Learning Visual Concepts for Image Classification

by

Aparna Lakshmi Ratan

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1999

Author .....
Department of Electrical Engineering and Computer Science
April 14, 1999

Certified by .....
W.E.L. Grimson
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .....
W.M. Wells
Research Scientist, Artificial Intelligence Lab
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Learning Visual Concepts for Image Classification

by

Aparna Lakshmi Ratan

Submitted to the Department of Electrical Engineering and Computer Science
on April 14, 1999, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

This thesis presents an example-based approach for learning visual concepts that encode the properties of an image class. Extracted concepts are simple relational templates that capture the color and spatial properties of the class. We show that for a range of image classes, simple representations/concepts that encode the properties of a small set of image regions and relations between them can be learned from a small set of examples. We also show that (1) the learned concept often captures the set of salient properties and relations that define the image class and (2) these learned representations are sufficient to detect a variety of image classes efficiently and reliably (with few false positives). Such class detectors can be used for applications such as (a) image database retrieval and (b) object detection and localization.

In this thesis we explore an automatic way to extract "visual concepts" from a few example images which can be used in content-based image retrieval. We adapt the Multiple Instance Learning paradigm as a way of modeling the ambiguity in images and learning "visual concepts" to classify new images. In this framework, a user labels an image as positive if the image contains the concept. Each example image is a bag of instances (sub-images) where only the bag is labeled — not the individual instances (sub-images). From a small collection of positive and negative examples, the system learns the concept and uses it to retrieve images that contain the concept from a large database.

We also describe a matching technique that uses one-dimensional warps of simple templates to detect and localize objects while allowing for clutter and pose variations. The method recovers three of the six degrees of freedom of motion (2 translation, 1 rotation), and accommodates two more degrees of freedom in the search process (1 rotation, 1 translation).

Thesis Supervisor: W.E.L. Grimson
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: W.M. Wells
Title: Research Scientist, Artificial Intelligence Lab

Acknowledgments

I have many people that I'd like to thank for making my experience as a graduate student most enjoyable, stimulating and exciting. I would like to start with Eric Grimson who has been my thesis supervisor and mentor since I started graduate school. I wish to thank Eric for his continual support and encouragement. Eric provided me with excellent advice, new ideas and helped keep me focused on this project while giving me to the freedom to try new things. I have benefitted greatly from his knowledge and experience. William (Sandy) Wells, who co-supervised this project is another important person that I would like to thank. Part of this thesis work came out of a collaborative project with Sandy. Sandy has been a great source of innovative ideas and working with him has been a memorable learning experience for me. I have also learned a lot from discussions with my committee members, Tomas Lozano Perez and Paul Viola. I wish to thank them for their valuable feedback, ideas and suggestions at various stages of this thesis.

Over the years, I have benefitted a lot from discussions with many of my fellow graduate students. I would like to thank Oded Maron in particular for collaborating with me to build the framework that adapted Multiple Instance Learning to the problem of scene classification. The work in Chapter 3 was done jointly with Oded.

Many people have made my stay at the lab very memorable. I have enjoyed interacting with Pam Lipson, Mike Leventon, Chris Stauffer, Gideon Stein, Raquel Romano, Lily Lee, Polina Golland, Liana Lorigo, Tina Kapur, Holly Yanco, Pawan Sinha and Kinh Tieu among others. Special thanks to Pam Lipson whose work on flexible templates for scene classification motivated this thesis, Mike Leventon for his help with TCL/TK and the Hausdorff distance code, Greg Klanderman for providing access to his super-fast image processing library, Jeremy DeBonet and Kinh Tieu for help with the web interface for the scene classification demo and for providing data for comparison experiments and Raquel Romano for providing data for comparing some of the face detection results. Thanks also to Jill Fekete for all her help over the years.

I would like to thank my friends for making these past few years really wonderful. My immeasurable gratitude to my parents for their upbringing and love and for their emphasis on education; to Maithreyi and Aishwarya, my sisters, for just being themselves and there for me at all times. Finally, I would like to thank Sanjay whose unfailing love, encouragement and enthusiasm was crucial in making this possible.

This report describes research done at the Artificial Intelligence Laboratory. This research was sponsored by grants from ARPA under ONR contract N00014-95-1-0600.

To: Nayana and Amma for making this a reality.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Image Classification

This thesis presents an example-based approach for learning visual concepts that encode the properties of a class of images. Extracted concepts are simple relational templates that capture the color and spatial properties of the class. We show that for a range of image classes, simple representations/concepts that encode the properties of a small set of image regions and relations between them can be learned from a small set of examples. We also show that (1) the learned concept often captures the set of salient properties and relations that define the image class, (2) these learned representations are sufficient to detect a variety of image classes efficiently and reliably (with few false positives) and (3) such templates can be matched efficiently to detect and localize objects in new images while accommodating pose variations.

People (even young children) can recognize objects at the level of some basic categories. We can recognize a chair or a car whether or not we have seen that particular one before. Changes in internal configuration and individual variations are largely irrelevant to categorization. For example, we don't have trouble recognizing people despite changes in shape, size, height and color. Philosophers (e.g. Quine [Quine1960] among others) have suggested many theories and psychologists have used several procedures to study how people make classifications and learn concepts. Some of the early psychological studies in categorization were done by Rosch [Rosch1978].

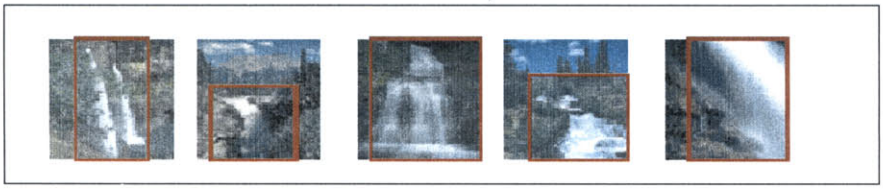


Figure 1-1: Example showing the within-class variability in images. Images in the same class (waterfalls) can vary in their color, texture, scale and illumination

She studied the hierarchical organization of natural categories by using three levels to describe them (super-ordinate, basic and subordinate). Rosch's findings also provide some evidence that the basic-level categories (e.g. car, chair, bird, mouse, etc.) are most differentiated from each other, they are the first categories we learn and that some items are more representative of the category than others.

In order to build machines to perform this task, we need a theory of object recognition that can recognize objects at the level of abstract categories (classes) in addition to recognizing specific individual instances. Although there have been advances in object recognition techniques, current systems are not general enough to detect a variety of objects under a variety of conditions like humans can. We need model representations that capture the commonalities in the class while tolerating in-class variations and have the ability to generalize to new instances of the class. Given the large number of possible classes and the variations that could occur, it might not be practical to build detailed models that capture all the relevant properties and variations for all objects.

The problem of automatically interpreting a large and diverse set of visual images has proved to be extremely challenging for machine vision researchers over the past 20 years. This is difficult because there is a great deal of variability in images. The same object can appear very different in different images due to changes in pose, lighting, occlusion and sensor noise. In addition, to these variations, a good detection system has to deal with variations within the class of objects as well. For example, Figure 1-1 shows how images in the same class can differ in their color, texture, shape and size.

1.2 Flexible templates for image classification

The flexible/qualitative template approach [Sinha1994],[Lipson *et al.*1997] provides a way to capture the within-class variablity in images by encoding the qualitative relations in color and luminance between image patches. The ratio-template constructed by Sinha [Sinha1994] captures the changes in luminance between fixed image patches. These templates have been used successfully to detect faces under varying illumination conditions. Lipson [Lipson *et al.*1997] extended the ratio-templates approach by constructing flexible templates that encode the qualitative color, luminance and spatial relations between image patches that can vary in location. The flexible templates constructed by Lipson have been used successfully to capture concepts like fields, waterfalls and cityscapes for the task of scene classification. Each template is defined as a set of regions with color and texture properties and the spatial and photometric relations between those regions. A template can be deformed to represent a class of images. Lipson pre-defined a number of these templates and showed that they can be used to classify natural scenes (e.g. fields, waterfalls, sunsets, etc.) efficiently and reliably while accommodating variations in absolute size, color and location.

1.3 Thesis

In this thesis, we build on this idea of flexible templates and develop methods to learn these templates automatically from a small set of positive and negative example images and to match such templates efficiently to detect and localize objects in new images while accommodating class variations and pose changes. We discuss how such methods can be used effectively in application domains of natural scene classification and object localization.

1.3.1 Application: Natural Scene Classification

With the recent explosion of digital image and video information due to the World-Wide Web, there is a need for automated techniques to index into these large collec-

tions of images based on content. Currently there exist systems that index into large text databases with keywords that are related to the content of the document. While we can annotate images with keywords and descriptions based on their content, this is impractical because (1) it is time-consuming for a person to go through all the images and annotate them and (2) images are inherently ambiguous and could have many interpretations depending on the user's experiences. For example, given the image in Figure 1-2, it is difficult to say what is relevant about the image in order to retrieve other images that are "visually similar" to it. Do we retrieve images of Mt. McKinley, any snow covered peak, deer, grassy fields or some combination of these objects? It is thus challenging for a classification system to interpret this image even if all the objects in the image can be recognized separately. While we cannot say much about the "visual content" of an image from a single image, we can use a few examples (labeled positive or negative) to make the ambiguity explicit and attempt to find commonalities within the positive examples that do not appear in the negative examples.

It would be useful to have a flexible, automated way to extract "visual concepts" from example images which can then be used to index into these large image databases. Examples of "visual concepts" include "mountains", "cars", "sunsets" etc.

Many existing image classification strategies depend crucially on the representation and features picked. Most current systems require the user to specify salient regions in an image as well as the relevant cues (e.g. color, shape, texture etc.) for a particular query. For example, Figure 1-3 shows the results obtained for a classification task using the Virage system [Bach *et al.*1996] where search is performed using 4 cues (color, texture, composition and structure). These results are not consistent with what we expect for mountains even though the cues chosen seem reasonable. The figure illustrates the importance of choosing relevant cues and the appropriate weights for combining these cues for good classification performance. Thus, for this application, it would be valuable to have the system automatically learn the relevant combination of features from example images.

In this thesis we describe methods to learn "visual concepts" in the form of flexi-

Figure 1-2: Images are inherently ambiguous and can have many interpretations based on the user's experiences. For instance, this image can be interpreted as an example of (a) Mt. McKinley, (b) any snow covered peak, (c) deer, (d) grassy fields or some combination of these objects

ble, relational templates that capture the relative photometric and spatial properties of classes of natural scenes (e.g. snowy mountains, waterfalls, sunsets, lakes) from a few examples. These learned templates be used to classify new images while accommodating within-class variations in color, texture and scale. We also provide a framework to (a) automatically select/learn the relevant features from a few examples at query time, (b) evaluate the effects of user interaction with positive and negative feedback and (c) evaluate the role of positive and negative examples for classification.

1.3.2 Application: Object Localization

Model-based object recognition can be formulated as follows: "Is there an instance of a particular object in this image, and if so where is it?" This is difficult to do because the same object can appear very different in different images due to changes in viewpoint, lighting, occlusion and sensor noise. In addition to these variations, a good detection system has to deal with variations within the class of objects as well.

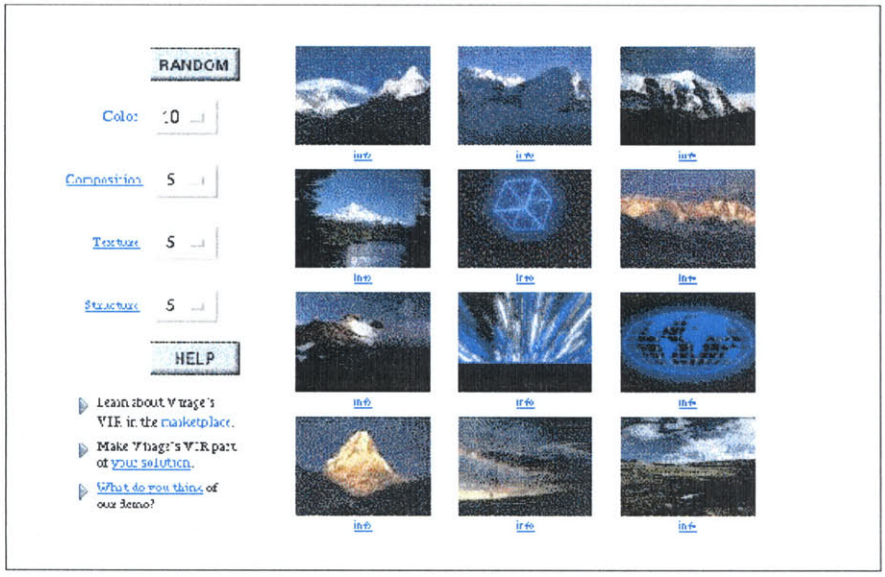


Figure 1-3: Classification results given an example of a mountain scene. This illustrates the importance of picking the right cues together with the right representation of features for that cue.

For example, in face detection, faces of two different individuals in the same pose can appear quite different due to variations in color, texture, shape and size between them.

One way to account for variations in object classes due to changes in pose and lighting conditions is to model them using examples. This approach to training systems to account for variations using examples is called "example-based" learning in the literature. These learning techniques have been used successfully in drug design [Jain *et al.*1994, Dietterich *et al.*1994] and object detection [Sung and Poggio1994, H. Rowley and Kanade1995, Turk and Pentland1991, Murase and Nayar1994] among other applications. Exisiting detection methods model the object as a set of 2D views [Sung and Poggio1994, H. Rowley and Kanade1995] or in terms of their projection into a low-dimensional subspace [Turk and Pentland1991, Murase and Nayar1994, Pentland *et al.*1994] which capture variations in the appearance of the object due to pose changes, lighting changes and within-class changes. At detection time, each of the input patterns are matched against each image location to find the closest match of the input to the image. These detection techniques typically need a large

number of examples to model all these variations and detect the object class efficiently and reliably. Another problem is that the examples typically need to contain the object without any background variations or distractors. It might not be possible to find such examples for every object class. Ideally, we would like to just give the system a *few* examples of images containing the object without having to perfectly segment the object from the background first and let the system (a) pick the relevant regions in the image corresponding to the object (b) represent the object using a set of features that can handle variations within the object class as well as other external variations and (c) match the object model to retrieve new instances of the object while accommodating changes in lighting, pose and noise.

In addition to detecting the presence or absence of an instance of the model object in an image, an object recognition system might need to verify or refute the hypothesis that a particular object exists in the image. Most existing verification methods find the alignment of model with data by minimizing some error criterion [Huttenlocher and Ullman1990, Viola and Wells1995, Ullman and Basri1991]. Some of these methods [Huttenlocher and Ullman1990, Ullman and Basri1991] extract features like points and lines from the modle and data and identify pairings between model and data features that yield a consistent transformation of the model object into image coordinates. If we have a cluttered image, the large number of pairings between model and image features makes this search combinatorially explosive. Other methods [Viola and Wells1995] find the alignment of model with data by locally minimizing some error criterion. An example of this is the Alignment by Maximization of Mutual Information [Viola and Wells1995]. While these techniques register the model with the image very accurately, the problem is that they have narrow capture radii. We need methods for representing and matching object classes that will converge to the optimal solution from inaccurate initial measurements and in the presence of clutter and occlusions.

In this thesis, we introduce a matching and detection technique combines the general detection flavor of patch correlation with the ability to accommodate more degrees of freedom of object motion that are traditionally associated with feature-

based recognition systems. We provide a detection technique that localizes objects in an cluttered images and extracts rough pose information which can then be fed into existing alignment engines for accurate verification. We demonstrate our method on face detection, as faces naturally suit the approach (heads are usually oriented upwards, but appear facing in varying directions). We show performance comparable to some current face detectors, and demonstrate that our method generalizes naturally to other classes of objects, such as vehicles.

1.3.3 Thesis contributions

In this thesis, we address these issues by suggesting methods to build simple detectors for classes of objects. Given a small number of examples, we build "class concepts" which capture the photometric and spatial relations between regions in the object class. We demonstrate the use of these detectors in two domains (1) natural scene classification (2) detection and localization of objects for recognition. We show that

- For a range of image classes, simple concepts/templates that encode the properties of a small set of image regions and relations between them can be learned from a small set of examples.
- The learned concept often captures the set of salient properties and relations that define the object class.
- For certain classes of objects, one dimensional warps in these simple templates can be used to detect and localize objects efficiently while allowing for within class variation and pose changes.

We will describe the primitives used and discuss the process of learning visual representations/templates by building detectors for 3 specific classes (1) natural scenes, (2) faces and (3) cars. We will also describe a matching technique that uses one-dimensional warps of a simple template to detect and localize objects while allowing for clutter and variations in pose. At runtime, these class detectors can be used for

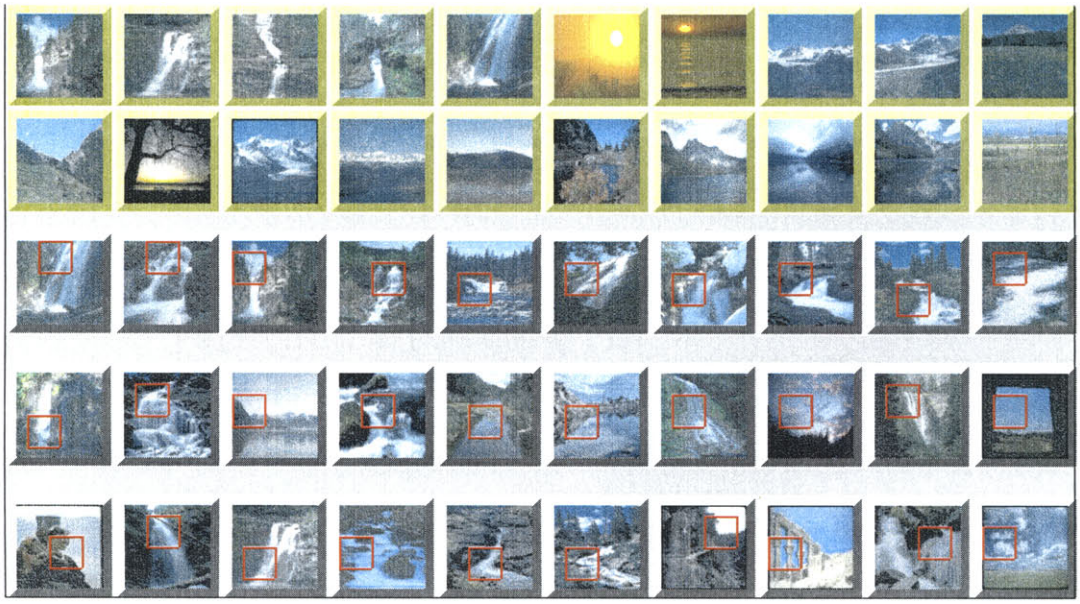


Figure 1-4: Results for the waterfall concept using the `single blob with neighbors` concept. The dataset contained 2600 images of a variety of scenes (e.g. mountains, lakes, sunsets, flowers etc.) from the COREL image library. Top row: Initial training set–5 positive and 5 negative examples. Second Row: Additional false positives. Last three rows: Top 30 matches retrieved from the large test set. The red squares indicate where the closest instance to the learned concept is located.

(a) image classification or image retrieval and (b) detection/pose estimation. We will demonstrate the use of these "object class detectors" in both these domains.

1.3.4 Thesis Overview

The thesis is organized into four chapters which describe methods for (a) capturing flexible, relational templates from examples and using them for classification and (b) matching these templates for localization of objects in images.

In Chapters 2, 3 and 4, we discuss methods for capturing relational templates from a small set of examples using a concrete natural scene classification example. Natural scene classification is an open problem in machine vision and has applications in image and video database indexing. In Chapter 5 we discuss a method for matching templates to detect and localize instances of objects like heads while accommodting more degrees of freedom of motion than existing template-based correlation methods.

In Chapter 2, we describe the features used and the template extraction method using a few positive examples of the scene class. In this chapter, we describe a simple reinforcement scheme for learning the flexible templates that Lipson hand-crafted and show that the learned templates can classify new images effectively. The scheme proposed here is a variation on simple discrete intersection of common color and spatial properties in all the example images. The experiments in this chapter demonstrate proof of concept that this scheme classifies natural scenes efficiently and reliably.

In Chapter 3, we formulate the extraction of scene concepts using color and spatial properties as a multiple instance learning problem. We use a general architecture that employs a method called Diverse Density [Maron1998] to learn scene concepts from a set of positive and negative examples. A key part of our system is the bag generator: a mechanism which takes an image and generates a set of instances where each instance is a possible description of what the image is about. We show that simple learned concepts perform well in this task. We also evaluate the effects of user interaction with positive and negative feedback. Figure 1-4 shows a preview of the results from Chapter 3. As we see in the figure, this system for query learning and classification can be used effectively to classify waterfalls. The system is able to learn the concept from the small set of positive and negative examples automatically without any need for the user to mark out salient regions.

In Chapter 4, we extend the bag generator described in Chapter 3 to generate more complex instances using multiple cues and show that this extension can be used to learn object concepts (e.g. cars). We demonstrate that rough segmentation of high resolution images into salient connected components greatly reduces the number of instances per bag and the running time of the algorithm. Our experiments on "car classification" show that a combination of cues (color, texture and primitive shape), some feature selection and more complicated concepts (conjunctions of instances with spatial relations) play a significant role in improving classifier performance. Figure 1-5 shows a preview of the car detection results. The learned concept is a 2-blob concept and the red squares in the figure indicate the location in the image closest to

Figure 1-5: Results for the cars concept using the `cc2` concept. The database contained 538 images from of a variety of objects and scenes from the COREL library. The training set and the test set were disjoint i.e. the examples used to learn the concept were not included in the test set. Top row: Initial training set–5 positive and 5 negative examples. Positive examples are highlighted in red. Last three rows: Top 30 matches retrieved from the test set. The red squares indicate where the closest instance to the learned concept is located.

the learned concept. The results indicate that the "wheel" regions are closest to the learned "car" concept in the retrieved images.

In Chapter 5, we discuss a method for detection and localization of objects. We develop a new correlation-based method for building detectors for object classes using a single, simple, stretchable 2D template. This detection scheme allows extraction of useful 3D pose information from the template to image match which can then be used to recognize the object accurately using conventional recognition methods. Figure 1-6 shows an example of the detection and localization process.

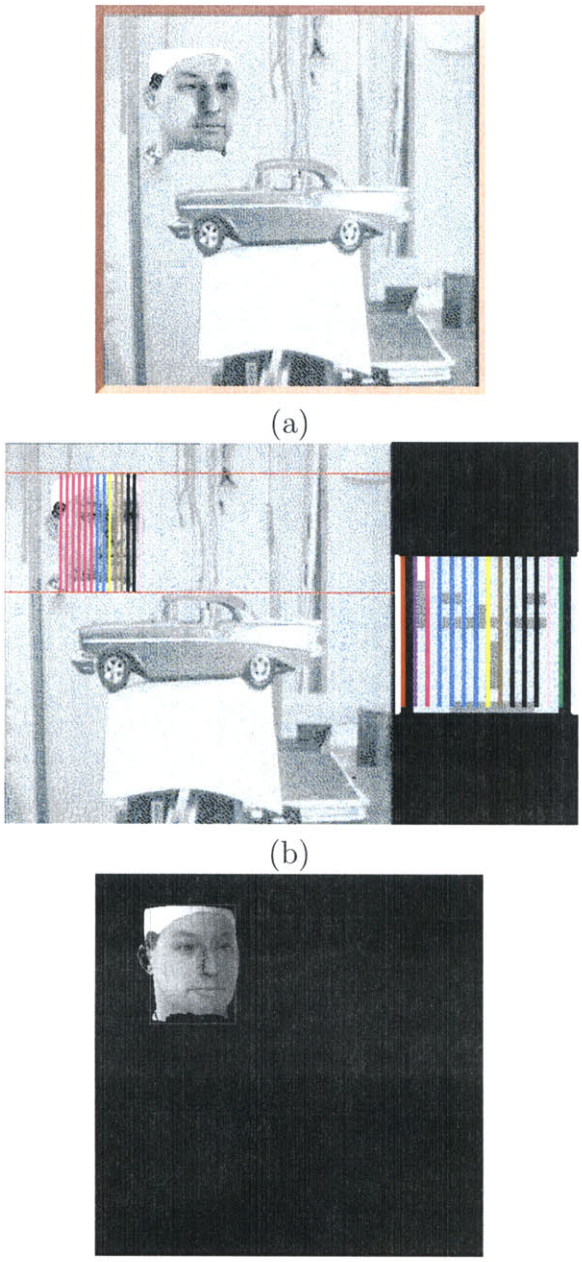


Figure 1-6: (a) Input image (b) Detection results showing the mapping between template features and image features for the detected instance (c) results of the partial pose solver which uses the mapping from (b) to compute 3 DOF pose (x and z translation, y rotation). The model is rendered using the recovered pose. Visual inspection shows that the recovered pose matches the input pose quite well.

Chapter 2

Natural Scene Classification

In the past few years, a new field has emerged for the understanding of natural images. The growing number of digital image and video libraries has led to the need for flexible, automated content-based image retrieval systems which can efficiently retrieve images from a database that are similar to a user's query. Because what a user wants can vary greatly, we also want to provide a way for the user to explore and refine the query by letting the system bring up examples. For example, given a picture of a snowy mountain scene, what is it about the image that causes it to be labeled as a snowy mountain. Is it the sky, the mountain, the snow or the house in the corner? It is hard to determine this by looking at only one image. The best we can say is that at least one of these constituent features or some combination of them makes this image a snowy mountain scene. However, given a few examples, we can extract those features that are common in all the examples as the ones that cause them to be labeled as snowy-mountain scenes. In this chapter and the next, we suggest methods to learn concepts that capture the relevant properties (features) of classes of natural scenes from a few examples.

2.1 Previous Work

In this section, we review some of the existing techniques for content-based image retrieval.

2.1.1 Histogramming methods to capture color distributions

One of the most popular global techniques for indexing is color-histogramming which measures the overall distribution of colors in the image [Swain and Ballard1991]. While histograms are useful because they are relatively insensitive to position and orientation changes, they do not capture the spatial relationships of color regions and thus have limited discriminating power. Many of the existing image-querying systems work on entire images or in user-specified regions by using distribution of color, texture and structural properties. The QBIC system [Flickner *et al.*1995] and the Virage system [Bach *et al.*1996] are examples of such systems. Figure 2-1 shows an example of images retrieved using color histogram matching. The results in the figure show the top few matches that are closest in color composition to the first image.

Examples of some recent systems that try to incorporate some spatial information into their color feature sets are [Smith and Chang1996, Huang *et al.*1997, Belongie *et al.*1998, Pass *et al.*1996, Stricker and Dimai1996, Hsu *et al.*1995] among others. Pass et al. [Pass *et al.*1996] have a technique that is better than color histogramming by splitting the global histogram into coherent and scattered components which identify connected color regions. However, it does not support querying with spatial locations of the color regions. Stricker et al. [Stricker and Dimai1996] divide the image into five overlapping regions and compute the first three moments of the color distributions in each region. They allow querying by using the five absolute region locations and properties and the method is insensitive to small rotations. VISUALSeek [Smith and Chang1996] is another system that supports query by color, texture and spatial layout. It also partitions the image into regions, but allows a region to contain multiple colors and lets pixel belong to several different categories.

Promising work by Rubner [Rubner *et al.*1998] using the Earth Mover's Distance provides a metric that overcomes the binning problems of existing definitions of distribution distances for indexing.

All of the methods described above use the absolute color and texture properties of the whole image or fixed regions in the image and some incorporate spatial relations

Figure 2-1: This figure shows the set of images retrieved by the QBIC system [Flickner *et al.*1995] using global color histogramming. These images are the top few matches that are closest in color composition to the query (first image). The results illustrate that images with very different content can have similar color composition.

between connected regions having similar color properties. If we consider the domain of natural scene classification, these absolute properties can vary between images of the same class.

These methods also use a single image as the query to the search engine. As we mentioned earlier, it is hard to extract the relevant properties of a scene from just one image and this results in false positives from matching the irrelevant parts of the image. Most of these techniques overcome this issue by requiring the user to specify the salient regions in the query image. One of the goals of our system is to use a small set of examples to automatically learn (a) the relevant regions in the image that best describe the example class and (b) the color and spatial properties of these regions in order to classify natural scenes efficiently and reliably (with few false positives).

2.1.2 Flexible templates to capture color and spatial layout

More recently, work by Lipson [Lipson1996, Lipson *et al.*1997] illustrates that predefined flexible templates that describe the relative color and spatial properties in the image can be used effectively for scene classification. Figure 2-2 shows an example of a template that captures the "snowy-mountain" concept. The flexible templates constructed by Lipson [Lipson *et al.*1997] encode the scene classes as a set of image patches and qualitative relationships between those patches. This is similar to the ratio-template constructed by Sinha [Sinha1994] to detect faces under varying illumination conditions. The ratio template captures the changes in luminance between fixed image patches.

The flexible templates constructed by Lipson for the task of scene classification include spatial and photometric relationships between image patches. Each image patch has color and luminance properties, and the templates describe the relationships between the patches in terms of color (relative changes in the R,G,B channels), luminance (relative changes in the luminance channel) and spatial layout. Lipson hand-crafted these flexible templates for a variety of scene classes and showed that they could be used to classify natural scenes efficiently and reliably. In this section, we would like to create an interactive system that allows the user to build his own templates by selecting a set of example images and letting the system extract a set of templates that capture the common relations within the set of images.

Example of a predefined template

Figure 2-2 is an example of a predefined template used to classify scenes (Lipson [Lipson *et al.*1997]). The template consists of a set of regions, with a set of spatial relationships (above, below, left, right). For example, two regions A and B could have the spatial relation A is above B. The template also describes the color and luminance relationship between the image regions. The relations used for the color (R, G, B) and luminance channels are $(>, <, =)$. For example, two regions A and B could be related by color relations such as $Ar > Br, Ag < Bg, Ab = Bb$. These hand-

built templates capture the relevant color and spatial relations of classes of natural images and have been used to classify scenes efficiently.

These templates are matched to candidate images by measuring the extent to which a similar representation of an image meets the relative relationships of the template, and the extent to which patches in the image template must be spatially altered to best match these other constraints. The deformation of a template is defined in terms of

- Inter-region spatial layout which measures the extent to which the template regions need to be moved to match the image,
- Intra-region spatial layout which measures the extent to which the boundary of the template needs to be stretched to fit the image,
- Inter-region photometry which measures the extent to which the photometric properties between template regions must be varied to match the image and
- Intra-region photometry which measures the extent to which the absolute photometric values in a region of the template needs to be altered to fit the image.

More details on this matching algorithm can be found in [Lipson1996].

2.1.3 Complex features to capture image patterns

Texture properties have been used successfully in segmentation and discrimination. Some examples include [J.Malik and P.Perona1990, H.Greenspan1994, W.Forstner1994, J.Garding and T.Lindeberg1996]. Texture properties have also been used in object recognition [Rao and Ballard1995, Viola1996]. These texture properties are commonly described as responses to a set of oriented filters and many existing image retrieval systems use the distribution of these texture properties (texture histograms) in addition to color to discriminate between images. Some examples of such systems include [Manjunath and W.Y.Ma1991, J.Malik and P.Perona1990, Belongie *et al.*1998, Pentland *et al.*1996, Minka and Picard1996]. Salesin et al. [Jacobs *et al.*1995] use a wavelet based approach for indexing into image databases. They use the information

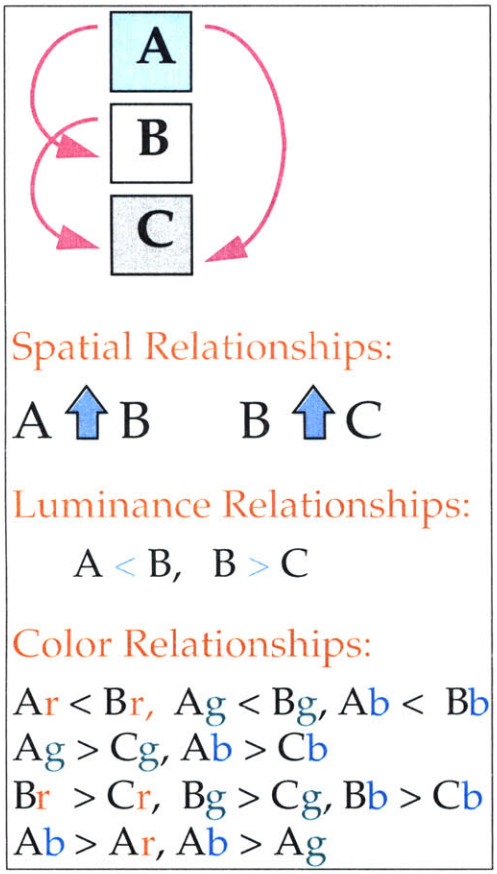


Figure 2-2: Example of a pre-defined template. This figure is taken from [Lipson *et al.*1997] with permission of the author.

encoded in the dominant wavelet coefficients to obtain a signature which can then be used to retrieve images that are visually similar to the query. This algorithm pre-processes all the images in the database by transforming them using the Haar wavelet decomposition in each of the color channels. For each image, the average color and the indices of the top 40 largest wavelet coefficients are stored. The same wavelet transform is applied to the query image and the indices of the top 40 coefficients are retained. The query is a single image or a sketch and the algorithm retrieves images that are similar in structure to the query image. This algorithm is very sensitive to small rotations (5 degrees). and small translations (5 pixels) when the query images have a reasonable amount of detail. Experiments analyzing this algorithm can be found in [Ratan and Dang1996]. Using just the top 40 coefficients is probably insufficient for queries in a large database since they could be discarding many of the significant coefficients in the process. While the idea of using wavelet coefficients and other filter outputs for representing the structure of the image is a good one, what we need is a good way to extract the relevant coefficients for an image.

DeBonet et al. [J.S.DeBonet and P.Viola1997] introduce a method which extracts thousands of complex features that represent color, edge orientation and other local properties at several resolutions. Their approach uses both the first order properties (oriented edges etc.) and the relationships between the first order properties. The process of finding relationships between features with specific local properties leads to complex features which are more discriminating and more specific. They show that these complex features can be used effectively to retrieve images that are structurally similar to the query (e.g. planes,cathedrals etc.).

2.1.4 Learning for image database indexing

Most of the systems described in the previous sections require users to specify precisely the desired query. Minka and Picard [Minka and Picard1996] introduced a learning component in their system by using positive and negative examples which let the system choose image groupings within and across images based on color and texture cues; however, their system requires the user to label various parts of the

scene, whereas our system only gets a label for the entire image and automatically extracts the relevant parts of the scene. Forsyth et al. [D. Forsyth1997] learned representations for horses by training the system using the appropriate color, texture and edge configuration. Cox et al. [Cox *et al.*1998] modeled relevance feedback in a Bayesian framework that uses an explicit model of the user's selection process. More recently, Nastar et al. [C.Nastar1998] introduced a relevance feedback system for query refinement over time using a set of positive and negative images. They estimate the distribution of relevant images and minimize the probability of retrieving non-relevant images. Our system uses the Multiple Instance learning framework (which we will describe in the next chapter) to learn a query concept and feature weights automatically from a small set of positive and negative examples. Our system finds the simplest concept (template) that can be used to explain the query set and retrieves similar images from the database by finding the location of the concept in the image.

2.2 Our Natural Scene Classification System

In this chapter and the next, we describe our approach to learning queries of natural scenes like sunsets, waterfalls, fields etc. Natural scene classification is a difficult problem for computer vision since classes of natural scenes can vary significantly in viewpoint, scale, brightness and color. Figure 2-3 shows an example of the color and spatial variations that can occur in the class of waterfall scenes.

