKSVM to the original 69,552 features is quite time-consuming. Table 3.2 compares the speeds of the 3 classification approaches shown in Figure 3.16. Note that the numbers provided in Table 3.2 account for the classification time only, and not the time required to calculate the features. Projection onto the subspace followed by classification by a quadratic classifier is very fast compared to the other two approaches. Applying the fast approximate intersection kernel is much slower than the other two methods.

Overall, our two stage vehicle detection system is able to process approximately 5600 detection windows per second. Our system is implemented in MATLAB and all our experiments are run on a an Intel Xeon 2.8 GHz processor. While the machine we use has multiple processing cores, our program currently makes use of only a single core. The

Table 3.2: Comparison of classification speeds (windows/second) while using 69,552 features

	PLS method	LibLinear	Fast IKSVM
Detection speeds (win/sec)	8565	308	95

number of detection windows processed per second can be improved by taking advantage of multi-core architectures. All three stages of our detection system - feature calculation, projection onto a subspace as well as classification - can be parallelized to yield a faster detection system.

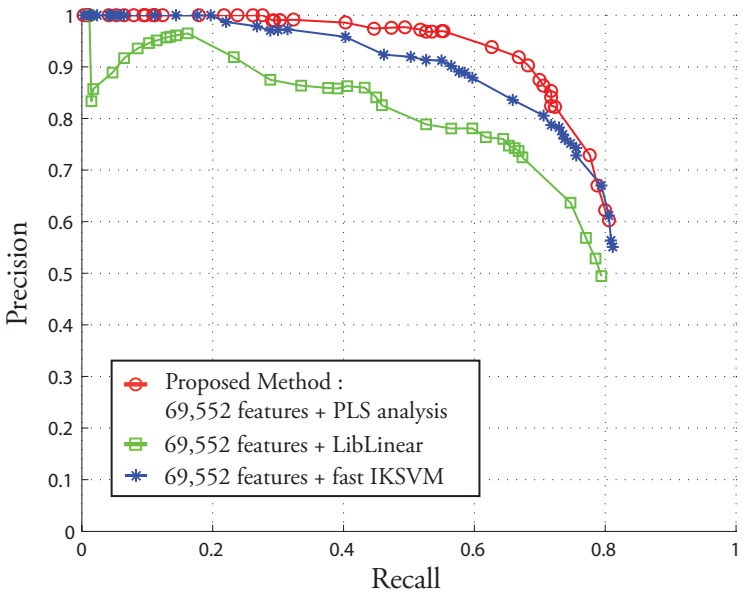


Figure 3.16: Performance of the proposed vehicle detector compared to detectors composed of the proposed features and SVMs as classifiers.

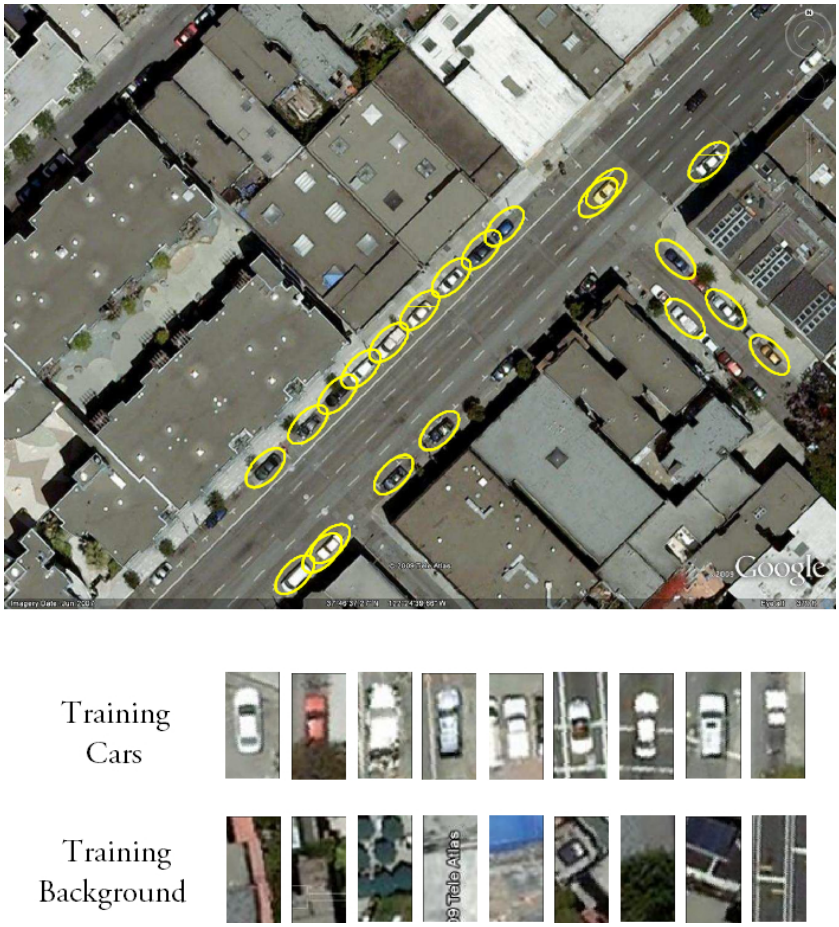


Figure 3.17: Sample vehicle detection results from the Google Earth San Francisco dataset.

3.5.6 Speedup using Feature Selection

Figure 3.18 compares the performance of the 2 stage vehicle detector with a detector when only the high resolution stage (Stage 2) is used. The Precision-Recall curves of the two detectors are quite comparable. This enables us to obtain the speedup given by the 2 stage approach, without a significant loss of performance.

Table 3.3 shows the percentage of windows that are processed in each stage, over the entire Google Earth San Francisco dataset. All windows are processed by the fast Stage 1, but only a very small number are passed on to the second stage. Note that the image

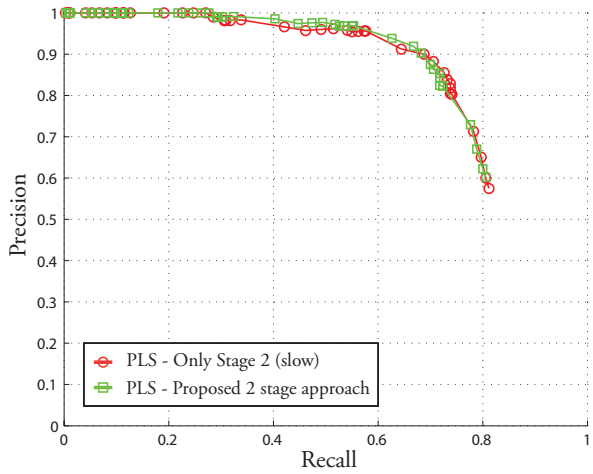


Figure 3.18: Performance of the 2 stage vehicle detector compared to the performance of the detector when only the the high resolution stage (Stage 2) is used.

Table 3.3: Percentage of windows that are processed by each stage of the 2 stage system

	STAGE 1	STAGE 2
Number of Features	175	2,000
Percentage Windows Processed	100%	0.48%

patches that obtain a high detection probability in stage 3, undergo further processing by rotating them at finer angular intervals. This processing is done at the full resolution.

Figure 3.17 shows some sample vehicle detection results from the Google Earth San Francisco dataset. As can be seen, false alarms are typically caused by rectangular structures on top of buildings such as air-conditioning units.

3.5.7 Overhead Imagery Research Data Set

We also test the performance of our system on the publicly available Overhead Imagery Research Data Set (OIRDS)[7]. The OIRDS is a large collection of almost 900 overhead images, captured using aircraft mounted cameras. The total number of vehicles annotated in the dataset is around 1800.

The OIRDS dataset is a very challenging dataset. The images in this set have varying levels of zoom and a wide degree of difficulty. The images have been captured primarily in suburban settings. The presence of a large number of trees in these images often causes vehicles to be partially occluded. We divide this dataset into 3 parts (with roughly 300 images in each part). We label these parts OIRDS 1, OIRDS 2 and OIRDS 3. OIRDS 1 contains images that have the best picture quality and vehicles that are clearly visible. These images are similar in quality to the images in the Google Earth San Francisco dataset. OIRDS 2 contains images that are of a poorer quality and the vehicles are also harder to find. Some of these vehicles are partially occluded. OIRDS 3 contains images in which vehicles are very difficult to find. Many of these images have vehicles that were almost fully occluded. Figure 3.19 shows sample images from these three sets.

We compare the performance of the five vehicle detectors on OIRDS 1 and OIRDS 2. No new training was carried out for this dataset. The detectors trained on the Google Earth San Francisco training dataset were directly used. Figures 3.20 and 3.20 shows the Precision-Recall curves for all three detectors. We outperform all detectors on OIRDS1 and obtain a comparable performance to the HOG + kernel SVM approach on OIRDS 2.

[7]Downloaded from http://sourceforge.net/apps/mediawiki/oirds

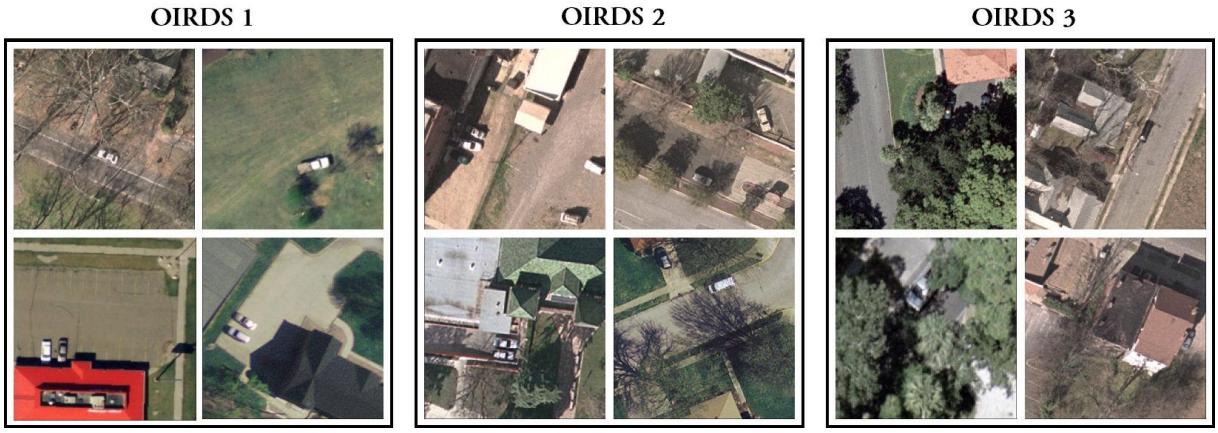


Figure 3.19: Sample images taken from the OIRDS dataset. We divide this dataset into three parts based on the degree of difficulty.

As expected, the performance of all the detectors drops for OIRDS 2.

Finally, Figure 3.22 shows the performance of our vehicle detector on a large panoramic image (5007×7776 pixels). This was obtained by stitching a large number of images obtained from Google Earth. The image overlooks the parking lot and adjacent areas of the San Francisco Giants stadium in San Francisco city. Our vehicle detector is able to accurately locate a large number of vehicles in this image.

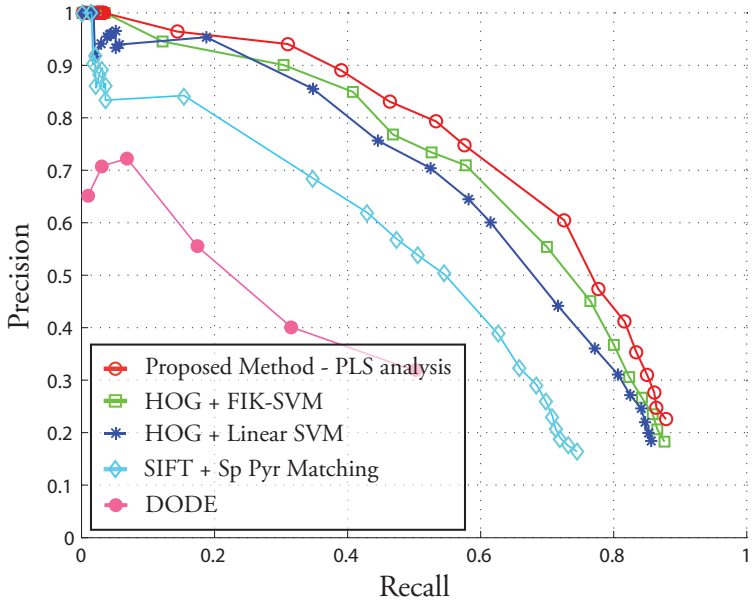


Figure 3.20: Performance of the three vehicle detectors on the OIRDS 1 dataset.

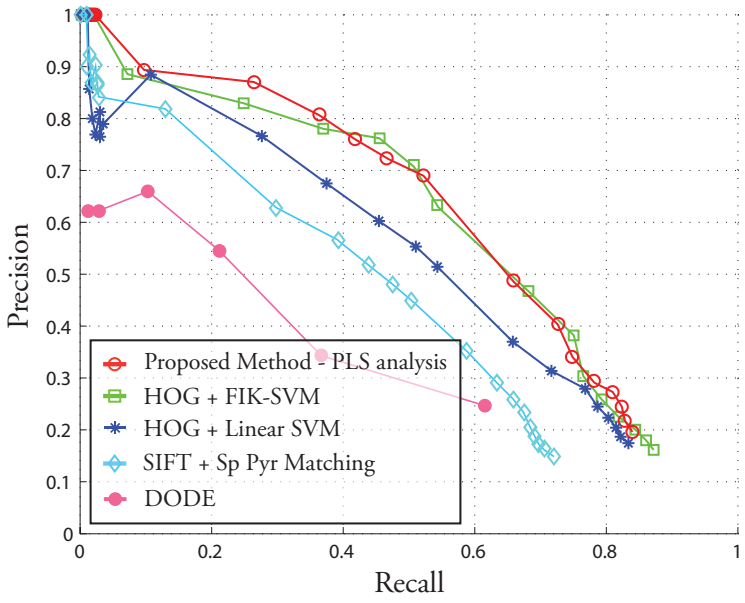


Figure 3.21: Performance of the three vehicle detectors on the OIRDS 2 dataset.