In order to classify new images, we need methods to compare the query image/images to other images in the database. However, correlation cannot be used directly on the high-resolution images since even a small shift of a few pixels can cause large changes in the correlation coefficients. Correlation on smoothed and down-sampled images is less sensitive to translation and scale variations between images but works well when there is a single object in the image without any background clutter. In more complex images, there are often multiple objects and the object of interest occupies only a small part of the image. In this case, we need to compare

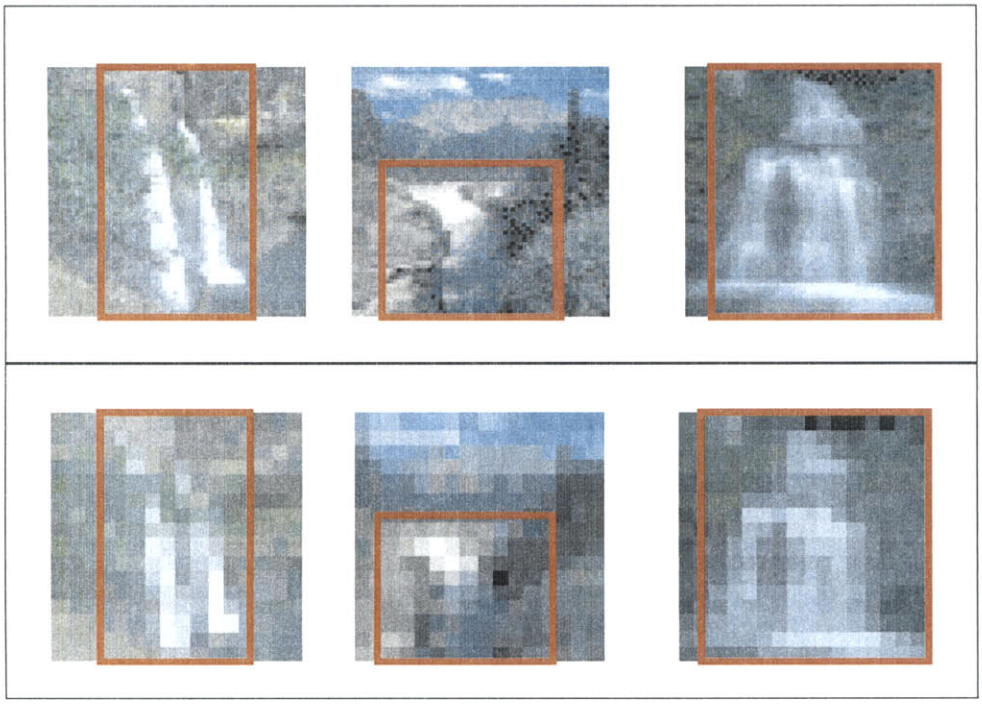


Figure 2-3: Example showing the variations that can occur in the class of waterfalls. The waterfalls vary in size, color and brightness values. These variations are preserved even in smoothed and down-sampled images. Since the waterfall forms only a small part of the image in some examples, correlation on the whole image gives poor results.

sub-regions in one image with sub-regions in the other image in order to classify them. Now we are left with the problem of choosing the relevant sub-regions in a particular image. As we mentioned earlier, it is difficult is say which regions are relevant from a single example. If we have a few examples, however, we can pick the common regions in all the examples as being relevant to the query concept. In our approach, we consider all possible sub-images in an image and pick those that are common in all the examples.

This still leaves the problem of how to compare the sub-images: by directly using correlation on the pixels or by extracting other features. As Figure 2-4 shows, the absolute color in classes of natural scenes varies significantly. For example, we see in the figure that the absolute color for the class of fields varies from green to yellow to brown. These variations are preserved in the smoothed and down-sampled images even when the colors are quantized into a few bins (3 bins in each of the R,G,B channels). Direct correlation on these sub-images even after smoothing and down-sampling will give poor results. We need methods that can generalize well by accommodating these variations. However, while the absolute color varies in the images of the fields, the relations between the regions (sky and field regions) are preserved in all the examples (Figure 2-4). The features used in our system are based on the color and spatial relations between image regions. The next few sections describe how we extract the set features from images, pick the key features that describe the query class from a set of examples and use these key features to classify new images.

The query learning system described here was motivated by Lipson's pre-defined flexible templates [Lipson *et al.*1997] described in section 2.1.2 which captured the color and spatial layout of the image which could then be used for classification. The flexible templates constructed by Lipson [Lipson *et al.*1997] encode the scene classes as a set of image regions and the qualitative relationships between those regions.

Our goal is to automatically construct such flexible templates that capture the common color and luminance patterns and their spatial relationships present in a small set of positive example images.

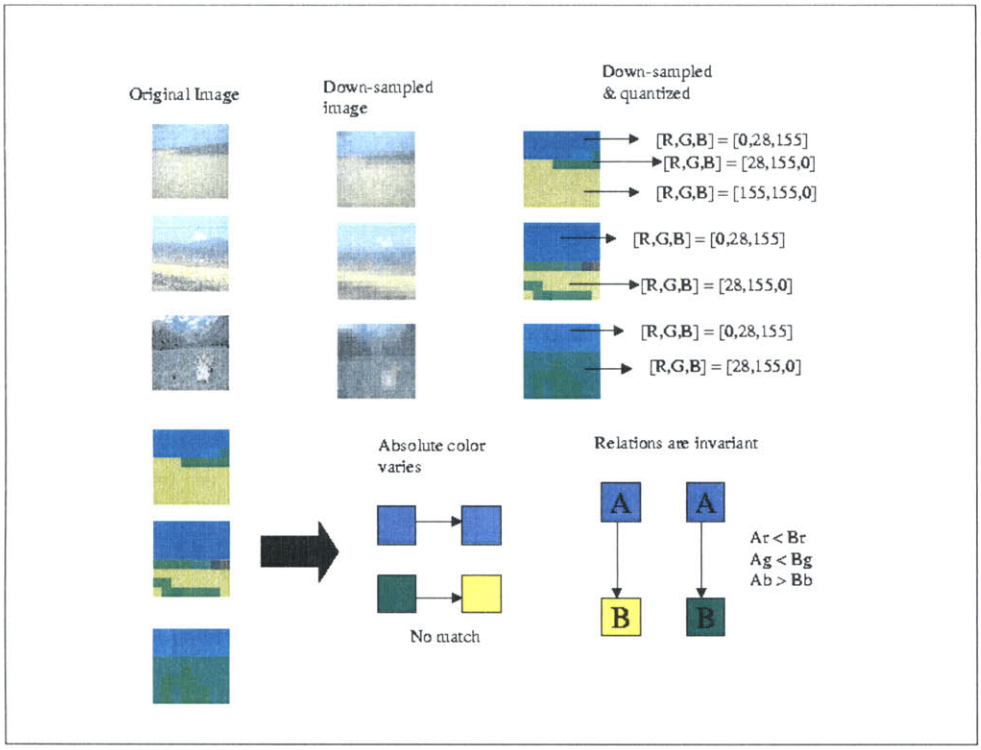


Figure 2-4: This example illustrates the differences in absolute color in the class of fields. The example also shows that the color and spatial relations are invariant across the set of images even when the absolute color and spatial properties change.

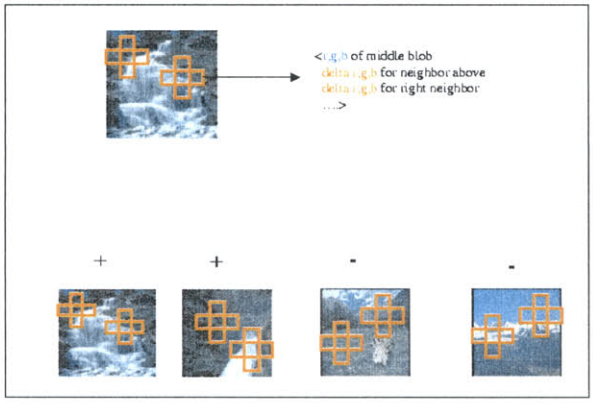


Figure 2-5: Framework for generating feature nodes.

2.2.1 Extracting templates using reinforcement of color relations

The method we use to construct these flexible templates from a training set of images consists of the following steps:

1. extracting features from example images
2. extracting template units from features
3. building templates
4. classifying new images (template matching)

We now introduce some terminology and the general framework for extracting features from an image. In this framework, each example image is divided into many sub-images (blobs) which capture the color relationships within pixel neighborhoods. Each blob is a *node*. As shown in Figure 2-5, we transform images into a set of feature *nodes*. We experimented with a couple of hypothesis classes which we will describe in the next few sections.

2.3 Hypothesis Classes

Since we want to capture the commonalities in a set of images while allowing for spatial and color variations, we cannot just use correlation on the smoothed and down-sampled images as we discussed above. Instead, we extract features (pixel and its neighbors) that are reinforced in all the example images and build a template from these features by linking them with appropriate color and spatial relations. We experimented with two different hypothesis classes of features which are given below. All the images were smoothed using a Gaussian filter and down-sampled to 16x16 images. We use the $[R, G, B]$ color space to represent color values. There are other color spaces that have been explored in the literature like HSV (hue,saturation,value) in [Mahmood1993, Das *et al.*1997] and YIQ ("W-Bk", "magenta-green", "R-cyan") in [Jacobs *et al.*1995] among others. Details on the different color spaces can be found in [Foley and Dam1984]. In our experiments, we did not experiment with other color spaces since we found that the R, G, B color space gave satisfactory results. The luminance (brightness) is can be expressed as a weighted sum of the R, G, B values ($L = 0.30R + 0.59G + 0.11B$). In all of the images, the $[R, G, B]$ values for each pixel are quantized into a few bins. Figure 2-6 shows some examples of smoothed, down-sampled and quantized images of natural scenes.

2.3.1 `single blob with mean color`

- Each 2x2 pixel neighborhood P_i in the low resolution (16x16) image is now a unit represented with (a) its mean color and luminance value and (b) the color and luminance difference between P_i and its neighboring 2x2 regions in the 4 directions. Figure 2-7 shows an example of this feature class.

- P_i is described as a vector $[x_1, x_2, \ldots, x_{20}]$, where x_1, x_2, x_3, x_4 are the mean RGB and luminance values of the central blob, x_5, x_6, x_7, x_8 are the differences in mean RGB and luminance values between the central blob and the blob above it, etc.

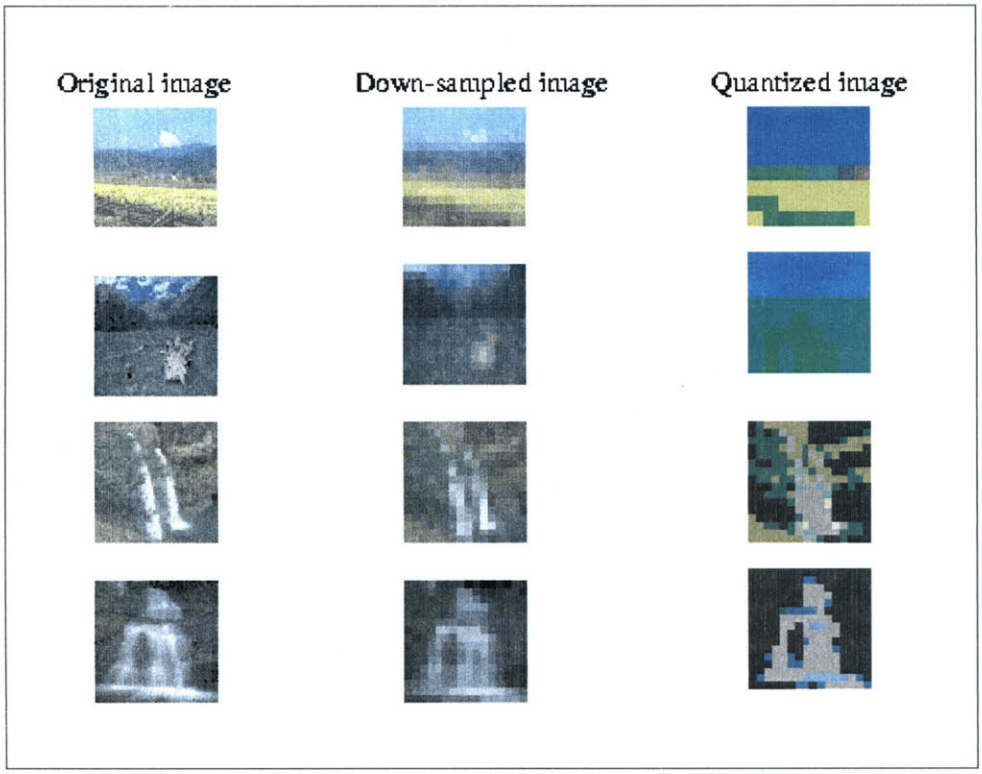


Figure 2-6: Examples of smoothed, down-sampled and quantized images of some natural scenes.

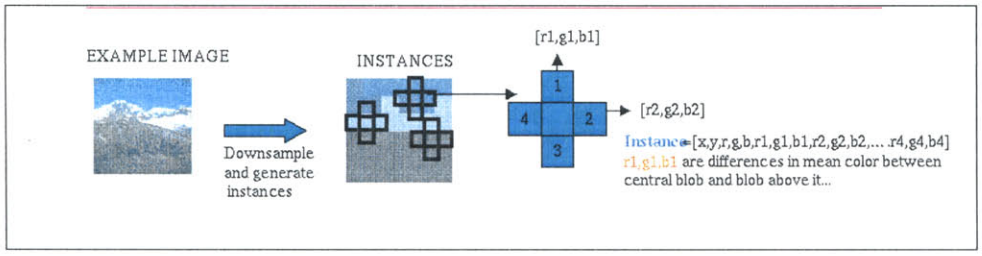


Figure 2-7: Instances of `single blob with mean color` class

- The distance between two instances P_i and P_j is given by

$$\| P_i - P_j \|^2 = \sum_k (P_{ik} - P_{jk})^2 \tag{2.1}$$

where P_{ik} is the kth feature in the instance P_i.

2.3.2 `single blob with relational histograms`

- We smooth the images with a Gaussian filter and down-sample them to low-resolution (16x16) images.
- We then quantize all the images as described above.
- Each 2x2 pixel neighborhood P_i in the low resolution image is now a unit represented with (1) a rough color $P_i(col)$ and (2) histograms of luminance and color relationships in 2 directions (below it and to the right of it) $P_i(hist_d), d = 1, 2$ (Figure 2-8).
- Histograms of color and luminance relationships:

  The relationships are defined on k channels for every pixel (e.g. red, green and blue color channels, luminance, texture, etc.) and there are 3 choices of symbols for a relationship between a pair of pixels (>, <, =). This implies that there are 3^k bins into which a pair of pixels can cast a vote.

  For each pixel in the low resolution image, take a neighborhood of pixels in 4 directions relative to it. For example, in the North-South (NS) direction, the

neighborhood for a pixel $P(i,j)$ is given by the set of pixels $P(i+k,j+l)$ where $k = -1,0,1$ and $l = 1...6$. For each pixel, record the signature of the color and luminance relations between the key pixel and pixels in the neighborhood corresponding to a particular direction (i.e. the set of relationship symbols as defined above). Histogram these signatures into bins. Each bin in the histogram represents a particular relationship defined on the red, blue, green and luminance channels using the relations $(>,=,<)$. For each pixel, build these histograms for different directions separately. We ignore the bin which represents the equality relation i.e. uniform regions in the image do not contribute to the histogram peaks. Thus:

- Distance between nodes P_i and P_j is measured by the

$$D(P_i,P_j) = \texttt{CDiff} + \sum_{k=0}^{2} \texttt{HistDiff}$$

where

$$\texttt{CDiff} = ((P_i(R) - P_j(R))^2 + (P_i(G) - P_j(G))^2 + (P_i(B) - P_j(B))^2)$$

$$\texttt{HistDiff} = \sum_{k=0}^{bins} (P_i(\texttt{hist}_k) - P_j(\texttt{hist}_k))^2$$

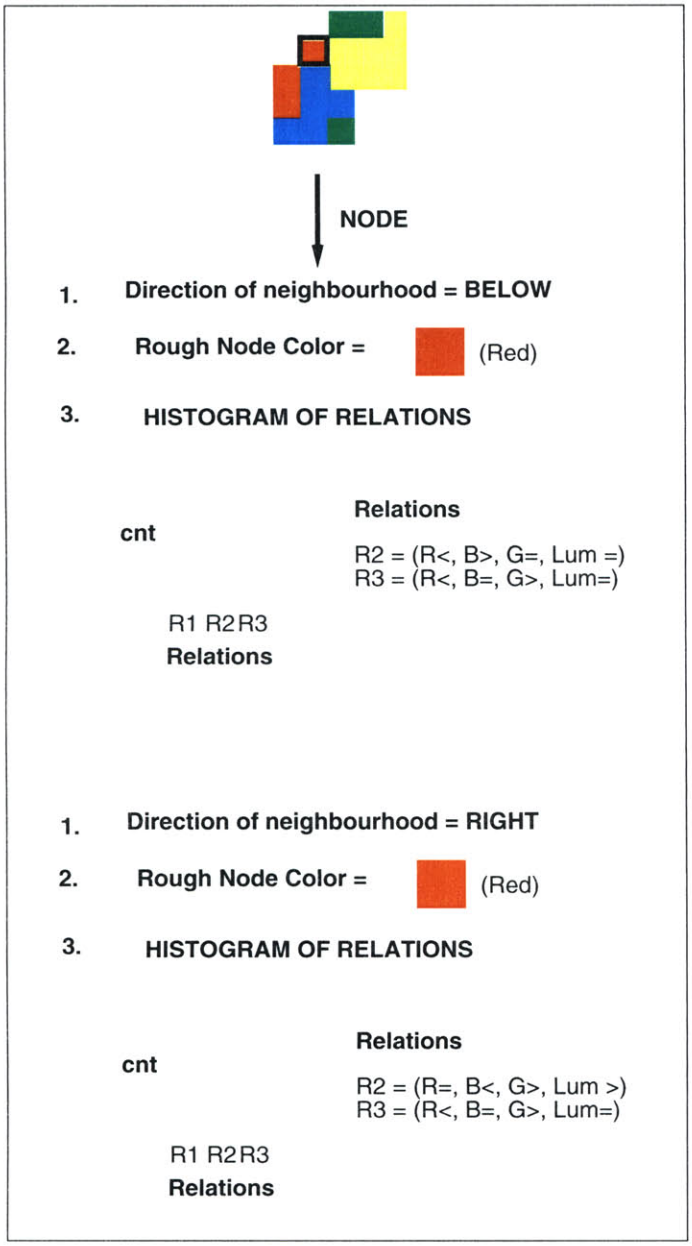


Figure 2-8: A single node in `single blob with relational histograms` hypothesis class. The blob *node* has a (1) rough node color and (2) a histogram defining the relations between the blob and other pixels in the neighborhoods below it and to the right of it.

2.4 Extracting the template (largest matching sub-graph)

Each example image can be viewed as a graph with $16*16$ *nodes* and pairwise spatial relations between those nodes. The extracted template is then the largest sub-graph which has consistent color properties and pairwise spatial relations across the set of examples. Each sub-graph consists of a set of nodes with color properties and a set of edges with the spatial relations between pairs of nodes. The largest matching sub-graph is found as follows:

1. Find all the *distinct* features (nodes) in the first image that have matches in *all* of the example images. The nodes in the examples are compared using their color properties given above in section 2.3. This is equivalent to doing a discrete intersection of the nodes to get a set of nodes N_t whose color and luminance properties are reinforced in all of the n example images $I_k, k = 1..n$

2. If there is just one matched node then this node is the template.

3. Prune the nodes in N_t such that

   - For every pair of nodes $(u, v) \in N_t$ such that $u \in N_t$ matches $u'_k \in I_k$ and $v \in N_t$ matches $v'_k \in I_k$, the spatial relation between u and v in preserved in the corresponding pair of nodes u'_k and v'_k for $k = 1...n$ (i.e. the spatial relationship between u and v in N_t is preserved in the matching pair of nodes in each of the example images).

4. The model template now consists of the largest matching sub-graph in all the example images and has color, luminance and spatial properties that can be used to describe all of the positive examples (Figures 2-9,2-10). The template has a set of nodes N_t and a set of edges E_t such that

   - Every node in N_t has color and luminance properties that are reinforced in all of the example images

- Every pair of nodes in N_t have consistent color and spatial relations in all of the example images. The relations between every pair of nodes in N_t form the edges in E_t.

2.5 Model Graph

The template is represented as a graph T_G where each node in the intersection set N_t forms a node in the graph and pairs of nodes are connected with edges that describe the color and spatial relations between the two nodes.

$$T_G = [N_t, E_t, C_t, RC_t, RL_t, RS_t]$$

where

- N_t gives the set of nodes in the graph (these nodes are reinforced in all the positive examples as described in section 2.4)
- E_t gives the set of edges in the graph (all pairs of nodes that have color and spatial relations)
- $C_t(u)$ gives the color (R, G, B) and neighborhood color relations of a node $u \in N_t$ as described by the hypothesis-class.
- $RC_t(u, v)$ gives the color differences between two nodes u and v where $(u, v) \in E_t$
- $RL_t(u, v)$ gives the distance between the (x, y) coordinates of two nodes u and v where $(u, v) \in E_t$
- $RS_t(u, v)$ gives the spatial relation between u and v where $(u, v) \in E_t$.

$$RS_t(u, v) = \begin{cases} 1 & \text{if } v \text{ is to the left of } u \\ 2 & \text{if } v \text{ is to the right of } u \\ 3 & \text{if } v \text{ is above } u \\ 4 & \text{if } v \text{ is below } u \end{cases}$$

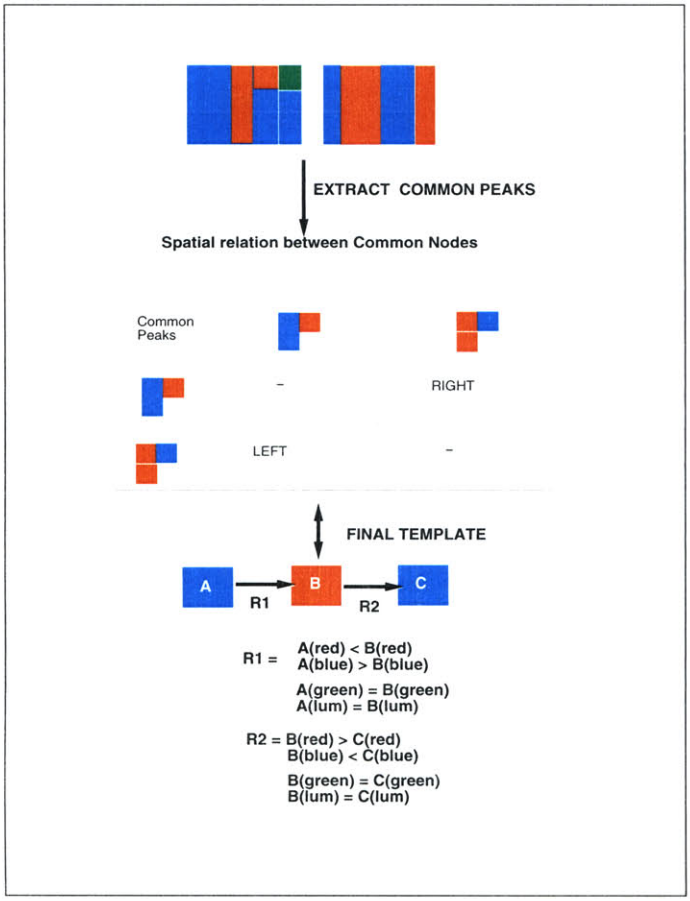


Figure 2-9: Example of a template extraction process for the `single blob with relational histogram` class. Note that the patches can vary in size and shape. Patches that are not common in both images do not appear in the template.

Figure 2-9 and Figure 2-10 illustrates this process for the two hypothesis classes described above. Graph representations have been used successfully in model-based object recognition before in [Mahmood1993] among others. In [Mahmood1993], the model color region information is represented as a region adjacency graph (M_G) and matched to similar region adjacency graphs extracted from the image regions I_G at recognition time. The model is located in the image by finding subgraphs in the image that satisfy the model description. M_G forms a sub-graph of I_G if an instance of the model is in the image. [Mahmood1993]. Graph representations have also been used successfully in image database retrieval as can be seen in [Lipson1996, Das *et al.*1997] among others. In [Lipson1996], Lipson matches a model (flexible template as shown in

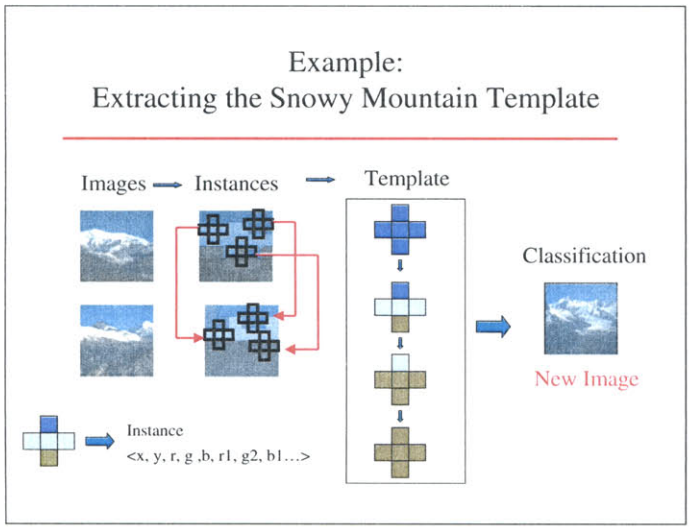


Figure 2-10: Example of the template extraction process for the `single blob with mean color` class on the "snowy mountain" query.

section 2.1.2) with color and spatial relations to subgraphs extracted from the image which satisfy the absolute color properties of the model nodes, the spatial relations between pairs of model nodes and the color and luminance relations between pairs of model nodes.

2.6 Classifying new images: The template matching process

We match the template against all the images in the database and rank the new images based on how well they are explained by the template. This can be viewed as a graph matching problem where the nodes are the color properties and edges between nodes are the color and spatial relations between two nodes. More formally, this graph matching problem between a model graph and an image graph is an instance of the subgraph isomorphism problem. The subgraph isomorphism problem (finding the largest matching subgraph) can be formally defined as follows: given a graph $G = (V_1, E_1)$ with V nodes and E edges and a graph $H = (V_2, E_2)$, does G contain a subgraph isomorphic to H i.e. a subset $V \subseteq V_1$ and $E \subseteq E_1$ such that $\| V \| = \|$

$V_2 \parallel$, $\parallel E \parallel = \parallel E_2 \parallel$ and there exists a one-to-one function $f : V_2 \longrightarrow V$ satisfying $u, v \in E_2$ if and only if $f(u), f(v) \in E$. This problem is known to be NP-complete [Corman *et al.*]. If G has m nodes and H has n nodes, then the number of subgraphs that need to be hypothesized in order to find the isomorphism is $\begin{pmatrix} m \\ n \end{pmatrix} n!$ which grows exponentially [Corman *et al.*]. In our example, if the template graph has 5 nodes and the image graph has 100 nodes, the number of subgraphs that have to be hypothesized is approximately 100^5 which is computationally impractical. However, as will discuss in the next section, if the nodes have to satisfy certain constraints, we can significantly reduce the number of feasible subgraphs that need to be matched. In our system, we use low resolution images with a few nodes and additional constraints (color and spatial constraints) to prune the search space.

A new image is classified by searching for sub-graphs in it that satisfy the model description. We will describe the matching process in more detail below.

Given a new image, convert it into a bag of nodes as described in section 2.3. The image is represented as a graph $I_G = [N_i, E_i, C_i, RC_i, RL_i, RS_i]$. N_i is the set of nodes, E_i is the set of all edges connecting every pair of nodes, $C_i(u)$ is the color description of node $u \in N_i$ as specified by the hypothesis class, $RC_i(u, v)$ is the color difference in the [RGB] channels between u and v, $RL_i(u, v)$ is the distance between the coordinates of u and v and $RS_i(u, v)$ is the relative spatial relation between u and v as given above in the model graph description.

2.6.1 Constraints

1. Node color constraint: For each $u \in N_t$ in the template graph T_G, find the matching nodes $u' \in N_i$ in the image such that $C_t(u) = C_i(u')$. $C_t(u)$ and $C_i(u')$ are matched using the similarity measures given in section 2.3 depending on their hypothesis class.

2. Spatial relations constraint: For each pair of nodes $(u, v) \in E_t$ in the template graph T_G, find matching pairs of nodes in the image such that the spatial relation between the pair of nodes is maintained i.e. $RS_t(u, v) = RS_i(u', v')$.

2.6.2 Obtaining image subgraphs

If the template graph has just one node, find the node in the image graph that best satisfies the node color constraint.

If the template graph has more than one node, enumerate all subgraphs from the image graph and pick the ones that satisfy the color and spatial constraints given above. Find $I_k = [N_k, E_k, C_k, RC_k, RL_k, RS_k]$ such that

- $\| N_k \| \leq \| N_t \|$, $\| E_k \| \leq \| E_t \|$ and
- $\forall (u', v') \in N_k, C_t(u) = C_k(u'), C_t(v) = C_k(v'), RS_t(u, v) = RS_k(u', v')$

This step prunes the number of subgraphs to be considered by the matching step by only considering those subgraphs that satisfy the color and spatial constraints.

2.6.3 Subgraph Matching

Out of the set of image subgraphs $I_k = [N_k, E_k, C_k, RC_k, RL_k, RS_k]$ that satisfy the contraints given above, we want to find the one that best matches the template graph. To do this we match each candidate subgraph I_k in the image with the template graph T_G to find the one that minimizes the score metric S given below. The best subgraph gives the location of the extracted template (concept) in the new image.

$$S(T_G, I_k) = (1 - \frac{\| N_k \|}{\| N_t \|}) + \frac{\sum_{(u,v) \in E_t : C_t(u) = C_k(u') and C_t(v) = C_k(v')} (\Delta C_u + \Delta C_v + \Delta RL + \Delta RC)}{\| E_t \|}$$

where

$\Delta C_u = (C_t(u) - C_k(u'))^2$

$\Delta C_v = (C_t(v) - C_k(v'))^2$

$\Delta RL = (RL_t(u, v) - RL_k(u', v'))^2$

$\Delta RC = (RC_t(u, v) - RC_k(u', v'))^2$

This best (lowest) score is obtained when the number of nodes in the template is equal to the number of nodes in the subgraph and the color and spatial relations between the template regions are preserved well in the image subgraph.

We reduce the search space in the subgraph matching process since(a) there are a small number of nodes in the low resolution images and (b) the number of feasible subgraphs in the image is reduced significantly after applying the two constraints.

2.7 Experiments and Results

We have built an interactive classification system which allows the user to pick a few examples from a random set of images (top two rows of Figure 2-16) from the database. The system extracts the dominant luminance and color relations between image patches and their spatial relations from the query images and searches the database for other images that have the same relationships. The system returns an ordered set of images that belong the query-class. The dataset consists of images of natural scenes like "sunsets", "lakes and rivers", "snowy mountains", "fields", "coasts" etc. from the COREL[1] libraries. All the images are smoothed using a Gaussian filter and down-sampled to 16x16 images. All the templates were built using coarsely quantized 16x16 images.

2.7.1 Experimental setup

We tried to learn three different concepts: waterfall, mountain, and field. For training and testing we used natural images from the COREL library, and the labels given by COREL. These included 100 images from each of the following classes: waterfalls, fields, mountains, sunsets and lakes. We also used a larger test set of 2600 natural images from various classes.

We created a *potential training set* that consisted of 20 randomly chosen images from each of the five classes mentioned above. This left us with a *small test set* consisting of the remaining 80 images from each of the five classes. We separated the potential training set from the testing set to insure that results of using various

[1]We would like to thank the Corel©whose preview thumbnails, comprise our image database. These 120x80 preview thumbnails were acquired from Corel©in accordance with their Image and Pricing Specifications.

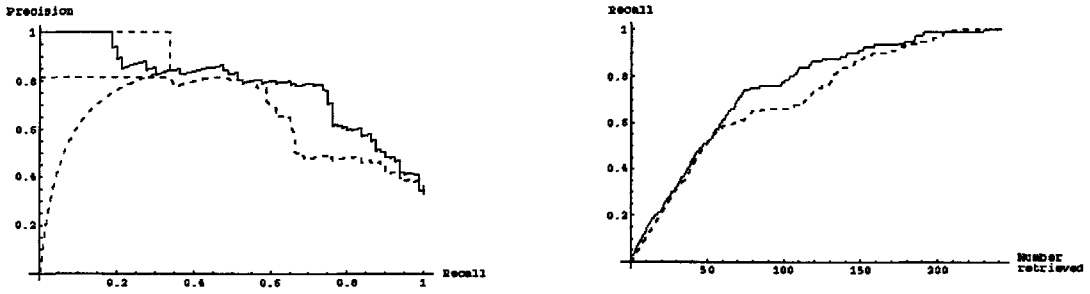


Figure 2-11: Comparison of learned concept (solid curves) with hand-crafted templates (dashed curves) for the mountain concept on 240 images from the small test set. The top and bottom dashed precision-recall curves indicate the best-case and worst-case curves for the first 32 images retrieved by the hand-crafted template which all have the same score.

training schemes and hypothesis classes can be compared fairly. Finally the *large test set* contained 2600 natural images from a large variety of classes.

2.7.2 Proof of concept

In order to test the training procedure, we performed the following experiments to compare its templates with hand-built templates. We took a dataset of 240 images and compared the performance on the "mountain" concept. There were 80 mountain images in the dataset and the images used to build the template were not included in the test set. We used precision-recall and recall curves to compare the methods. Precision is the ratio of the number of correct images to the number of images seen so far. Recall is the ratio of the number of correct images to the total number of correct images in the test set. The graphs in Figure 2-11 shows that this method is competitive with the hand-crafted templates. The false positives for the mountains in both cases included pictures of fields with clouds in the sky.

2.7.3 Results using trained templates

We compared the performance of the trained templates with global histogramming on the small test set of 400 images. The global histogram is computed for each of

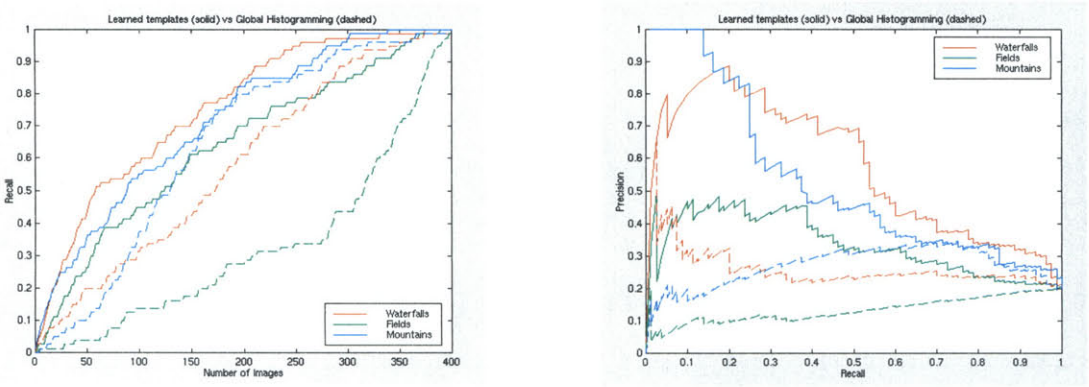


Figure 2-12: Comparison of simple trained templates averaged over the hypothesis classes (solid curves) with global histogramming method (dashed curves) for 3 concepts on 400 images from the small test set.

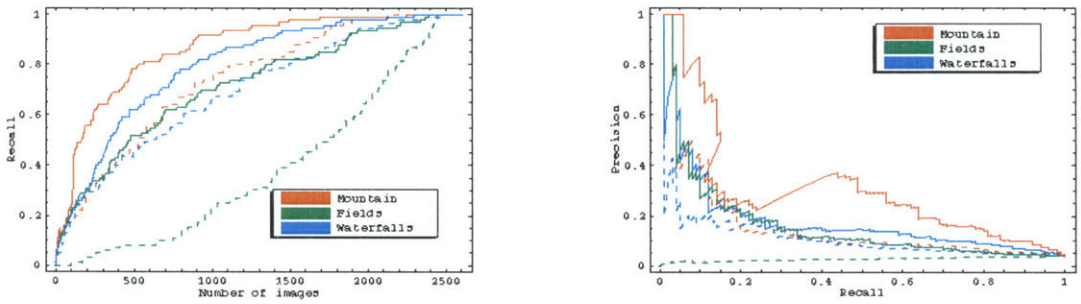


Figure 2-13: Comparison of simple trained templates (solid curves) with global histogramming method (dashed curves) for 3 concepts on 2600 images images from the large test set.

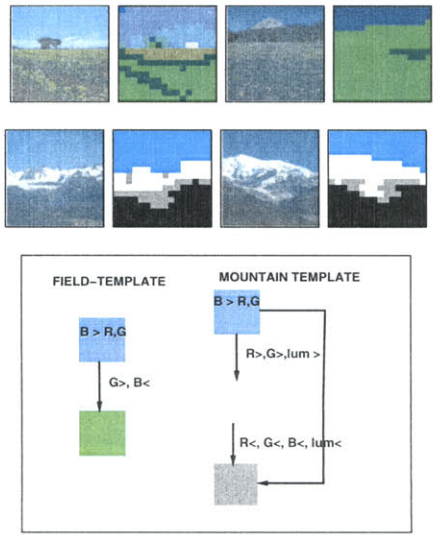


Figure 2-14: Top Rows: Examples of images in fields and snowy-mtns classes and the same low resolution quantized images. Bottom Row: Templates for fields and snowy mountains built from these examples.

the query images. Each image is represented by three histograms, one for each color channel (R,G,B). Figure 2-12 shows the best precision recall and recall curves for 3 concepts (mountains, fields and waterfalls) over both the hypothesis classes. The dashed curves show the performance on global histogramming using the same queries. We see that the trained templates perform much better than global histogramming on this dataset. This holds even for the larger dataset of 2600 images as seen in Figure 2-13.

Figure 2-15(a) shows the results of training the template on the example field images selected in red in the top two rows of the figure. The query images are selected from a random set of 20 images that the user can bring up from the database. The database is the small test set of 400 images which is disjoint form the training images. The top 30 matches that matched the field-template are also shown. The false positives (Figure 2-15)(b) indicate that the simple color relationships defined by the template might not be sufficient to make finer distinctions when the color and spatial relations are satisfied by 2 classes (for example, the difference between a beach and a brown field with sky). Figure 2-14 shows examples of the quantized patches in the low resolution images for fields. The templates were built from these low-

Class	FP in top 30	FP in top 100
Fields	5	28
Snowy Mountains	3	26

Table 2.1: Summary of Results using trained templates. The table has the average number of false positives in the top 30 and the top 100 matches over 50 runs.

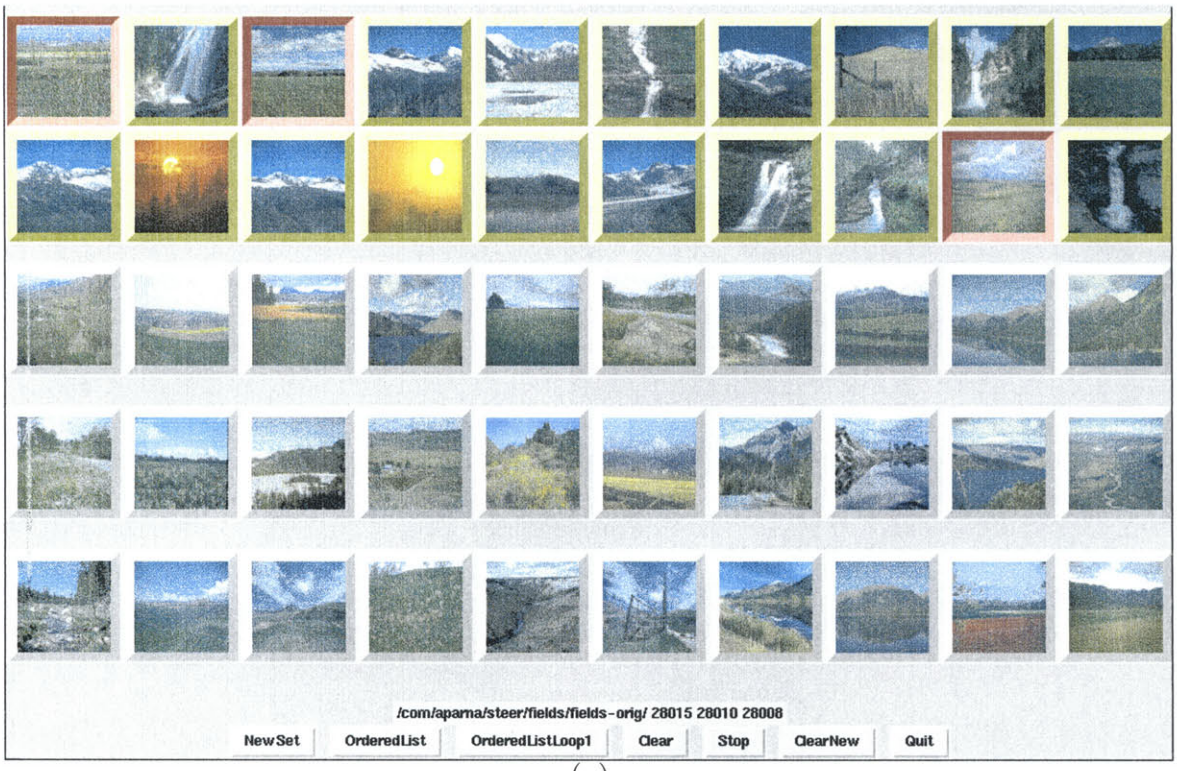
resolution images by extracting the dominant luminance and color relations (units of the template) and assembling the template units using spatial relations.

Figure 2-16 shows the results of running our system on the example snowy-mountain images which are the ones selected in red in the top two rows of the figure. The database has 2600 images. The top 30 matches that matched the snowy-mountain-template in the order they were found are also shown in the last three rows of the figure. The false positives for this query were mostly fields with clouds. Figure 2-14 shows examples of the quantized patches in the low resolution images for snowy mountains. The snowy-mountain template extracted from the examples is a 3 patch template in the N-S direction (see figure 2-14).

Table 2.1 summarizes the average performance of the system over 50 trials for two classes (fields and snowy-mountains). There were 110 field images and 100 snowy mountain images in the dataset. These were classified manually for the purpose of evaluating the performance of the system.

2.8 Summary

In this chapter, we have described a simple reinforcement scheme for learning the flexible templates that Lipson hand-crafted and have shown that the learned templates can classify new images effectively. The scheme proposed here is a variation on simple discrete intersection of common color and spatial properties in all the example images. This scheme uses only positive examples and classifies natural scenes surprisingly well. The system extracts the most complex graph (all instances) that is



(a)

(b)

Figure 2-15: (a) The query images are the ones selected in red in the top two rows. The next 3 rows are the top 30 matches returned by the system, based on the constructed template. (b) Some examples of the false positives.

Figure 2-16: Training images for mountains are the ones selected in red in the top two rows. Bottom Rows: Top 30 matches for mountain template. The dataset has 2600 images.

common to all positive examples, emphasizes both the color and spatial relations in the examples by treating the classification as a graph matching problem. The system finally does an exhaustive subgraph match at classification time. This is feasible here since we are using very low resolution images and the number of nodes in the graph is small. In the next chapter, we will explore the following issues.

- the role of optimizing for the target concept in a continuous domain instead of the discrete intersection used in this chapter
- the role of adding negative examples to train the system

We will see that, for more general concepts, moving to the continuous domain and adding negative examples help reduce the false positives and hence improve classifier performance.

Chapter 3

Pictorial Query Learning as a Multiple Instance Learning Problem

As we mentioned earlier, current training schemes for image database indexing require the user to specify exactly what he wants by labeling parts of the query image and specifying the importance of the features like color composition, color layout, texture, shape and location from a set of preferences. The selected region and the feature weights are then used to retrieve new images that match the user's query preferences. Our goal is to have a framework which will automatically pick the relevant regions in the image as well as the weights for the different features.

While a user can create templates corresponding to the different visual categories (e.g. waterfalls, mountains) by hand, this is not very practical for a diverse set of queries. In this thesis, our goal is to allow the user to train the system by selecting positive and negative examples and letting the system construct a template that captures the relevant parts of the scene from those examples. The user can then refine the template by providing the system with positive and negative feedback from the set of retrieved images. Our approach to training image retrieval systems builds on the idea of Multiple Instance Learning [Dietterich *et al.*1997, Dietterich *et al.*1994] using the Diverse Density algorithm [Maron and Lozano-Pérez1998, Maron1998].

The method described in the previous chapter extracts templates using a small set of positive examples by a simple intersection of feature instances common to all the examples. The template is then used to classify new images from the database. The experimental results are encouraging and show that simple learned concepts can be used effectively in this domain. In order to extend the current system, we need to understand (1) the role of negative examples in the initial training process and (2) the role of positive and negative feedback in the template refinement process through user interactions. We encode the influence of negative examples on the scene concept as follows: color and spatial properties encoded in the scene concept are common to regions in the positive examples and their importance is reduced if they share attributes with regions in the negative examples.

To deal with these issues, we consider an alternative framework for learning queries of visual concepts. We apply the Multiple-Instance learning framework to extract and refine scene concepts from a small number of positive and negative examples. Multiple Instance Learning explicitly addresses the ambiguity inherent in learning concepts from images. In this chapter, we describe the Multiple Instance Learning framework and show that it is applicable for learning query concepts for image retrieval. We also show that (a) very simple features that encode color and spatial properties of pixel neighborhoods are sufficient to classify images of natural scenes efficiently and reliably, (b) complicated concepts (e.g. conjunctions) perform better that simple ones and (c) user interaction helps improve classifier performance.

3.1 Example Based Learning

Example Based Learning is a branch of machine learning which allows programs to learn a task from examples and use it to adapt to new situations. In example-based learning, the system is trained with input values and their corresponding output values and the system "learns" the mapping between the inputs and outputs for that task and adapts this mapping to predict outcomes for new input values that it has not been trained on. These learning techniques are particularly useful in problems where

it is hard to precisely model all possible outcomes and have been used in a variety of applications like speech recognition, pedestrian detection [Oren *et al.*1997], face detection [Sung and Poggio1994, H. Rowley and Kanade1995, Pentland *et al.*1994] etc.

In a typical machine learning problem, the task is to learn a function $y = f(x_1, x_2, ...x_n)$ given a set of input-output values so that the approximated function f maps the input values to the correct output values and interpolates between the outputs in regions where there is no input value provided. For example, in a face detection task, input values can be pixel values from a face image and the output value y is a boolean value indicating whether the input is a face. f is the function which we would like to learn from examples so that it can be applied to new input values. A large data sample gives more information about f since it has the output values for f for a large number of input points. The choice of input values also affects how well we can learn an approximation of f. Ideally, we want the training data to span the input space and provide a good representation of f. In order to learn a good approximation of f at all locations, we need (1) a well distributed training data set that spans the input space and (2) more data at input points where f changes rapidly. Thus, the quality and the number of training examples affects the performance of the learning algorithm.

In typical supervised learning, the examples are given in terms of $(y_i, x_{i1}, x_{i2}, ..., x_{in})$, where each set of input values is labeled with the correct output value y_i. In traditional unsupervised learning, the set of input values is given to the learner with no information about the output value. In the context of image classification, a *Supervised Learning* approach would give a class label to every pixel in the example images and use these labels to retrieve new images. On the other extreme, an *Unsupervised Learning* approach would not use any labels on the images or the pixels which indicate if the example is good or bad. The approach we describe in this chapter is called Multiple Instance Learning [Dietterich *et al.*1997, Long and Tan1996, Auer *et al.*1996, Blum and Kalai1998] and falls in between these two extremes. In the Multiple-Instance learning paradigm, the learner gets a set of positive and negative labeled examples of the concept to be learned. Each example is a collection

of instances. An example is labeled positive if at least one instance in it is positive and labeled negative if all the instances in it are negative. In our image classification example, the learner gets a small set of labeled images indicating whether they are positive or negative examples. Thus, given a positive example, the learner knows that somewhere in the example is the class concept. Since the instances are unlabeled, the system does not know which instance in the positive example is relevant and deduces this automatically.

3.2 Multiple-Instance Learning

In traditional supervised learning, a learning algorithm receives a training set which consists of individually labeled examples. There are situations where this model fails, specifically when the teacher cannot label individual instances, but only a collection of instances. For example, given a picture containing a waterfall, what is it about the image that causes it to be labeled as a waterfall? Is it the butterfly hovering in the corner, the blooming flowers, or the white stream of water? It is impossible to tell by looking at only one image. The best we can say is that at least one of the objects in the image is a waterfall. Given a number of images (each labeled as waterfall or non-waterfall), we can attempt to find commonalities within the waterfall images that do not appear in the non-waterfall images. Multiple-Instance learning is a way of formalizing this problem, and Diverse Density is a method for finding the commonality.

In Multiple-Instance learning, we receive a set of *bags*, each of which is labeled positive or negative. Each bag contains many *instances*, where each instance is a point in feature space. A bag is labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least one instance in it which is positive. From a collection of labeled bags, the learner tries to induce a concept that will label unseen bags correctly. This problem is harder than even noisy supervised learning because the ratio of negative to positive instances in a positively-labeled bag (the noise ratio) can be arbitrarily high.

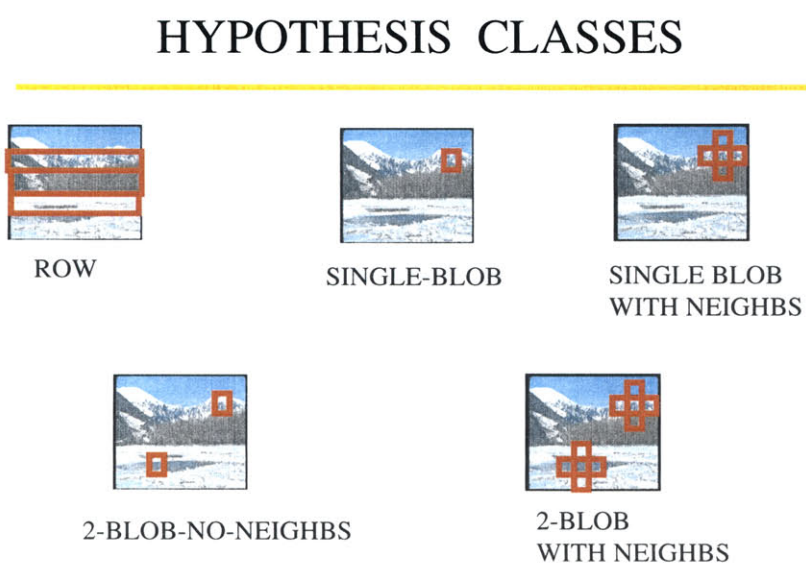
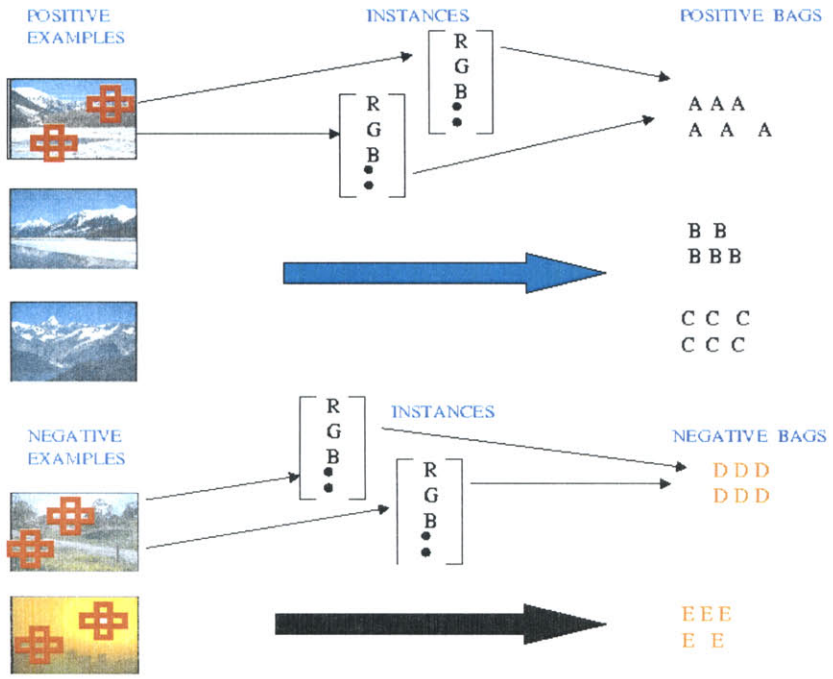


Figure 3-1: Hypothesis classes used for the experiments.

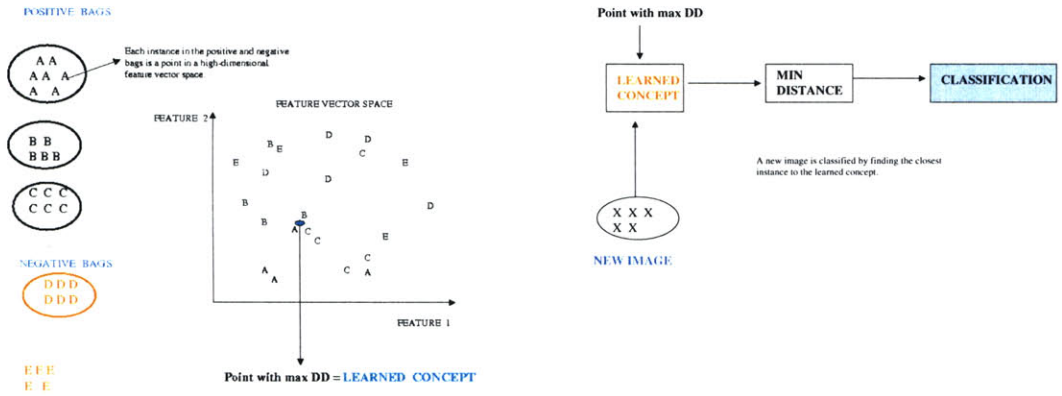
The multiple-instance learning model was only recently formalized by Dietterich et al. in [Dietterich *et al.*1997], where they develop algorithms for the drug activity prediction problem. This work was followed by [Long and Tan1996, Auer *et al.*1996, Blum and Kalai1998], who showed that it is difficult to PAC-learn in the Multiple-Instance model unless very restrictive independence assumptions are made about the way in which examples are generated. Auer [Auer1997] shows that despite these assumptions, the MULTINST algorithm performs competitively on the drug activity prediction problem. Maron et al. [Maron and Lozano-Pérez1998] develop an algorithm called *Diverse Density*, and show that it performs well on a variety of problems such as drug activity prediction, stock selection, and learning a description of a person from a series of images that contain that person.

3.2.1 Multiple-instance learning for scene classification

In our framework, each training image is a bag. The instances in a particular bag are various sub-images. If the bag is labeled as a waterfall (for example), we know that



(a)



(b)

Figure 3-2: (a) The positive and negative examples are converted to bags of feature instances (b) each instance is a point in a high dimensional space. The "ideal" feature vector is the point with maximum Diverse Density. This point corresponds to the learned concept. To classify new images, find the location of the instance closest to the learned concept.

at least one of the sub-images (instances) is a waterfall. If the bag is labeled as a non-waterfall, we know that none of the sub-images contains a waterfall. Each of the instances, or sub-images, is described as a point in some feature space. As discussed in section 3.5, we have experimented with several ways of describing an instance. Figure 3-1 shows the hypothesis classes that we used in our experiments. We will discuss one of them (`single blob with neighbors`) in detail: a sub-image is a 2x2 set of pixels (referred to as a *blob*) and its four neighboring blobs (up, down, left, and right). The sub-image is described as a vector $[x_1, x_2, \ldots, x_{15}]$, where x_1, x_2, x_3 are the mean RGB values of the central blob, x_4, x_5, x_6 are the differences in mean RGB values between the central blob and the blob above it, etc. One bag is therefore a collection of instances, each of which is a point in a 15-dimensional feature space. We assume that at least one of these instances is the template that contains the waterfall.

We would now like to find a description which will correctly classify new images as waterfalls or non-waterfalls. This can be done by finding what is in common between the waterfall images given during training and the differences between those and the non-waterfall images. The main idea behind the *Diverse Density* (DD) algorithm is to find areas in feature space that are close to at least one instance from every positive bag and far from every negative instance. The algorithm searches the feature space for points with high Diverse Density. Once the point (or points) with maximum DD is found, a new image is classified positive if one of its sub-images is close to the maximum DD point. As seen in Section 3.5, the entire database can be sorted by the distance to the learned concept. Figure 3-2 is a schematic of how the system works.

In the following subsection, we will describe a derivation of Diverse Density and how we find the maximum in a large feature space. We will also show that the appropriate scaling of the feature space can be found by maximizing DD not just with respect to location in feature space, but also with respect to a weighting of each of the features.

3.2.2 Diverse density

In this section, we derive a probabilistic measure of Diverse Density. More details are given in [Maron1998]. We denote positive bags as B_i^+, and the j^{th} instance in that bag as B_{ij}^+. Likewise, B_{ij}^- represents an instance from a negative bag. For simplicity, let us assume that the true concept is a single point t in feature space. We can find t by maximizing $\Pr(t \mid B_1^+, \cdots, B_n^+, B_1^-, \cdots, B_m^-)$ evaluated for t at all points in feature space. Using Bayes' rule and a uniform prior over the concept location, we see that this is equivalent to maximizing the likelihood:

$$\arg\max_t \Pr(B_1^+, \cdots, B_n^+, B_1^-, \cdots, B_m^- \mid t). \tag{3.1}$$

By making the additional assumption that the bags are conditionally independent given the target concept t, this decomposes into

$$\arg\max_t \prod_i \Pr(B_i^+ \mid t) \prod_i \Pr(B_i^- \mid t) \tag{3.2}$$

which is equivalent (by similar arguments as above) to maximizing

$$\arg\max_t \prod_i \Pr(t \mid B_i^+) \prod_i \Pr(t \mid B_i^-) \tag{3.3}$$

This is a general definition of Diverse Density, but we need to define the terms in the products to instantiate it. In this paper, we use the noisy-or model as follows:

$$\Pr(t \mid B_i^+) = 1 - \prod_j (1 - \Pr(t \mid B_{ij}^+)). \tag{3.4}$$

The noisy-or model makes two assumptions: one is that for t to be the target concept it is caused by (hence close to) one of the instances in the bag. It also assumes that the probability of instance j not being the target is independent of any other instance not being the target.

Finally, we estimate the distribution $\Pr(t \mid B_{ij}^+)$ with a Gaussian-like distribution

of

$$\exp(- \parallel B_{ij}^{+} - t \parallel^2).$$

A negative bag's contribution is likewise computed as

$$\Pr(t \mid B_i^-) = \prod_j (1 - \Pr(t \mid B_{ij}^-)).$$

This leads to an algorithm which performs a kernel density estimate with the instances from each bag. The algorithm then multiplies the contribution of each density estimate to get the Diverse Density at a point. A supervised learning algorithm such as nearest-neighbor or kernel regression would average the contribution of each bag, computing a density of instances. This algorithm computes a product of the contribution of each bag, hence the name Diverse Density. Note that Diverse Density at an intersection of n bags is exponentially higher than it is at an intersection of $n-1$ bags, yet all it takes is one well placed negative instance to drive the Diverse Density down.

The initial feature space is probably not the most suitable one for finding commonalities among images. Some features might be irrelevant or redundant, while small differences along other features might be crucial for discriminating between positive and negative examples. The Diverse Density framework allows us to find the best weighting on the initial feature set in the same way that it allows us to find an appropriate location in feature space. If a feature is irrelevant, then removing it can only increase the DD since it will bring positive instances closer together. On the other hand, if a relevant feature is removed then negative instances will come closer to the best DD location and lower it. Therefore, a feature's weight should be changed in order to increase DD. Formally, the distance between two points in feature space (B_{ij} and t) is

$$\parallel B_{ij} - t \parallel^2 = \sum_k w_k (B_{ijk} - t_k)^2 \tag{3.5}$$

where B_{ijk} is the value of the k^{th} feature in the j^{th} point in the i^{th} bag, and w_k is a non-negative scaling factor. If w_k is zero, then the k^{th} feature is irrelevant. If w_k is

large, then the k^{th} feature is very important. We would like to find both t and w such that Diverse Density is maximized. We have doubled the number of dimensions in our search space, but we now have a powerful method of changing our representation to accommodate the task.

We can use also use this technique to learn more complicated concepts than a single point. To learn a 2-disjunct concept $t \vee s$, we maximize Diverse Density as follows:

$$\arg\max_{t,s} \prod_i (1 - \prod_j (1 - \Pr(t \vee s \mid B_{ij}^+))) \prod_i \prod_j \Pr(t \vee s \mid B_{ij}^-) \tag{3.6}$$

where $\Pr(t \vee s \mid B_{ij}^+)$ is estimated as $\max\{\Pr(t \mid B_{ij}^+), \Pr(s \mid B_{ij}^+)\}$. Other approximations (such as noisy-or) are also possible.

Finding the maximum Diverse Density in a high-dimensional space is a difficult problem. In general, we are searching an arbitrary landscape and the number of local maxima and size of the search space could prohibit any efficient exploration. In this paper, we use gradient ascent (since DD is a differentiable function) with multiple starting points. This has worked successfully because we know what starting points to use. The maximum DD point is made of contributions from some set of positive points. If we start an ascent from every positive point, one of them is likely to be closest to the maximum, contribute the most to it and have a climb directly to it. Therefore, if we start an ascent from every positive instance, we are very likely to find the maximum DD point. When we need to find both the location and the scaling of the concept, we perform gradient ascent for both sets of parameters at the same time (starting with all scale weightings at 1). The number of dimensions in our search space has doubled, though. When we need to find a 2-disjunct concept, we can again perform gradient ascent for all parameters at once. This carries a high computational burden because the number of dimensions has doubled, and we perform a gradient ascent starting at every pair of positive instances.

Our goal in the next section is to show that: (1) Multiple-Instance learning by

maximizing diverse density can be used in the domain of natural scene classification, (2) simple concepts in low resolution images are sufficient to learn some of these concepts, (3) negative examples help improve performance , (4) working in the continuous domain helps when the query class has a lot of variation (e.g. images of fields) and (5) adding false positives and false negatives over multiple iterations (user interaction) can be used to improve the classifier performance. We also discuss how this general architecture can be extended to object classification and show that rough segmentation and feature selection (scaling) play an important role in this domain.

3.3 Discrete Intersection

Discrete intersection is a special case of the current formulation of Diverse Density when we do not optimize for either location or scaling of the target concept. Finding the target concept using Discrete Intersection consists of the following steps:

- Convert each example image into a bag of instances
- Every positive bag has at least one positive instance and every negative bag has only negative instances.
- Start the minimization from every positive instance but run it without optimizing for location or scaling i.e. find the positive instance that maximizes Diverse Density after one iteration.
- The instance with the maximum DD is the target concept In this case, the target concept is the positive instance which minimizes the Euclidean distance to at least one instance in each of the positive bags and maximizes the distance to all of the negative instances.

The algorithm we described in Chapter 2, can also be formulated as a special case of Diverse Density since the target concept (template graph) is made up of positive instances that are reinforced in *all* of the positive examples. A template graph with k nodes from Chapter 2 can be viewed as the target concept obtained from a discrete

intersection of positive bags each containing many $k - blob$ instances where each $k - blob$ instance is a point in the feature space and the target concept is the "ideal" point that minimizes the distance to at least one instance in each positive bag.

In the next section we show that while simple discrete intersection is computationally less expensive, the ability to interpolate between positive examples by optimizing for position using diverse density helps improve performance and get better generalization capabilities to accommodate variations in the query class.

3.4 Finding structure in the feature space

In our image database, each image is represented by a collection of instances. Each instance is a k dimensional vector $< x_1, x_2, ..., x_k >$ where x_i represent the color, texture, geometric and spatial properties of an image region and its neighbors. For the task of classification we would like the space to be made up of several tight clusters, each belonging to a different class. In reality, however, the space of images has a lot of overlapping clusters and features belonging to the same class do not cluster well in all dimensions. Figure 3-3 (a) and (b) shows an example where 10 images of the same class cluster well in some projections and do not cluster well in other projections. Choosing the right dimensions for a class can improve classification performance since the features are consistent in the examples in these projections and this leads to good clustering. Scaling the feature space also gives us a way of determining which properties are salient for a particular query and learning a concept that focuses on these relevant image properties. However, the important dimensions in the feature space vary from class to class and need to be determined only at query time. Figure 3-3 (c) shows an example of 10 images of a different class projected onto the same axes which caused the examples to cluster well in 3-3. Thus, for this application of image classification, it is hard to determine which dimensions are important prior to analyzing the query.

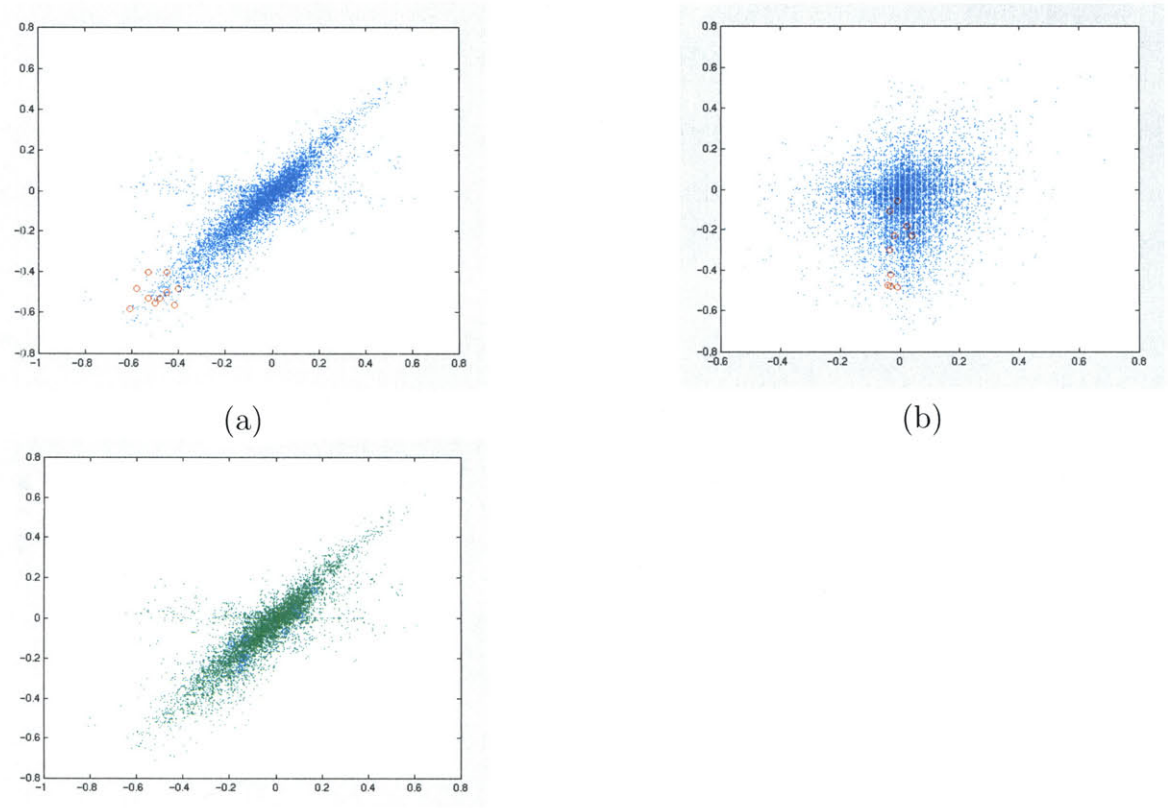


Figure 3-3: (a) 10 feature instances from one class cluster well when projected in these two dimensions (b) the same feature instances do not cluster well in other dimensions (c) 10 instances from another class do not cluster well when projected in the same dimensions as (a). The other dots represent all of the instances from all the images.

3.4.1 Feature selection using Principal Components Analysis

Many existing feature extraction techniques involving linear transformations are based on eigenvector analysis of the covariance matrix. These techniques rely on the input data entirely without using any information about the target data and are thus a form of *unsupervised* learning. These techniques fall under the broad category of principal components analysis (PCA). They represent the sample space with the vectors (directions) for which the variances (eigenvalues) are the largest. These directions which have the largest uncertainty are retained as features. For example, for the task of image classification, we want to explore a low dimensional representation of the image feature space that captures the relevant properties of the image class.

As we mentioned in Chapter 1, Principal Components Analysis (PCA) has been used effectively for applications like face detection and recognition [Pentland *et al.*1994] among others. PCA and other subspace methods project the high dimensional data onto a few principal components (eigenvectors) of large variance. These components form a low dimensional subspace which captures the key sources of variation in the class. At detection time, these methods compute the distance between the new image and this low dimensional subspace.

If we think of an image as an NxN array of pixels, then we can represent this array as an observation vector with N^2 values.

Given a set of observations

$$x_k \in R^N, k = 1, ..., M$$

such that

$$\sum_{k=1}^{M} x_k = 0,$$

PCA diagonalizes the covariance matrix

$$C = \frac{1}{M} \sum_{j=1}^{M} x_j x_j^T$$

To do this one has to solve the eigenvalue equation

$$Cv = \lambda v$$

for eigenvalues

$$\lambda \geq 0$$

and eigenvectors

$$v \in R^N$$

In the case of image database indexing, the features are distributed such that there is a lot of spread between classes. The dimensions we want to preserve are those that separate the data well and retain the clustering within classes. We did the following experiment to demonstrate that a linear and unsupervised method like PCA is not suited for feature selection in image classification. We computed the principal components from a database of 400 images containing 5 classes of natural scenes. Each feature was a 15 dimensional vector capturing the color and spatial properties of a pixel and its neighbors. We kept 8 eigenvectors having the largest eigenvalues and tried to classify new images by projecting onto this 8-dimensional subspace. Figure 3-4 shows a 3D plot of the projections of 20 images of two classes (mountains and waterfalls) onto the top three principal components. This figure illustrates a case where the largest eigenvectors do not separate the classes and hence do not provide good classification results. Figure 3-5 shows the results of classifying 400 images using the distance from the 8 dimensional subspace given by the principal components. We see that the results are no better than random. Given the complexity of the space of images and given that the salient dimensions could vary from class to class, a linear and unsupervised method like PCA is not well suited to capture an adequate low dimensional representation of the feature space a priori. As a consequence, we explore the alternative of extracting the salient dimensions at query time.

While PCA is not suited to finding salient dimensions a priori, can we use it in a *supervised* mode to extract the consistencies between labeled examples in a class at

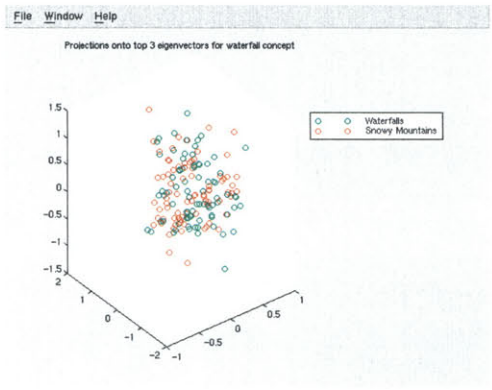


Figure 3-4: Plot showing the projections of 20 images from the waterfall and snowy mountain class onto top 3 principal components.