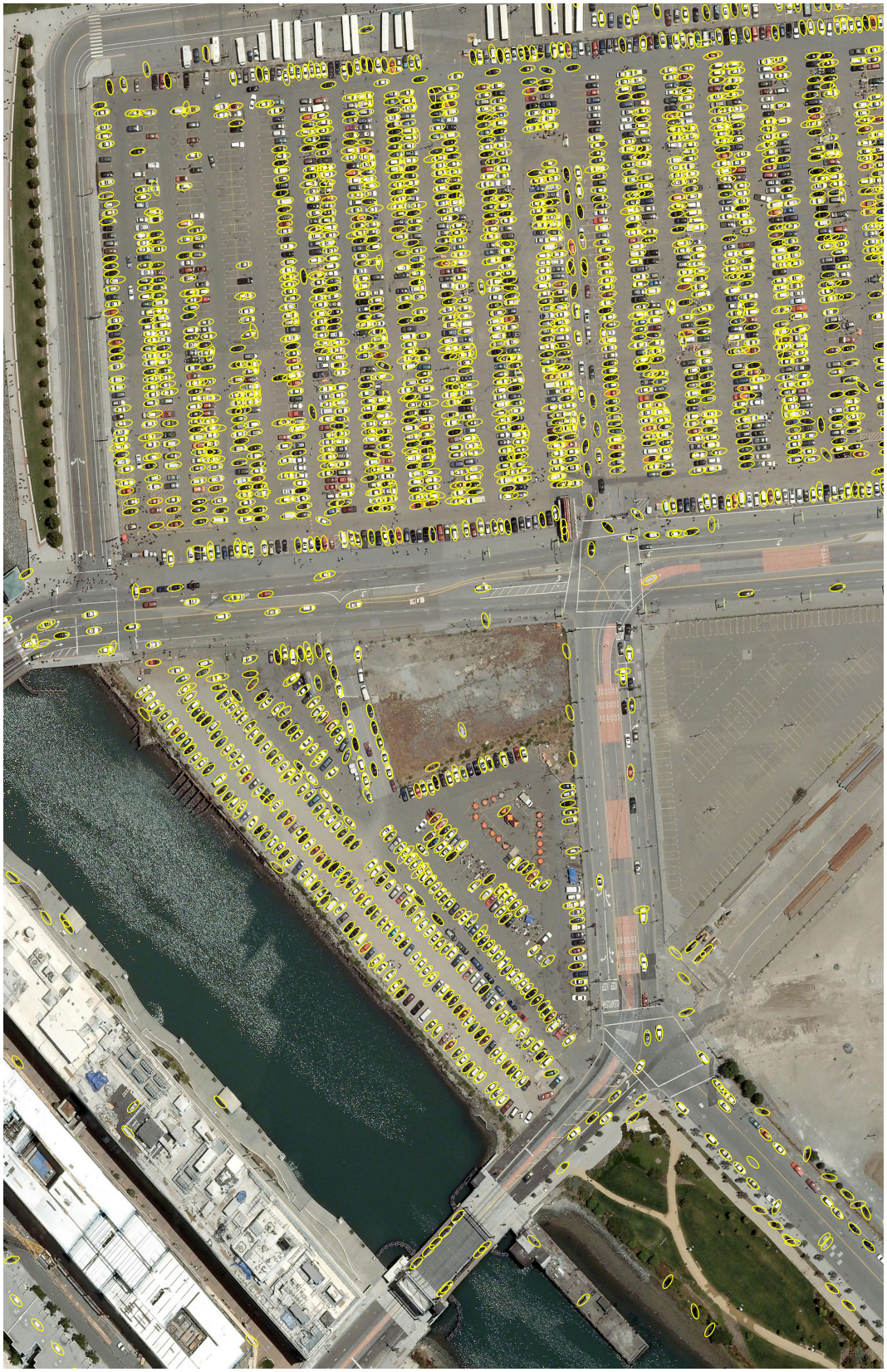


Figure 3.22: Performance of our vehicle detector on a large panoramic image overlooking the parking lot of the San Francisco Giants stadium. © 2009 Google

Chapter 4

Detecting Vehicles using Incremental Multiple Kernel Learning

4.1 Introduction

Early works on object recognition used global features such as color or texture histograms [69]. However these features were not robust to view-point changes, clutter and occlusion. Over the years, more sophisticated approaches such as *part-based* [28] and *bag-of-features* [79] methods have become more popular.

Increased interest in object recognition has resulted in new feature descriptors and a multitude of classifiers. Inspired by the pyramidal feature matching approach of [53], Bosch et al. proposed two new region descriptors - the Pyramidal Histogram of Oriented Gradients (PHOG) and Pyramidal Histogram of Visual Words (PHOW) [9]. These features were then used with Random Forests as a multi-way classifier [8]. Zhang et al. used the Geometric Blur (GB) feature [5] and proposed using a discriminative nearest neighbor classification for object recognition [101]. Wu et al. [95] used edgelet features to capture the local shape of objects and were able to simultaneously detect and segment objects of a known category.

Zhang et al. [102] combined multiple descriptors and obtained improved results for texture classification and object recognition. They provided equal weights to each descriptor. Similarly, Bosch et al. [8] linearly combined the PHOG and PHOW descriptors to obtain improved performance. The linear combination weights were, however, ob-

tained by a brute force search using a validation dataset. Since the number of features was small, their search space had few dimensions, thus making the brute force computationally feasible. Wu et al. [96] combined multiple heterogeneous features for object detection by using cascade structured detectors in a boosting framework. Features were combined using their classification powers and computational cost.

Lanckriet et al. [51] introduced the MKL procedure to learn a set of linear combination weights, while using multiple sources of information with a kernel method, such as an SVM. Their problem formulation, however, resulted in a convex but non-smooth minimization problem. Bach et al. [3] considered a smoothed version of the problem. Their Sequential Minimal Optimization (SMO) algorithm was significantly more efficient than the previous formulation in [51]. Sonnenburg et al. [80] reformulated the problem as a semi-infinite linear program and solved it efficiently by recycling the standard fast SVM implementations. Their algorithm worked for hundreds of thousands of examples or hundreds of kernels. Rakotomamonjy et al. [71] formulated the problem using a 2-norm regularization formulation to a smooth and convex optimization problem. Their method provided the additional advantage of encouraging sparse kernel combinations. Varma et al. [89] combined multiple features using MKL and showed a considerable increase in the performance of their visual classifier.

A number of unsupervised, online learning algorithms have been used for computer vision applications. Li et al. [55] used a non-parametric graphical model in an incremental approach for automatic dataset collection from the Internet (OPTIMOL). Their iterative framework simultaneously learns object category models and collects object category datasets. We compare our IMKL method with OPTIMOL in Section 4.4. Boosting tech-

niques for incremental learning have also been popular. Javed et al. [44] used *co-training* to label incoming data and used it to update a boosted classifier. Co-training [7] is a method for training a pair of learners, given that the two algorithms use different *views* of the data. The two classifiers are used to provide additional informative labeled examples to one another, which improves the overall performance. Wu et al. [94] extended the online boosting algorithm and proposed an online framework for cascade structured detectors. An automatic labeler called the *oracle*, with a high precision rate, provided samples to update the online object detector. In order to prevent the boosting algorithm from overfitting noisy data (provided by the *oracle*), they employed two noise resistant strategies from variants of the Adaboost algorithm designed to be robust to outliers. Our initial object classifier, built from a generic training dataset, is tuned similar to this *oracle*. Our work builds on MKL and fits well into the SVM framework. It also provides the useful property of being able to adapt kernel weights over time in addition to updating the training database.

4.2 An Incremental Solution

4.2.1 The Multiple Kernel Learning Problem

Kernel based learning methods have proven to be an extremely effective discriminative approach to classification as well as regression problems. Given multiple sources of information, one might calculate multiple basis kernels, one for each source. In such cases, the resultant kernel is often computed as a convex combination of the basis kernels,

$$\Phi(x_i, x_j) = \sum_{k=1}^{K} d_k \Phi_k(x_i, x_j), \quad \sum_{k=1}^{K} d_k = 1 \quad , d_k \geq 0 \tag{4.1}$$

where x_i are the data points, $\Phi_k(x_i, x_j)$ is the k^{th} kernel and d_k are the weights given to each information source (kernel). Learning the classifier model parameters and the kernel combination weights in a single optimization problem is known as the Multiple Kernel Learning problem [51]. There have been a number of formulations for the MKL problem, as noted in Section 4.1. Our incremental approach builds on the MKL formulation of [71], known as SimpleMKL. This formulation enables the kernel combination weights to be learnt within the SVM framework. The optimization equation is given by,

$$\begin{aligned} \min \quad & \sum_k \frac{1}{d_k} w_k w_k^T + C \sum_i \xi_i \\ \text{such that} \quad & y_i \sum_k \phi_k(x_i) + y_i b \geq 1 - \xi_i \ \ \forall i \\ & \xi_i \geq 0 \ \ \forall i, \ \ d_k \geq 0 \, \forall k, \ \ \sum_k d_k = 1 \end{aligned} \tag{4.2}$$

where b is the bias, ξ_i is the slack afforded to each data point and C is the regularization parameter. The solution to the above MKL formulation is based on a gradient descent on the SVM objective value. An iterative method alternates between determining the SVM model parameters using a standard SVM solver and determining the kernel combination weights using a projected gradient descent method.

4.2.2 Karush-Kuhn-Tucker Conditions

The support vectors returned by the training algorithm of an SVM generally represent a small fraction of all the training examples, but are able to summarize the decision boundary between the classes very well. Thus, one way to increment an SVM is to retain only the support vectors, to reduce the computational load required at every successive training step [84]. The same approach could be used for the MKL problem. However,

this gives only approximate results.

The first exact online approach to train SVMs was proposed by Cauwenberghs et al. [15]. New data points are presented to the SVM one at a time. The new data point is added to the solution while ensuring that the Karush-Kuhn-Tucker (KKT) conditions are retained on all the previous data points. Our proposed approach to IMKL is inspired by this work.

The key idea behind the Incremental SVM is that the SVM optimization problem is convex. Thus, the KKT conditions are not only *necessary* but also *sufficient*. Thus, maintaining the KKT conditions on all old points, as well as the new point, indicates that a new solution has been obtained. The optimization problem given by the SimpleMKL framework in Equation 4.2 is also convex, making it suitable for our purposes.

The KKT conditions for our problem are derived from the Lagrangian function corresponding to Equation 4.2,

$$L = \frac{1}{2}\sum_{k}\frac{w_k w_k}{d_k} + C\sum_{i}\xi_i - \sum_{i}\nu_i\xi_i - \mu_k d_k - \sum_{i}\alpha_i(y_i w_k \phi_k(x_i) + y_i b - 1 + \xi_i) - \lambda(\sum_{k} d_k - 1) \tag{4.3}$$

where α_i is the Lagrange multiplier corresponding to the first constraint in Equation 4.2, ν_i and μ_k are the Lagrange multipliers associated with the non-negativity constraints on ξ_i and d_k respectively, while λ corresponds to the Lagrange multiplier of the l_1-norm equality constraint on d.

The optimal solution of the multiple kernel system in Equation 4.2 occurs at the saddle point of Equation 4.3. The saddle point is obtained by differentiating the Lagrangian equation with respect to the primal variables (w_k, d_k, ξ_i, b) and the dual variables (α_i, ν_i, μ_k). A small amount of algebraic manipulations yields the KKT conditions given

below,

$$g_i = \sum_j \sum_k d_k \alpha_j Q_{ij}^k + y_i b - 1 = 0, \quad \sum_i \alpha_i y_i = 0$$
$$\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j Q_{ij}^k + \mu_k - \lambda = 0, \quad \mu_k d_k = 0, \quad \sum_k d_k = 1 \tag{4.4}$$

where $Q_{ij}^k = y_i \Phi_k(x_i, x_j) y_j$.

Note that $g_i = y_i f(x_i) - 1$, where $f(x_i)$ is the solution of the multiple kernel SVM given by,

$$f(x_{new}) = \sum_j \sum_k d_k \alpha_j y_j \Phi_k(x_j, x_{new}) + b \tag{4.5}$$

4.2.3 Algorithm

Consider a set of data instances $(x_1, x_2, \ldots, x_n)$ with corresponding class labels $(y_1, y_2, \ldots, y_n)$. Let $\Phi_k(x_i, x_j)$ be the set of K kernels. The MKL solution for the given data is obtained by SimpleMKL and it thus satisfies the KKT conditions in Equation 4.4. The data points are divided into three disjoint sets based on their Lagrange multipliers ($\alpha_i's$): set L containing the set of points lying on the correct side of the margin vectors ($\alpha_i = 0$), set S containing the support vectors ($0 < \alpha_i < C$) and set E containing the points lying on the wrong side of the margins ($\alpha_i = C$). We also divide the kernels into two sets: set D_+ containing kernels with positive weights and set D_0 with kernels having zero weight. These sets are illustrated in Figure 4.1.

When a new point x_q is added to the solution, we need to calculate its Lagrange multiplier α_q ($0 \leq \alpha_q \leq C$) such that the KKT conditions are satisfied once again. We begin with a value $\alpha_q = 0$ and keep increasing it until we reach the updated solution.