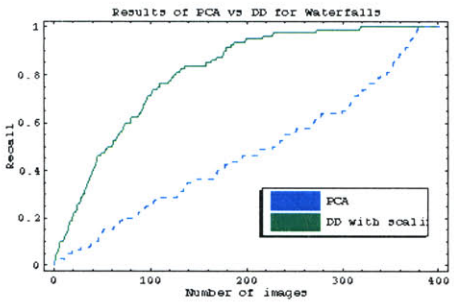


Figure 3-5: Classification results of PCA on a database of 400 images

query time? In order to explore this, we took a set of positive examples belonging to the waterfalls class. We extracted the instance (15 dimensional vector) corresponding to the waterfall concept in each example and computed the eigenvectors. We kept the 8 smallest eigenvectors (low variance) and classified new images by projecting them onto these 8 eigenvectors. Figure 3-6 shows the classification results on a database of 400 images. We see that the classifier gets all the members in the class but also gets a number of false positives. Since we did not use any negative examples in the training, the subspace could capture the salient properties of all the positive examples as well as many negative examples. As we mentioned before, the distribution of the negative examples in this image feature space make it difficult for linear methods like PCA to separate the positives from the negatives. In this chapter, we discuss a method for classifying images that brings the positive examples close together while pulling them away from all the negative examples. The method which we will describe maximizes diverse density rather than just minimizing the variance between the positive examples. Figure 3-6 shows that the classification performance of our method using positive and negative examples is better than the classification using PCA. Until now we have been discussing ways of learning structure in the feature space by reducing the dimensionality. In our treatment of the image classification problem, not only do we need to find the salient dimensions for a given query set, we also need to find the location of the concept that adequately captures the consistencies in the positive examples and separates it from the negative examples. For example, given two images of a tiger in a lake and a tiger in the woods, we want to be able to say the "tiger" concept is common in both images and that color and the striped texture are important feature dimensions. In the PCA experiments described above, we had to train the system with instances (sub-images) that contained the concept (object) and nothing else. In our tiger example, this means giving the system images that contained the tiger without the background. Our system finds both the location of the target concept ("tiger") as well as the salient dimensions for the "tiger" concept from examples with varying backgrounds and in the presence of other distractor objects.

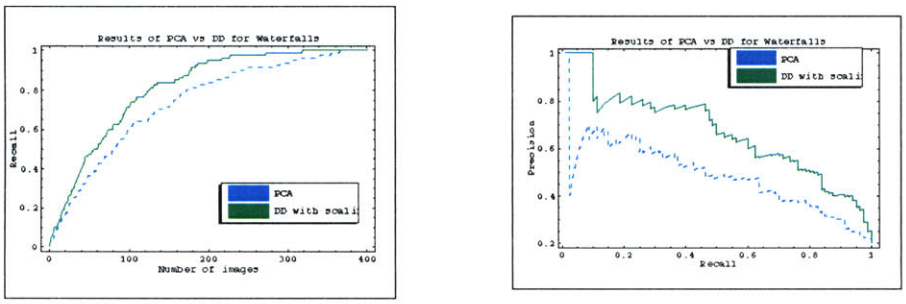


Figure 3-6: Classification results of PCA vs DD

3.5 Experiments: Natural Scene Classification

In this section, we show four different types of results from running the system: one is that Multiple-Instance learning is applicable to this domain. A second result is that one does not need very complicated hypothesis classes to learn concepts from the natural image domain. We also compare the performance of various hypotheses, including the global histogram method. Finally, we show how user interaction would work to improve the classifier.

3.5.1 Experimental setup

We tried to learn three different concepts: waterfall, mountain, and field. For training and testing we used natural images from the COREL library, and the labels given by COREL. These included 100 images from each of the following classes: waterfalls, fields, mountains, sunsets and lakes. We also used a larger test set of 2600 natural images from various classes.

We created a *potential training set* that consisted of 20 randomly chosen images from each of the five classes mentioned above. This left us with a *small test set* consisting of the remaining 80 images from each of the five classes. We separated the potential training set from the testing set to insure that results of using various training schemes and hypothesis classes can be compared fairly. Finally the *large test set* contained 2600 natural images from a large variety of classes.

For a given concept, we create an *initial training set* by picking five positive

examples of the concept and five negative examples at random, all from the potential training set. After the concept is learned from these examples (by finding the the location of the target concept (point with maximum DD) and scaling of feature space which maximized DD), the unused 90 images in the potential training set are sorted by distance from the learned concept. An image/bag's distance from the concept is the minimum distance of any of the image's subregions/instances from the concept. This sorted list can be used to simulate what a user would select as further refining examples. Specifically, the most egregious false positives (the non-concept images at the beginning of the sorted list) and the most egregious false negatives (the concept images at the end of the sorted list) would likely be picked by the user as additional negative and positive examples.

We attempted four different training schemes: `initial` is simply using the initial five positives and five negative examples. `+5fp` adds the five most egregious false positives, after initial learning and sorting of the training set. `+3fp+2fn` adds 3 false positives and 2 false negatives, after initial training. `+10fp` repeats the `+5fp` scheme twice i.e. with a second sorting.

3.5.2 Features/Instances

All images were smoothed using a Gaussian filter and sub-sampled to 8×8. We used the RGB color space in these experiments. The RGB values were normalized to be in the $[0,1]^3$ cube. For every class and for every training scheme, we tried to learn the concept using one of seven hypothesis classes (Figure 3-1 shows some examples):

- `row`: An instance is the mean color of a row in the image and the color difference in the rows above and below it given by a vector $(x_1, ..., x_9)$, where (x_1, x_2, x_3) are the mean RGB values of row j, (x_4, x_5, x_6) are the mean RGB values of row j+1 minus the mean RGB values of row j , and (x_7, x_8, x_9) are the mean RGB values of row j-1 minus the mean RGB values of row j.

- `single blob with neighbors`: An instance is the mean color of a 2×2 blob and the color difference with its 4 neighboring blobs given by a vector $(x_1, ..., x_{15})$,

where (x_1, x_2, x_3) are the mean RGB values of the blob centered at $(i+2, j)$, (x_4, x_5, x_6) are the mean RGB values of the blob centered at (i, j) minus the mean RGB values of the blob centered at $(i+2, j)$, (x_7, x_8, x_9) are the mean RGB values of the blob centered at $(i, j+2)$ minus the mean RGB values of the blob centered at (i, j), (x_{10}, x_{11}, x_{12}) are the mean RGB values of the blob centered at $(i-2, j)$ minus the mean RGB values of the blob centered at (i, j) , and (x_{13}, x_{14}, x_{15}) are the mean RGB values of the blob centered at $(i, j-2)$ minus the mean RGB values of the blob centered at (i, j).

- `single blob with no neighbors`: An instance is the color of each of the pixels in a 2×2 blob given by a vector $(x_1, ..., x_{12})$ where (x_1, x_2, x_3) are the RGB values of pixel (i, j), (x_4, x_5, x_6) are the RGB values of pixel $(i-1, j)$, (x_7, x_8, x_9) are the RGB values of pixel $(i-1, j-1)$, (x_{10}, x_{11}, x_{12}) are the RGB values of pixel $(i, j-1)$.

- `two blob with neighbors`: An instance is the mean color of two descriptions of two `single blob with neighbors` and their relative spatial relationship (whether the second blob is above or below, and whether it is to the left or right, of the first blob). An instance (i, j, k, l) is given by the vector $(x_1, ..., x_{32})$ where $(x_1, ..., x_{15})$ is a description of one single blob with neighbor instance, $(x_{16}, ..., x_{30})$ is a description of the second single blob with neighbor instance which can be in any direction relative to the first (x_{31}) and (x_{32}) gives the distance between the blobs $(i-k)$ and $(j-l)$ respectively.

- `two blob with no neighbors`: An instance is the mean color of two descriptions of two `single blob with no neighbors` and their relative spatial relationship.

- `disjunctive blob with neighbors`: An instance is the same as the single blob with neighbors but the concept learned is a disjunction of two single blob concepts and can be expressed as $(c_t^1, c_s^1) \vee (c_t^2, c_s^2)$. The 2-disjunct concept has different feature weightings in each disjunct. A new bag is labeled positive if at

least one of its instances is in either of the disjuncts (in (c_t^1, c_s^1) or in (c_t^2, c_s^2)).

- `disjunctive blob with no neighbors`: An instance is the same as the single blob with no neighbors but the concept learned is a disjunction of two single blob concepts.

In our experiments, learning a concept took anywhere from a few seconds for the simple hypotheses to a few hours for the 2-blob and disjunctive hypotheses. The more complicated hypotheses take longer to learn because of the higher number of features and because the number of instances per bag is large (and to find the maximum DD point, we perform a gradient ascent from every positive instance). Because this is a prototype, we have not tried to optimize the running time; however, a more intelligent method of generating instances (for example, a rough segmentation using connected components) will reduce both the number of instances and the running time by orders of magnitude. We will discuss some of these methods in the next chapter.

3.5.3 Results

In this section we show results of testing the various hypothesis classes, training schemes, and concept classes against the small test set and the larger one. The small test set does not intersect the potential training set, and therefore more accurately represents the generalization of the learned concepts. The large test set is meant to show how the system scales to larger image databases.

3.5.4 Precision and Recall Graphs

The graphs shown are precision-recall and recall curves. Precision is the ratio of the number of correct images to the number of images seen so far. Recall is the ratio of the number of correct images to the total number of correct images in the test set. Figure 3-7 shows the perfect precision-recall and recall curves. For example, in Figure 3-9, the waterfall precision-recall curve has recall 0.5 with precision of about 0.7, which means in order to retrieve 40 of the 80 waterfalls, 30% of the images retrieved are not

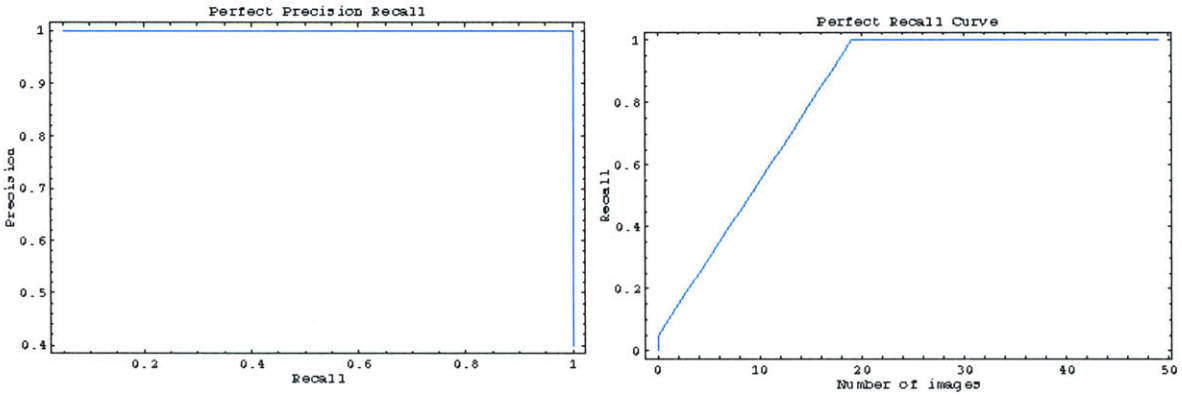


Figure 3-7: The perfect precision-recall and recall curves

waterfalls. We show both curves because (1) the beginning of the precision-recall is of interest to applications where only the top few objects are of importance, and (2) the middle of the recall curve is of interest to applications where correct classification of a large percentage of the database is important.

3.5.5 Experiments

Figure 3-8 shows that the performance of the learned mountain concept is competitive with a hand-crafted mountain template (from [Lipson *et al.*1997][1]). The test set consists of 80 mountains, 80 fields, and 80 waterfalls. It is disjoint from the training set. The hand-crafted model's precision-recall curve is flat at 84% because the first 32 images all receive the same score, and 27 of them are mountains. We also show the curves if we were to retrieve the 27 mountains first (best-case) or after the first five false positives (worst-case).

In Figure 3-9, we show the performance of the best hypothesis and training method on each concept class. The dashed lines show the poor performance of the global histogram method. The global histogram method [Flickner *et al.*1995, Bach *et al.*1996] retrieves new images based on the color composition of the query image. As we discussed earlier, this method does not capture any spatial information. The solid lines

[1]Lipson's classifier was modified to give a ranking of each image, rather than its class.

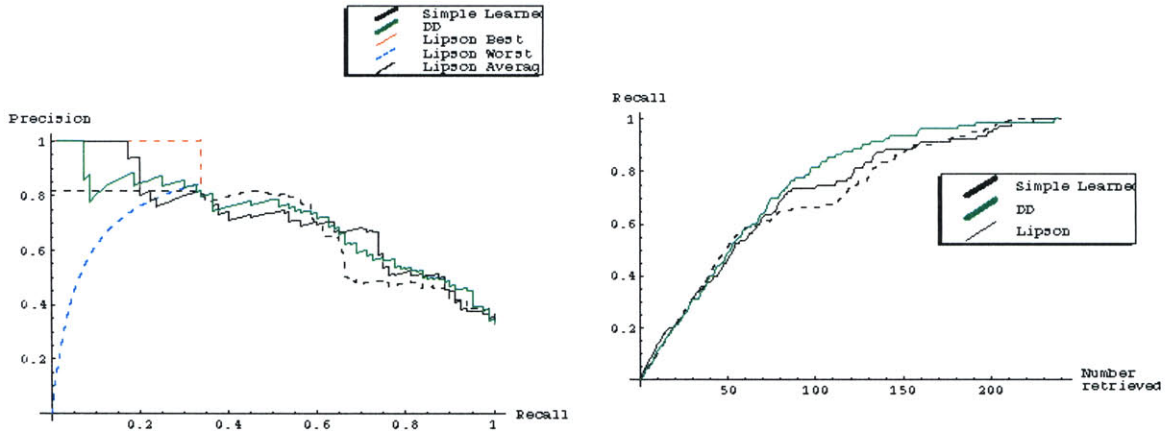


Figure 3-8: Comparison of learned concept (solid curves) with hand-crafted templates (dashed curves) for the mountain concept on 240 images from the small test set. The top and bottom dashed precision-recall curves indicate the best-case and worst-case curves for the first 32 images retrieved by the hand-crafted template which all have the same score.

in the precision-recall graph show the performance of `single blob with neighbors` with `+10fp` for waterfalls, `row` with `+10fp` for fields, and `disjunctive blob with no neighbors` with `+10fp` for mountains. The solid lines in the recall curve show the performance of the `single blob with neighbors` with `+10fp` for waterfalls, `single blob with neighbors` with `+3fp+2fn` for fields, and `row` with `+3fp+2fn` for mountains. This behavior continues for the larger test set.

In Figure 3-10, we show the precision-recall curves for each of the four training schemes. We average over all concepts and all hypothesis classes. We see that performance improves with user interaction. This behavior continues for the larger test set as well.

In Figure 3-12, we show the precision-recall and recall curves for each of the seven hypotheses averaged over all concepts and all training schemes. Note that these curves are for the larger 2600 image database. We see that the single blob with neighbors hypothesis has good precision. We also see that the more complicated hypothesis classes (i.e. the disjunctive concepts and the two-blob concepts) tend to have better recall curves.

In Figure 3-13, we show a snapshot of the system in action. The system is trained using training scheme `+10fp` for the waterfall concept. It has learned a waterfall

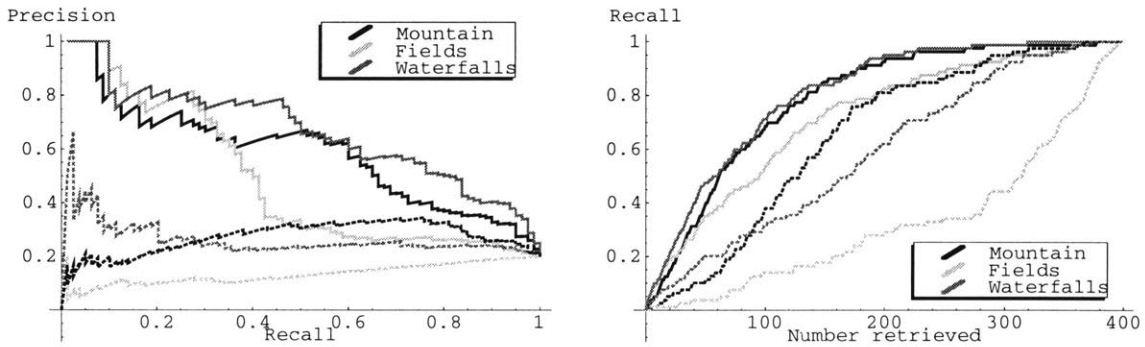


Figure 3-9: The best curves for each concept using a small test set. Dashed curves are the global histogram's performance.

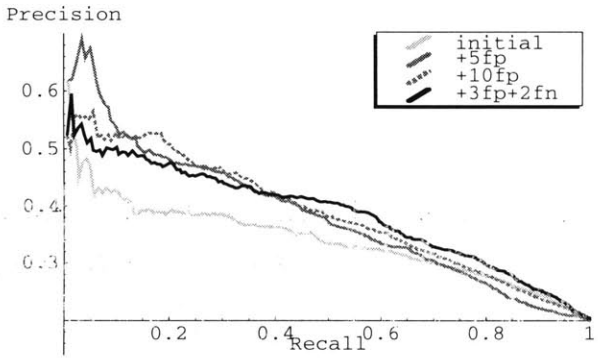


Figure 3-10: Different training schemes, averaged over concept and hypothesis class, using a small test set.

concept using the `single blob with neighbors` hypothesis. The learned waterfall concept is shown in Figure 3-11. The concept can be interpreted as somewhere in the image there is a blob with the properties given in Table 3.1.

These properties are weighted in the order given, and any irrelevant features have weight 0.0. A new image has the rating of the minimum distance of one of its instances to the learned concept, where the distance metric uses the learned scaling to account for the importance of the relevant features. As we can see in Figure 3-13, this simple learned concept is able to retrieve a wide variety of waterfall scenes.

The top 20 images in the figure are the training set. The first 10 images are the initial positive and negative examples used in training. The next 10 images are the false positives added. The last 30 images are the top 30 returned from the large

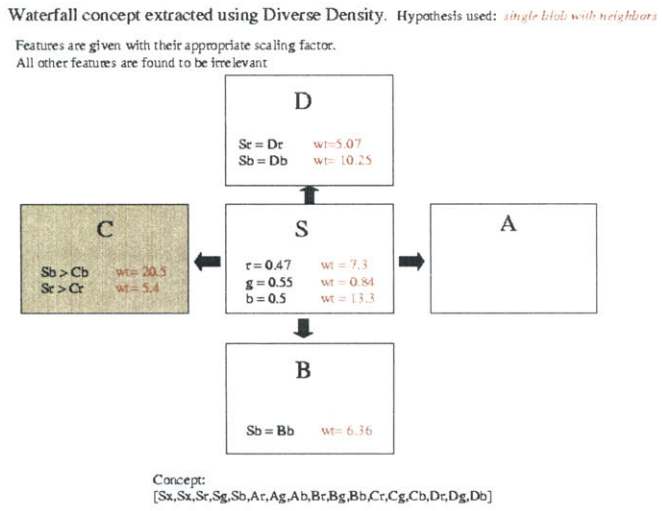


Figure 3-11: Illustration of the learned waterfall concept extracted by DD for Figure 3-13 dataset after retraining.

Role of feature selection

In Figure 3-14, we compare the performance of the system with and without the scaling option. As we discussed in Section 3.2.2, the architecture proposed here allows us to compute the scaling as well as the location of the target concept. The results indicate that scaling doesn't help very much in this domain of natural scene classification with this particular choice of features (color and spatial relations). We will show in the next chapter on object classification that the feature selection plays an important role when we have a more diverse set of cues.

Role of negative examples

In Figure 3-16, we compare the performance of the system when trained with only positive examples with its performance when trained with both positive and negative examples. The training in both cases was done without the scaling option. We see that adding negatives improves the recall and precision.

Feature description	Value	Weight
Blue change to left	0.277	20.57
Blue in blob	0.50	13.33
Blue change above	0.08	10.25
Red in blob	0.47	7.30
Blue change below	-0.03	6.36
Red change to the left	0.62	5.43
Red change above	0.07	5.07
Green in blob	0.55	0.84

Table 3.1: Learned waterfall concept. The table shows the feature descriptions, the feature value and the feature scaling. The features are listed in order of decreasing importance. All other features were found to be unimportant (i.e. with weight 0)

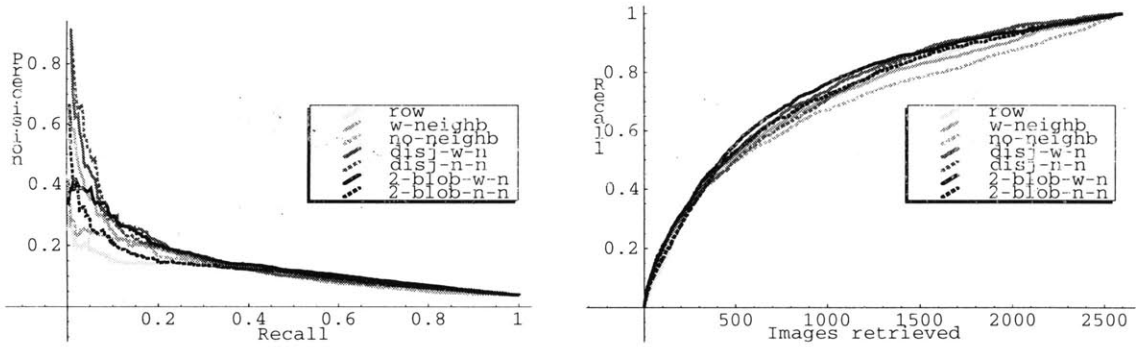


Figure 3-12: Different hypothesis classes averaged over concept and training scheme, using a large test set with 2600 images.

Discrete intersection vs. Diverse Density

Figure 3-15 compares the performance of DD vs discrete intersection as described in Section 3.3. We see that discrete intersection is competitive with diverse density for the "mountains" and the "waterfalls" but diverse density performs significantly better for the "fields" class. The "fields" class has the most variation. Optimizing for the location of the target concept allows for interpolation between the positive instances and this results in a more general target concept which accommodates a wider set of variations.

Figure 3-13: Snapshot of the system working. The database contained 2600 images of varied scenes from the COREL library. The figure shows the results for the waterfall class using the `single blob with neighbors` concept with `+10fp`. Top row: Initial training set–5 positive and 5 negative examples. Second Row: Additional false positives. Last three rows: Top 30 matches retrieved from the large test set. The red squares indicate where the closest instance to the learned concept is located.

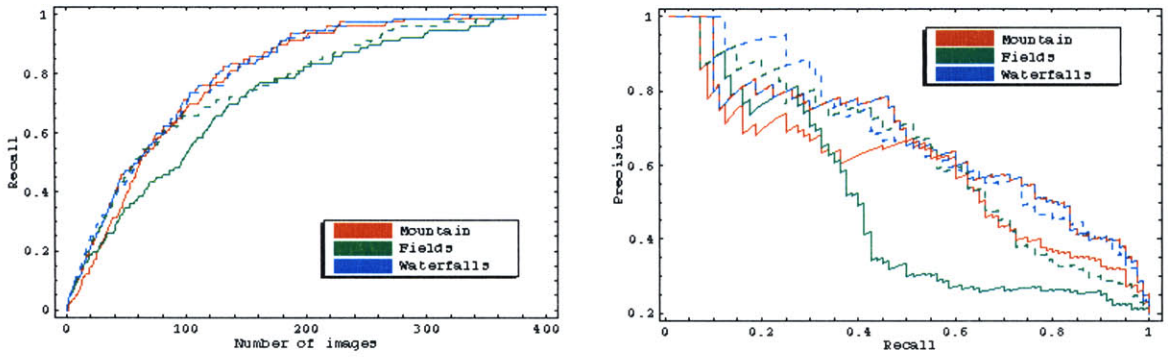


Figure 3-14: Comparing performance with and without feature selection. Dashed curves indicate no scaling.

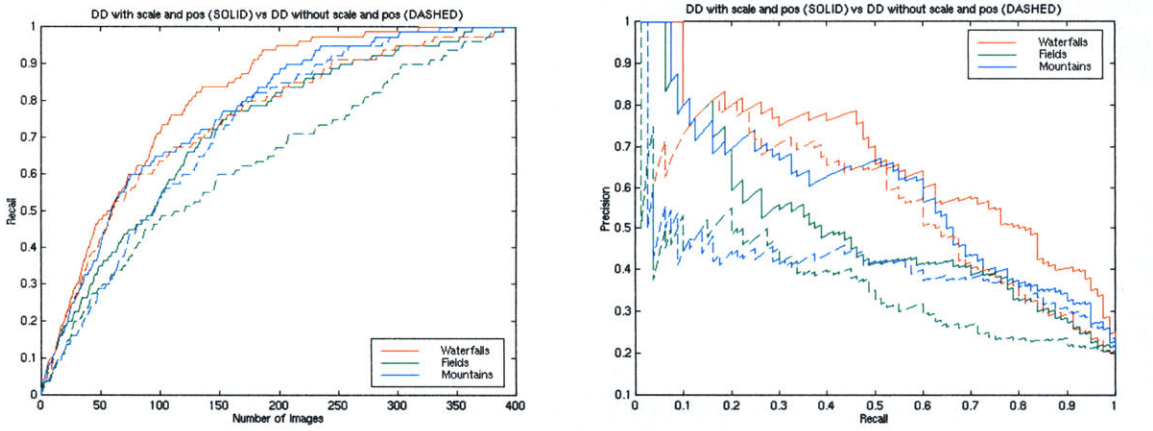


Figure 3-15: Comparing performance of DD (solid curves) with discrete intersect with no position or scale optimization (dashed curves)

3.6 Our Indexing System

We have built an indexing system which retrieves images of natural scenes from a few positive and negative examples. The system uses the Multiple Instance Learning paradigm to learn scene concepts in the form of relational templates from the examples. The system presents the user with a random set of images from which the user selects some positive and negative examples. The system then extracts the class concept using the framework we described in this chapter and retrieves images from the database that satisfy the concept. The user can then modify the query in suc-

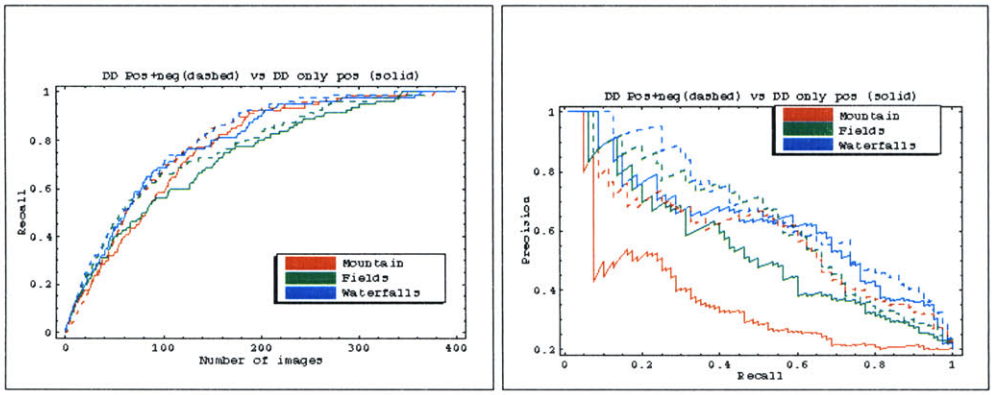


Figure 3-16: Comparing performance of DD with only positives (solid) vs positives and negatives (dashed)

cessive iterations by adding more true positives from the retrieved set and/or adding the false positives as negative examples in order to further refine the concept learned and improve the performance of the system.

3.7 Results on the Corel database

Figure 3-19, figure 3-21 and figure 3-20 show the results of running the indexing system on different queries. In each of these figures, we see the set of positive and negative examples selected by the user and the top 30 images retrieved by the system form a database of 3000 assorted images from the COREL database.

3.8 Results on Greenspun Photo Collection

We also ran the system on another collection of photos from the Greenspun stock photo database. This collection has 5849 assorted images of people,places and events. Figure 3-22, figure 3-23 and figure3-24 show the results of running the system on the "mountains", "sunsets" and "bear in waterfall" queries respectively. The figures show the positive and negative examples selected by the user and the top 20 images retrieved. Figure 3-25 show the recall and precision curves for the mountain and sunset queries. These class labels were obtained manually by going through the database for the purpose of evaluating the performance of the system. There were 92

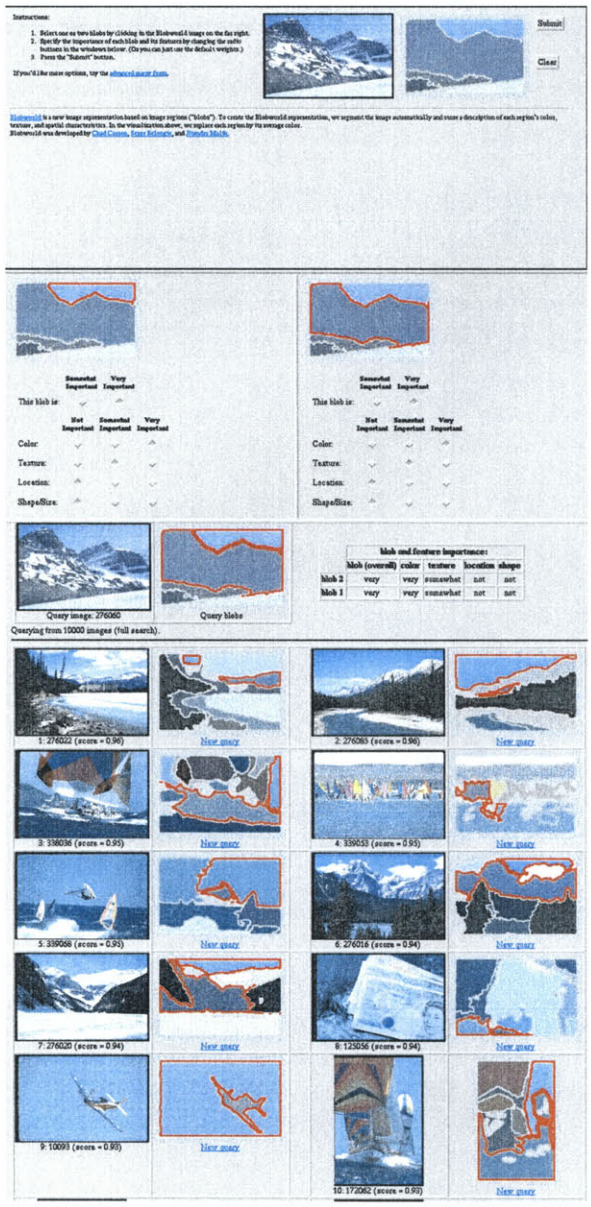


Figure 3-17: Blob World: Interface and Results for a snowy mountain query. These results were taken from the Blob World demonstration program at http://elib.cs.berkeley.edu/photos/blobworld/

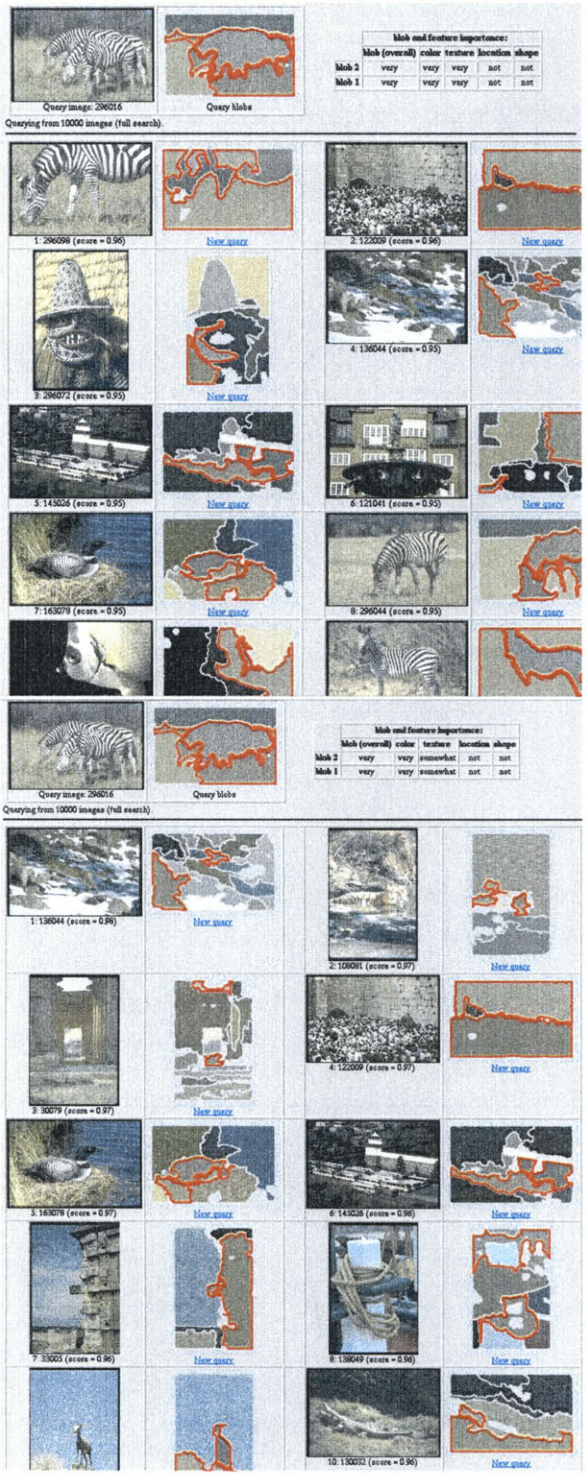


Figure 3-18: Blob World: Results of the system vary significantly when small changes are made to the features selected. These two results for the zebra query have texture selected to be very important in the first case and somewhat important in the second case with all the other features remaining the same. These results were obtained by running the Blob World demonstration program at http://elib.cs.berkeley.edu/photos/blobworld/

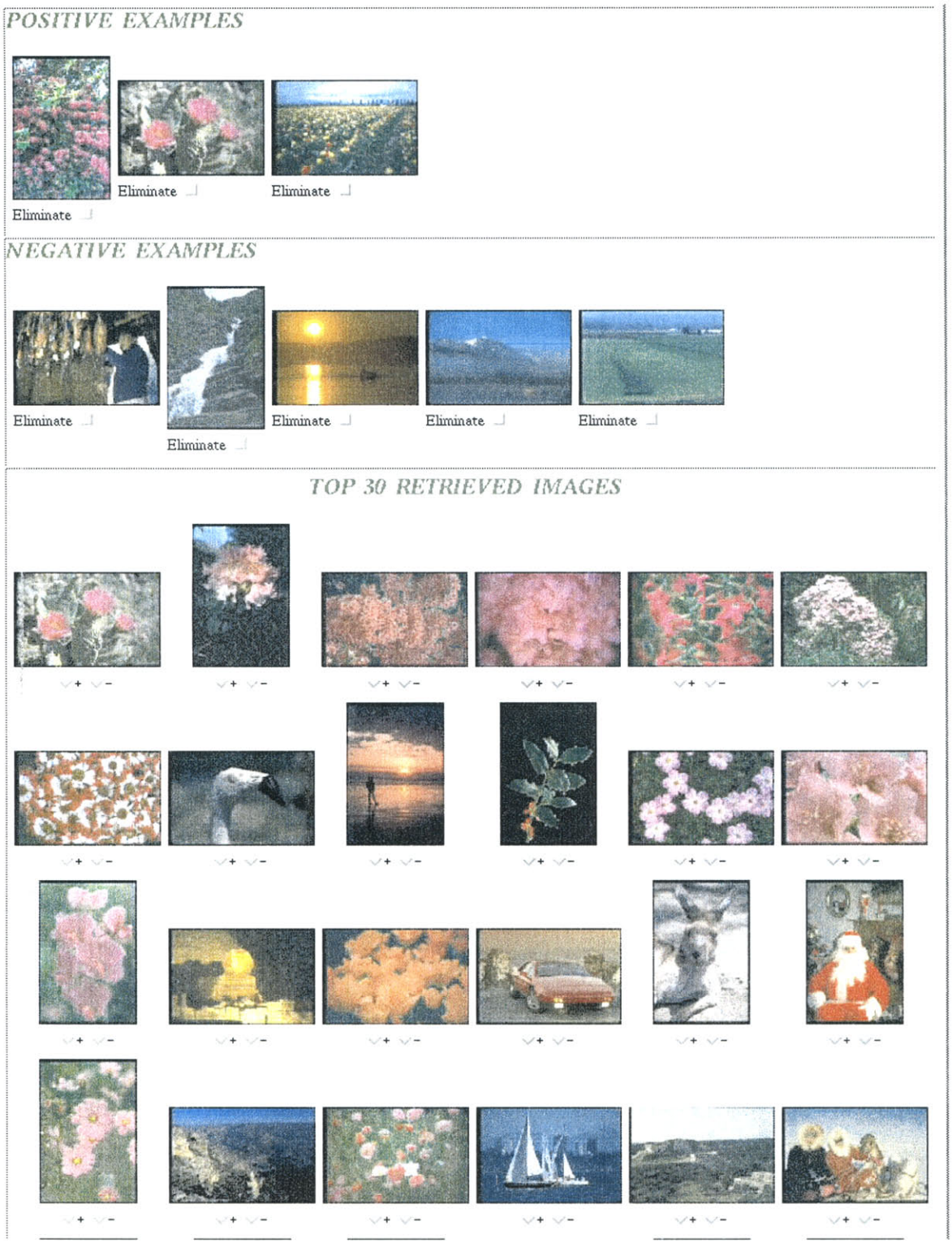


Figure 3-19: Results on the flower query.

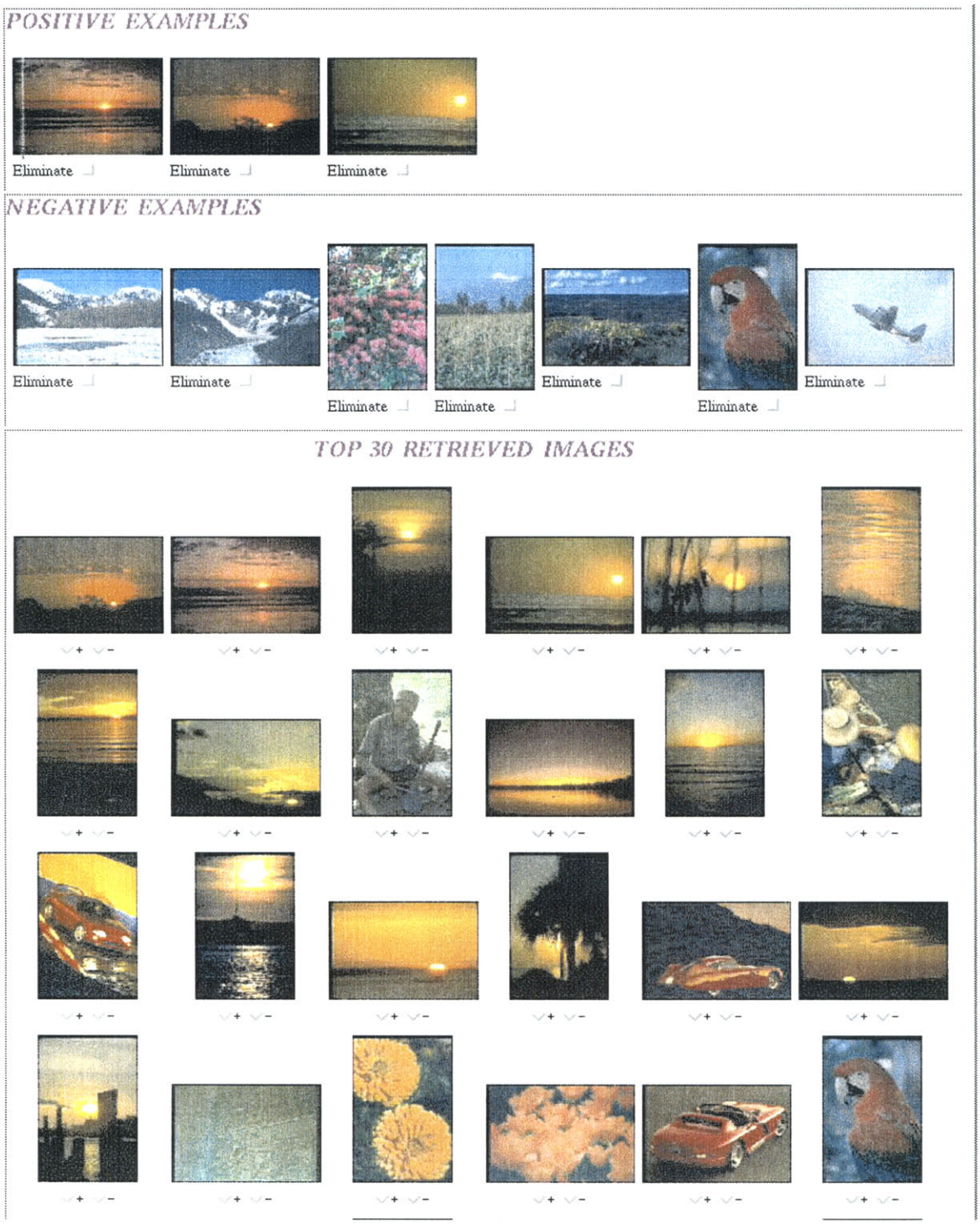


Figure 3-20: Results on the sunset query.

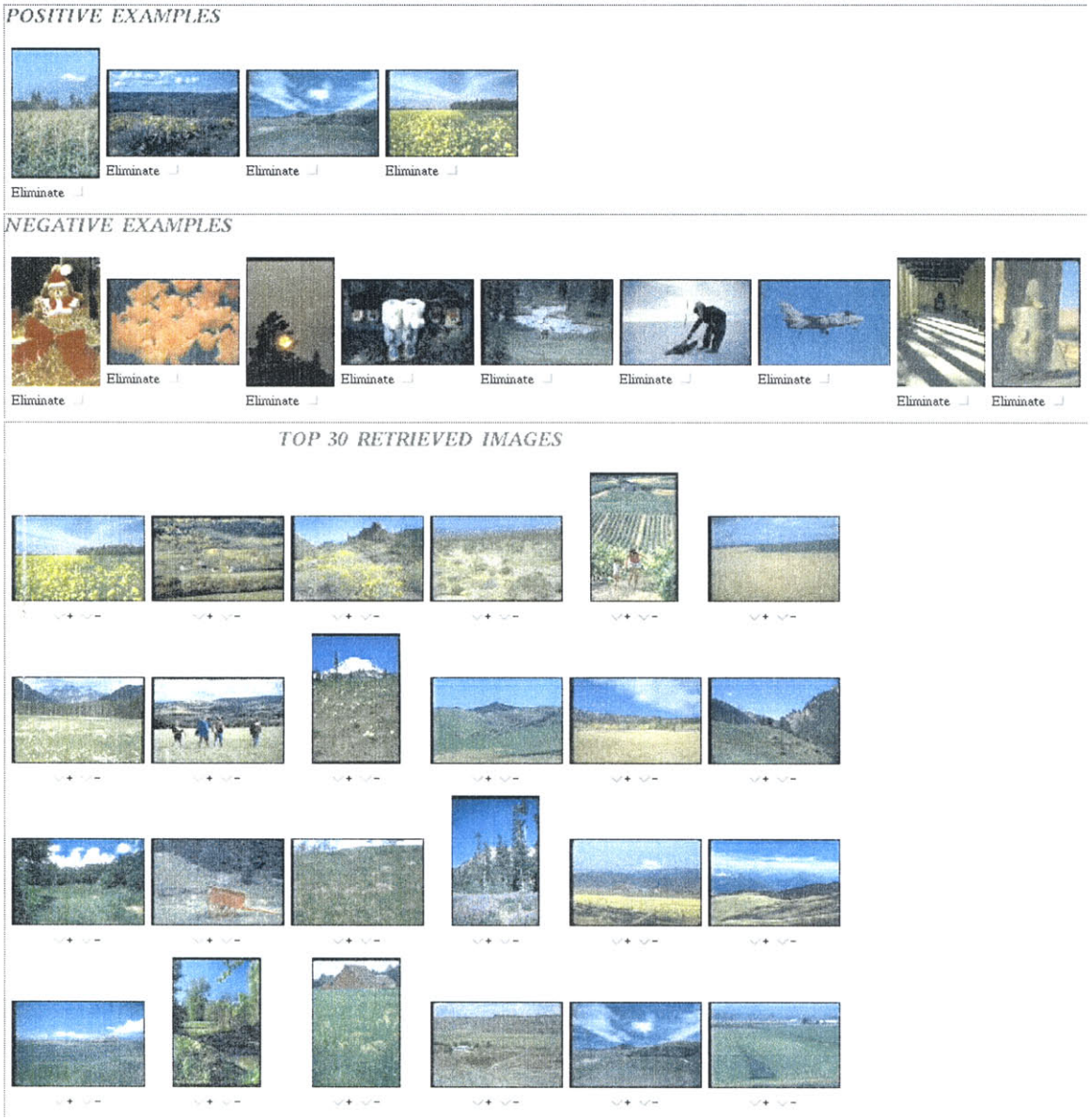


Figure 3-21: Results on the fields query.

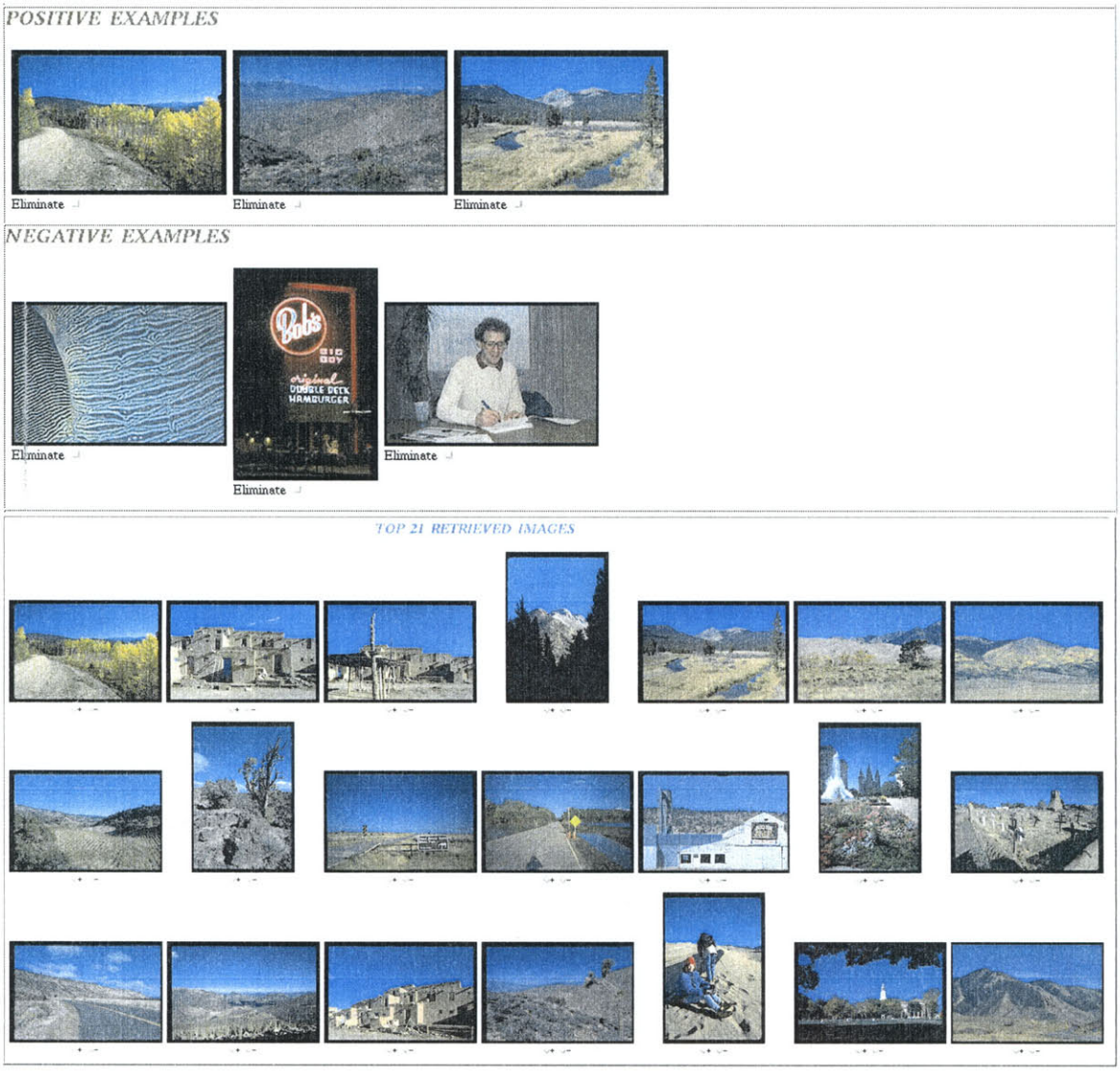


Figure 3-22: Results on the mountain road query.

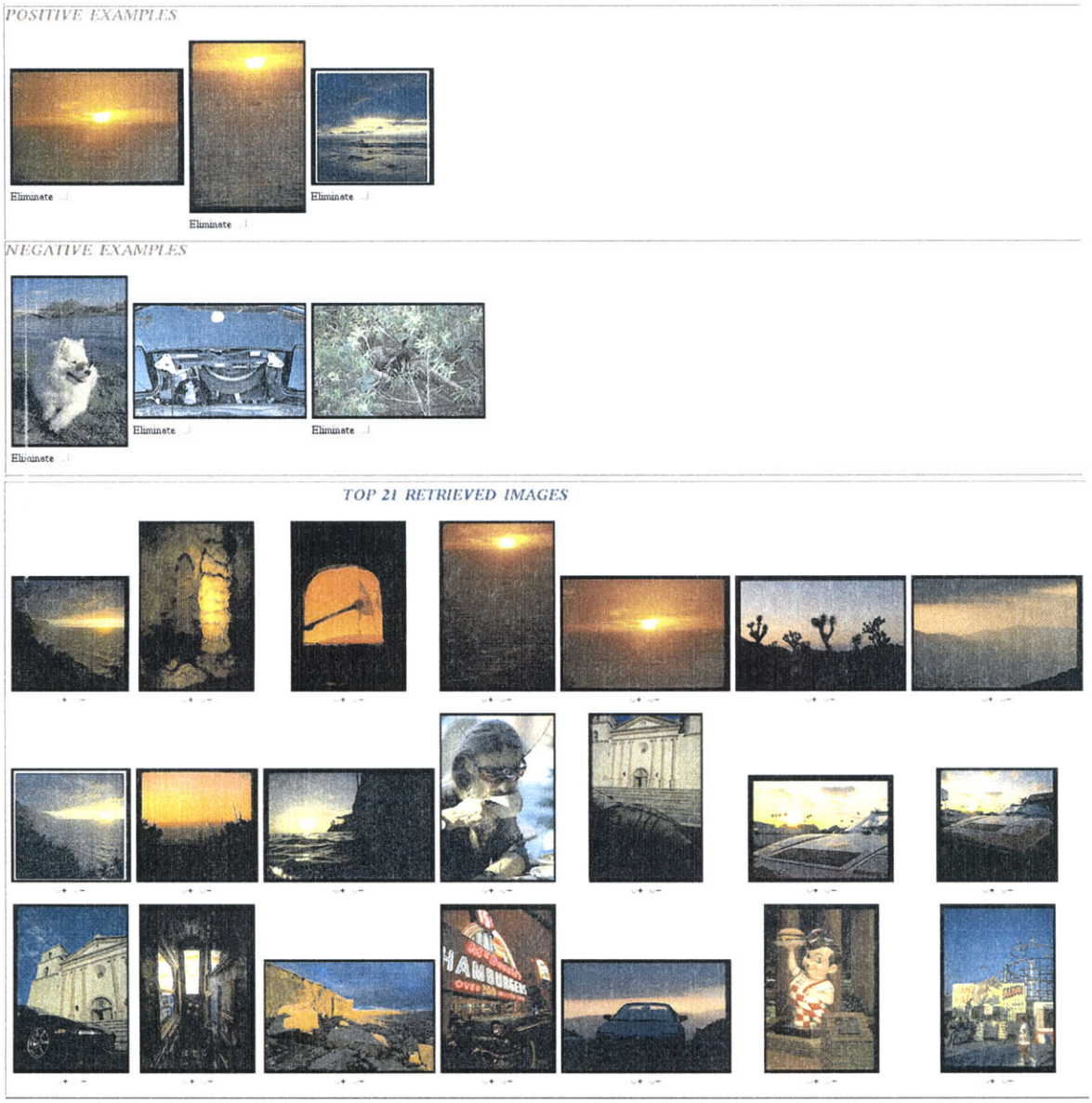


Figure 3-23: Results on the sunsets1 query.

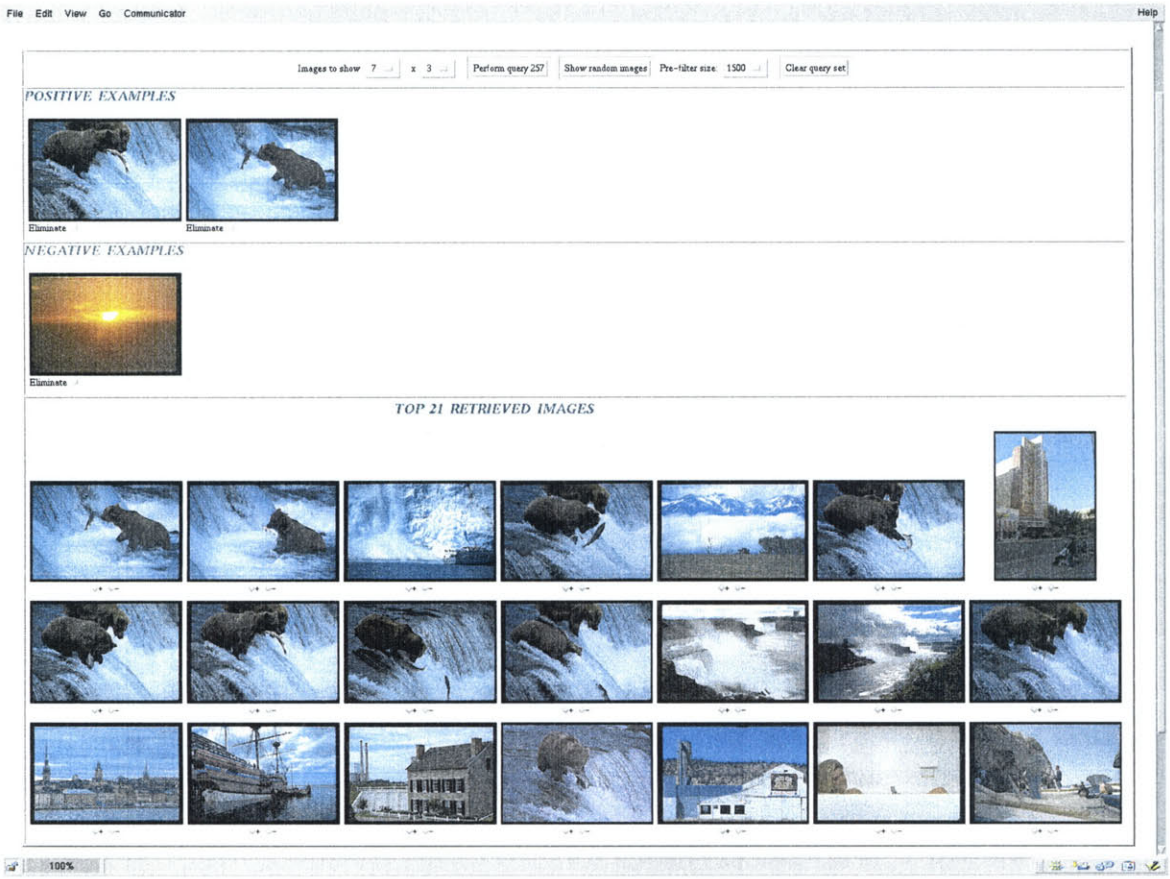


Figure 3-24: Results on the "bears in waterfall" query.

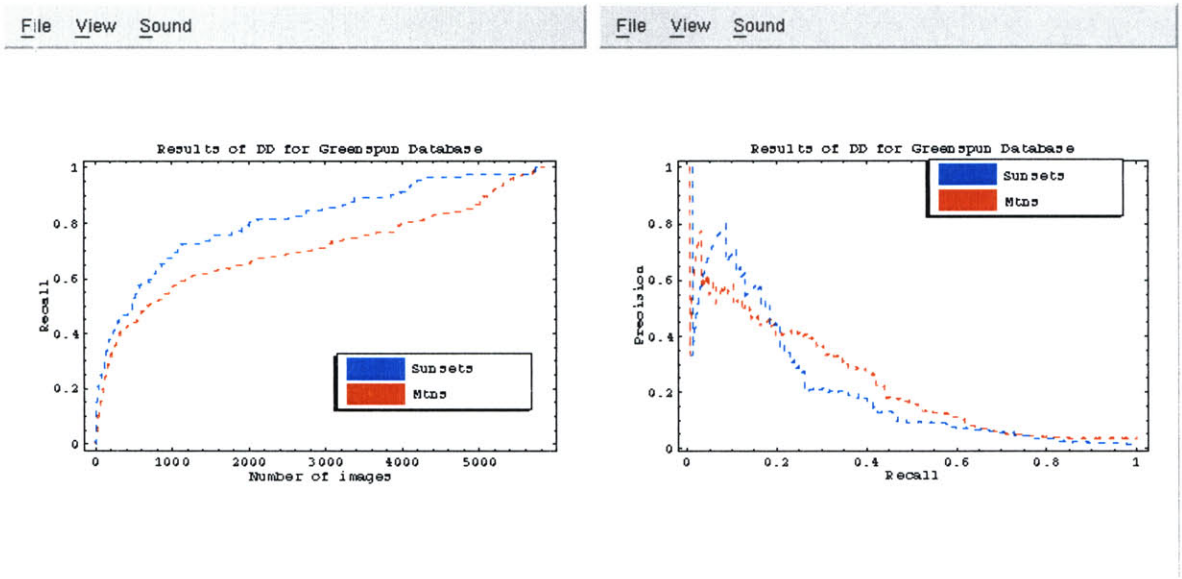


Figure 3-25: Recall and precision recall curves for the waterfall and sunset queries on the Greenspun database of 5849 images.

House in the Sunset District of San Francisco, California.

from Photo CD 0923

canned HTML for use on your site

(a)

Sky.

from Photo CD 4229

canned HTML for use on your site

(b)

Water falling off the cliffs near the glaciers on the west coast of the South Island of New Zealand.

from Photo CD 1645

canned HTML for use on your site

(c)

Figure 3-26: Examples from a text keyword based search on the Greenspun database for 3 queries. (a) "sunset" (b) "cloudy skies" (c) "coast"

sunsets and 222 mountain images using this labeling.

While text based retrieval based on captions can be used to index into collections like these, text is often insufficient by itself for this task. Annotating every photograph is time consuming and every image can have a variety of interpretations which the annotator has to describe exhaustively. Figure 3-26 shows some examples of such captions from running a text based search on the same database. Thus, it is useful to have an automatic way to search through images based on visual content in addition to textual keywords.

3.9 Other Systems

In this section, we discuss some of the other existing image database indexing systems.

3.9.1 Blob World

Blob-World is an image representation based on image regions ("blobs") developed by Belongie et al. [Belongie *et al.*1998]. In this representation, the image is segmented into regions based on color and texture characteristics. Figure 3-17 shows the blob-world representation where each region is replaced by its average color for visualization purposes. Figure 3-17 also shows the query interface. The user picks one image from a random set and is then presented with the query image and its blob-world representation as shown in the figure. The user then picks one or two blobs from this new representation and specifies the importance of each blob and its features (color, texture, shape and location) from a set of preferences. The system then uses the selected blobs and the feature weights selected for each blob to retrieve other images that are closest to it based on the user's preferences. Figure 3-17 shows the results to the earlier query.

In a querying task, the user can access the regions directly in order to see the segmentation of the query image and specify which aspects of the image are central to the query. When query results are returned, the user sees the Blob-world representation of the returned images; this assists greatly in refining the query. While

these features are all useful in the query and refinement stages of database indexing, the system depends on the user for input on the weights of the different features. Different combinations of feature weight selections can result in very different results as we can see in Figures 3-18. In these figures, the input image and specified blobs are the same but the feature weights for color and texture are different and this leads to very different results. In many applications, it would be valuable to have a system that can automatically extract the regions that are salient and select the relevant features for these regions based on a small set of user-selected examples. Our system addresses these issues of query learning and refinement. Since images are inherently ambiguous and it is hard to say what is salient in an image given one example and no prior knowledge, we use a small set of examples instead of one to automatically extract those regions and features that are salient in the examples. These salient regions and their relationships are learned as the target concept which can then be used to retrieve other images that belong to the same concept-class.

3.9.2 Alta Vista - Photo Finder

This system[2] uses a combination of text annotations and image content to retrieve new images that belong to the user specified class. The first stage of the query engine is text based and the system crawls the web to bring up images which have the user specified keyword either in its name or description. The second stage allows the user to pick one of the returned images as a query in order to return other images that are "visually" similar to the query image. This system also uses a single image as the query and uses a combination of color and texture properties of the image for classification. The system does not explicitly use any spatial information between image regions. The idea of using text and image content in a two-stage retrieval process is a nice one since text alone might not be sufficient especially if the concept cannot be described easily by a keyword. Text-based image retrieval alone can also result in false positives when the description of the image contains words that are not

[2]http://www.altavista.com

associated with the image.

3.9.3 QBIC

The QBIC system [Flickner *et al.*1995] provides an interface where the user can select a query image from a set of random images. The user can also choose one of a few options for the cueing/retrieval technique. These options are color composition (color histograms), color layout and texture. Different cueing techniques (features) can give very different results for the same query. It would also be useful to allow combinations of features like Blob-World does [Belongie *et al.*1998] instead of asking the user to pick one of the allowable features (e.g. color histograms or texture or layout). In this application, it would be valuable to have the system automatically learn the relevant combination of features from example images.

3.9.4 Complex feature based indexing

The approach proposed by DeBonet et al. in [J.S.DeBonet and P.Viola1997] uses both the first order properties (color, edges etc.) and the relationships between the first order properties. The process of finding relationships between image regions with local properties generates features which are more complex and more specific. They show that these complex features are more discriminating and can be used effectively to retrieve images that are structurally similar to the query (e.g. crowds of people, gothic cathedrals etc.). Given a set of positive and negative examples, the search is performed by comparing the signature of the new image to the mean signature of the query images. The importance of each element is inversely proportional to its variance across the examples. This approach is complementary to the flexible template based approach that we focus on in this thesis. The flexible templates model color properties in regions and the relationships between regions. These templates are good at capturing the global configuration of scenes by explicitly using the spatial relationships between relevant image regions while the complex features can specify structural queries that do not depend on global scene properties.

3.10 Summary

In this chapter, we have demonstrated a general system for query learning and for image classification using a small number of examples. We describe an architecture that allows the user to train the system, by selecting positive and negative examples, letting the system create and use an initial template based on those examples and finally refine the template by adding incorrect matches (false positives). Our approach to training indexing systems treats query learning as a Multiple Instance Learning problem and builds on the method of Diverse Density. The system has been tested for a few classes on a database of 2600 images of natural scenes from the COREL photo library. The scenes in the database include "sunsets", "coasts", "fields", "mountains and glaciers" among others.

We show that Multiple-Instance learning by maximizing diverse density can be used to classify images of natural scenes. Our results are competitive with hand-crafted models, and much better than a global histogram approach. We have also demonstrated that simple learned concepts that capture color relations in low resolution images can be used effectively in the domain of natural scene classification. Our experiments indicate that complicated concepts (e.g. disjunctive concepts) tend to have better recall curves and that user interaction (adding false positives and false negatives) over multiple iterations can improve the performance of the classifier. Our experiments also show that optimizing for position of the target concept (in the continuous domain) works better than discrete intersection for classes that have variations (e.g. fields) and need the target concept to be general enough to accommodate the variations. We also note that the role of position and scale optimization for the features depends on the positive query set used and the amount of generality needed to describe the class adequately. For example, in the case of queries that have very little variation, discrete intersection without any feature selection works as well as Diverse Density.

3.11 Discussion

In this section, we discuss the possible extensions of our current classification system using diverse density.

- Speed

  As we mentioned earlier, the time taken to learn a concept ranges from a few seconds to a few days. We have used this prototype system to demonstrate that DD can be used to learn scene concepts which can be used effectively in classification. In the next chapter we introduce a more intelligent method of generating instances (for example, a rough segmentation using connected components) will reduce both the number of instances and the running time by orders of magnitude.

- Classification Performance

  As we saw in the earlier section, the classification results of the prototype system are encouraging but we would like to investigate ways of improving the classification performance. The current framework extracts the simplest concept that separates the positive examples from the negatives. While it is doing the correct thing from a machine learning point of view, we have run some tests on hand-built concepts that indicate that the simplest concept is not the most discriminating one that we would like to have for classification. We discuss extensions to the current framework in the next chapter by using a combination of cues to describe each instance so that the extracted concept does not overfit in order to return the simplest concept.

  Example selection plays a crucial role in classifier performance. Ideally, we would like to have a large number of training examples that give us a dense and comprehensive sampling of the input space. We can expect to get better classifier performance by generating representative positive and negative examples rather than randomly sampling to get the training set. Our experiments show that adding negative examples definitely helps classifier performance. However,

it can be extremely difficult to obtain a small and relevant sample of negative examples as training data. The number of negative examples can also grow very large without a reasonable selection strategy. One such strategy is to pick the "near misses" from initial classification as the negative examples. In this case, the system will return the concept that separates the positives and negatives (false positives that confused the system) which in turn will be a more discriminating concept that reduces the number of false positives in future iterations. We discuss a few ways to extend the current system and improve classifier performance below.

1. Iterative refinement

   The learned scene concept can be refined with user interaction. The user can select some of the returned images or select some new positive and negative examples. The system can refine the existing concept by adding and modifying the relations in the template so that the new concept to explain the new set of examples.

2. Feature Selection

   The Diverse Density algorithm finds the target feature vector t and the feature weight values w which maximize Diverse Density. From our experiments, we see that the algorithm tends to push most of the weights to zero leavin very few large weight values. This tendency to overfit leads to the simplest concept that separates the positive examples from the negative examples. It might not be desirable to overfit in the domain of image classification since we have very little training data and such a simple concept might not have the power to discriminate between classes in the larger test set. One way to modify the algorithm to overcome this overfitting problem is to force all the weights to be 1 (not perform any feature selection). As we discussed earlier in this chapter, this modification performs well in the domain of natural scene classification with our simple color features. However, if we had more diverse features (e.g. color and

texture), the initial feature space might not be the best one to separate the positives form the negatives. Cheng [Yang1998] discusses an alternate approach to controlling the feature weights. He constrains the weight factors during the optimization process. He controls the change in weights w_k by setting a lower bound for the sum of weights and shows that the constraint improves classification performance. We would like to explore other constraints based on the selectivity of features. We plan to estimate the selectivity of concepts (starting with a single blob concept to more complicated k-blob concepts) on a large, unlabeled database. Since the database is large and diverse enough, we assume that it is fairly representative of the set of digital images. Weighting learned concepts by their selectivity could help improve the discrimination power of the concept and hence improve classification performance.

Chapter 4

Extending the framework: Learning object concepts using complex instances

4.1 Introduction

In chapter 3, we introduced the Multiple-Instance Learning framework for the domain of natural scene classification and demonstrated that the Diverse Density algorithm can be used effectively to learn templates for a range of natural scene classes. We also used it to explore the effects of positive and negative feedback on the queries.

In the domain of natural scenes, the predominant color and spatial relations explained most parts of the image and were preserved even in very low resolution images. For many other types of queries, we might want to retrieve new images based on smaller sub-regions of the query image. For these queries, we need to preserve some detailed high resolution properties of image sub-regions for classification. In this chapter, we explore a few extensions to the previous system:

- we start with high resolution images and segment them into a few coherent connected components,

- we use the segmented regions to generate complex instances [1] (2 and 3 blob conjunctions) where each instance is a point in feature space and is a possible description of what the image is about,

- we show that the new system can be used to extract and retrieve images of object classes like cars in addition to images of natural scenes like fields and waterfalls described in Chapter 3. While natural scene concepts generally spanned the entire image and the color and spatial properties were preserved in low resolution images, object concepts typically tend to occupy only a small portion of the image we need to preserve some high resolution properties of the sub-images in order to classify them.

We test our approach on images from the COREL photo library. We begin by extending the flexible template representation introduced by Lipson [Lipson *et al.*1997] to the class of cars and then describe how one can learn this template from examples using Multiple Instance Learning.

4.2 Flexible templates for Car Detection

As we discussed in Chapter 2, the original flexible templates constructed by Lipson [Lipson *et al.*1997] used color and other photometric relationships between image regions. Here, we extend this template representation to incorporate simple geometric properties. As proof of concept, we construct a template for the class of "cars" which uses color and simple geometric properties of image regions. Figure 4-1 shows the template and illustrates how it can be used to represent the relevant properties of cars. The car template is made up 6 regions (A,B,C,C1,C2,D1,D2,E) and relations between them. Some of these regions have color properties and others have color and geometric properties (wheel instances in this case). The geometric properties are defined on on high resolution images (384x256) and the color relations are defined on low resolution images (48x32). Specifically, the template has the following relations.

[1]The definition of the instance is the same as the one described in Chapter 3 when we introduced the Multiple Instance Learning framework

- Color relations defined on low resolution images:

  $A(col) = C1(col)$

  $B(col) = C2(col)$

  $D1(col) = D2(col)$

  $D1(col) \neq C1(col)$

  $D2(col) \neq C2(col)$

  $C(col) \neq E(col)$

- Shape properties on high resolution images:

  D1 and D2 contain instances of "wheels" where the wheel is detected in the image using shape properties.

Color properties alone are not sufficient to represent the class since there could be other objects in the image that satisfy the same color relations. Using only the geometric properties (wheels) can lead to many false positives as well as seen in the Figure4-1. For example, we see in the figure that non-wheel regions like the sun is detected as a wheel by the circle detector. The combination of these two cues, however, captures the essential properties for a wide variety of cars with varying size, pose and shape. We use the Hausdorff matching technique [Huttenlocher *et al.*1993, Rucklidge1994] to match the geometric properties in the template (circle) and edge features extracted from an image. We will describe the the wheel detection algorithm and the template matching process below.

4.2.1 The Hausdorff distance for matching shapes

Given two sets of points $A = a_1, ..., a_m$ and $B = b_1, ..., b_n$, the Hausdorff distance [Huttenlocher *et al.*1993, Rucklidge1994] is defined as

$$H(A, B) = max(h(A, B), h(B, A))$$

where

$$h(A,B) = \max_{a \in A} \min_{b \in B} \| a - b \|$$

where $\| a - b \|$ represents Euclidean distance.

The function $h(A,B)$ is called the *directed* Hausdorff distance from A to B. It identifies the point $a \in A$ that is farthest from any point in B, and measures the distance from a to its nearest neighbor in B. Thus, $H(A,B)$ measures the degree of mismatch between the two sets of points. For example, if the Hausdorff distance is d, then every point of A must be within a distance d of some point in B and vice versa. A nice property of the Hausdorff distance is that it allows for the image to contain many instances of the model, or no instances. It also accommodates transformations of the model (2 dimensional translation, rotation and scaling) and partial occlusion.

In our application, we use an extension of the Hausdorff distance [Rucklidge1994] which defines a fraction of points in set A that lie within c of some point in B.

$$F_c(A,B) = min(f_c(A,B), f_c(B,A))$$

$$f_c(A,B) = \frac{\sharp(A \wedge B^c)}{\sharp(A)}$$

where B^c is the point set B dilated by c. $f(A,B)$ is the fraction of points in A that lie within c of some point in B. The Hausdorff fraction maximizes the fraction of model edge points that lie near image edge points.

4.2.2 Wheel detection using the Hausdorff fraction

The model template that we use to detect "wheels" in images is a bitmap of a circle. Given a new image, we smooth it using a Gaussian filter and detect edges using the Canny edge detector [Canny1986].

We use the Hausdorff fraction to detect "wheels" in the image since it (a) tolerates errors as well as the presence of extra or missing data points between data sets and (b) operates on an arbitrary, user defined transformation function. In the case of

wheel detection, we search for the circle template over 2 dimensional translation and scale. The ability of the Hausdorff distance to accommodate these transformations is useful since it accommodates deformations in the wheel (from a circle to an ellipse) due to change in pose and viewpoint.

To match the model (circle template) with the image, we find all instances of circle/ellipses in the image that have a with a Hausdorff fraction above a certain threshold (currently 85%). The Hausdorff matcher gives the location (x, y) of each instance and the scaling of the model in x and y directions.

Figure 4-1 shows an example of the wheel detection using the Hausdorff fraction.

4.2.3 The car detection algorithm

We now describe the steps involved in matching the car template to a new image in order to determine if the image has an instance of a car in it.