Every time we increment α_q, the remaining Lagrangian multipliers, the kernel weights and the bias must be changed to maintain the constraints in Equation 4.4. These changes are given by the differential form of the constraints,

$$\begin{aligned}
\sum_j \alpha_j \sum_k \Delta d_k Q_{ij}^k + \sum_j \Delta\alpha_j \sum_k Q_{ij}^k \\
+ \sum_j \sum_k \Delta d_k \Delta\alpha_j Q_{ij}^k + y_i \Delta b = 0, \ \forall i \in S, \forall j \in \{S, E, L, q\} \\
\sum_i \sum_j \Delta\alpha_i \alpha_j Q_{ij}^k + \frac{1}{2} \sum_i \sum_j \Delta\alpha_i \Delta\alpha_j Q_{ij}^k \\
+\Delta\mu_k - \Delta\lambda = 0, \qquad \forall k \in K \\
\sum_i \alpha_i y_i = 0 \forall i \in \{S, E, L, q\}, \quad \sum_k \Delta d_k = 0 \\
\Delta\mu_k d_k + \mu_k \Delta d_k + \Delta\mu_k \Delta d_k = 0, \qquad \forall k \in K
\end{aligned} \tag{4.6}$$

For a given step size $\Delta\alpha_q$, Equation 4.6 is a set of (num_S+2K+2) equations in (num_S+2K+2) unknowns. Here, num_S is the number of points in set S and K is the number of kernels. The unknown variables are: $\{\Delta\alpha_1 \ \ldots, \Delta\alpha_{num_S}, \Delta d_1, \ldots, \Delta d_K, \Delta\mu_1, \ldots, \Delta\mu_K, \Delta b, \Delta\lambda\}$. These non-linear equations can be solved using a standard non-linear equation solving package. Since an addition of a new point may not alter the system significantly, a good initial solution for all the unknowns in Equation 4.6 is 0.

The above differential equations only hold when $\Delta\alpha_q$ is small enough to ensure that there is no change in set membership for either the points or the kernels. Thus, when set membership changes, the differential equations are updated and the process is repeated. The conditions for a change in the set membership are described in Figure 4.1.

The algorithm is terminated when any of the following conditions occur.

- $g_q > 0$ at $\alpha_q = 0$: x_q is a correctly classified point. Added to set L.

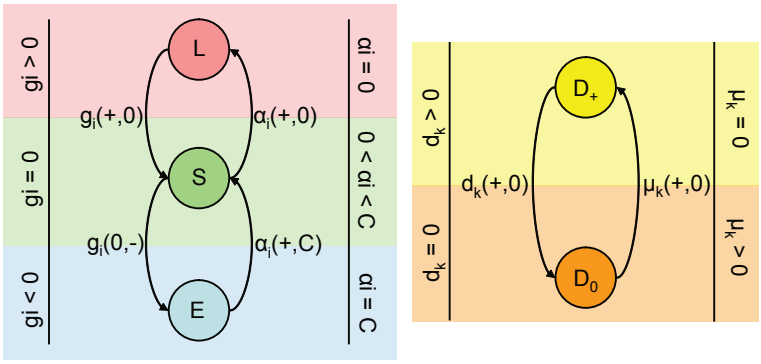


Figure 4.1: Categorization of the data points and kernels. The image on the left shows the values of the Lagrange multipliers ($\alpha's$) and the output of the system ($g's$) for each of the sets: L, S and E. It also shows the conditions that are checked to detect a set transition. (Notation: $g_i(+,0)$ denotes the value of g_i changing from a positive value to 0.) The image on the right shows the two kernel sets, the corresponding values of the weights ($d's$) and their Lagrange multipliers ($\mu's$) and the set change conditions.

- $g_q = 0$ before $\alpha_q = C$: x_q is a support vector. Added to set S.
- $\alpha_q = C$ and $g_q < 0$: x_q is on the wrong side of the margin. Added to set E.

A similar procedure can be used for removing data points from the classifier (decremental unlearning).

The number of computations required by the IMKL algorithm depends on the computations to solve the non-linear system and the number of steps taken to reach the final value of $\Delta\alpha_q$. In our experiments, we have observed that setting the initial solution of the non-linear solver to a zero vector, reduces the computational cost significantly. The number of steps taken to reach the final solution is lower bounded by the number of set changes that are required to arrive at the final solution. We use a large step size at every time instant and backtrack our solution if we observe a set change for the given step size. The IMKL algorithm can also be sped up by ignoring the higher order terms in Equation 4.6 to obtain linear equations. However this provides only an approximate solution.

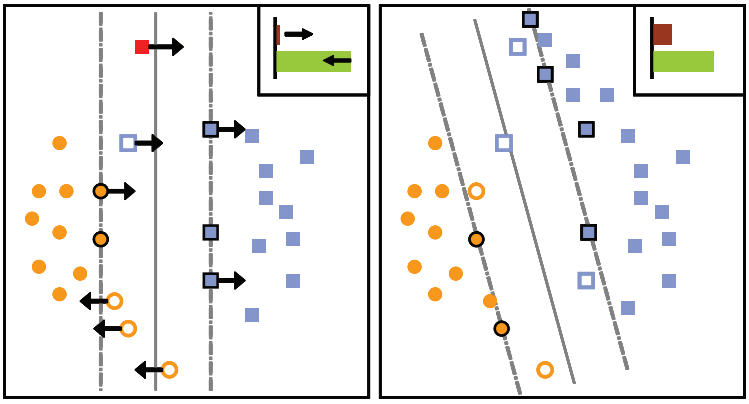


Figure 4.2: A 2-class classification example. Points in class 1 are shown in orange and points in class 2 are shown in blue. Points in set S are marked with a black border. Points in set L are solid colored while points in set E are not filled with color. Kernel 1 (weight shown by the brown bar) captures the similarity between the y-coordinates of the points, while Kernel 2 (green bar) captures the similarity between the x-coordinates. The left figure shows the effect of adding a new point (shown in red) on the original points and the weights. A change in set membership is observed for some points. The figure on the right shows the final classifier after adding 7 new points close to the first new point.

Consider the two class classification problem shown in Figure 4.2. A new point q, marked in red, is added to the system, and it initially gets misclassified. As the Lagrange multiplier α_q is incremented upwards from a value of 0, the distance between the new point and the margin reduces, while some of the other points change set membership. At the same time, the kernel combination weights also change.

4.3 Object Recognition Framework

A training database, representative of the expected test points, is an essential component of any classification system. In a practical object recognition framework, a good training database is one that contains images of the expected objects in their more likely poses and illumination conditions. It must also contain a representative set of images in

the negative set, which, in an object recognition framework, is usually the background. Obtaining such a set of good training examples can often be a tedious process. On the other hand, it is easier to obtain a generic training dataset of images of the expected object classes. Our object detector is initialized on a generic training dataset and tunes itself towards the objects and background in the scene.

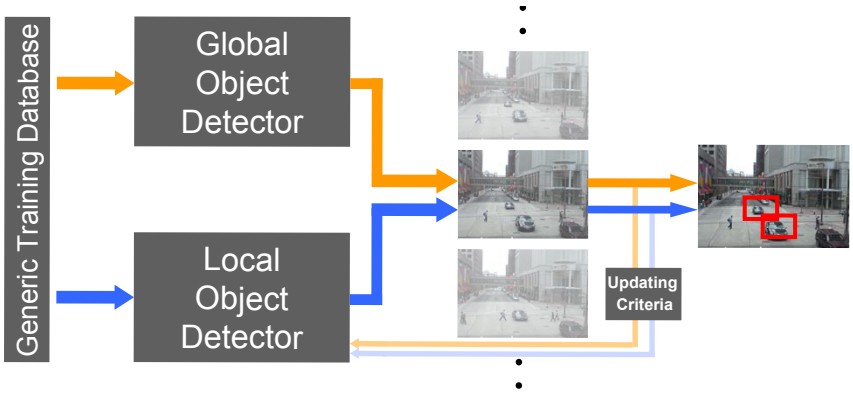


Figure 4.3: Object recognition framework.

Figure 4.3 provides an overview of our visual categorization framework. Training images from a generic training dataset are used to train an initial object detector which we call the *global* detector. The *global* detector is not updated at any time and serves as a generic object classifier. The generic training dataset is also used to train a *local* object detector, which runs in an online mode throughout the duration of analysis. Incoming images from a video stream are scanned using overlapping windows and each window is classified into one of the classes by both the detectors. The classification results returned by the global detector keep updating the training image sets of the local object detector.

The updating criterion differs for the foreground classes (buses and cars) and the background class. The image windows that are classified by the *global* detector as be-

longing to one of the foreground classes are thresholded so as to retain only very high confidence detections. Such windows are considered reliable detections and used to update the foreground training sets of the *local* object detector. Since the purpose of the local object detector is to train on typically observed appearances and poses, updating it with high confidence samples works well. The high precision of the *global* detector comes at the cost of a lower recall. Updating the *local* detector with false positives can lead to a significant drop in the performance of the system, and the probability threshold is set sufficiently high to minimize this.

On the other hand, for the background class, such an updating criterion leads to the addition of a large number of image patches from a single portion of the scene. This is because background patches with very similar appearances repeat over several frames. Thus if a patch gets classified with a very high probability of belonging to the negative set, several similar images also get added to the *local* training set. Ideally, one would like the entire scene to be well represented in the background class of the local detector. Thus, we first threshold image windows classified by the *global* detector as belonging to the background class. Then, for every image patch passing this initial criterion, we evaluate its positional entropy with respect to the distribution of the positions of all image patches currently in the *local* background training set. This is given by,

$$H(I) = -\sum_{w \in \{BG_{local}\}} p(w_{(x,y)}|I_{(x,y)}) \log p(w_{(x,y)}|I_{(x,y)}) \tag{4.7}$$

where w represents an image patch in the current background set, I represents the new image patch and (x, y) represent the co-ordinates of an image patch in the scene. Image patches passing the initial background threshold, as well as having a high entropy

with respect to the current local training set, form good candidates to improve the diversity of the *local* background set and are added to it. Over time, the object classes get updated with images of objects in their typical observed appearances and poses and the background class gets updated with image patches from different parts of the scene. Figure 4.4 demonstrates the image patches in the *local* background set which has been updated using both criteria. Using the entropy criteria in addition to a probability threshold, samples the entire scene well. Li et al. [55] used a similar criteria to update their dataset. While their entropy is calculated in the feature space, our measure is calculated in the image co-ordinate space.

The *local* detector fits itself towards image patches observed in the recent past, improving its performance. However, it also has the tendency of misclassifying objects that are atypical in the scene, due to overfitting on the observed data. The more generically trained *global* detector helps classify such atypical objects. The outputs of both detectors are combined to obtain the final detections. The resultant object detections are used to update the *local* detector.

In order to fit the local detector towards a dynamically changing scene, it is also important to discard image patches from the local training dataset. For every image patch added to the local set, we retain a timestamp indicating the frame it was obtained from. We use this to discard training samples based on the length of their stay in the training set. Thus the classifier adapts itself towards changing illumination conditions, particularly when day transitions to night.

Our IMKL algorithm described in Section 4.2 is used to update the *Local* classifier with new training images. This also results in an update of the kernel combination weights

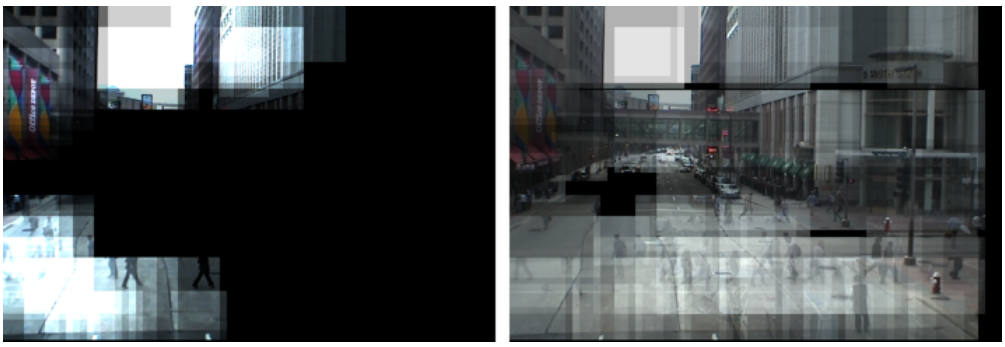


Figure 4.4: Representation of the *local* negative training set using two sampling methods to update the training set. For this display, all image patches in the set are added together at the appropriate locations in the scene. Thus brighter regions corresponds to more patches in that portion of the scene, black regions indicate that no image patches represent that portion of the scene. (Left) High probability criteria - Only certain portions of the scene are represented. (Right) High probability + high entropy criteria - Most portions of the scene are represented equally.

based on the training data. We use multiple 1-Vs-All classifiers for our purpose of multi-class classification. This enables us to compute a separate set of kernel combination weights, one for each object class. In Section 4.4 we show an example of the evolution of these kernel weights over time.

4.4 Experiments

We test the performance of our system on the task of object detection on videos taken from a traffic dataset. This dataset consists of 11 challenging videos (480 x 704 pixels at 15 frames/second), of a busy intersection, taken from a traffic surveillance camera. The total number of frames is more than 120,000. Our task is to detect two classes of objects, cars and buses. We have ground truth marked for every tenth frame in this dataset.