Start with a high resolution (384x256) image and a model image (bitmap of a circle). Smooth the image using a Gaussian filter and detect edges in it using the Canny edge detector [Canny1986]. Match the circle model with the image to search over all 2 dimensional translations and scales to find all instances of circles/ellipses in the image.

For each detected instance of the circle, we have the location (x, y) in the high dimensional image, the scaling in the x and y directions and the size of the model template (m_x, m_y).

For each pair of detected circles in the image

- find the corresponding locations in a low resolution 48x32 image $(a1, b1)$ and $(a2, b2)$
- find the bounding boxes $D1$ and $D2$ for the wheels using the location $(a1, b1)$, scaling and model size (m_x, m_y)
- find the mean color [R,G,B] of (1) the wheel regions ($D1$ and $D2$), (2) the region between the wheels (C), (3) the region below C, (4) sub-region of C adjacent

to $D1$ ($C1$), (5) the sub-region of C adjacent to $D2$ ($C2$), (6) the (3x3) region above $D1$ (A), (7) the (3x3) region above $D2$ (B)

- A car is detected if the following color relations are satisfied: $A(col) = C1(col)$
  $B(col) = C2(col)$
  $D1(col) = D2(col)$
  $D1(col) \neq C1(col)$
  $D2(col) \neq C2(col)$
  $C(col) \neq E(col)$

Figure 4-1 shows an example of the detection process using the car template. Figure 4-2 shows the detection performance on a random set of images form the COREL library.

An alternate way to match the template to the image would be to find all instances in the image which satisfy the color and spatial constraints and then verify the existence of a car in a hypothesized location with the wheel detector. In our experiments, we found that, for the Hausdorff wheel matcher, the false positive rate was higher than the false negative rate i.e. the matcher always found the wheels but also detected circles/ellipses in non-wheel regions. There were also only a few wheel regions detected in every image. Thus, we used the color and spatial constraints to weed out the false matches from the shape matcher instead of using them to hypothesize car instances.

We would now like to learn such a representation from examples. In the next few sections, we discuss how the learning system from Chapter 3 can be extended to extract complex templates using multiple cues (e.g. the car template) from examples.

4.3 Extending the Bag Generator

As we saw in Chapter 3, a key part of our system is the bag generator: a mechanism which takes an image and generates a set of instances, where each instance is a possible

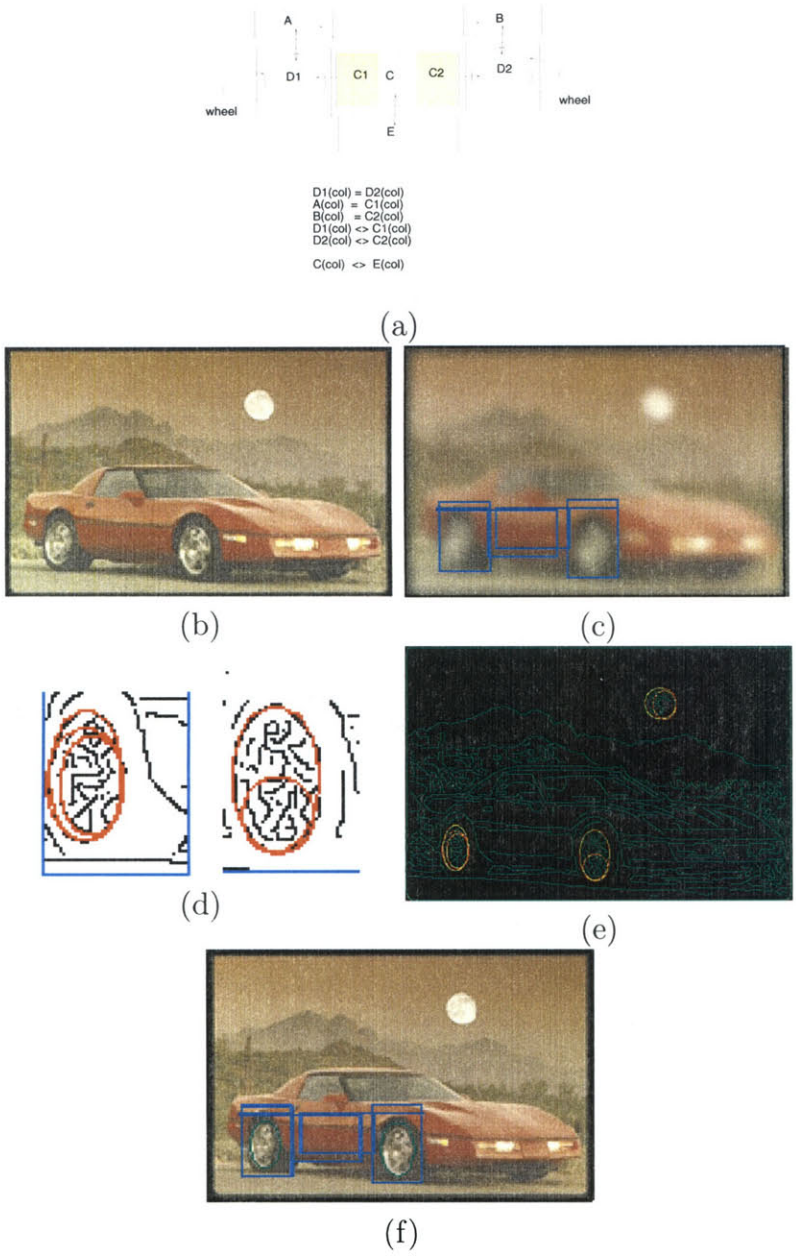


Figure 4-1: (a) Car template (b) Original image (c) region selection using color and spatial relations (d) and (e) wheel detection using hausdorff matching (f) final results superimposed on the original image

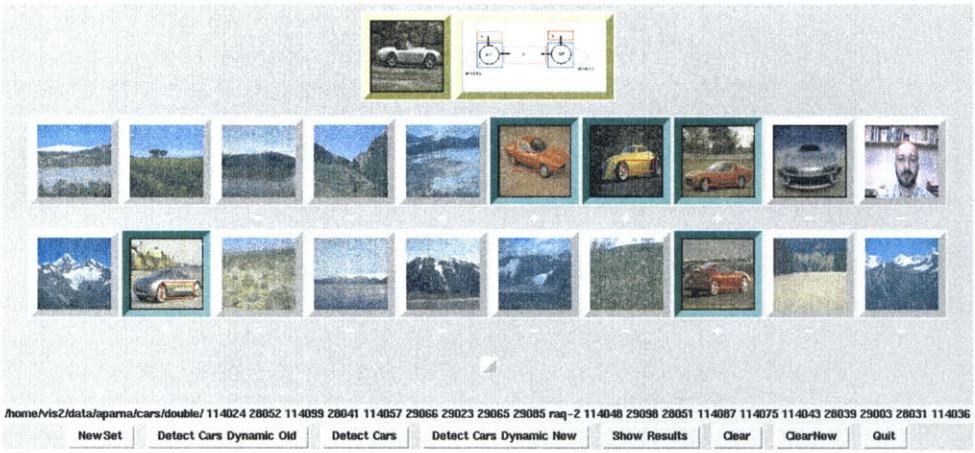


Figure 4-2: Car detection results on a set of 20 random images. The images with cars are highlighted in green.

description of what the image is about.

The instances in a particular bag are various subimages in a very low resolution image (8x8). Each of the instances, or subimages, is described as a point in some feature space. For the (`single blob with neighbors`) instance: a subimage is a 2x2 set of pixels (referred to as a *blob*) and its four neighboring blobs (up, down, left, and right). The subimage is described as a vector $[x_1, x_2, \ldots, x_{15}]$, where x_1, x_2, x_3 are the mean RGB values of the central blob, x_4, x_5, x_6 are the differences in mean RGB values between the central blob and the blob above it, etc. One bag is therefore a collection of instances, each of which is a point in a 15-dimensional feature space. Even with these very low resolution images, the number of instances grows very large especially when we consider conjunctions. For example, let us consider a 2-blob conjunction. For the `two blob with neighbors` concept, each instance is the mean color of two descriptions of two `single blob with neighbors` and their relative spatial relationship (whether the second blob is above or below, and whether it is to the left or right, of the first blob). An instance (i, j, k, l) is given by the vector $(x_1, ..., x_{32})$ where $(x_1, ..., x_{15})$ is a description of one single blob with neighbor instance, $(x_{16}, ..., x_{30})$ is a description of the second single blob with neighbor instance which can be in any direction relative to the first (x_{31}) and (x_{32}) gives the distance

between the blobs $(i-k)$ and $(j-l)$ respectively.

Each instance is a 32 dimensional vector and there are $36^2 = 1296$ instances per image. As we mentioned before, we double the number of dimensions in order to find both the location and the scaling of the target concept. Performing gradient descent in a high dimensional space (64 dimensions) with this many instances can take a long time with more complicated concepts (conjunctions and disjunctions). In this chapter, we use a bag generator that generates subregions with color, texture and simple shape properties, which limits the number of instances per bag and therefore improves the efficiency of an algorithm such as Diverse Density. We show that by using a more complex representation and features within the same framework, we can learn certain object classes (e.g. cars) in addition to natural scene classes. Our experiments on "car" classification show that a combination of cues (color, texture and simple shape), some feature selection, and more complicated concepts (conjunctions) play a significant role in improving classifier performance for this class. We also show that rough segmentation of high resolution images into salient connected components greatly reduces the number of instances and the running time of the algorithm especially when low resolution pixel-instances are not sufficient (e.g. when we move from the domain of natural scenes to objects).

4.4 Segmentation

In the domain of natural scenes, the predominant color and spatial relations in the target concept captured the global configuration of the image and were preserved even in very low resolution images. For many other types of queries, we might not be interested in the whole image and might want to retrieve new images based on specific parts of the query image. For example, we might want to retrieve images that contain a certain object class like cars. For these queries, we need a representation that retains some detailed high resolution properties of image sub-regions in order to be able to (a) extract the query object as the target concept from the positive examples and (b) retrieve new images containing the object.

To do this, the system needs to ignore the background and clutter and select out the object of interest from the positive examples. We extend the architecture described in the previous chapter to generate instances that can capture object concepts. The instances here differ from the ones used in the previous system in that they are segmented components of the high resolution image. These segmented regions have a richer set of descriptive properties and the number of segmented regions (connected components) in the image is small on the order of 10–15. There have been many methods proposed in the literature for segmenting images into multiple regions with coherent properties [Zahn1971, Z.Wu and R.Leahy1993, Shi and Malik1997, Felzenszwalb and Huttenlocher1998, Comaniciu and Meer1997]. We use the method by Felzenszwalb et al. [Felzenszwalb and Huttenlocher1998] to roughly decompose the image into the dominant set of regions using its color properties. Figure 4-3 shows an example of the segmentation results using this algorithm. Each image in these examples has $10 - 15$ components.

The segmentation helps generate instances that correspond to salient regions in the image. This reduces both the number of instances and the running time for more complicated concepts (e.g. conjunctions). For example, let us compare the number of instances for a 2-blob conjunction concept using segmented components vs pixel neighborhood in an 8x8 image. As we mentioned before, each each instance is a 32 dimensional vector and we double the number of dimensions in order to determine the scaling. Each bag has $36^2 = 1296$ instances when we consider pixel neighborhoods in the low-resolution (8x8) image and $15^2 = 225$ instances when we use segemented components. This reduction in the number of instances reduces the running time while performing the gradient descent in the 64 dimensional space to get the location and scaling of the target concept.

4.5 Bag Generator using Multiple Cues

In this section, we will demonstrate how the Diverse Density algorithm can be used to learn a representation for the class of cars. We would like to show that we can

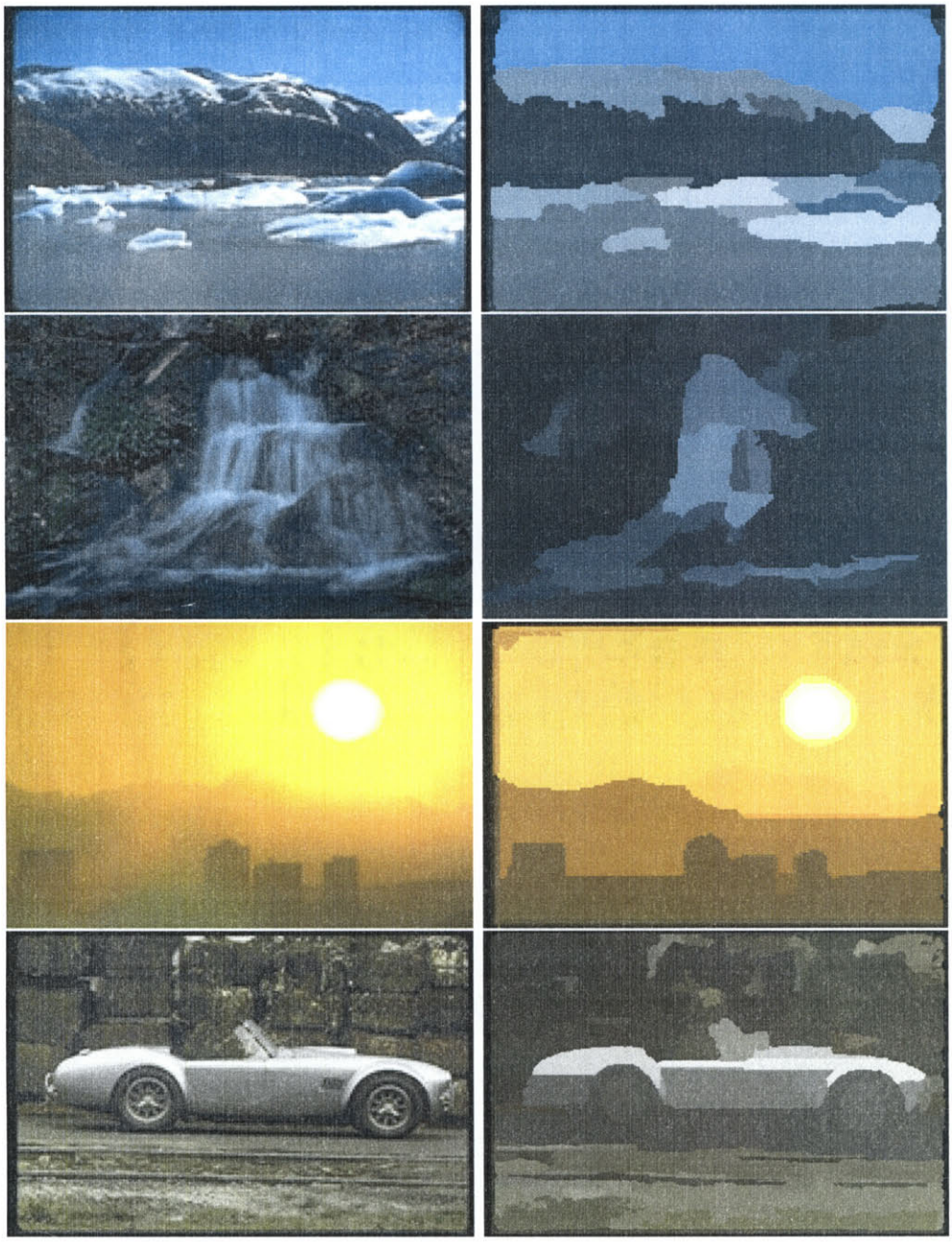


Figure 4-3: Segmentation results on a few images from the COREL database

use the Multiple Instance Learning framework to learn the car template described in section 4.2 which encodes the geometric properties of the two wheels and the color and texture relations between the wheel regions and its neighbors. Such a template could be used to detect cars while accomodating within-class variations in color, texture and pose. We describe a new bag generator which uses a combination of cues (which includes color, texture and simple shape properties) and rough segmentation to generate instances.

4.5.1 Hypothesis Classes

In our experiments, all images were pre-segmented using the algorithm described in [Felzenszwalb and Huttenlocher1998]. All images were also smoothed and edge detected. We tried to learn the concept using the hypothesis classes given below:

- `cc1`: each instance is a 13 dimensional vector $< x_1, x_2, ...x_{13} >$ which describes the color, texture and basic shape properties of a segmented region in the image.

    1. Color properties of segmented components.
       x_1, x_2 gives the position of the centroid of the component, x_3, x_4, x_5 gives the representative color of the component.
    2. Geometric primitives

       We use the Hausdorff matching technique [Rucklidge1994] which was described in section 4.2 to detect instances of a simple shape primitive (circle) in the segmented component. The circles are detected in the edge detected image. As we mentioned in section 4.2, the Hausdorff matcher searches for the circle over all 2 dimensional translations and scale thus detecting ellipses as well as circles. x_6, x_7, x_8, x_9 gives the location and scaling of the circle in the image. x_6 gives the distance to the detected circle, x_7 gives the Hausdorff fraction that says how well it matched the model circle, x_8, x_9 gives the scaling of the model circle in the image.

3. Oriented filter responses

The oriented derivative-of-Gaussian operators have been used as a set of natural basis functions for image segmentation and classification by [J.Malik and P.Perona1990, Manjunath and W.Y.Ma1991] and in image indexing by [J.S.DeBonet and P.Viola1997, Pentland *et al.*1996] among others. The image is filtered using a set of oriented Gaussian derivative basis filters denoted by:

$$G_n^{\theta_n}, n = 3, \theta_n = 0, ..., \frac{k\pi}{n+1}, k = 1, ..., n$$

where n is the order of the filter and θ_n gives the orientation of the filter. The response of an image patch I centered at (x_0, y_0) to a filter $G_i^{\theta_j}$ is obtained by convolving the image patch with the filter:

$$r_{i,j}(x_0, y_0) = \int\int G_i^{\theta_j}(x_0 - x, y_0 - y) I(x, y) dx dy$$

The steerable pyramid is an invertible image representation. It is an oriented transform that decomposes the image into several spatial frequency bands. In addition it divides each frequency band into a set of orientation bands. We use the output of the Steerable Pyramid to get a multi-scale, multi-orientation image decomposition with a 4 orientation band filter set in a neighborhood around the segmented component [Freeman and Adelson1991]. We include these responses to give some measure of texture where the segmented regions have are non-uniform and have a lot of scatter (e.g. snow capped mountains, wheel regions of a car). These filter responses are obtained using the Steerable Filter Software Library designed by Simoncelli et al.[Freeman and Adelson1991, E. Simoncelli and Farid1995].

The features $x_{10}, x_{11}, x_{12}, x_{13}$ give the sum of the squared oriented filter

responses for each of the 4 orientations across 3 scales respectively.

$$S_i(I) = \sum_{\sigma=1}^{3} \sum_{pixels} (R_{\sigma,i}(I))^2$$

where I is the sub-image, i indexes over the 4 orientations, $R_{\sigma,i}(I)$ is the response of the oriented filter F_i at scale σ refers to $\sigma = 1$ is the original I, $\sigma = 2$ refers to I downsampled by 2 and $\sigma = 2$ refers to I downsampled by 4,

Figure 4-4 shows the oriented filters and the image pyramid consisting of the responses of these filters on an image across 3 scales. The pyramid was built using a 4-orientation band filter set which are essentially the third derivatives. Such multi-scale responses of oriented filters have been used successfully in image indexing in [J.S.DeBonet and P.Viola1997]. DeBonet and Viola use a more comprehensive set of filters at various orientations across 3 scales. For each image, they use the first order properties (oriented edges) and the relations between first order properties at several resolutions to get a complex set of features (45000 features) which can then be used to index into the database.

We now summarize the process of generating `cc1` instances. Each image is segmented into coherent color regions. The image is also smoothed and the edge image is computed using the Canny edge detector [Canny1986]. For each segmented component in the image, record the location of the centroid of the component, the representative [R,G,B] color of the component, the results of circle/ellipse detection using the Hausdorff distance on the component and the responses of a set of oriented filters on the component. Each instance describes the location and the rough color, shape and texture properties of a particular component. Figure 4-5 illustrates how the `cc1` instances are generated.

- `cc2` : an instance is the conjunction of two connected components (`cc1` vectors). Each instance is a vector $[x_1, x_2, ..., x_{26}]$ where $[x_1..., x_{13}]$ is a `cc1` instance of the first connected component. x_{14}, x_{15} are the differences in the x, y location of the second component with respect to the first. x_{16}, x_{17}, x_{18} are the color differences (RGB) between the second component and the first, $x_{19}, x_{20}, x_{21}, x_{22}$ gives the location and scaling of the closest circle and $x_{23}, x_{24}, x_{25}, x_{26}$ gives the oriented filter responses around the second component.

4.6 Experiments

While one does not need very complicated hypothesis classes to learn concepts from the natural image domain [Maron and Lakshmi Ratan1998], one can learn a more diverse set of queries (e.g. object queries) by using complex instances which describe a combination of properties. In the next section, we evaluate the performance of the system on four query classes ("fields", "mountains", "waterfalls", "cars") and show that complicated concepts (conjunctions) and feature selection improve classifier performance.

4.7 Experimental setup

We tried to learn each of four concepts ("fields", "mountains", "waterfalls", "cars") using the new bag generator. For training and testing we used natural images from the COREL library, and the labels given by COREL. These included 100 images from each of the following classes: waterfalls, fields, mountains, sunsets, lakes, cars, race-cars. We had a training set of 140 images with 20 images from each of the classes, a small test set of 538 images which was disjoint from the training set, and a larger test set of 2600 images. The training scheme used five positives and five negative examples. We attempted different training schemes: `initial` is simply using the initial five positives and five negative examples. `+5fp` adds the five most egregious false positives after a round of testing on a held-out set of images. `+10fp` repeats the

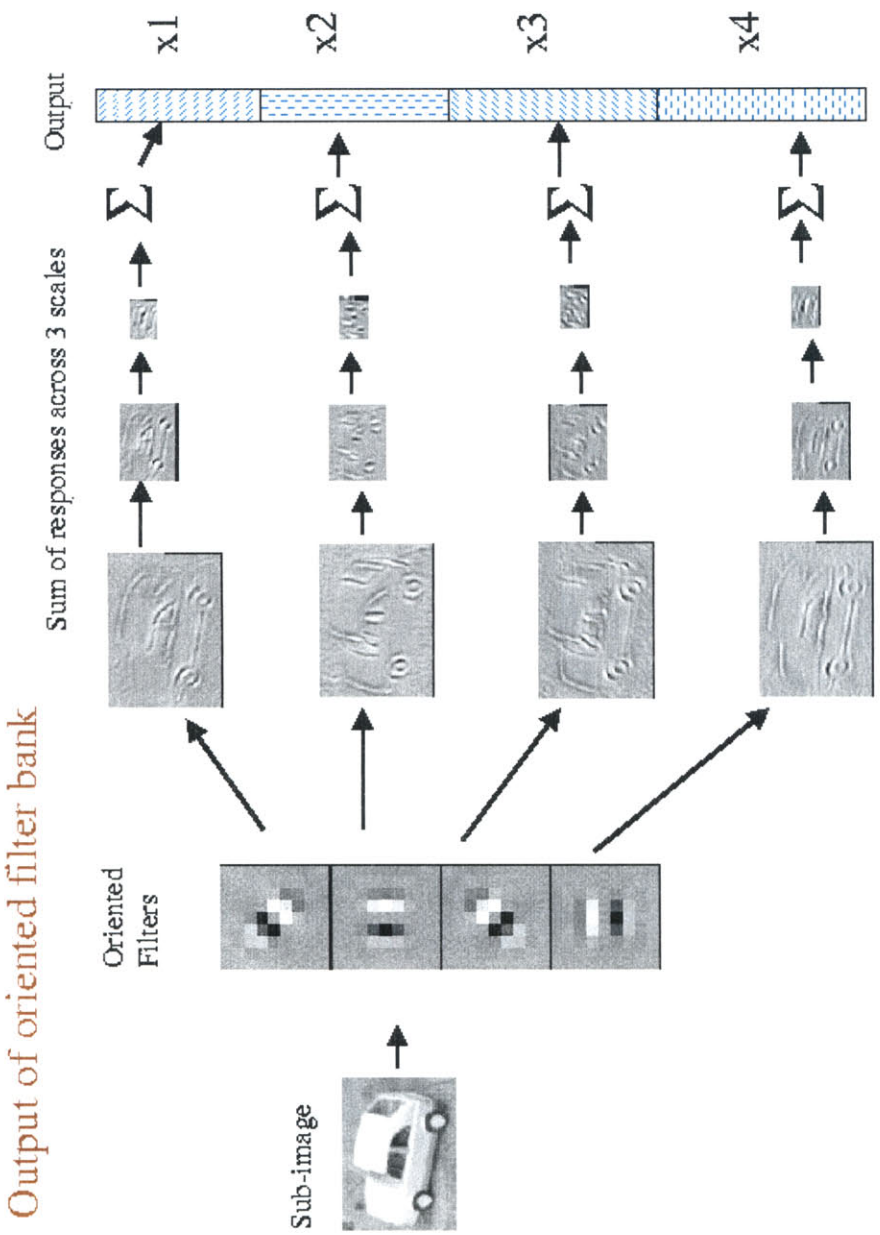


Figure 4-4: Response of an oriented filter bank with 4 oriented filters on an image of a car across 3 scales. The output is a four dimensional vector corresponding to the sum of the filter responses in the sub-image for each orientation across all scales.

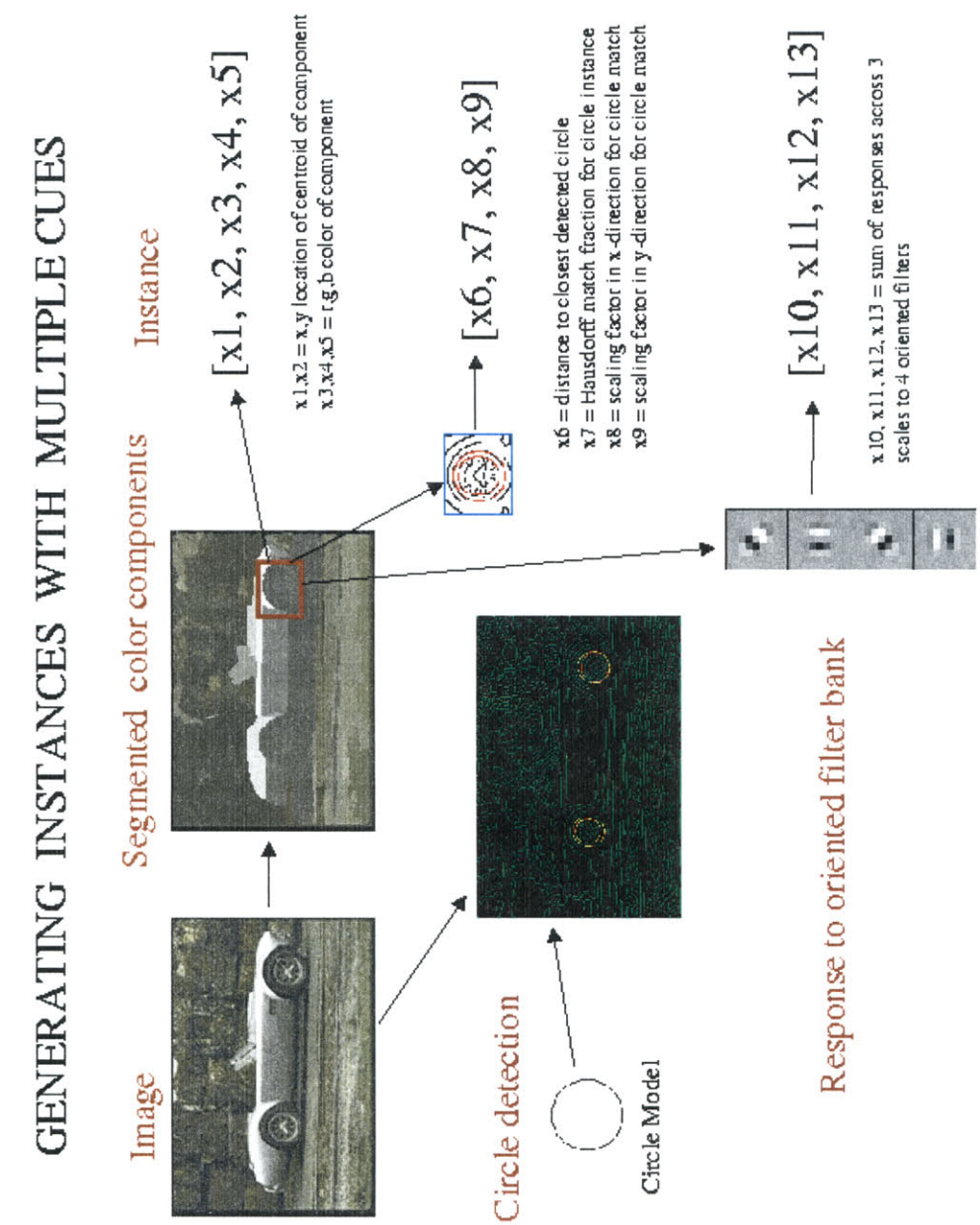


Figure 4-5: The `cc1` instances are generated using a combination of cues (color, texture and simple geometry) on a roughly segmented image.

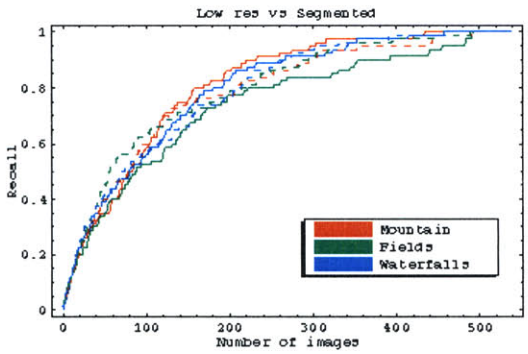


Figure 4-6: Comparison of system using segmentation (solid curves) and without using segmentation (dashed curves) for three concepts (mountains, fields and waterfalls) on 538 images from the small test set.

`+5fp` scheme twice i.e. select the second set of 5 negative instances after the second round of classification. This simulates the behavior of a user interacting with the system.

4.8 Natural Scene Classification

We compared the best curves for three natural scene classes ("fields", "mountains", "waterfalls") with the previous version of the system that uses low resolution sub-images as instances. Figure 4-6 shows that the performance is comparable to our previous results for the natural scene concepts, which is significantly better than global histogramming. While the classification performance is about the same for the "snowy-mountains" and the "waterfalls" classes, we see the the new system does worse for the "fields" class. The "fields" class is the most general of the natural scene classes and has a lot of variability in color (ranging from green to brown) and texture. Smoothing and down-sampling to low-resolution images the images (as we did in Chapter 3) accommodates a wider set of variations and avoid false matches due to the effects of high-frequency sensor noise. In addition, the current system has the advantage of better running time and a more general framework which can be extended to other classes of objects.

Even when using extremely low resolution images (8x8), learning a concept using the previous system took anywhere from a few seconds for the simple hypotheses to a few days for the more complicated hypotheses (conjunctions and disjunctions). The more complicated hypotheses take longer to learn because of the higher number of features and because the number of instances per bag is large (and to find the maximum DD point, we perform a gradient ascent from every positive instance). However, the current system uses a better method of generating instances (a rough segmentation using connected components) and this reduces both the number of instances and the running time by orders of magnitude. There are roughly $10 - 15$ components per image and the system takes a few seconds to learn the simple `cc1` concept and tens of minutes to learn the more complicated ones

Figure 4-9 shows a snapshot of the new system working on the larger dataset of 2600 images for the waterfall concept using the `cc2` hypothesis class. The red square indicates the location of the instance that is closest to the target concept in the retrieved image.

4.9 Classification of Cars

Figure 4-8 (a) compares the performance of the two hypothesis classes `cc1` (denoted by the solid lines) and `cc2` (denoted by the dashed lines) on the "cars" class. We see that both the recall and precision-recall[2] are better for the conjunction concept `cc2`. This indicates that a single circle detector alone is not sufficient to classify cars.

As we discussed in Chapter 3, all the attributes in an instance may not be equally important. The Diverse Density framework allows us to find the best weighting (scaling) on the initial feature set that emphasizes the relevant attributes for a particular query. Figure 4-8 (b and c)shows the role of scaling (feature selection) in the presence of multiple cues. The solid lines represent the case where there is no feature selection and the dashed lines represent the case where there is feature selection. Both

[2]Precision is the ratio of the number of correct images to the number of images seen so far. Recall is the ratio of the number of correct images to the total number of correct images in the test set

the recall and precision-recall is significantly better when there is feature selection. This indicates that for this class certain dimensions are redundant or irrelevant and selecting the salient dimensions helps improve classifier performance.

Figure 4-7 compares the performance of DD with feature selection against the structure driven image database retrieval using complex features by DeBonet in [J.S.DeBonet and P.Viola1997] on the small dataset of 538 images. The training set for this example consisted of 5 positive examples and 5 negative examples. We see that our method performs competetively with this method and has the advantage of being able to locate the target concept position and its salient feature dimensions from examples. Our method performs better in the case of cars since we use a shape-primitive (wheel detector) that is specific to cars in addition to color and texture properties. As we demonstrated earlier, using our hand-crafted car template, the combination of shape and color properties can effectively classify side views of cars. The use of relevant shape primitives in instances together with other general low level properties can improve classifier performance.

Figure 4-10 shows a snapshot of the system working on the cars concept on the test set of 538 images which are disjoint from the training set. The query is a set of 5 positive car images and 5 non-car images. A `cc2` concept is extracted using these examples. A new image is converted into a bag of `cc2` instances. The new is image rated on the distance from the learned concept to the closest instance in the image weighted by the learned scaling factor for each of the features (color, position, shape etc.) in the concept. The figure also shows the top 30 images retrieved by the system for this query. The location of the learned 2-component concept `cc2` is marked by the red-boxes. The closest matching instance to the learned concept corresponding to the "wheel" regions in this example. As we see the system is able to retrieve images of cars given a few examples of cars with varying backgrounds without having the user select out "car regions" in the examples.

4.10 Summary

In this chapter, we have demonstrated a general system for query learning and for image classification using a small number of examples. We described an architecture that allows the user to train the system, by selecting positive and negative examples, letting the system create and use an initial template based on those examples and finally refine the template by adding incorrect matches (false positives). Our approach to training indexing systems treats query learning as a Multiple Instance Learning problem and builds on the method of Diverse Density. The system has been tested for a few classes on a database of 2600 images from the COREL photo library.

We show that rough segmentation of high resolution images into salient connected components greatly reduces the number of instances and the running time of the algorithm especially when low resolution pixel-instances are not sufficient (e.g. when we move from the domain of natural scenes to objects). We also have experiments to show that by using a more complex representation and features within the same framework, we can learn certain object classes (e.g. cars). Our experiments on "car" classification show that segmentation, combination of cues (color, texture and simple shape), some feature selection and more complicated concepts (conjunctions) help improve classifier performance.

We have used a small combination of primitives which were suitable for natural scenes and vehicles to demonstrate the learning, feature selection and retrieval capabilities of our system. However, the issue of which primitives/invariants we need to use to generate the instances is still open. In the future, we would like to explore this issue and extend the bag generator to use a base set of low-level primitives that captures a larger set of class concepts.

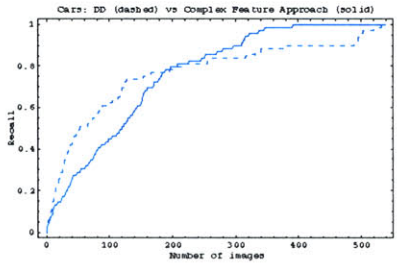
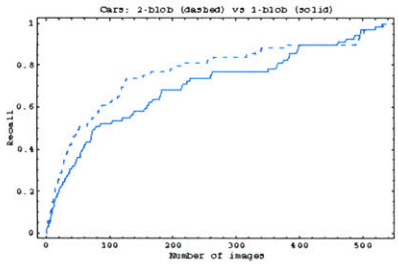
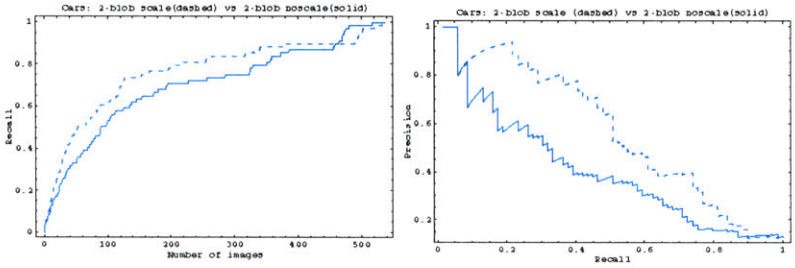


Figure 4-7: Comparing performance of DD (dashed curves) vs the complex feature approach (solid curves)



(a)



(b)

Figure 4-8: (a) Recall for single component concept (solid curves) vs conjunction of components (dashed curves) for cars. (b) Recall and precision curves using feature selection i.e. scaling (dashed) vs noscaling (solid) for cars.