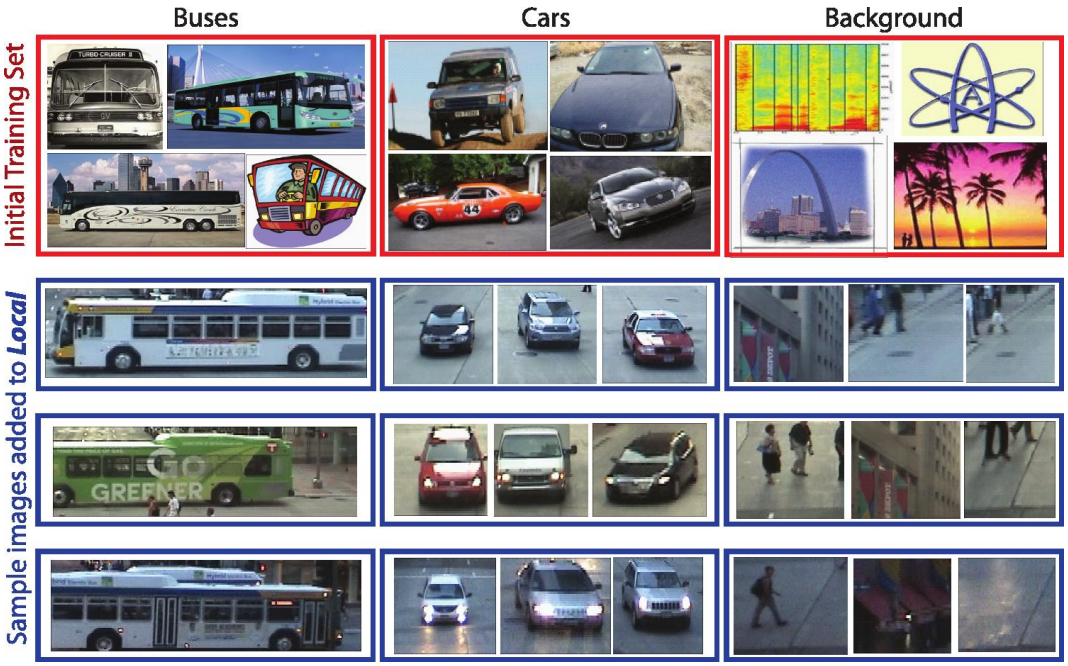


Figure 4.5: Snapshots of the training set at 4 time instants. Top row shows the initial training set. The next 3 rows show sample images added to *local* over time. The illumination change is noticeable at each time instant. The dataset gets updated with many objects in similar poses and representative background patches.

Due to the camera location and traffic restrictions in the scene, cars in the video typically have a frontal view, while buses typically appear in a profile view. Other views are also observed, but they are less common. The *car* category includes cars of varying sizes as well as SUV's and trucks. With a few exceptions, buses have a similar appearance, since most of them are public transportation buses. The dataset consists of videos captured at different times of the day, resulting in a variety of illumination conditions as shown in Figure 1.3, including street-lights at night. For videos captured during the transition of day to night, the appearances of the vehicles also change (most prominently, vehicles in the dark have their headlights turned on).

4.4.1 Kernel Matrices

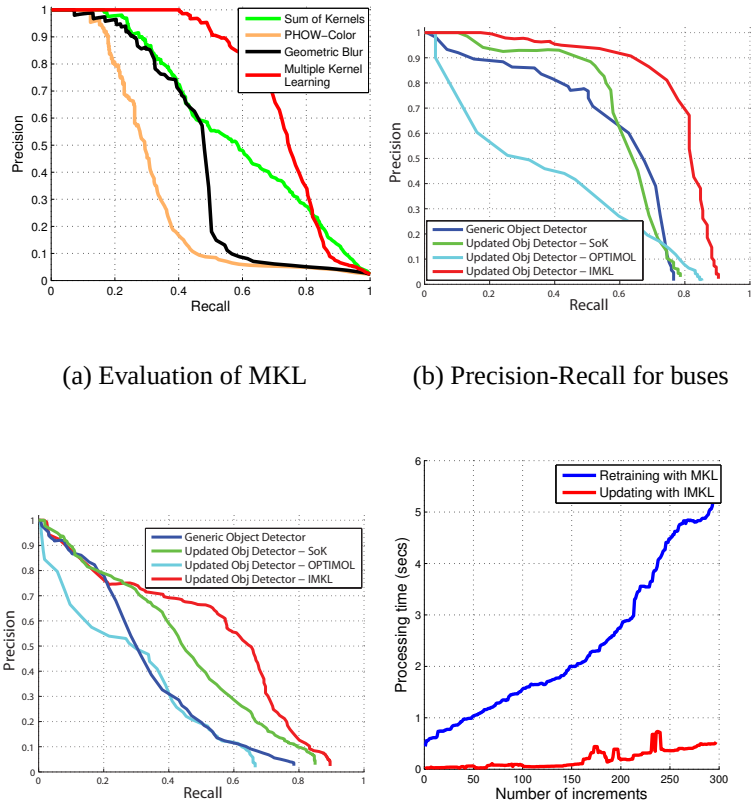
We use 5 kinds of features in our system, giving rise to a total of 17 kernel matrices. The first feature used is the Pyramidal Histogram of Oriented Gradients (PHOG-180) [9] to represent local shape. This consists of HOG features calculated over increasingly finer spatial grids. The orientations are calculated over the interval $[0, 180]$.We set the number of levels of the pyramid to 4. HOG features calculated for grids within the same level of the pyramid are concatenated to form a long feature vector, but feature vectors calculated at different levels are treated independently. Our IMKL algorithm automatically weights each level of the pyramid based on the training dataset. Histogram intersection is used as the similarity metric for all features in this paper. The first feature gives rise to 4 kernels, one for each level of the pyramid. The second feature is the PHOG-360. It only differs from PHOG-180 in that orientations are calculated over the interval $[0, 360]$. This also gives rise to 4 kernels.

The third feature, PHOW-Gray [9], encodes appearance. SIFT features are densely sampled at 10 pixel intervals in each direction and quantized to a 300 visual words vocabulary. Histograms of visual words are calculated over an increasing number of grids at each pyramidal level. We use 3 levels. The fourth feature is PHOW-Color. The only difference from PHOW-Gray is that it is calculated on the 3 channels of the HSV image. These give rise to 6 kernels.

The fifth feature is Geometric Blur (GB) [5], which captures shape information of the objects and also accounts for the geometric distortion between images. The unquantized GB feature was used with an expensive correspondence based distance metric

in [101]. However, in order to speed-up computations, we quantized the GB feature to a set of 300 visual words. We then calculated histograms of GB words in the same pyramidal framework to enforce some measure of spatial constraints. We used a 3 level pyramid. Thus we obtained a total of 17 kernel matrices.

4.4.2 Analysis



(a) Evaluation of MKL

(b) Precision-Recall for buses

(c) Precision-Recall for cars

(d) Efficiency of IMKL

Figure 4.6: (a) shows the evaluation of the individual kernels, combination using SoK and combination using MKL. MKL outperforms all other schemes. The best performing individual kernel is GB. (b) and (c) show the Precision-Recall curves for the *bus* and *car* classes respectively. Using our incremental object detector consistently increases performance in both cases. (d) compares the processing time of our incremental approach to retraining the MKL system at every step using all available images.

Evaluation of MKL. We first evaluate the power of using multiple kernels and using MKL to determine kernel weights for the given classification task. For this purpose, we created a validation dataset consisting of images of buses, cars and background extracted from the ground truth as well as the initial training set (obtained from Google). We then individually evaluated each kernel as well as the combination of kernels using a Sum of Kernels (SoK) approach (such as in [53]) and an MKL approach for both object classes over the validation set. The SoK approach assigns equal weights to all kernels. SoK has been known to provide good results when kernels are carefully chosen for the given data, but its performance degrades in the presence of noisy kernels. In our experiments, the MKL approach performs better than all other methods where as the SoK approach comes in second, outperforming both GB (the best performing individual feature) and the popular SIFT feature. Figure 4.6(a) shows the results for the *Buses* class.

Local dataset snapshots. We now demonstrate results of our IMKL approach on the video dataset. Starting from a generic training dataset, our IMKL algorithm simultaneously updates the training dataset as well as the kernel combination weights. Figure 4.5 shows snapshots of the training database at different time instants for one video.

Kernel weights over time. Figure 4.7 demonstrates the change of kernel combination weights over time. For this experiment, we chose a video where the scene is bright in the beginning but gets very dark by the end. We do not display kernel weights 1 to 8, since they do not show considerable change over time. Time 1 refers to the initial training dataset obtained from Google. Between times 1 and 2, we do not update the foreground classes to study the effect of updating only the background training set. This also causes a non-trivial change of weights (Time 2). After time 2, we update all object

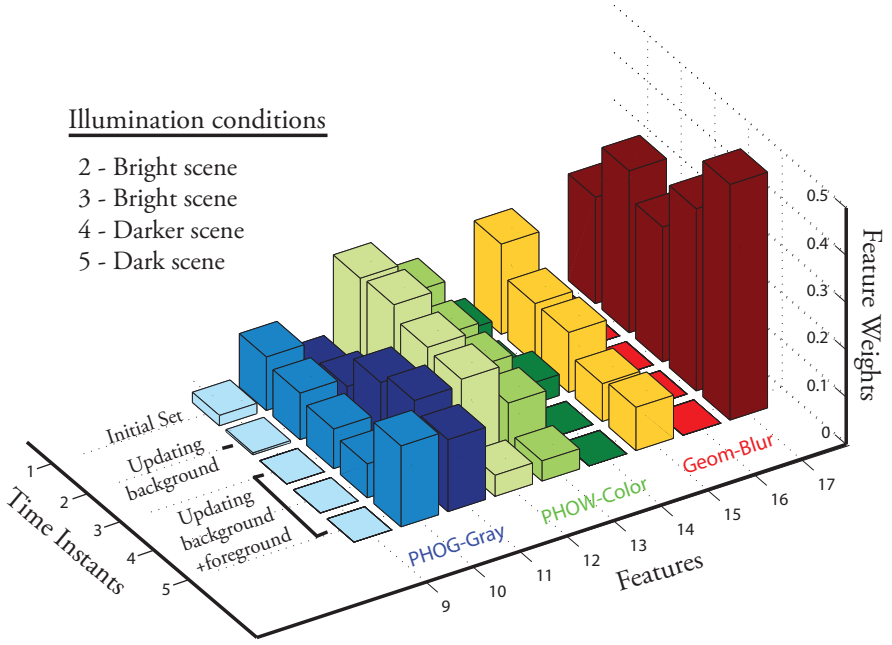


Figure 4.7: Kernel combination weights sampled at multiple time instants. Results shown for *Buses* class. *(see text for details)*

classes. Between times 2 and 3, the scene is bright. In this period, the detector tunes itself towards objects of specific poses and background patches. Beyond time 3, the scene gets darker. Here, PHOW-Color weights show a considerable drop (kernels 12-14), since color information in the video deteriorates, while PHOW-Gray kernels get higher weights. GB at fine spatial resolution (kernel 17) gets high weights with decreasing illumination, indicating added importance to positional information (such as importance given to the position of vehicle headlights).

Performance evaluation. Figures 4.6(b) and 4.6(c) show the performance of our system for the *bus* and *car* classes respectively, averaged over all videos in the dataset. We compare our IMKL object detector with 3 other detectors. Our baseline detector (which we call the *Generic* detector), represents an object detector built offline using only the generic training dataset and is not updated over time. It uses all 17 kernels and MKL

to obtain the kernel weights. Our second comparison is to an object detector built on the generic Google dataset and updated over time, but using SoK (equal kernel weights). Since these kernel weights are fixed over time, an incremental SVM approach suffices as the classifier. Our third comparison is to OPTIMOL [55], an incremental model learning approach, recently proposed for automatic object dataset collection.[1] The OPTIMOL algorithm is run independently of the IMKL system with a single change. In [55], Li et. al use SIFT as their feature descriptor. But given the superior performance of GB in our validation set (Figure 4.6(a)), we use histograms of GB based visual words as our feature descriptor for OPTIMOL.

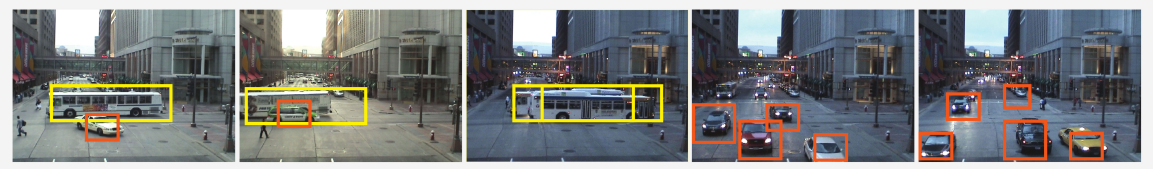


Figure 4.8: Sample results from a video sequence showing the ability of our system to adapt to gradual illumination changes.

Our IMKL approach outperforms the other 3 methods, especially at high recalls. Figure 4.9 provides some more insight into the results. This plot shows the performance of the various methods over time for one of the videos in our dataset for which illumination changes. The images at the bottom show a sample frame within the specified time interval. OPTIMOL starts off slowly but as it gets updated, it catches up with the rest of the object detectors. As the scene gets darker, however, its performance deteriorates. OPTIMOL uses GB, and even our IMKL approach begins to reduce the importance given to this

[1] We obtained code for OPTIMOL from the authors.

kernel when the scene becomes dark. We also noticed a low overall performance of OPTIMOL (on a subset of the data) while using other kernels such as PHOW-Gray and PHOW-COLOR. This is because no single kernel has been able to provide consistently good results in all scene conditions. Using multiple kernels with fixed weights (SoK) was also sub-optimal. Our IMKL approach provided the best results because it was able to dynamically change kernel weights based on the current object and scene characteristics. IMKL's performance decreases at times 4 and 6 since the scene changes, but recovers at instants 5 and 7, once it updates itself sufficiently.