Figure 4-9: Snapshot of the system running on a database of 2600 images of varied scenes from the COREL library. The figure shows the results for the waterfall concept using the `cc2` concept. Top row: Initial training set–5 positive and 5 negative examples. Last three rows: Top 30 matches retrieved from the large test set. The red squares indicate where the closest instance to the main component of the learned concept is located.

Figure 4-10: Snapshot of the system running on a database of 2600 images of varied scenes from the COREL library. The figure shows the results for the cars concept using the `cc2` concept. Top row: Initial training set–5 positive and 5 negative examples. Positive examples are highlighted in red. Last three rows: Top 30 matches retrieved from the large test set. The red squares indicate where the closest instance to the learned concept is located.

Chapter 5

Object detection and localization using dynamic template warping

5.1 Introduction

In Chapters 2,3 and 4, we discussed methods of extracting simple template representations form examples and showed that they can be used to learn a diverse set of class concepts and tolerate within-class variations in position, color, luminance and scale. In this chapter, we change the focus a little from template generation to template matching and explore ways to exploit simple template representations to search for instances of a given model template in the image.

Many recent approaches to object recognition are really verification and localization methods, because they address the question: "Is there an instance of a specific object in this image, and if so, where is it?" Thus, such methods verify or refute the hypothesis that a particular object exists in the image[1], but often don't address:

- Detecting and localizing instances from classes of objects (e.g., faces or cars), rather than specific objects.

[1]Of course not all current recognition methods are so restricted. For example, [Murase and Nayar1994] and related methods attempt to find the best object from a library that accounts for the data.

- Constructing object models that support object detection from 2d images while allowing for three dimensional pose changes. One could use an explicit 3D model, or a large set of 2D images (real or virtual). Ideally, one would like to use a very small set, e.g., one image.

- Efficiently seeding the search for an instance of an object in an image. Many recognition methods find the alignment of model with data by minimizing some error criterion. Such methods often have narrow capture radii. An alternative is to provide multiple seed hypotheses, e.g., by considering matches of minimal pairings of data and model features, but this runs the danger of being overwhelmed by the search combinatorics. We need methods that will converge to optimal solutions from highly inaccurate initial estimates.

To address these issues, we present a method that:

- detects instances of a class of objects, while allowing for object class variation, and pose changes;

- extracts approximate pose data that can seed more detailed verification, and

- verifies the existence and pose of an object by the Mutual Information (MI) [Viola and Wells1995] approach to recognition, which supports a very general matching of model and image data, without direct feature correspondence.

The technique we develop has several critical properties that make it robust in extracting the 2D image location and 3D pose estimate of an object whose pose in the image is unknown. It differs from the view-based correlation methods in that the model is a single, simple, flexible template extracted from a nominal view of the object and it is used in a matching method that can handle a wide range of pose variations.

Key properties of our approach include:

- The model for an object class is a single, simple template that may be stretched or warped in one dimension.

- This warped template can be matched to instances of an object under widely different poses, accommodating scaling, translation and out of plane rotations about the vertical axis.
- The method converts the search for object instances and poses into one (or a few) one dimensional matching problem(s).
- Dynamic programming is used to establish a mapping from the deformable template representing the object class into the image.
- While the method can recover good estimates of three degrees of freedom of object motion (x and z translation, y rotation), it can also accommodate up to two more DOFs of motion in the search process (x rotation, y translation). Extensions allow one to also search for the final rotation. Depending on the implementation, some information about y translation may be available from knowledge of the actual match of template to iamge.

Our approach combines the general detection flavor of patch correlation with the ability to accommodate more degrees of freedom of object motion that are traditionally associated with feature-based recognition systems. We demonstrate our method on face detection, as faces naturally suit the approach (heads are usually oriented upwards, but appear facing in varying directions). We show performance comparable to some current face detectors, and demonstrate that our method generalizes naturally to other classes of objects, such as vehicles.

The flexible shape model we propose deals with 2D deformations that are more severe than traditional transformations. Figure 5-1 shows an example of 2D transformations that the method can handle. The first image in the set is the prototype (nominal) image and the rest of the images illustrate warps due to y-translation, vertical shifts, scaling and shearing variations that are accomodated as long as the overall ordering of the features are maintained. The experiments described above show how this method can be used to accomodate class variations due to variation of features across a population or variations of an individual over time (e.g. facial expressions).

Figure 5-1: This class of cartoon faces shows the 2D deformations handled by the system. The last row shows the resulting of matching the template (nominal view) to a novel image.

5.1.1 Background

A primary difficulty in object detection is accommodating the wide range of variations in object appearance due to pose, lighting and class variations, while maintaining good detection performance. Existing approaches for dealing with these variations are described below.

In view-based methods, one computes the similarity between a template and image locations and returns all matches above a threshold. Since most classes of objects cannot be modeled by a single template, these techniques typically use a set of templates to accommodate variations in pose, lighting and shape, by synthesizing views from a set of examples [D.Beymer1993], or by using networks to learn an object class from several examples with varying pose and lighting [Sung and Poggio1994, H. Rowley and Kanade1995]. Most of the existing methods for face detection perform extremely well on upright, frontal faces and do not handle 3D pose variations. Recently, some systems like [Rowley *et al.*1998, Schneiderman and Kanade1998] detect faces with arbitrary rotations in the image plane. However, these systems are not as robust with off-plane rotations of faces (profiles). In this chapter, we propose a method that can handle off-plane rotations (rotations about the vertical axis for

faces) efficiently and reliably.

A related set of methods to the view-based correlation techniques is the linear combination approach for modelling classes of objects. Early work in this field was done by Ullman and Basri [Ullman and Basri1991] who showed that an intermediate view of an object can be written as a linear combination of three 2D views that span it assuming orthographic projection and barring self occlusion. This was followed by work by Vetter and Poggio [Vetter and Poggio1996] who showed how to model, learn and match classes of objects (faces) by using a linear combination of prototypes. In these methods, images are represented by the locations of a small set of labeled features. Work by [D.Beymer1993, Jones and Poggio1996] extend this approach by modeling classes of objects (faces) by a linear combination of a small number of prototypes where the example images are represented by dense pixel-wise correspondence between the examples and a reference image. These methods depend crucially on a good set of features and accurate correspondence.

Betke [Betke and Makris1995] uses simulated annealing for fast 2D object recognition (traffic signs) in noisy images by matching a new scene to a set of templates generated by transforming model images.

In deformable template matching, templates are constructed to model the non-rigid features of the object, and at recognition time the templates are aligned to the image by minimizing an energy function for the individual features. Some examples of this method include [A. Yuille and Cohen1992, Cootes and Taylor1993]. These methods work well when the deformations are small – they provide a detailed analysis of the image in terms of the template, but do not address non-local search issues.

Eigenspace methods represent the varying appearances of objects by using principal components analysis (PCA) on a set of sample views to identify a low-dimensional subspace of the view-space. Examples of this methods are [Turk and Pentland1991, Pentland *et al.*1994, Murase and Nayar1994, Huttenlocher *et al.*1996]. Images are accepted or rejected based on their distance from the pre-computed subspace. These methods need a large set of views of the object to be sampled under varying pose and lighting conditions.

The ratio-template [Sinha1994] detects faces under varying illumination conditions, by capturing changes in luminance between fixed image patches. While this method performs well for frontal faces under varying lighting conditions, it is not as robust under 3D pose variations.

Hornegger [Hornegger1995] converts feature based object recognition under orthography into several 1D search subproblems. The technique constructs 1D images from the more conventional 2D images by a process of *marginalization*, which projects image features onto the x axis (and discards their y coordinates). This effectively discharges two degrees of freedom (DOF) of object motion, namely x rotation and y translation[2]. The remaining three DOFs are then searched exhaustively. The resulting candidate partial solutions are then completed with a full 5 DOF localization.

In this paper, we advocate the use of dynamic programming to address the 3 DOF search that remains after conversion to 1D matching subproblems. Dynamic programming (DP) has been used successfully in speech recognition to produce an optimal time alignment of an actual speech stream to be recognized to an acoustic model, via the Viterbi algorithm [Viterbi1967, Forney1973, Sakoe and Chiba], and we were motivated to apply it to search problems in object recognition. DP is well established as a methodology for solving the stereo matching problem, e.g., [Baker and T.O.Binford1981, Ohta and Kanade1985]. It has also been used for contour adjustment and other boundary oriented scoring or identification tasks e.g., [Ballard and Brown1982, H.G.Barrow1976, Shashua and Ullman1988].

DP matching techniques typically will accommodate fairly general 1D warpings in the matches that are found, subject to order constraints. This might seem to be a drawback because in object recognition, it would allow distortions that can not be generated by a rigid object moving in front of a camera. However, within the warps that DP accepts are embedded some useful mappings, for example x translation and the overall scale changes that can occur as an object moves away from a camera. In addition, y rotations of a vertically oriented cylindrical object will induce (primarily)

[2]To make the descriptions concrete, we use the convention that the 1D searches are carried out along the x-axis in the image.

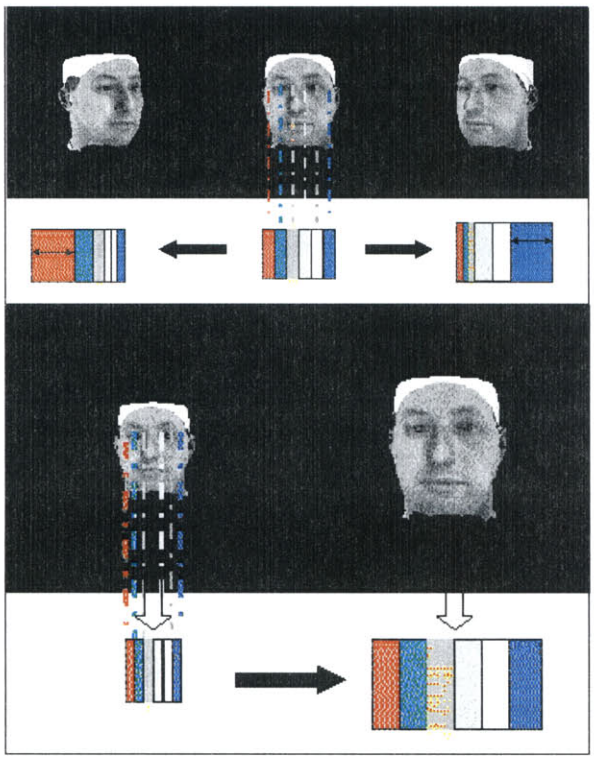


Figure 5-2: Rotations and scale changes induce 1-D warps.

stretches and shrinks along x, which may be accommodated in a DP match. Figure 5-2 illustrates the 1D-warps induced by rotations and scale changes. We have found in our experiments that this feature can be used to accurately recover the y rotation when the object in question is approximately cylindrical.

5.2 Comparison with Affine Models

There has been considerable work in the computer vision literature on recognition systems that accomodate affine transformations on objects [Huttenlocher and Ullman1990, Rucklidge1994] among others. Part of the motivation is that for planar objects, affine transforms are equivalent to rigid body motion under orthography. The flexible shape model that we propose handles five of the six DOF of affine transformations. Figure 5-3 shows the allowable set of affine deformations and Figure 5-4 shows a shear that is not handled by the system. Our technique finds a match between members of a

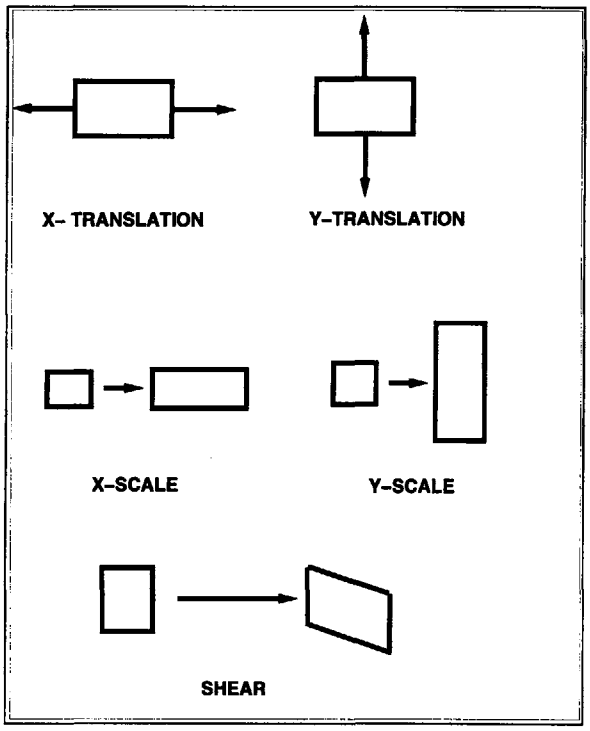


Figure 5-3: 2D affine warps handled by the system.

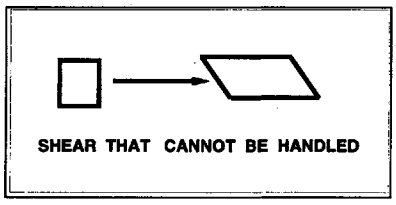


Figure 5-4: 2D shear that cannot be handled by the system.

class by relating new images in the class to the prototype (template) through the set of allowable deformations based on stretches and shifts in the template features, subject to order contraints. The ability of our system to handle these affine warps is similar to the constrained-affine shape model proposed by T.F. Syeda-Mahmood in [Mahmood and Zhu1998]. The constrained-affine shape model is a region-topology based model that captures the spatial layout similarity between members of a class by a set of affine deformations from a prototypical member.

5.3 Dynamic Template Warping

The information flow used in our system including 3D pose recovery is outlined in Figure 5-5. In brief, the class model consists of many 3D surface patches. A template (whose details are provided below) is constructed from a characteristic (typically frontal) view of a representative object in the class. The actual mapping from surface patches to elements of the template is determined once for the model. At recognition time, dynamic programming (DP) is used to solve one-dimensional matching subproblems and thus determine mappings from elements of the image to corresponding elements of the template. A partial solution for the unknown object pose is obtained by solving an optimization in which the projection of the model surface patches into template columns, via the hypothesized pose and the runtime column mapping determined by DP, are made to best agree with their true values.

Below, we provide more details about the detection and localization process.

5.3.1 Representing objects

Our goal is to represent classes of objects in a flexible manner, capturing 3D pose changes and class variations in a simple system. Initially, we assume that the images we will be processing are taken in an upright position, i.e., the gravity vector runs vertically through the image. As a consequence, we choose to represent a class of objects by a sequence of image columns, each column consisting of a set of intensity patterns. To construct our model template, we begin with an image of the object rendered in the nominal pose (Figure 5-8) – e.g., for the case of faces, we use an image of a face model in a roughly frontal view. This image is smoothed, downsampled and intensity-normalized, then separated into columns. Each column is described as a sequence of intensity classes obtained by coarsely quantizing the intensities present in the smoothed, downsampled and intensity-normalized image. The rough quantization allows for class variations in intensity, and as we shall see shortly, the use of separated columns will allow us to accomodate 3D pose variation.

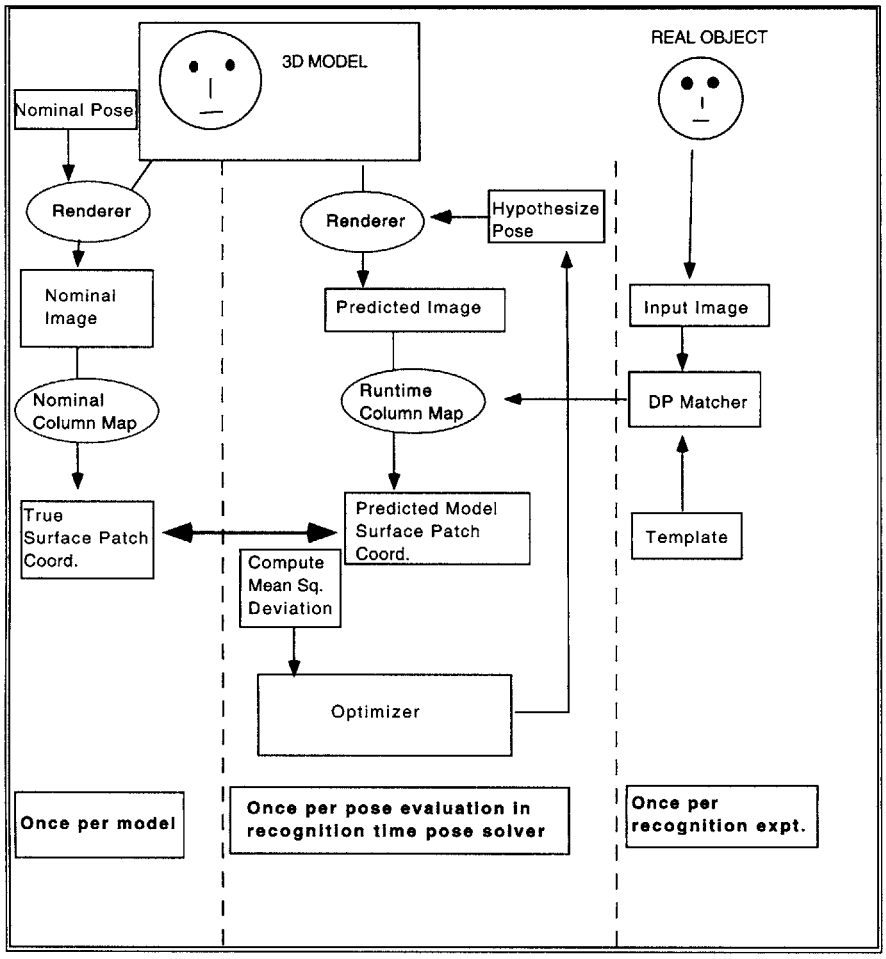


Figure 5-5: Information flow in the DTW pose solver. Template is derived from the nominal image.

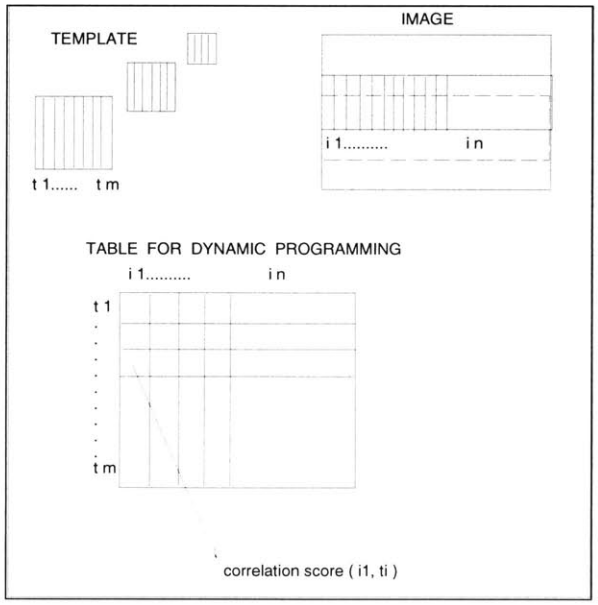


Figure 5-6: DP Matcher data structures

5.3.2 Finding template matches in the image

To find matches of such a template in an image, we need to consider the variations that a template may undergo as a function of class variation and pose variation. Class variations will be discussed shortly, when we describe the measurement of similarity. We break pose variation into several pieces, initially focusing on three of the six degrees of freedom. Translation of the template in the two image dimensions can be straightforwardly handled through a search process. Rotation about the vertical axis will have the effect of inducing a warping on the columns of the template. In particular, some of the template's columns will be stretched out to cover several image columns, and other columns will be foreshortened and thus compressed into fewer columns. This implies that our matching of template columns to image columns is no longer a one-to-one map, and cannot be dealt with by simple techniques such as correlation. On the other hand, such a warping of the template naturally lends itself to dynamic programming methods.

Dynamic Programming (DP) is a search technique that has been applied to many problems that are formulated as the minimization of a cost function over paths in

a graph. It may also be applied to find optimal matchings among features if certain monotonicity constraints are met. The dynamic programming algorithm we have used is derived from the longest common subsequence algorithm described in [Corman *et al.*]. In the present application, sequences of image and template features are compared by a measure of their degree of match rather than the binary-valued measure used in the standard subsequence algorithm.

For a given model template, a set of overlapping horizontal strips is extracted that cover the input image (Figure 5-6). The strips are 1.5 times higher than the template height, and overlap by three fourths of their height – they have been designed so that some strip will entirely cover an instance of the object with nominal height. Within each strip, the template column features are matched with the image column features by a correlation-like process that locates the best matched section of the image column feature. This embedded problem of measuring the agreement of template and image column features is somewhat similar to the surrounding problem of matching image and template, and it is similarly useful to address it with a mechanism that is able to tolerate vertical shifts, and moderate scaling and stretching.

In this case, we are interested in finding a many to one matching between elements of the image and template column features. The process operates by searching for equivalent structure in terms of uniform intensity class subsequences while accomodating some variation in subsequence length (up to a factor of two). In more detail, each template and image column is a vector of pixels given by $(tc_1 ... tc_m)$ and $(ic_1 ... ic_n)$. Starting from all pairs of pixels in the template and image column vector, match the content of pairs of pixels based on their intensity or color. The match score between pairs of pixels (tc_i, ic_j) are used to fill a cost table. Dynamic programming can then be used to find the optimal subsequence that satisfies the following constraints: (1) the overall spatial ordering of elements in the template column is preserved in the image column and (2) the extent of stretch of a template column element (tc_i) in the image column is bounded by a fixed threshold δ [3]. This column matching algorithm gives the longest common set of corresponding elements between

[3]This stretching is is caused by many image column elements matching the same template element

the template and image columns which match in their content, have the same overall spatial ordering while allowing for some bounded vertical shifts and scale variations.

To match across the sequence of column matches, we use the standard subsequence algorithm for dynamic programming, which assumes that matches are order-preserving and have scores that are independant of one another. The algorithm constructs two tables, the cost table, and the index table, that we index by every pair of template ($t_1 ... t_m$) and image ($i_1 ... i_n$) column-features (the cost table is illustrated in (Figure 5-6)). The dynamic programming approach to finding the optimal subsequence proceeds from left to right. By consulting the cost table, and by examining previously computed solutions to problems ending in the previous position, the algorithm efficiently computes and stores in the tables the scores and sub-sequence predecessor information for the optimal solutions to the problem at the current position.

Our algorithm extends the standard one to return all subsequences having scores above a threshold (rather that a single best subsequence), by preserving information defining the previously most promising subsequence when a new subsequence is initiated.

The algorithm needs more space to store this information, but the time complexity remains the same ($O(MN)$) for M model features and N image features. In addition, we employ a look-ahead technique similar to those used in DP stereo-matching systems to accomodate partial occlusions, by requiring several columns of match failure before the initiation of a new subsequence. When combined with the column matcher described above, and run at a single scale, the DP template matching algorithm has equivalent complexity to simple template correlation while accomodating larger rotations about the y-axis.

The DP matching algorithm described above is used to find ordered image strip columns that are deemed to be explained by the template by having scores greater than a given threshold. The matches found by the DP algorithm are ranked according to score.

We can repeat this process for each horizontal strip of the image. Note that using

multiple strips covers the y translation, and finding the optimal column match using DP directly covers the x translation and rotation about the y axis. Note that the flexiblity of the matching – both the roughly quantized intensity sequence matching within each column, and the flexible matching across columns – allows for a range of class variations as well.

When the DP matcher has finished, the following information is available for use by the partial pose solver (which will be described below):

- The horizontal position of the matched instance of the model in the image if the model is present.
- The "active patch" in which the match was found i.e. the horizontal position together with the vertical position given by the horizontal strip.
- A detailed column mapping between image columns and template columns where many image columns can map to the same model column (to account for horizontal stretching caused by rotations about the y-axis and scaling). Image columns may also map to the "no" model column (this accommodates partial occlusions).

5.3.3 Scale Hierarchy

Our initial strategy uses a fixed template height of 64 pixels. Because the column matcher is able to accomodate some variation in scale, and the DP matching algorithm can accomodate scale variation as a "stretching" of the match, this mechanism can accomodate object instances with apparent heights ranging from roughly 40 to 80 image pixels, and the scale may be later recovered from information recorded in the detailed column mapping.

A heirarchical implementation was used for the rest of the experiments. A set of three templates were constructed with heights of 16, 32, and 64 pixels by pixel replication, and three sets of image strips were prepared, as described above. The DP matching algorithm was then run at the three scales, in order to accomodate

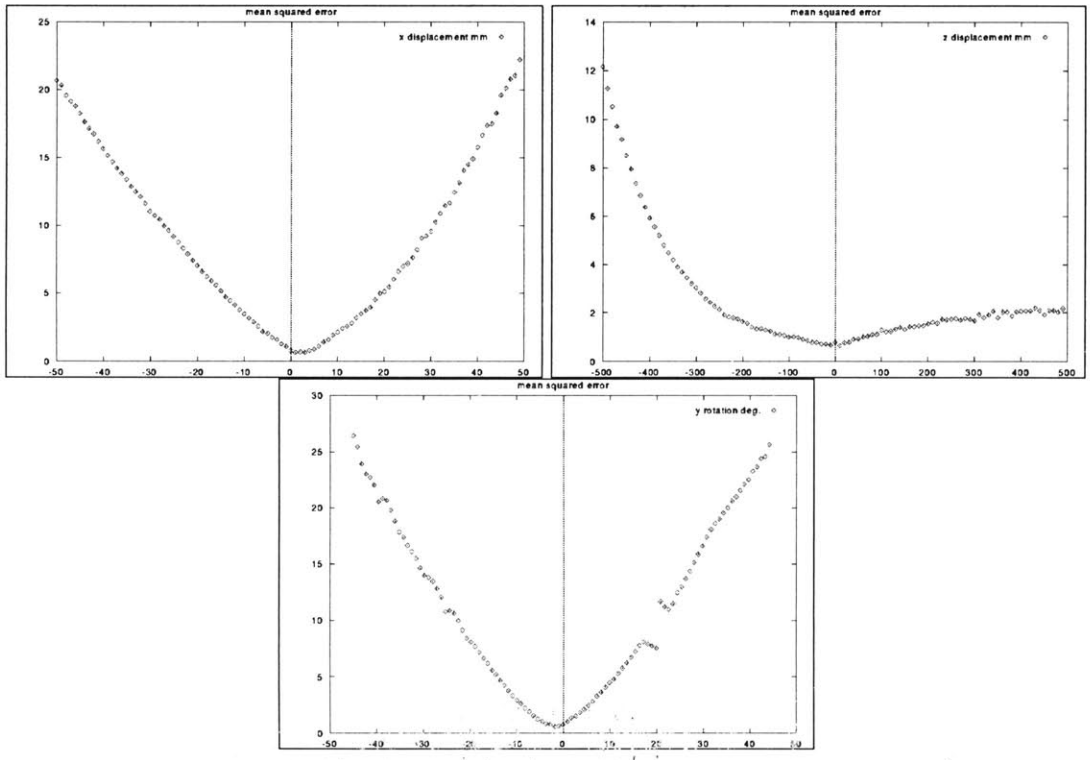


Figure 5-7: Probed values of the objective function used in the recognition-time pose estimation (the mean squared difference between predicted and true template columns for projected model surface patches). The plots display the objective function against the difference between the three pose adjustment parameters and their known true values.

objects instances with apparent heights ranging from roughly 10 to 80 image pixels. Note that this implictly allows us to solve for the third translational component of the pose.

5.3.4 Partial Pose from DP Match Results

In this section we describe a method for recovering a rough, or partial, object pose from the detailed column mapping that is obtained as a result of the DP match from image to template columns. Figure 5-5 shows the flow of information that is used in the pose determination process.

We seek a pose such that a rendering of the model by that pose looks like the input image. We assume that the DP matcher has determined the correct column map (the

"runtime column map") that maps image columns to their corresponding template columns. Thus we want to find a pose such that when the model is rendered by that pose, the resulting synthetic image will (like the input image) be correctly related to the template columns via the runtime column map. This suggests the following approach for evaluating a hypothesized model pose:

- Project the model surface patches into the image by rendering them according to the hypothesized pose, and send the resulting image coordinates through the runtime column mapping. This process will produce a mapping from model surface patches to their corresponding template columns.
- Evaluate the consistency of the hypothesized pose by comparing these predicted surface patch coordinates (template indices) with their true values.

An objective function may be constructed to measure this consistency – we have used the mean squared difference between the predicted and true template coordinates of the surface patches.

As it happens, there is a simple method available for establishing an approximation to the true mapping from object surface patches to template columns – we simply render the model at a nominal (in this case, frontal) pose, and use the DP matcher to establish a nominal column map from the nominal image to the template. We then establish the mapping from surface patches to template columns in the manner described above.

The mechanism outlined above provides detailed information relating to x and z translations, and also the y rotations of the model.

Useful, but less precise, information related to the x and y translations is available from our knowledge of the the "active patch" in the image (the section of the image strip this is involved in the match that just includes the columns at the beginning and and of the DP match to the template).

In our experiments, the partial pose solution is determined in the following way.

- Starting with the nominal pose, an initial pose estimate is obtained by applying a motion to the object that adjusts the apparent azimuth and elevation of the

object in order to center it in the active patch. While this motion is applied to the object, in terms of the relation of the object and the camera, it is equivalent to a rotation of the camera about x and y.

- The initial pose estimate is used as a starting value for a second stage of pose estimation. An additional motion is applied to the object that consists of a rotation of the object about y (measured in degrees), and translations along x and z (in millimeters). These parameters are optimized with respect to the objective function described above, using the first stage estimate as a starting value.

The effect of the first stage described above is to move the object from the nominal pose in order to place it in approximately the right part of the image, while the second stage provides relatively precise adjustment of some of the pose parameters from the match.

After this process, x and z translation and y rotation have been determined relatively precisely from the match, and y translation has been approximately determined from knowledge of the active patch. The remaining parameters of motion, x and z rotation are not recovered by the method, and their values will be whatever is obtained from the application of the two stages of adjustment to the nominal pose (the adjustments have minor effects in this regard).

Some plots of the pose objective function are shown in figures 5-7, for a match done against a synthetic image (top left image in Figure 5-9) where the true pose is known. The plots suggest a prominent minima near the true value for the adjustment parameters, although some noise and local minima are apparent.

5.4 Image Matching Results using DP

In this section we show some simple experiments to demonstrate that dynamic programming with a simple, stretchable template can be used as an efficient search strategy that can perform better than normalized correlation for detection and lo-

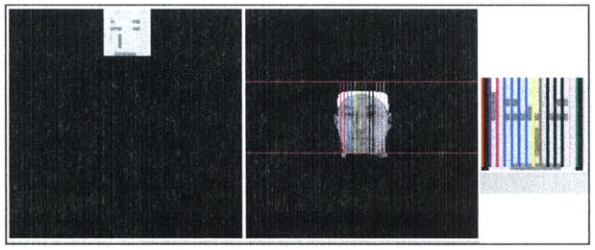


Figure 5-8: Results of the DP matcher on the nominal pose image. The left image shows the low resolution model columns copied into the appropriate matching image columns. The images in the middle and right show the column mappings. The color code on the model template and the image lines show the mapping. The red box on the image indicates the horizontal strip in which the best match was found. The middle image shows the nominal view (frontal face) from which the template is extracted.

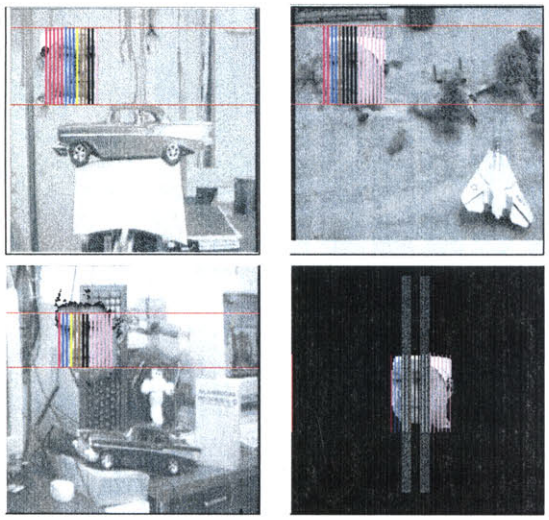


Figure 5-9: Results of DP matcher on images with varying backgrounds. The colored vertical lines indicates the mapping from template columns to input image columns. The box on the image indicates the strip in which the best match was found.

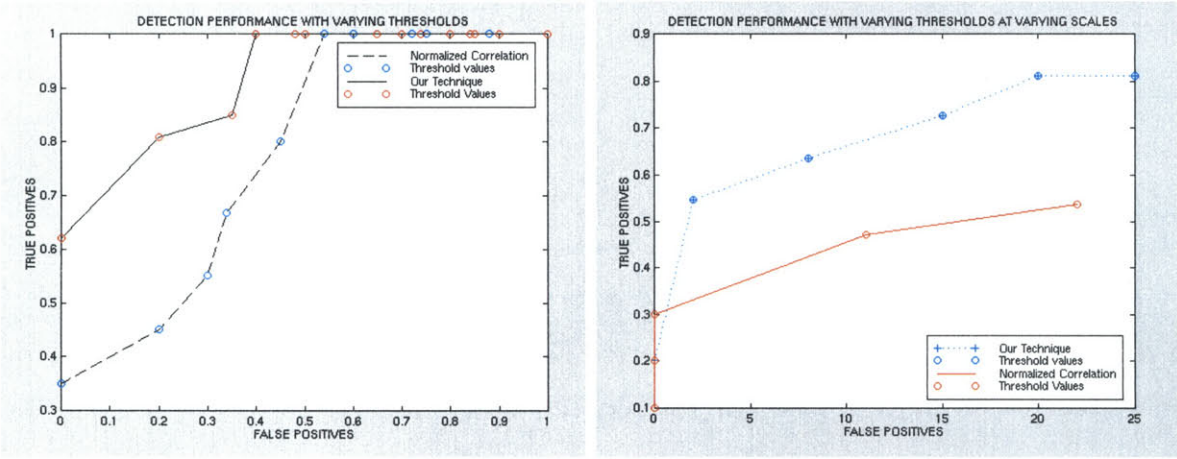


Figure 5-10: ROC Curves: Plot of False Positives vs True Positives for varying threshold values. Each point represents the (False Positive, True Positive) pair for a particular threshold value. The threshold values increase from 0 to 1 from the upper right corner to the lower left. LEFT: The curves show the detection performance of the DP matcher (solid line) and Normalized correlation (dotted line) on the synthetic dataset with rotated images of the head in cluttered backgrounds. RIGHT: The curves show the detection performance of the DP matcher (dotted line) and Normalized correlation (solid line) on a synthetic dataset with images where the scale varies upto a factor of two.

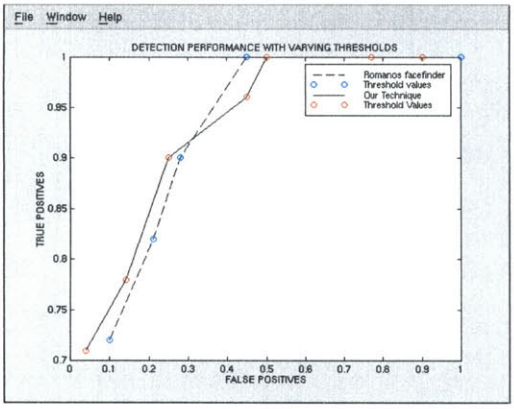


Figure 5-11: ROC Curves: Plot of False Positives vs True Positives for varying threshold values. Each point represents the (False Positive, True Positive) pair for a particular threshold value. The threshold values increase from 0 to 1 from the upper right corner to the lower left. The curves compare the detection performance of the DP matcher (solid line) and Romano's facefinder (dotted line) on a dataset of frontal faces in cluttered backgrounds with varying lighting.

calization tasks under translations, y-rotations and partial vertical occlusions. The method also provides a detailed column mapping from matched image columns onto model columns which can then be used for 3D pose estimation. We will describe experiments that address these issues below.