Figure 4.8 shows sample results. Overall, we detect buses more reliably than cars. We are unable to consistently detect cars smaller than 60x60 pixels, which is the case for cars approaching from a distance, giving rise to a number of false negatives. Finally, Figure 4.6(d) illustrates the computational efficiency of the IMKL algorithm as compared to retraining the entire system using SimpleMKL.

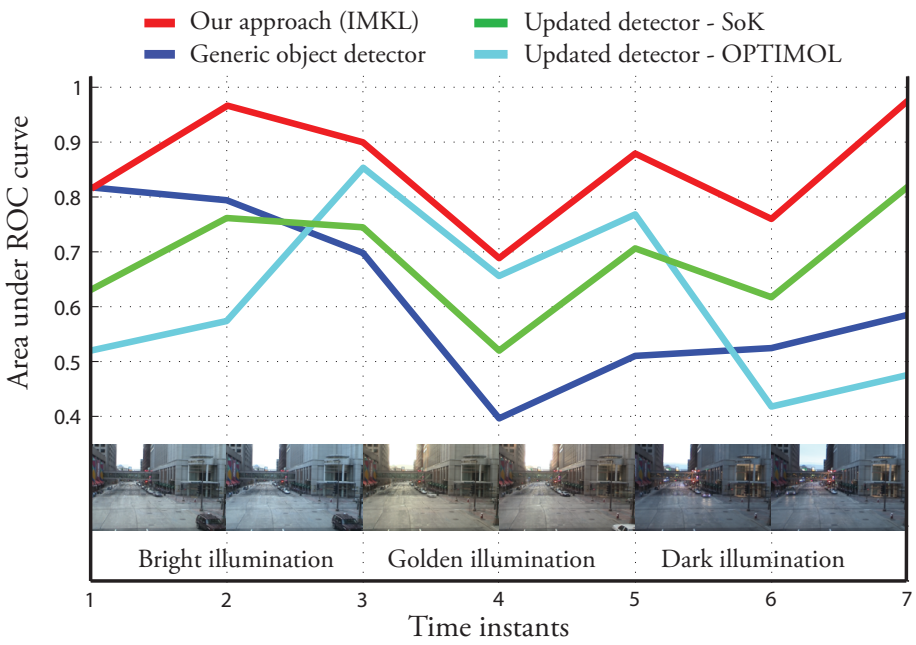


Figure 4.9: Performance comparison of object detectors over time for a single video for *Buses* class. *(see text for details)*

Chapter 5

Scene Understanding

5.1 Introduction

We develop a video understanding system for scene elements, such as bus stops, crosswalks, and intersections, that are characterized more by qualitative activities and geometry than by intrinsic appearance. The domain models for scene elements are not learned from a corpus of video, but instead, naturally elicited by humans, and represented as probabilistic logic rules within a Markov Logic Network framework. Human elicited models, however, represent object interactions as they occur in the 3D world rather than describing their appearance projection in some specific 2D image plane. We bridge this gap by recovering qualitative scene geometry to analyze object interactions in the 3D world and then reasoning about scene geometry, occlusions and common sense domain knowledge using a set of meta-rules.

5.2 Related Work

Methods to categorize scenes from single images by completely bypassing the tasks of image segmentation and object detection are described in [66, 27, 11]. Oliva et al. [66] represented holistic image structure using low level features that captured the degree of naturalness, openness, ruggedness, etc. whereas Fei-Fei et al. [27] represented scenes as

bags of codewords of texture measures. More recently, there have been attempts to jointly solve the tasks of object recognition and scene classification. Bosch et al. [10] detected objects and then used the object distribution for scene classification. Murphy et al. [64] combined the holistic image representation of [66] with local object detectors using a tree-structured graphical model. Li et al. [56] proposed a framework to deal with three problems simultaneously: object detection, segmentation and scene categorization.

There has also been progress in recovering surface orientations [40, 32] and occlusion boundaries [16], given just a single image. Recently, Hoiem et al. [41] proposed a framework in which estimates of surface orientations, occlusion boundaries, objects, camera viewpoint and relative depth are combined, enabling automatically reconstructed 3D models.

Research in the domain of scene understanding from videos has mostly focused on building models of motion patterns of objects and using these to detect anomalous behaviors [82, 60, 42, 75]. While Hu et al. [42] propose a parametric approach to model typical scene behaviors, Saleemi et al. use non-parametric density functions. Building such typical behavior models can help to improve foreground detection, detect areas of occlusion and identify anomalous motion patterns. There have also been attempts to learn activity based semantic region models for locations such as roads, paths, and entry/exits, most notably by Makris et al. [60] and Swears et al. [83]. These approaches both involved designing a detector for every scene element.

Research in object category recognition has typically focused on building visual classifiers trained on annotated datasets. Recently however, there has been a growing interest in building object category models directly from human elicited descriptions [50,

26, 91]. Such approaches have the potential to learn unseen object categories based on their descriptions in terms of known visual attributes.

5.3 Image Analysis

Figure 5.1: System overview. Our scene understanding system consists of an image analysis module (Section 5.3) that takes an input video and outputs a set of events and zone characteristics as observational evidence, a knowledge base (Section 5.4) that stores human elicited domain models and general rules about scene geometry and occlusion as a set of first-order logic rules, and an inference engine (Section 5.5) based on Markov Logic Networks that uses the logic rules and observational evidence to infer the labels of visible scene elements.

Our scene understanding framework has three components: an image analysis module, a knowledge base and an inference module (refer to Figure 5.1 for a system overview). The image analysis module first segments the scene into a set of neighborhoods called *zones*. It then analyzes appearance characteristics of each zone as well as motion properties of objects passing through them, to generate a set of zone attributes that characterize local scene geometry and capture occlusion relationships between zones. A set of dynamic events is then generated for every zone, at every time instant, to describe the behavior of objects in the scene. The knowledge base consists of domain models describing the scene elements of interest, as well as a set of meta-rules that capture general knowledge about scene geometry and occlusion. The inference module, based on Markov Logic Networks (MLN), integrates events generated by the image analysis component with the rules in the knowledge base to label scene elements. The knowledge base and inference module are described in Sections 5.4 and 5.5 respectively. The components of the image analysis module are described below.

5.3.1 Detection and Tracking

We detect and track three classes of objects: humans, cars and buses. Detection is carried out using the object detection method proposed in [78][1]. For the purposes of human detection, we directly used a trained model provided along with the code, which was trained on the INRIA pedestrian dataset [20]. The car detector is trained using the Caltech Car Rear Training Set and the ETHZ Car Side Training Set [54]. The bus detector is trained using images from Bing Image Search.

[1]Code obtained: http://www.umiacs.umd.edu/∼schwartz/softwares.html

A two level association based tracking method is used to link object detections into tracks. At the low level, detections are linked to form tracklets using appearance and proximity features. At the second level, these tracklets are associated into longer tracks using appearance and motion features. Figure 5.2b shows car and human tracks obtained for one of the videos in our dataset.

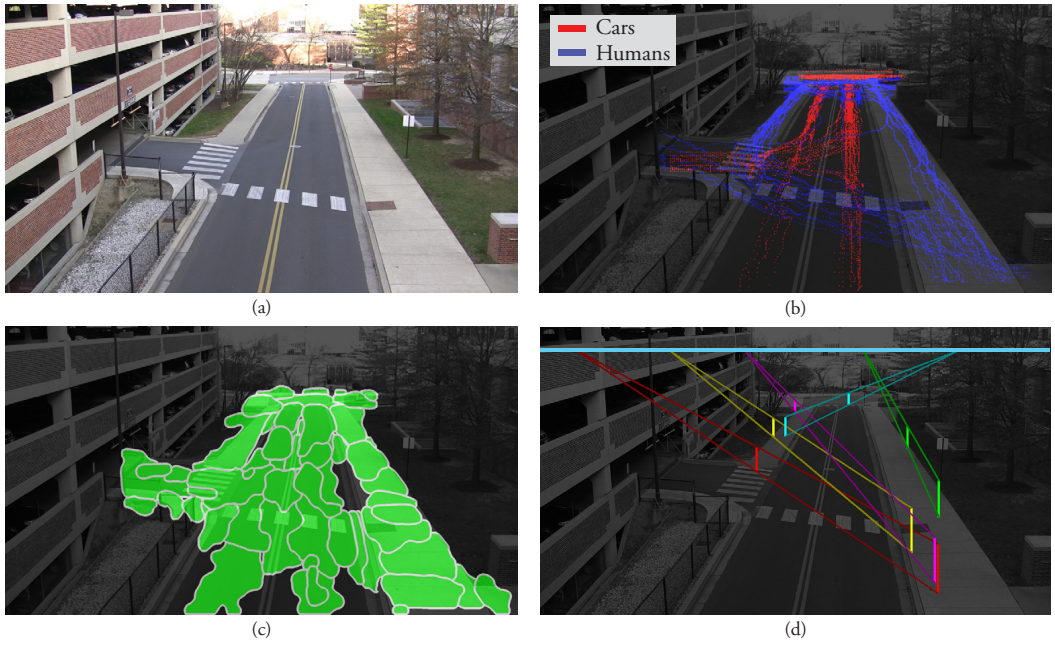


Figure 5.2: Components of the image analysis module. (a) Background image for Scene I. (b) Car and human trajectories (Section 5.3.1). (c) Zone segmentation (Section 5.3.2). (d) Horizon line estimate (Section 5.3.3).

5.3.2 Zone Segmentation

The MLN based reasoning module utilizes events generated by the image analysis framework to assign labels to each part of the scene. To avoid performing inference at the pixel level, we segment the scene spatially into a set of zones, and perform inference on each zone. Zone segmentation groups pixels based on their appearance, location and

the motion characteristics of objects passing through them. This results in a set of zones in which objects display distinct behaviors. Examples include locations where people gather and stand still for a long time (at bus stops), locations where vehicles drive in specific directions (along drive lanes), locations where cars and people cross each other (at cross walks), etc.

We begin by obtaining a background image by simply constructing an image for which a pixel $p(i, j)$ is the median of all pixels in the video at that location. This image is oversegmented by an image segmentation algorithm [2] to create a set of superpixels[2]. A set of features are computed for each superpixel, including

1. Appearance - 3 histograms (one each for R,G,B).

2. Motion - Velocity magnitude histogram and velocity orientation histograms (weighted by magnitude) for each class of passing objects.

An affinity matrix that includes the similarity between all pairs of superpixels is created for each feature. The distance metric used for all histograms is the Earth Mover's Distance (EMD). In addition, a location based affinity matrix is also created. This captures the minimum Euclidean distance between all pairs of superpixels and is calculated efficiently using the distance transform.

Spectral clustering is then used to group superpixels into zones. We used the self-tuning method proposed by Zelnik-Manor et al. [100][3], since it automatically selects the scale of analysis as well as the number of clusters. Figure 5.2c shows zones obtained for one of the scenes in our dataset.

[2]Code obtained: http://www.wisdom.weizmann.ac.il/∼ronen/index_files/segmentation.html

[3]Code obtained: http://www.vision.caltech.edu/lihi/Demos/SelfTuningClustering.html

5.3.3 Scene Geometry Analysis

5.3.3.1 Surface Layout

An estimate of the scene surface layout supports reasoning about the location of many scene elements. For example, entrance and exit zones (such as doors into buildings) are typically located where horizontal and vertical surfaces meet. We obtain a rough surface layout using the method of [40][4] which classifies pixels into three primary classes: *horizontal*, *vertical* and *sky*.

This estimate uses information extracted from individual images. However, we also have the additional knowledge of object trajectories that can help us obtain better surface estimates. Our meta-rules (discussed in Section 5.4) encode common sense knowledge about surfaces such as: *Objects are supported by a horizontal surface. Objects might appear out of and disappear into vertical surfaces.* Such rules allow us to correct some of the erroneous surface estimates provided by [40]. Figure 5.3 shows a surface layout before and after inference by our system.

5.3.3.2 Proximity Measures

Models of scene elements typically contain predicates corresponding to notions of proximity in the world, such as *nearby*, *far away*, *next to*, etc. Distances measured directly in the image plane, however, do not maintain these scene proximity relationships. Under a unit aspect ratio perspective camera model, we show how to compare segment lengths measured at different parts of the image based on their *true lengths* in the 3D

[4]Code obtained: http://www.cs.uiuc.edu/homes/dhoiem/

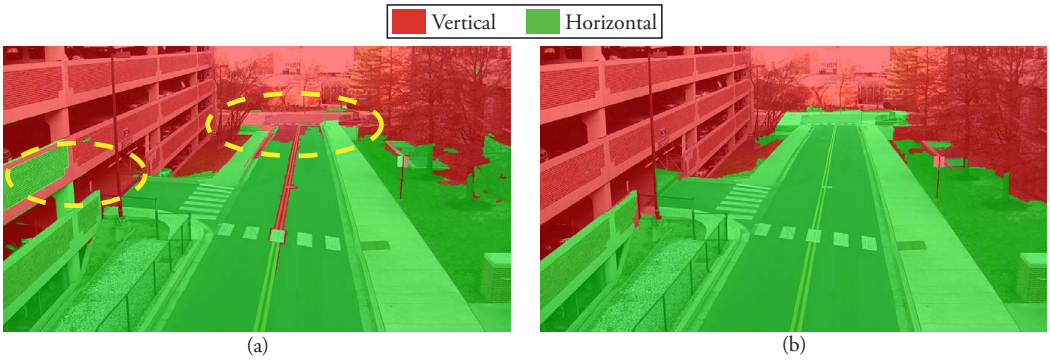


Figure 5.3: Surface layout estimates before and after inference by our system. The road visible in the far distance is erroneously labeled as a vertical surface (in (a)), but corrected after inference (in (b)), due to the presence of objects passing over it.

world. We break the problem down into two components: segments parallel to the camera axis (lengths measured along a column of pixels) and segments parallel to the camera image plane (lengths measured along a row of pixels), shown in Figures 5.4a and 5.4b respectively.

Consider Figure 5.4a. As in [39], we translate our image co-ordinates (u, v) to $(\hat{u}, \hat{v})$ so that $\hat{v} = 0$ for every point on the horizon line and $\hat{v} > 0$ below the horizon line. In this new co-ordinate system f_1 represents the foot location in the image of a person at a distance z_1 from the camera and f_1' is the foot location when the person takes a step Δz_1 parallel to the camera axis to be located at a distance z_1' from the camera. Now, $f_1 z_1 = f_1' z_1' = f y_c$. Consider a person at a second location in the scene taking a step Δz_2. This gives us: $f_2 z_2 = f_2' z_2' = f y_c$. A little algebra yields,

$$\frac{(f_1' - f_1) f_2 f_2'}{(f_2' - f_2) f_1 f_1'} = \frac{\Delta z_1}{\Delta z_2} \tag{5.1}$$

Now consider Figure 5.4b. Here the person moves from foot location f_1 to a new location f_1' parallel to the camera image plane. One can obtain: $\Delta i_1 y_c = \Delta z_1 f_1$, where

Δi_1 represents the image plane distance between the two feet locations. For a second person at a new location, we obtain: $\Delta i_2 y_c = \Delta z_2 f_2$. This yields,

$$\frac{\Delta i_1}{\Delta i_2}\frac{f_2}{f_1} = \frac{\Delta z_1}{\Delta z_2} \tag{5.2}$$

Given the location of the horizon line, Equations 5.1 and 5.2 relate distances (segment lengths) measured at different locations in the image plane, based on the true 3D measurements. Measures such as *nearby*, *far away*, *large distance away*, when defined at one location in the image, can be transformed to equivalent measures at other locations.

The horizon line is estimated using the method of Lv et al. [43]. Consider two vertical poles of the same height in the scene. The two lines joining their foot locations and head locations, respectively, intersect at a point on the horizon line. Thus, three non-coplanar poles of the same height uniquely determine the horizon line. In practice, we have a large number of people walking through each scene. Each pair of detections (from the same human track) provides us with an estimate of a point lying on the horizon line. A least squares estimate of many such detection pairs yields a good horizon line estimate (shown in Figure 5.2d).

5.3.3.3 Zone Transitions

While the distance measures described above help define notions of proximity in the scene, they do not capture the restrictions imposed on object trajectories due to the scene layout. For example, a sidewalk is located adjacent to a road, yet vehicles typically do not traverse between roads and sidewalks. We characterize typical object traffic patterns in the scene in terms of the average transition times of objects between one zone and another.

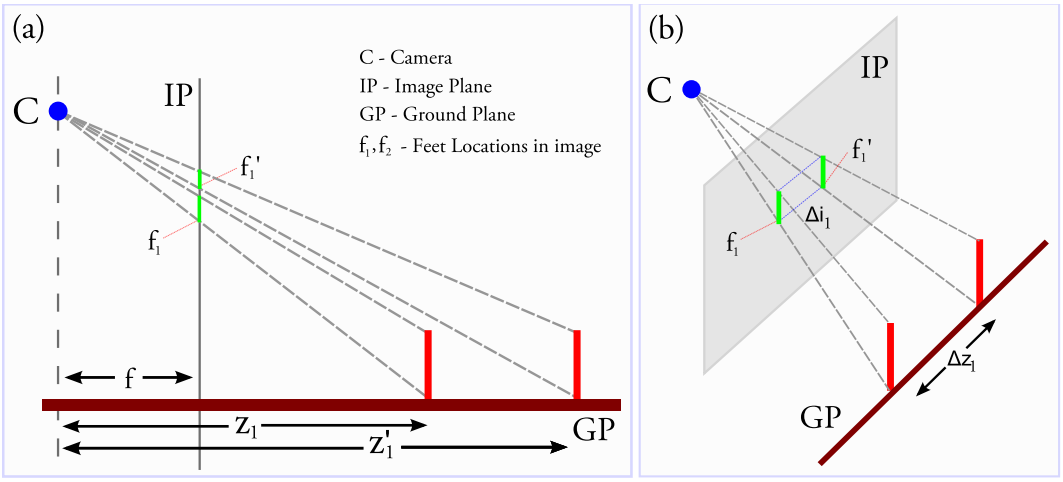


Figure 5.4: Schematic relating image plane distances to ground plane distances.

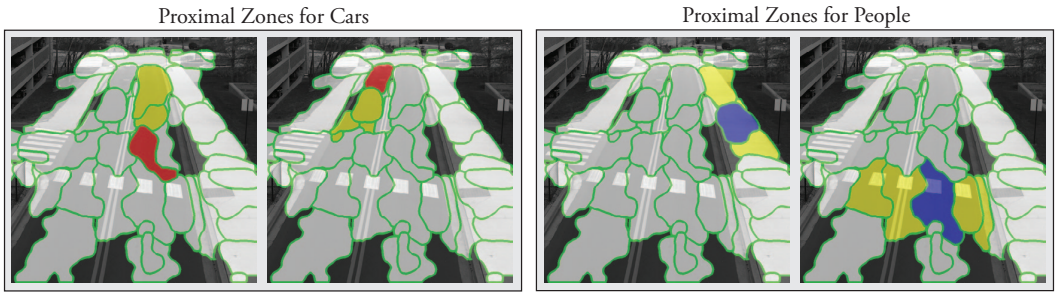


Figure 5.5: Examples of proximal zones based on zone transition matrices. (a) Vehicles travel from red zones onto yellow zones within a short time span. (b) People walk from blue zones onto yellow within a short time span.

These patterns are represented as transition matrices $T^k(z_i, z_j)$, one for each object class k. Zone pairs that do not have any traffic flowing between them, are assigned a large transition time by default. Figure 5.5 shows examples of proximal zones based on the zone transition measure. Note that cars typically conform to fixed directions along road lanes, where as people walk along paths in both directions.

5.3.3.4 Directionality

User descriptions of scene elements often involve spatial prepositions which provide a notion of directionality, such as *in front of, behind, to the left of,* etc. Under the assumption that objects move in the direction in which they are facing, we define four directions with respect to the motion of the object: left, right, front and behind. Furthermore, some zones in the scene exhibit a single dominant direction of motion (based on the objects that pass through them). This is especially true of zones located on the road, on which vehicles strictly follow a single direction of motion. The four directions defined above are also noted for such zones, with respect to the centroid of the given zone.

5.3.4 Zone Occlusion Relationships

As vehicles and humans move through the scene, they occlude different areas of the scene as well as objects present at those locations. This is a common source of detection and tracking errors in a typical computer vision system. Knowledge about typical occlusion areas can provide valuable information to the scene understanding framework. For example, people trajectories ending at a location suggest the presence of a doorway/entrance to a building at that location. However, the observation of a vehicle parked nearby, with the knowledge that it may causes occlusions at the former location, can prevent such an inference error.

We represent occlusion relationships between zones using a binary matrix OC. For every object that passes through a zone z_i, we determine zones in the scene that intersect the object bounding box in the image plane (indicating potential occlusions), while the

object was within z_i. If a zone z_j consistently undergoes occlusion by objects in z_i, the indicator variable $OC(z_i, z_j)$ is set to 1. An occlusion relationship matrix is created for every object class.

5.3.5 Event Generation

Short time spans of 20 frames are grouped together to form a temporal window. A set of dynamic events is generated at every zone within each temporal window. These events characterize the location, motion and trajectory of objects in a given zone during the given window. This results in a large set of evidence ground atoms passed to the inference module throughout the duration of the video sequence. In addition, the image analysis module also generates a set of zone characteristics and inter zone relationships, as described above. These are also represented as evidence atoms and passed on to the inference module.

5.4 Knowledge Base

The knowledge base in our system consists of two components: a set of domain models describing the scene elements and a set of meta-rules that capture information about scene geometry, general occlusion reasoning as well as common sense knowledge that applies to many domains. We begin with a description of our approach to represent uncertain knowledge, and then proceed with outlining the two components of our knowledge base.

5.4.1 Knowledge Representation

Knowledge is represented in our system as first order production rules. The rules are represented in clausal form, whereby each rule is a conjunction of clauses and each clause is a disjunction of literals. Rules are constructed using variables such as *zone*, *time*, etc. Some of our variables are typed, in which case they range over a predetermined set of objects. All typed variables have mutually exclusive and exhaustive values. For example, the typed variable *appearPersonReason* signifies an explanation for the birth of a person track and must take one of the following values: {*NearBuildingEntrance, TrackingFailure, OcclusionByCar, ... , Other*}.

We use two types of predicates. The first represents events in the video and are associated with a particular zone and time instant. For example, *PersonAppear(zone,time)*, *CarStop(zone,time)*, etc. These are generated by the image analysis module at each time instant. The second represents properties of individual zones such as *ZoneIsVertical(zone)*, relationships between zones such as *ZoneNearZone(zone, zone)* and relationships between time instants such as *ShortlyAfter(time, time)*. These predicates are also generated by the image analysis module, but they need only be calculated once for the entire video sequence.

Each rule in our knowledge base is associated with a weight that indicates its confidence. We use three degree of confidence for our rules to indicate rules of absolute certainty ($weight = M$), ones with lesser certainty ($weight = 0.5M$) and rules that may be true a very small fraction of times ($weight = 0.25M$). In principle, one may infer the certainty of a human elicited rule by frequency adverbs such as always, never, sometimes,

rarely, etc.

Some of the predicates generated by the image analysis module, such as *ZoneIsVertical(zone)*, have a confidence value associated with them. Such uncertain predicates are integrated into the first order rules using the method employed in [88]. Consider a predicate P with a weight w. We introduce a dummy observation predicate O_P along with a rule $O_P \rightarrow P$ and associate the weight w with this rule, instead of assigning it with predicate P. The predicate O_P does not have any weight associated with it.

5.4.2 Scene Element Models

Each scene element is described by a logical model comprising a set of first order rules. These logical models describe a scene element on the basis of *what typically happens* in a scene at that element. For example, the logical model for a cross-walk consisting of first order logic rules with confidence measures is given in Figure 5.6. The numbers in parentheses represent the weight assigned to each rule (recall that M represents the highest weight assigned in the knowledge base). The presence of people walking on the road indicates that they might be passing over a crosswalk (Rule 1). However, pedestrians often disobey laws and cross the road at other locations. The presence of a car waiting for people to cross the road is a stronger indication of a crosswalk and is thus assigned a higher weight (Rule 2).

```
Crosswalk Model:
Rule1: (0.25M)  PeopleMove(z1,t1) ^ ZoneClassA(z1,Road) => ZoneClass(z1,Crosswalk)
Rule2: (0.5M)   PeopleMove(z1,t1) ^ ZoneClassA(z1,Road) ^ CarStop(z2,t1) ^
                ZoneTransitionCar(z2,z1) => ZoneClass(z1,Crosswalk)
Rule3: (0.5M)   ZoneClassA(z1,Road) ^ ZoneTransitionPeople(z2,z1) ^ ZoneClassA(z2,Sidewalk) ^
                ZoneTransitionPeople(z1,z3) ^ ZoneClassA(z3,Sidewalk) => ZoneClass(z1,Crosswalk)
Rule4: (1.0M)   !ZoneClass(z1,Road) => !ZoneClass(z1,Crosswalk)
```

Figure 5.6: First order logic rules representing a crosswalk model.

5.4.3 Meta-Rules

In addition to the scene element specific models, the knowledge base also consists of a set of meta-rules, which encode information relating to scene geometry, occlusion handling, common failures of low level computer vision modules as well as common sense knowledge about the world. They only need to be written once, but are then widely applicable over a large number of domains. For instance, consider the scene element *Building Entrance/Exit*. Entrances and exits are typically characterized as sources and sinks of person tracks. There are however, a variety of situations that may lead to an initiation of a person track such as: exiting a vehicle, occlusion by a vehicle, identity switching by the tracker, entering the image at its boundary, occlusion within a group of people, etc. Our meta rules encode such possibilities. This enables the inference module to reason about plausible explanations when it encounters a new person track. This reduces the number of false locations that might be labeled as an entrance-exit.

5.5 Inference using Markov Logic Networks

There has been a growing interest in problems related to knowledge representation and learning in domains that are rich in relational as well as probabilistic structure.

Markov Logic Networks (MLN) are one such representation that combine first order logic with probability theory in finite domains [74]. They support the specification of statistical models using intuitive and understandable first order rules. A first order knowledge base, by itself, is often impractical to use for real world problems. Each rule in such a knowledge base is a hard constraint. A world that does not satisfy a single formula gets assigned a zero probability. MLNs attempt to relax these hard constraints using weights for each formula. The probability of a world is dependent upon the number of formulae that the world satisfies and the weights assigned to those formulae.

MLNs can also be viewed as a template for constructing ordinary Markov networks. Given a set of formulae and constants, a MLN produces a Markov network. Based on the constructed network, marginal distributions of events given the observations can be computed using probabilistic inference. Since the resulting network may contain cycles, exact inference is intractable. Inference in MLNs is thus often performed using MCMC. For details on efficient inference algorithms in MLNs we refer the reader to [74]. We use the Alchemy system [48] to represent our rules and perform inference on the resulting MLN[5].

5.5.1 Local Inference Procedures

The image analysis module generates a large number of evidence ground atoms within every temporal window, for every zone in the scene. Over the entire video, the number of ground atoms gets prohibitively large, rendering inference intractable. However, the spatio temporal interactions between objects, that characterize the scene ele-

[5]Code available: http://alchemy.cs.washington.edu/

ments of interest are sufficiently local in nature, both spatially and temporally. For instance, consider the crosswalk model in Figure 5.6 described by the interaction between people walking on the crosswalk and vehicles waiting on the road adjacent to it. Interactions between objects at locations far away from the crosswalk do not affect inference about the given zone. Likewise, interactions between people and vehicles at the crosswalk, at other times in the video, are largely independent of the current interaction.

We break down the large inference problem into smaller ones, carried out in every zone and at regularly spaced time instants. For every such spatio temporal location, the inference procedure takes into consideration events generated at a set of neighboring zones and time instants. For each zone, votes for each label, which are generated over the duration of the video, are aggregated to determine the final scene element label associated with that zone.

5.6 Experiments

We demonstrate our scene understanding framework on a dataset of 5 videos of public spaces, totaling over 100,000 frames (about 58 minutes). The video data has been collected using cameras overlooking scenes from varying viewpoints. Each scene contains a large amount of pedestrian, car and bus traffic passing through it. Over the entire dataset, the number of pedestrians, cars and buses is approximately 700, 500 and 25 respectively. The data has been collected in high definition mode (1920x1080 pixels). Figures 5.7- 5.11 shows some representative frames.

The scene elements that we seek to identify are: Road, Sidewalk, Other Path (other

paths taken by people, which are not sidewalks), Bus-stops, Stop-sign Zones, Crosswalks, Entrances-Exits for People (typically buildings) and Entrances-Exits for Vehicles (typically garages). Figure 5.7- 5.11 shows the labels assigned to different regions of the scenes. The system is able to correctly identify a large number of the scene elements using the human elicited domain models.

Our scene understanding framework is effectively able to reason about the scene geometry and occlusions to identify scene elements from widely varying viewpoints. Recall the example of a bus-stop observed from two viewpoints (refer to Figure 1.4). Scene III contains a view of a bus-stop in which we are able to observe people entering and exiting the bus. Scene II and IV, on the other hand, contain views of bus-stops in which the doors of the bus are not visible. The system reasons about people that might have entered and exited the buses that stopped at the location and correctly identifies all bus stops. Note that in Scene III, two locations are marked as bus-stops. This is because camera III overlooks a bus depot at which multiple buses stop one behind the other.

Pedestrian crosswalks are also correctly identified in all scenes, with the exception of a partially visible crosswalk in Scene II. These include the three crosswalks visible in the far distance in Scene III. A fair number of people tend to cross roads at locations other than crosswalks. However, cars do not always stop for such jaywalking violations. The system correctly identifies crosswalk locations using this additional information and suppresses the false alarms.

Vehicle and pedestrian entrances are identified on the basis of track appearances and disappearances into vertical surfaces. Scene I shows an example of a correctly identified garage entrance. The other detections in Scene I are not garage entrances, but they

correspond to locations in the scene (away from the image boundary and close to vertical surfaces) where cars enter and exit the camera frame. Scene V shows a loading dock correctly marked as a potential entrance/exit for people. We fail to detect one of the doorways in Scene III (primarily due to an leafless, yet occluding tree). Another entrance in the same scene at a large distance away is correctly determined.

The scene elements: Roads, Sidewalks and Other Paths are also identified in each scene. Sidewalks are defined to be paths adjacent to roads and parallel to them on which people walk. Zones in the scene are considered parallel to one another if the orientations of objects passing through them are similar. Stop-sign zones are also correctly detected in the scenes. The system does not merely depend on locations where cars stop-and-go, but also uses information such as *Stop zones may be located adjacent to cross-walks and at intersections*. Scene V shows a false alarm caused by cars frequently stopping at a crosswalk with a very large amount of pedestrian traffic. Such false alarms can be reduced by analyzing a larger amount of data, possibly spanning different times of the day.

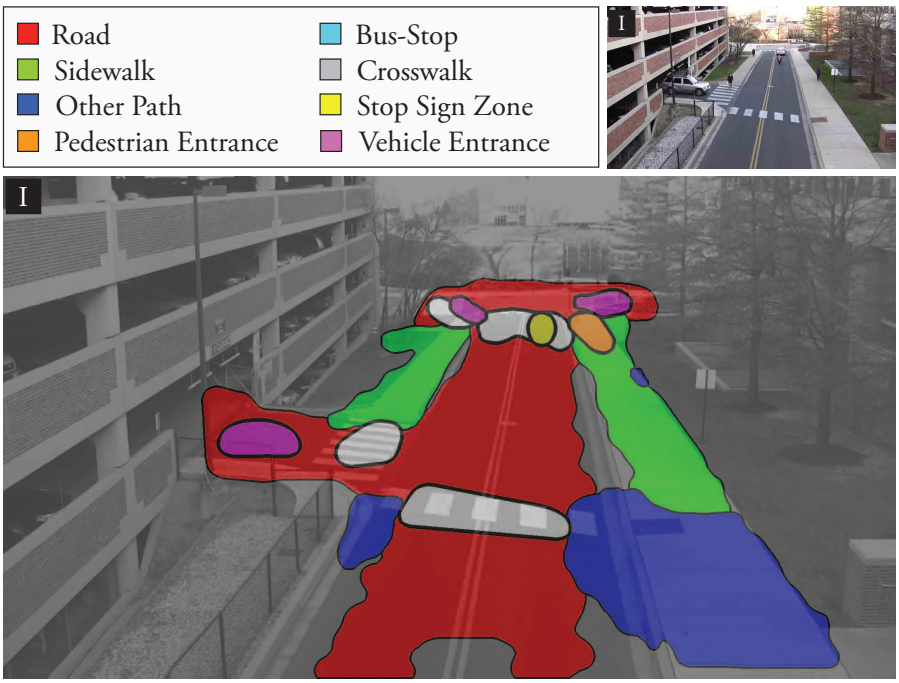


Figure 5.7: Scene element labels determined by our system for Scene I along with a representative image from the scene.

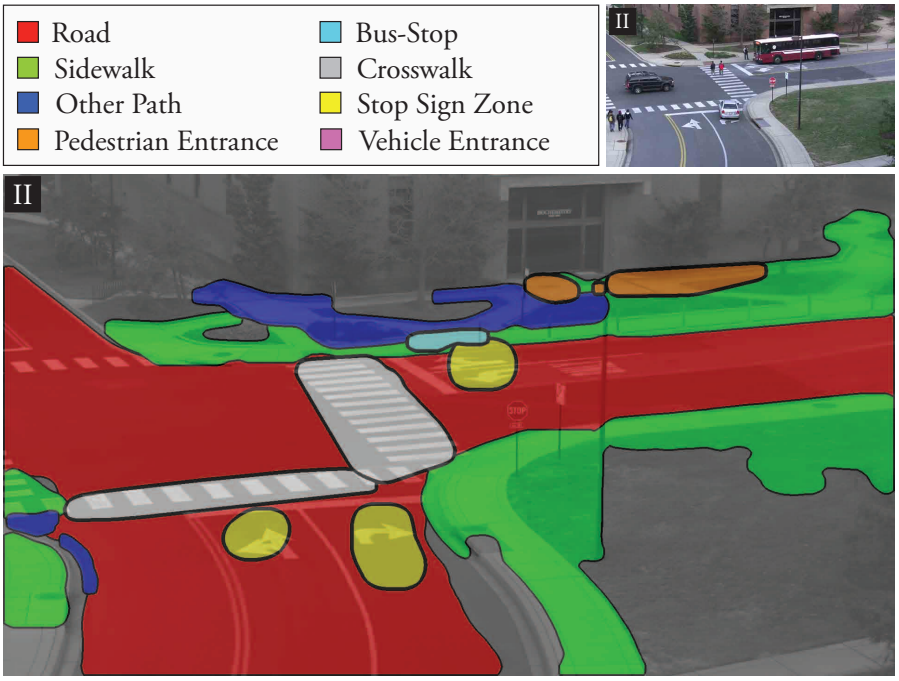


Figure 5.8: Scene element labels determined by our system for Scene II along with a representative image from the scene.

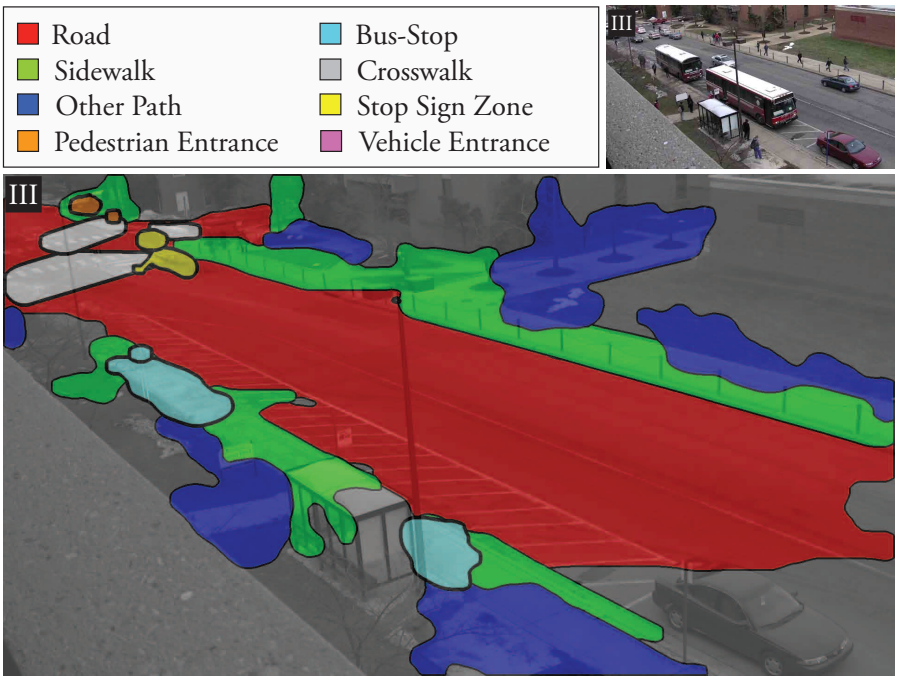


Figure 5.9: Scene element labels determined by our system for Scene III along with a representative image from the scene.

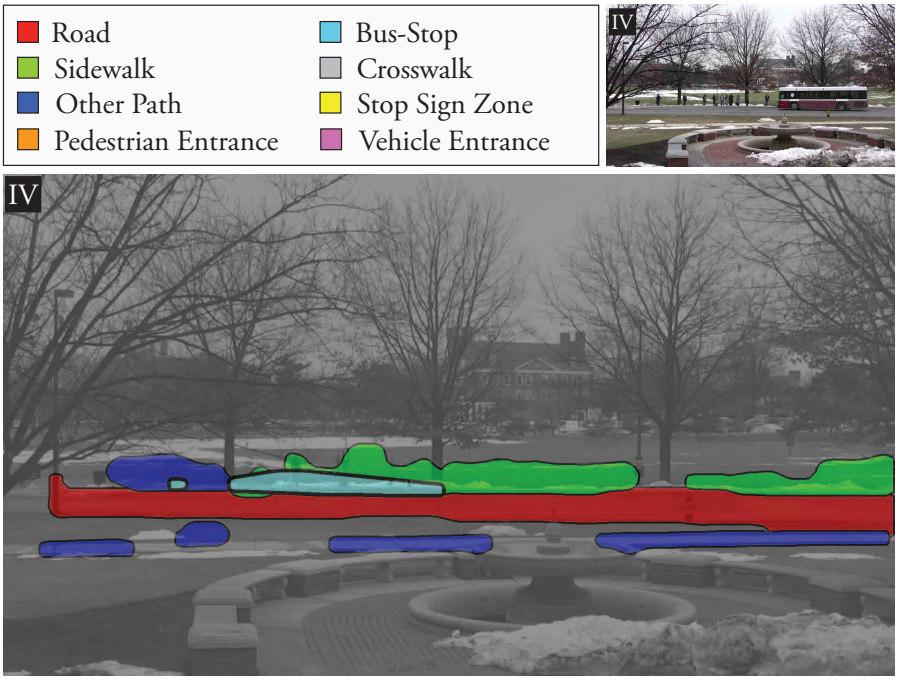


Figure 5.10: Scene element labels determined by our system for Scene IV along with a representative image from the scene.

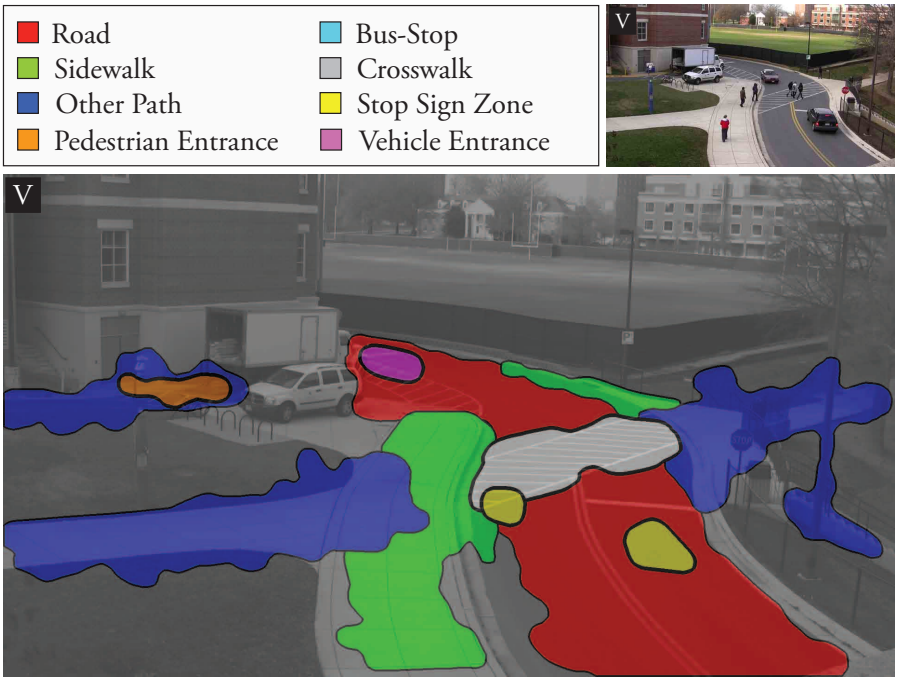


Figure 5.11: Scene element labels determined by our system for Scene V along with a representative image from the scene.

Chapter 6

Conclusions

Scene understanding is one of the fundamental objectives of computer vision. This task involves recognizing the different objects present in the scene, segmenting the scene into meaningful regions, as well as obtaining a holistic understanding of the activities taking place in the scene. Each of these problems has received considerable interest within the computer vision community. In this thesis, we presented contributions to two aspects of visual scene understanding.

First we explored multiple methods of feature selection for the problem of object detection. We demonstrated the use of Principal Components Analysis to select features in different parts of the scene, in order improve object detection in video. We then demonstrated the use of Partial Least Squares, a supervised dimensionality reduction tool, to detect vehicles in aerial and satellite imagery. We proposed two new feature sets: Color Probability Maps, to capture the color statistics of vehicles and their surroundings, and Pairs of Pixels, to capture the structural characteristics of objects. A powerful feature selection analysis based on Partial Least Squares was employed to deal with the resulting high dimensional feature space (almost 70,000 dimensions). We compared against state of the art approaches to object detection and consistently obtained superior results. We also proposed an Incremental Multiple Kernel Learning (IMKL) scheme to detect vehicles in a traffic surveillance scenario. Obtaining task and scene specific datasets of visual

categories is far more tedious than obtaining a generic dataset of the same classes. Our IMKL approach initialized on a generic training database obtained from Google Image Search and then tuned itself to the task of detecting vehicles at a busy traffic intersection.

Second, we developed a video understanding system for scene elements, such as bus stops, crosswalks, and intersections, that are characterized more by qualitative activities and geometry than by intrinsic appearance. The domain models for these scene elements were written down by humans and were represented as probabilistic logic rules within a Markov Logic Network framework. We analyzed object interactions in the 3D world and reasoned about qualitative scene geometry, occlusions and common sense domain knowledge.

Bibliography

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

[2] Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. *Computer Vision and Pattern Recognition (CVPR)*, 2007.

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

[4] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19, 1997.

[5] A.C. Berg and J. Malik. Geometric blur for template matching. *Computer Vision and Pattern Recognition (CVPR)*, 2001.

[6] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 1997.

[7] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *COLT: Proceedings of the Workshop on Computational Learning Theory*, 1998.

[8] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. *International Conference on Computer Vision (ICCV)*, 2007.

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

[10] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Scene classification via plsa. *European Conference on Computer Vision (ECCV)*, 2006.

[11] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2008.

[12] A.-L. Boulesteix and K. Strimmer. Partial least squares: A versatile tool for the analysis of high-dimensional genomic data. *Briefings in Bioinformatics*, 2007.

[13] Kristin Branson and Serge Belongie. Tracking multiple mouse contours (without too many samples). In *Computer Vision and Pattern Recognition (CVPR)*, pages 1039–1046, 2005.

[14] Kristin Branson, Vincent Rabaud, and Serge Belongie. Three brown mice: See how they run. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, 2003.

[15] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. *Neural Information Processing Systems (NIPS)*, 2000.

[16] Xiaofeng Ren Charless, Xiaofeng Ren, Charless C. Fowlkes, and Jitendra Malik. Figure/ground assignment in natural images. *European Conference on Computer Vision (ECCV)*, 2006.

[17] Li-Fen. Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 2000.

[18] Jae-Young Choi and Young-Kyu Yang. Vehicle detection from aerial images using local shape information. *PSIVT '09: Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, 2008.

[19] Seth W. Coleman, Gail L. Patricelli, and Gerald Borgia. Variable female preferences drive complex male displays. *Nature*, 428(6984):742–745, 2004.

[20] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[21] Piotr Dollar, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1971, 2006.

[22] L. Eikvil, L. Aurdal, and H. Koren. Classification-based vehicle detection in high-resolution satellite images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2009.

[23] Ahmed M. Elgammal, David Harwood, and Larry S. Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision (ECCV)*, pages 751–767, 2000.

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

[25] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[26] Ali Farhadi, Ian Endres, Derek Hoiem, and David A. Forsyth. Describing objects by their attributes. 2009.

[27] Li Fei-fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. *Computer Vision and Pattern Recognition (CVPR)*, 2005.

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

[29] John W. Fitzpatrick, Martjan Lammertink, Jr. Luneau, M. David, Tim W. Gallagher, Bobby R. Harrison, Gene Sparling, Kenneth Rosenberg, Ronald Rohrbaugh, Elliott Swarthout, Peter Wrege, Sara Barker Swarthout, Marc S. Dantzker, Russell A. Charif, Timothy R. Barksdale, Jr. Remsen, J. V., Scott D. Simon, and Douglas Zollner. Ivory-billed Woodpecker (Campephilus principalis) Persists in Continental North America. *Science*, 308(5727):1460–1462, 2005.

[30] P. H. Garthwaite. An interpretation of partial least squares. *Journal of the American Statistical Association*, 1994.

[31] P. Geladi and B.R. Kowalski. Partial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185, 1986.

[32] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. *International Conference on Computer Vision (ICCV)*, 2009.

[33] Helmut Grabner, T.T. Nguyen, Barbara Gruber, and Horst Bischof. On-line boosting-based car detection from aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(3):382 – 396, 2008.

[34] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 2003.

[35] Marko Heikkilä and Matti Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(4):657–662, 2006.

[36] I. Helland. On the structure of partial least squares. *Communication in Statistics., Simulation and Computation*, 1988.

[37] Geoffrey E. Hill, Daniel J. Mennill, Brian W. Rolek, Tyler L. Hicks, and Kyle A. Swiston. Evidence suggesting that ivory-billed woodpeckers (*Campephilus principalis*) exist in florida. *Avian Conservation and Ecology*, 2006.

[38] Stefan Hinz. Detection and counting of cars in aerial images. *International Conference on Image Processing (ICIP)*, 2003.

[39] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[40] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision (IJCV)*, 2007.

[41] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Closing the loop on scene interpretation. *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[42] Weiming Hu, Xuejuan Xiao, Zhouyu Fu, Dan Xie, Tieniu Tan, and Steve Maybank. A system for learning statistical motion patterns. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2006.

[43] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. *European Conference on Computer Vision (ECCV)*, 2008.

[44] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[45] Zia Khan, Tucker Balch, and Frank Dellaert. A rao-blackwellized particle filter for eigentracking. *Computer Vision and Pattern Recognition (CVPR)*, pages 980–986, 2004.

[46] Zia Khan, Tucker R. Balch, and Frank Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *European Conference on Computer Vision (ECCV)*, pages 279–290, 2004.

[47] Zia Khan, Rebecca A. Herman, Kim Wallen, and Tucker Balch. An outdoor 3-d visual tracking system for the study of spatial navigation and memory in rhesus monkeys. *Behavior Research Methods*, 37:453–463, 2005.

[48] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. The alchemy system for statistical relational ai. *Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA*, 2007.

[49] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2009.

[50] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[51] G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research (JMLR)*, 2004.

[52] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid match for recognizing natural scene categories. *Computer Vision and Pattern Recognition (CVPR)*, 2006.

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

[54] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision (IJCV)*, 2008.

[55] L. Li, G. Wang, and L. Fei-Fei. Optimol: automatic object picture collection via incremental model learning. *Computer Vision and Pattern Recognition (CVPR)*, 07.

[56] Li-Jia Li, R. Socher, and Li Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[57] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 2004.

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

[59] Subhransu Maji and Alexander C. Berg. Max-margin additive classifiers for detection. *International Conference on Computer Vision (ICCV)*, 2009.

[60] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man, and Cybernetics*, 2005.

[61] H. Moon, R. Chellappa, and A. Rosenfeld. Optimal edge-based shape detection. *IEEE Transactions on Image Processing*, 2002.

[62] H. Moon, R. Chellappa, and A. Rosenfeld. Performance analysis of a simple vehicle detection algorithm. *Image and Vision Computing*, 2002.

[63] Vlad I. Morariu, Balaji Vasan Srinivasan, Vikas C. Raykar, Ramani Duraiswami, and Larry S. Davis. Automatic online tuning for fast gaussian summation. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[64] Kevin Murphy, Antonio Torralba, and William T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. *Neural Information Processing Systems (NIPS)*, 2003.

[65] Sang Min Oh, James M. Rehg, Tucker Balch, and Frank Dellaert. Learning and inference in parametric switching linear dynamical systems. In *International Conference on Computer Vision (ICCV)*, pages 1161–1168, 2005.

[66] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision (IJCV)*, 2001.

[67] Toufiq Parag, Ahmed M. Elgammal, and Anurag Mittal. A framework for feature selection for background subtraction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1916–1923, 2006.

[68] Gail L. Patricelli, J. Albert C. Uy, Gregory Walsh, and Gerald Borgia. Sexual selection: Male displays adjusted to female's response. *Nature*, 415(6869):279–280, 2002.

[69] M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1998.

[70] Fatih Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. *Computer Vision and Pattern Recognition (CVPR)*, 1:829–836, 2005.

[71] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. More efficency in multiple kernel learning. *International Conference on Machine Learning (ICML)*, 07.

[72] Vikas C. Raykar and Ramani Duraiswami. Fast optimal bandwidth selection for kernel density estimation. In *SIAM International Conference on Data Mining (SDM)*, 2006.

[73] S.P. Reinikainen and A. Hoskuldsson. Covproc method: strategy in modeling dynamic systems. *Journal of Chemometrics*, 2003.

[74] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 2006.

[75] Imran Saleemi, Khurram Shafique, and Mubarak Shah. Probabilistic modeling of scene dynamics for applications in visual surveillance. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2009.

[76] Jean-François Savard, Jason Keagy, and Gerald Borgia. Spatial dynamics and modulation of courtship in satin bowerbirds, *Ptilonorhynchus violaceus*. 44th annual meeting of the Animal Behavior Society, 2007.

[77] C. Schlosser, J. Reitberger, and Stefan Hinz. Automatic car detection in high resolution urban scenes based on an adaptive 3d model. *GRSS/ISPRS Joint Workshop on Data Fusion and Remote Sensing Over Urban Areas*, 2003.

[78] W.R. Schwartz, A. Kembhavi, D. Harwood, and L.S. Davis. Human detection using partial least squares analysis. *International Conference on Computer Vision (ICCV)*, 2009.

[79] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and location in images. *International Conference on Computer Vision (ICCV)*, 2005.

[80] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research (JMLR)*, 2006.

[81] Chris Stauffer and W. Eric L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2246–2252, 1999.

[82] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[83] Eran Swears and Anthony Hoogs. Functional scene element recognition for video scene analysis. *Workshop on Motion and Video Computing (WMVC)*, 2009.

[84] N.A. Syed, H. Liu, and K.K. Sung. Incremental learning with support vector machines. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 1999.

[85] F. Tanner, B. Colder, C. Pullen, D. Heagy, M. Eppolito, V. Carlan, C. Oertel, and P. Sallee. Overhead imagery research data set an annotated data library and tools to aid in the development of computer vision algorithms. *IEEE Applied Imagery Pattern Recognition Workshop 2009*, 2009.

[86] R.F. Teofilo, J.P.A. Martins, and M.C. Ferreira. Sorting variables by using informative vectors as a strategy for feature selection in multivariate regression. *Journal of Chemometrics*, 2008.

[87] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In *International Conference on Computer Vision (ICCV)*, pages 255–261, 1999.

[88] Son D. Tran and Larry S. Davis. Event modeling and recognition using mlns. *European Conference on Computer Vision (ECCV)*, 2008.

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

[90] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision (IJCV)*, 2002.

[91] J. Wang, K. Markert, and M. Everingham. Learning models for object recognition from natural language descriptions. *British Machine Vision Conference (BMVC)*, 2009.

[92] H. Wold. Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, 1966.

[93] H. Wold. Partial least squares. In S. Kotz and N.L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 6, pages 581–591. New York, 1985.

[94] B. Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[95] B. Wu and R. Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[96] B. Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[97] Changjiang Yang, Ramani Duraiswami, and Larry S. Davis. Efficient kernel machines using the improved fast gauss transform. In *Neural Information Processing Systems (NIPS)*, pages 1561–1568, 2004.

[98] Ron Yeh, Chunyuan Liao, Scott Klemmer, François Guimbretière, Brian Lee, Boyko Kakaradov, Jeannie Stamberger, and Andreas Paepcke. Butterflynet: a mobile capture and access system for field biology research. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 571–580, 2006.

[99] Z. Yue, D. Guarino, and R. Chellappa. Moving object verification in airborne video sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 2009.

[100] Lihi Zelnik-manor and Pietro Perona. Self-tuning spectral clustering. *Neural Information Processing Systems (NIPS)*, 2004.

[101] H. Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. *Computer Vision and Pattern Recognition (CVPR)*, 2006.

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

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

[104] T. Zhao and R. Nevatia. Car detection in low resolution aerial images. *Image and Vision Computing*, 2003.

[105] H. Zheng, L. Pan, and L. Li. A morphological neural network approach for vehicle detection from high resolution satellite imagery. *International Conference on Neural Information Processing*, 2006.

[106] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. *Computer Vision and Pattern Recognition (CVPR)*, 2006.