We used five Cyberware[4] scans of heads rendered at various poses using a graphics package. The Cyberware scanner is a structured light scanning device that simultaneously obtains 3D surface and surface texture information. The rendered heads were then embedded in a variety of backgrounds as shown in Figure 5-9. The cyberware models were rendered with varying horizontal and vertical translations, a large range of y-rotations (-50 to 50 degrees from the nominal pose) and a small range of x-rotations (-5 to 5 degrees from the nominal pose). There were no z-rotations of the model. In addition to the cyberware scans, we tested the system on real images of heads and compared detection performance with normalized correlation (Figures 5-16 and 5-12).

This technique can handle rotations about the Y-axis since they are approximated by stretches and shrinks of the model columns in the image. The model can be thought of as a flexible template that consists of a set of ordered columns that can be stretched and shifted to fit the image.

The template is a set of columns extracted from the quantized frontal face (model face at the nominal pose) at low resolution. The color code on the model face (right) and the input image lines (left) show the mapping. The red box on the image indicates the horizontal strip in which the best match was found. Figure 5-8 describes the process of getting the nominal column mapping used to approximate the true column mapping for the model surface patches. A single template at 3 different scales is used for all the experiments. The results (Figure 5-9) illustrate how rotations about the Y-axis are handled as horizontal deformations of model columns. The best match finds the longest common subset of ordered model columns that explain an image region. Figure 5-9 also shows that simple vertical occlusions can be naturally accomodated by the matcher and that a single template can be used to locate heads in general.

[4]Cyberware Incorporated, CA

Figure 5-16 shows an example of scale changes handled by the system. This figure also illustrates that the system can localize the head and get accurate column mapping information in real images.

5.4.1 Comparison to Normalized correlation

These experiments compare the performance of the DP search to normalized correlation in a domain with rotations about the Y-axis and clutter.

Normalized correlation has been used in a variety of detection tasks to find instances of an object in the image. This technique has been used successfully in the face detection literature, for example in [Sung and Poggio1994, H. Rowley and Kanade1995], to locate instances of face-like patterns in cluttered scenes, with low computational cost. It works well when the object is not rotated or scaled much in the image and the lighting varies uniformly.

In this experiment we compare the DP matcher with normalized correlation with a single frontal template extracted from the nominal pose shown in Figure 5-8 (rendered face facing forward). For each threshold value, we ran normalized correlation on 20 images with varying backgrounds and varying rotations of the face (-50 to 50 degrees) about the Y-axis and rotations of (-5 to 5 degrees) about the X-axis. The template used was extracted from the image of the face facing forward with no rotation about the Y-axis as shown in the figure. The search window size was the size of the template. The search was centered at pixels (i, j) for $(i = 0,\ i < ImageWidth,\ i = i + 2)$ and $(j = 0,\ j < ImageHeight,\ j = j + 5)$ in all the images.

The detection performance of the two methods are compared using a "Receiver Operating Characteristic" (ROC) shown on the left in Figure 5-10. The ROC is frequently used in the detection literature to evaluate detection performance by plotting the percentage of true positives detected as a function of the false alarms due to clutter. A higher curve indicates better performance. We see that for detecting faces with cluttered backgrounds and varying rotations about the Y-axis this method performs significantly better than normalized correlation with a single template. Techniques like [Sung and Poggio1994] and [H. Rowley and Kanade1995] use several templates

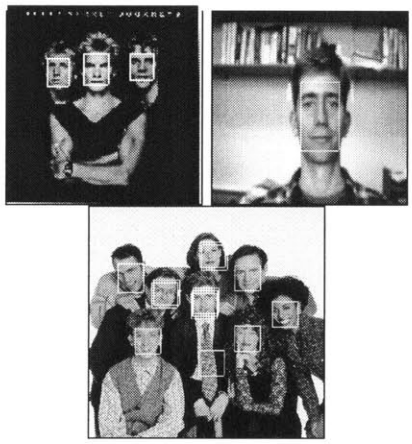


Figure 5-12: Some examples showing the detection performance of the system

extracted from several views of the object to accommodate y-rotations. The advantage with the DP matcher is that a wide range of y-rotations can be accommodated with a single stretchable template.

Figure 5-11 compares the DP-matcher with normalized correlation with a single frontal template on a dataset of synthetic images of varying scales (variations upto a factor of two in scale). Here again we see that the DP matcher is able to accomodate larger variations in scale using a single template than traditional normalized correlation.

5.4.2 Comparison with Romano's facefinder

We compared the detection performance of the DP matcher to a robust facefinder [R. Romano and Poggio1996] technique in a domain where normalized correlation is known to perform well.

The input consisted of a dataset of 50 images which contained frontal views of faces. The template is extracted from one of the faces in the dataset by specifying the coordinates of the locations of the left and right eye, the left and right nose lobes and the the left and right extremities of the mouth. This template was then matched against the images in the dataset using normalized correlation for different thresholds.

Our algorithm was run on the same dataset of 50 images. Our template was ex-

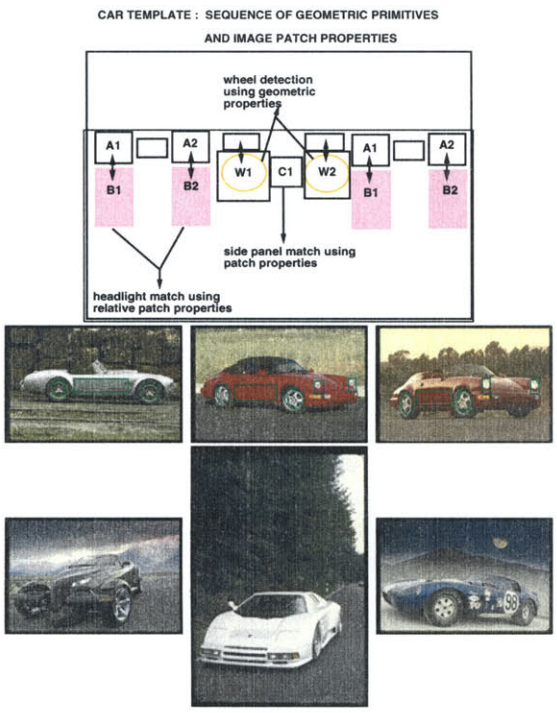


Figure 5-13: Top Row: Car template. Middle Row: Examples of cars detected. Bottom Row: Example of cars not detected.

tracted from the same face image that was used to extract the template for Romano's method. The input images were split into overlapping strips as described earlier and the output consisted of image regions with column mapping scores that were greater than the given threshold.

The ROC-curve in Figure 5-11 shows that the DP matcher performs slightly worse for this dataset for low thresholds but is comparable to normalized correlation for thresholds in the middle to high range.

5.4.3 Detection Experiments

We also ran the system on a set of face and non-face images, to determine is performance as a class detection system. Figure 5-12 shows examples of faces detected by the system. The database had a total of 188 faces. The true positive fraction of faces detected by the system was 0.91. The false positive fraction was 0.23.

5.4.4 Initial experiments on car detection

We conducted initial experiments to test the same general framework of feature sequence matching using dynamic template warping in the domain of car detection. Figure 5-13 illustrates the features used in this experiment. They are a combination of patch properties (color and spatial properties of image patches) and geometric primitives. Geometric primitives (e.g. wheels) are detected with an edge based shape matcher (using Hausdorff distance [Huttenlocher *et al.*1993, Rucklidge1994]). This model template for the car is similar to the hand-crafted car template described in Chapter 2. It encodes the same combination of features. The match strategy we use to detect new instances of cars differs here. Instead of considering pairs of wheel regions and matching the color properties around the wheels as we did earlier, we divide the image up into overlapping strips and look for subsequences of features in the image that match the template. The matching subsequence simultaneously searches for both the color and geometric features and preserves the order of features in the template. In addition, the matching process systematically handles variations in the image due to pose changes (stretching and shrinking of regions due to rotations and scale) as well as missing features due to errors in feature detection. For example if a wheel is missing due to occlusion or noise but the rest of the features are matched in the correct order, the match is accepted as an instance of the car.

The template is a sequence of features as in the head experiments. Figure 5-13 shows the template used in the detection experiments. A new image is divided into overlapping horizontal strips as in the head detection examples. The match between template features and image patch features accomodates scaling in the x-direction and rotations about the y-axis by stretching of the individual template features and finding the longest common subsequence of features. Detecting the individual features like the wheels without any information on their relationship to the rest of the car can lead to many false identifications. This method detects instances of cars by ensuring that a subset of individual features given in the car template are found in the image in the correct order (spatial configuration).

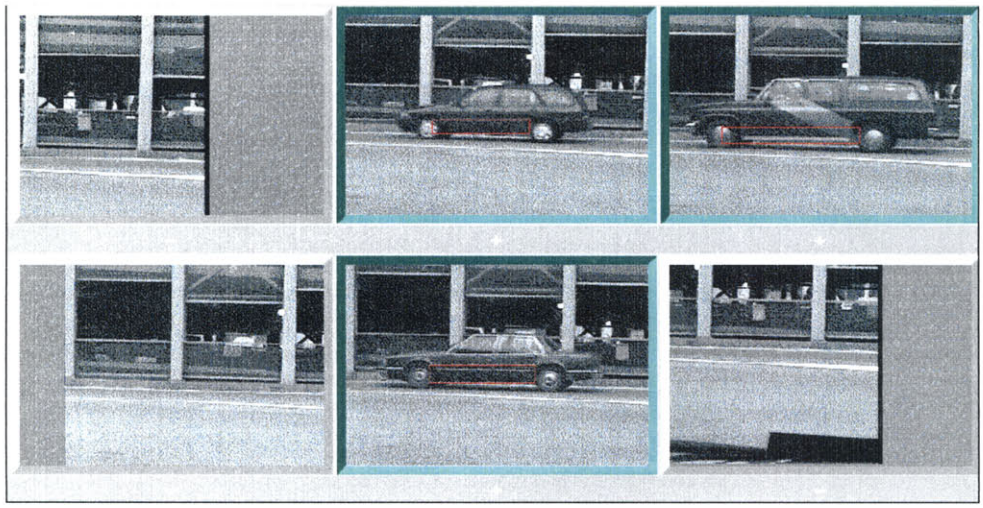


Figure 5-14: Snapshot of the car-detection system working on a real video sequence. The frames with cars are highlighted in red

In a preliminary detection experiment on a dataset of 100 cars from the COREL photolibrary there were 73 true positives and 27 false negatives. Figure 5-13 shows some selected results from the COREL database and Figure 5-14 shows a snapshot of the system working on a real video sequence.

5.4.5 Pose Solving Experiment

Partial Pose Solution Figures 5-15 and 5-16 shows examples of running the pose solver. For the experiment shown in Figure 5-15, we have optimized the adjustment parameters by using the downhill simplex method [Press *et al.*1992]. The initial simplex was established with displacements of 1 and 5 millimeters in x and z, and 3.5 degrees in y rotation. These values allow the optimization to step past small local minima in the initial movements of the optimization. For this (synthetic) test image, the ground truth for the object pose is available, and the resulting errors in the estimated pose were 3.3 mm in x position, 12.8 mm in z position, and 1.2 degrees in y rotation. Note that the error in depth is slight, considering that the object is 1000 mm from the camera. The x and z positions, and y rotation, have been well determined from the information in the DP match, while the approximate y position

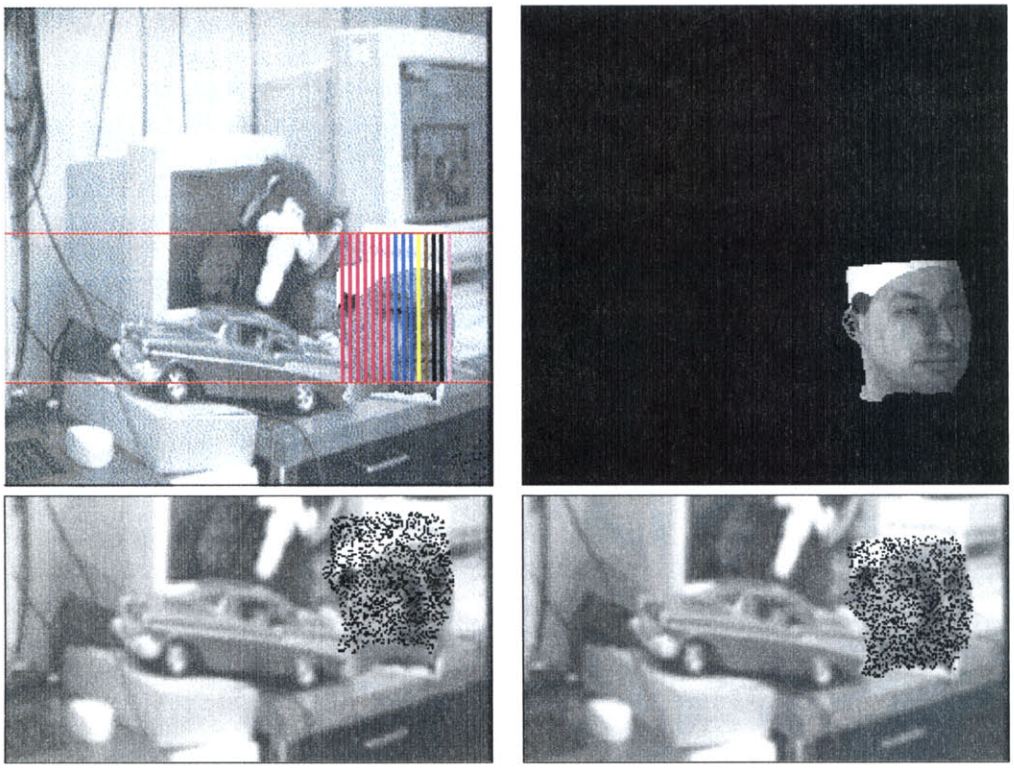


Figure 5-15: Complete System: Detection and Pose estimation. The input image with a 20 deg. rotation, and the DP match are shown at top left. Results of the partial 3D pose solution using the DP-method, which is used as the initial pose given to the MI recognition system are shown at bottom left. This pose is illustrated by an overlay of points subsampled from the object. The partial pose estimate is too high – the y estimate is off by a few cm. The y information was derived only from knowledge of which image strip was involved in the match. The estimated y-rotation is off by about 2 degrees from the ground truth pose. The final pose is illustrated by a rendering of the 3D model in the top right, and by an overlay of points subsampled from the object surface in the bottom right. The figure shows that the final solution given by the MI alignment has corrected the y-displacement, and the pose shown in the final rendered image agrees with the input pose.

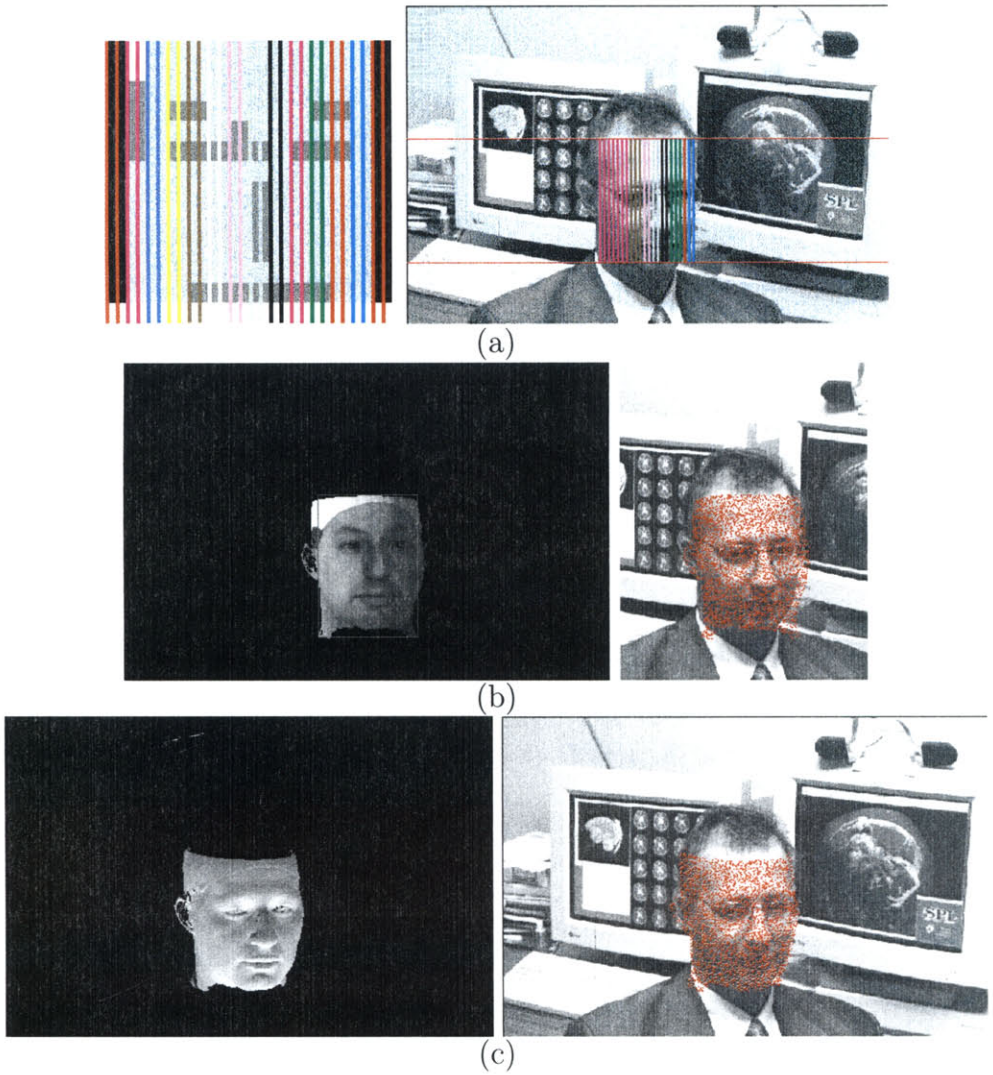


Figure 5-16: End to end system running on a real image. The template and results of the DP column match are shown in the top row. Results of the partial 3D pose solution using the DP-method, which is used as the initial pose given to the MI recognition system are shown in the next row. This pose is illustrated by a rendering and by an overlay of points subsampled from the object. The final pose is illustrated by a rendering of the 3D model, and by an overlay of points subsampled from the object surface in the next row. From the figure, we see that the final solution given by the MI alignment has corrected the x-rotation error, and the pose in the final rendered image agrees with the input pose.

has been determined from the vertical position of the "active patch" in the image.

Figure 5-16 shows the results of the pose solver running on a real image.

Two Stage Pose Determination using Mutual Information The pose determined by the pose solver was used as an initial pose for a final pose refinement by the method of Alignment by Maximization of Mutual Information (MI) [Viola and Wells1995]. The final pose determined by MI is illustrated on the right in Figures 5-15 and 5-16. This refinement was carried out as a local minimization over the six degrees of freedom of motion of the model. Upon visual inspection, the refinement has effectively corrected errors in the initial pose and aligned the model with the image.

5.5 Summary

We have demonstrated a simple visual search technique, based on dynamic programming, that is useful as a first stage for model based object recognition. The method has some of the characteristics of template correlation, but can solve for more DOF of object motion, while retaining similar complexity.

We have shown experiments to demonstrate that the method is nearly as robust as template correlation, in a domain where template correlation is successful. We have also shown that our method is more robust than simple template correlation in a domain having significant out of plane rotations.

The run-time efficiency of dynamic programming does not come for free – there are constraints that must be met in the design of the objective function, in order to get the guaranteed optimality properties in the solutions, along with the economy. One of these constraints implies that the matches between image and template columns be evaluated independently. This weakens the overall match strictness, by allowing column matches that have inconsistent vertical positions. One may ask: is the method is able to maintain adequate robustness despite this relaxation. The empirical evidence shown in Experiments 3 and 4 suggests that the answer is a qualified

"yes", at least in our test domain.

An additional benefit of our method is that partial pose information may be obtained for downstream use by local search and verification methods.

In speech recognition using DP, learning methods are well established for automatically deriving the templates (in the form of hidden Markov models) from training examples, and such a technique might be applicable here as well.

As we mentioned earlier, object recognition and localization is challenging due to the variations caused in the appearance of an object in the image due to pose and lighting variations as well as sensor noise and occlusions.

5.6 Future Directions

The templates for the object classes are currently hand crafted. We plan to learn these automatically from examples. In speech recognition using DP, methods using Hidden Markov Models are well established for learning models from training data. We would like to try some of these techniques to derive our class models for recognition.

Chapter 6

Conclusions

In this thesis, we have presented methods for (a) learning visual concepts in the form of flexible templates to model object classes from a small number of positive and negative examples and (b) matching templates efficiently to detect and localize new instances of the object class. These learned templates encode the relevant color and spatial properties of the class. We showed that these simple learned representations are compact, they can be extracted efficiently and reliably from a small number of examples, they can be matched efficiently and they can be used effectively in a number of image understanding tasks. We demonstrated the use of these learned templates in the application domains of image database indexing and object detection and localization.

As we discussed in Chapter 2, the growing number of digital image and video libraries has led to the need for automated, content-based image retrieval systems which can efficiently retrieve images from a database that are similar to a user's query. This is a difficult problem because (1) images are inherently ambiguous and can have many interpretations based on the users experiences and (2) there is a great deal of variability between images of the same class. Pre-defined flexible templates [Sinha1994, Lipson1996] have been used successfully to capture this variability for some classes of objects like faces and natural scenes. While a user can create templates corresponding to different visual concepts like waterfalls and mountains by hand, this is inconvenient when the user wants to browse and explore a wide variety of concepts.

In this thesis, we provide a framework that allows the user to train the system, by selecting a few example images and letting the system create a flexible template representation based on those examples. This "example-based" training framework provides a useful way for users to create and refine queries based only on a few examples. This framework also provides a way for the user to focus a query and refine the initial template by adding more positive and negative examples from the set of images retrieved by the system.

The learning approach presented here uses the Multiple Instance Learning paradigm and the Diverse Density algorithm [Maron1998] to address the ambiguity inherent in learning concepts from images. It differs from other "example-based" training schemes for image retrieval in that (1) it abstracts out the relevant properties that are reinforced in the examples and locates the learned concept in a new image, (2) learns the concept without requiring the user to select out the relevant regions in the examples, (3) it does not require expensive preprocessing and (4) it finds the scaling of the feature space that gives the best concept i.e. it weighs the different features based on how relevant they are to the extracted concept. This feature weighting could potentially be quite useful in learning salient features as well as reducing the dimensionality of very high dimensional image feature spaces. While we have used a basic, unconstrained version of this scaling in our experiments, we think this can be explored further to improve classification performance as we discussed at the end of Chapter 3.

A key feature of our architecture is the bag-generator which Generates features (instances) that capture the local color relations *within* subregions as well as color and spatial relations *between* subregions. In Chapter 4, we discussed how the bag generator can be extended using multiple cues to capture more complicated concepts. The bag generator separates the image representation from the learning technique thus allowing advances in machine vision (e.g. segmentation, object recognition etc.) to simplify the learning process. For example, if we can generate more sophisticated instances (e.g. objects) then the learning mechanism can be simplified significantly.

In addition to providing a framework to learn object class models in the form of

flexible templates, we also showed how to match such templates efficiently in order to detect objects in new images. In Chapter 5, we presented a simple method for detecting, localizing and recognizing classes of objects, that accommodates wide variation in an object's pose. The method utilizes a small two-dimensional template that is warped into an image, and converts localization to a one-dimensional sub-problem, with the search for a match between image and template executed by dynamic programming. The method recovers three of the six degrees of freedom of motion (2 translation, 1 rotation), and accommodates two more DOF in the search process (1 rotation, 1 translation), and is extensible to the final DOF. Our experiments demonstrate that the method provides an efficient search strategy that outperforms normalized correlation. This is demonstrated in the example domain of face detection and localization, and is extended to more general detection tasks. We also showed that we can integrate the matcher into an end-to-end system where we recover a rough object pose from the match results and use it in a two stage recognition experiment using maximization of mutual information. Our approach combines the general detection flavor of patch correlation with the ability to accommodate more degrees of freedom of object motion that are traditionally associated with feature-based recognition systems.

6.1 Future Directions

In this thesis we addressed the problem of representing and learning "visual concepts" in order to classify images based on their visual content. We explored the Multiple Instance learning framework to learn image queries represented as simple relational templates and showed that these learned concepts were effective in classifying natural scenes. We now look at ways to extend and adapt the current framework to capture a more comprehensive representation that could be used on a larger variety of images.

6.1.1 Combining image representations for classification

As we mentioned in Chapter 2, the flexible template representation [Lipson *et al.*1997] provided a way to capture global color and spatial properties of the image and were very effective for classifying natural scenes. The Complex-feature approach by DeBonet et al. [J.S.DeBonet and P.Viola1997], on the other hand, captures relations between local image properties and can be used to capture finer structure in queries like "cathedrals" or "people with hats". The framework we provide in this thesis treats images as a combination of instances and can be used to specify queries that span the whole image (e.g. natural scenes) or queries that occupy a small portion of the image (e.g. objects). We currently learn very simple low-resolution properties of the image using the flexible templates. We showed in Chapter 4, that by generating more complex instances, we can represent a wider set of queries. We used a combination of basic color, texture and shape there as proof of concept. Extending that framework to (a) generate instances using segmented components with more complex features that capture fine structure and (b) combining them using the explicit spatial relations as represented in the flexible templates could potentially capture compact representations for a variety of images.

6.1.2 The role of learned templates in model-based object recognition

Model based object recognition is a difficult task for a machine to solve directly. It is often convenient to divide up the recognition task into (a) model building which refers to the representation of the model and (b) matching which refers to the process of matching the model with the selected region. We think the work presented in this thesis can help address some of these issues as follows.

- Model Building: Model representation forms a key part of object recognition. We want a representation that can accommodate variations due to lighting, pose, within-class differences and noise among other things. One option is to

model all these variations explicitly and the other is to learn them from examples. Example based techniques have been used successfully in face recognition in [D.Beymer1993, Vetter and Poggio1996, Jones and Poggio1996] among others. These techniques however require the model images to be well segmented so that they contain just the model object (e.g. face) with a constant background. We would like to have a more flexible method for generating representations of model objects by extracting them from noisy examples. The framework provided in this thesis could be useful in generating image based models from a few examples. As shown in the thesis, very simple learned models that encode basic color and spatial relations between image regions can be quite effective in detecting a variety of objects (e.g. natural scenes, faces, cars). While these simple representations are useful for detecting object classes, we might need a more informative representation for recognizing objects. We could explore the use of a hierarchical approach starting with low resolution representations and adding richer descriptions (e.g. texture and shape) for further discrimination power at high resolutions. These hierarchical templates could be extracted from examples using a framework similar to the prototype system described in this thesis (in Chapters 3 and 4).

- Matching: We presented a technique to match simple templates in images to detect objects like faces. Instead of using a pre-defined template like we did there, we would like to explore if we can combine models learned from few examples using Diverse Density with this matching technique in order to detect instances of objects with varying pose.

Bibliography

[A. Yuille and Cohen1992] P. Hallinan A. Yuille and D. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 1992.

[Auer *et al.*1996] Peter Auer, Phil M. Long, and A. Srinivasan. Approximating hyper-rectangles: learning and pseudorandom sets. In *Proceedings of the 1996 Conference on Computational Learning Theory*, 1996.

[Auer1997] Peter Auer. On learning from multi-instance examples: Empirical evaluataion of a theoretical approach. In *Proceedings of the 14th International Conference on Machine Learning*, 1997.

[Bach *et al.*1996] J.R. Bach, C.Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R.C. Jain, and C. Shu. Virage image search engine: an open framework for image management. In *Symposium on Electronic Imaging: Science and Technology - Storage and Retrieval of Image and Video Databases*, volume 4, pages 76–87, 1996.

[Baker and T.O.Binford1981] H.H. Baker and T.O.Binford. Depth from edge and intensity based stereo. In *7th International Joint Conference on AI*, pages 631–636, 1981.

[Ballard and Brown1982] D. Ballard and C. Brown. *Computer Vision.* Prentice Hall, 1982.

[Belongie *et al.*1998] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture based image segmentation using em and its application to content-based image retrieval. In *International Conference on Computer Vision*, 1998.

[Betke and Makris1995] M. Betke and N. Makris. Fast object recognition in noisy images using simulated annealing. In *International Conference on Computer Vision*, pages 523–530, 1995.

[Blum and Kalai1998] A. Blum and A. Kalai. A note on learning from multiple-instance examples. *To appear in Machine Learning*, 1998.

[Canny1986] J.F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:34–43, 1986.

[C.Nastar1998] C.Meilhac C.Nastar, M.Mitschke. Efficient query refinement for image retrieval. In *Computer Vision and Pattern Recognition*, 1998.

[Comaniciu and Meer1997] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *Computer Vision and Pattern Recognition*, pages 750–755, 1997.

[Cootes and Taylor1993] T. F. Cootes and C. J. Taylor. Building and using flexible models incorporating gray level information. In *Int. Conf. on Computer Vision*, pages 242–246, 1993.

[Corman *et al.*] T. Corman, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* McGraw Hill.

[Cox *et al.*1998] I. Cox, M. Miller, T. Minka, and P. Yianilos. An optimized interaction strategy for bayesian relevance feedback. In *Computer Vision and Pattern Recognition*, 1998.

[D. Forsyth1997] M. Fleck D. Forsyth. Body plans. In *Computer Vision and Pattern Recognition*, 1997.

[Das *et al.*1997] M. Das, E. Riseman, and B. Draper. Focus: Searching for multi colored objects in a diverse image database. In *Computer Vision and Pattern Recognition*, 1997.

[D.Beymer1993] D.Beymer. Face recognition under varying pose. Technical report, MIT, 1993.

[Dietterich *et al.*1994] T. G. Dietterich, A. Jain, R. Lathrop, and T. Lozano-Pérez. A comparison of dynamic reposing and tangent distance for drug activity prediction. In *Advances in Neural Information Processing Systems 6*. Morgan Kauffman, 1994.

[Dietterich *et al.*1997] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal*, 89, 1997.

[E. Simoncelli and Farid1995] R. Buccigrassi E. Simoncelli and H. Farid. Shiftable pyramid software library. Developed at Computer Information Science Dept., University of Pennsylvania, 1995.

[Felzenszwalb and Huttenlocher1998] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *Computer Vision and Pattern Recognition*, 1998.

[Flickner *et al.*1995] M. Flickner, Harpreet S. Sawhney, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28:23–32, 1995.

[Foley and Dam1984] J.D. Foley and A. Van Dam. *Fundamentals of interactive computer graphics*. Addison Wesley, 1984.

[Forney1973] G.D. Forney. The viterbi algorithm. 61:268–278, 1973.

[Freeman and Adelson1991] W. Freeman and E.H. Adelson. The design and use of steerable filters. *PAMI*, 13(9), 1991.

[H. Rowley and Kanade1995] S. Baluja H. Rowley and T. Kanade. Human face detection in visual scenes. Technical report, Carnegie Mellon University, 1995.

[H.G.Barrow1976] H.G.Barrow. Interactive aids for cartography and photo interpretation. Technical report, Stanford research Institute, 1976.

[H.Greenspan1994] H.Greenspan. Learning texture discrimination rules in a multiresolution system. *PAMI*, 16(9):894–901, 1994.

[Hornegger1995] J. Hornegger. Statistical learning, localization and identification of objects. In *Int. Conf. on Computer Vision*, 1995.

[Hsu *et al.*1995] W. Hsu, T.S. Chua, and H.K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proc. ACM Intern. Conf. Multimedia*, pages 303–313, 1995.

[Huang *et al.*1997] J. Huang, S. Ravikumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *Computer Vision and Pattern Recognition*, 1997.

[Huttenlocher and Ullman1990] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[Huttenlocher *et al.*1993] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15:850–863, 1993.

[Huttenlocher *et al.*1996] D. Huttenlocher, R. Lilien, and C. Olson. Object recognition using subspace methods. In *European Conference on Computer Vision*, pages 537–545, 1996.

[Jacobs *et al.*1995] C. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. In *SIGGRAPH*, 1995.

[Jain *et al.*1994] A. N. Jain, T. G. Dietterich, R. H. Lathrop, D. Chapman, R. E. Critchlow Jr., B. E. Bauer, T. A. Webster, and T. Lozano-Pérez. Compass: A shape-based machine learning tool for drug design. *Journal of Computer-Aided Molecular Design*, 8:635–652, 1994.

[J.Garding and T.Lindeberg1996] J.Garding and T.Lindeberg. Direct computation of shape cues using scale adapted spatial derivative operators. *International Journal of Computer Vision*, 17, 1996.

[J.Malik and P.Perona1990] J.Malik and P.Perona. Pre-attentive texture discrimination with early vision mechanisms. *Journal of Opt. Society of America*, 1990.

[Jones and Poggio1996] M. Jones and T. Poggio. Model matching by linear combination of prototypes. Technical report, MIT, 1996.

[J.S.DeBonet and P.Viola1997] J.S.DeBonet and P.Viola. Structure driven image database retrieval. In *Neural Information Processing Systems*, volume 10, 1997.

[Lipson *et al.*1997] P. Lipson, E. Grimson, and P. Sinha. Context and configuration based scene classification. In *Computer Vision and Pattern Recognition*, 1997.

[Lipson1996] P. Lipson. *Context and Configuration Based Scene Classification.* PhD thesis, MIT, 1996.

[Long and Tan1996] P. M. Long and L. Tan. Pac-learning axis alligned rectangles with respect to product distributions from multiple-instance examples. In *Proceedings of the 1996 Conference on Computational Learning Theory*, 1996.

[Mahmood and Zhu1998] S.T.F. Mahmood and W. Zhu. Image organization and retrieval using a flexible shape model. In *Content Based Access of Image and Video Libraries*, 1998.

[Mahmood1993] S.T.F. Mahmood. *Attentional Selection in Object Recognition.* PhD thesis, MIT, 1993.

[Manjunath and W.Y.Ma1991] B.S. Manjunath and W.Y.Ma. Texture features for browsing and retrieval of image data. *Pattern Recognition and Machine Intelligence*, 1991.

[Maron and Lakshmi Ratan1998] O. Maron and A. Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *Machine Learning: Proceedings of the 15th International Conference*, 1998.

[Maron and Lozano-Pérez1998] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10*. MIT Press, 1998.

[Maron1998] O. Maron. Learning from ambiguity. Doctoral thesis, Dept. of Electrical Engeineering and Computer Science, M.I.T., June 1998.

[Minka and Picard1996] T. Minka and R. Picard. Interactive learning using a society of models. In *Computer Vision and Pattern Recognition*, 1996.

[Murase and Nayar1994] H. Murase and S. Nayar. Learning and recognition of 3-d objects from brightness images. In *AAAI Fall Symposium Working Notes, AAAI*, 1994.

[Ohta and Kanade1985] Y. Ohta and T. Kanade. Stereo by intra and inter-scanline search using dynamic programming. *Pattern Analysis and Machine Intelligence*, 1985.

[Oren *et al.*1997] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Computer Vision Pattern Recognition*, 1997.

[Pass *et al.*1996] G. Pass, R.Zabih, and J. Miller. Comparing images using color coherence vectors. In *ACM International Conference Multimedia*, 1996.

[Pentland *et al.*1994] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision Pattern Recognition*, 1994.

[Pentland *et al.*1996] A. Pentland, R. Picard, and S. Sclarof. Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 1996.

[Press *et al.*1992] W. Press W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: the art of scientific computing.* Cambride University Press, 1992.

[Quine1960] W.V. Quine. *Word and Object.* Cambridge MIT Press, 1960.

[R. Romano and Poggio1996] D.Beymer R. Romano and T. Poggio. Face verification for real time applications. In *ARPA Image Understanding Workshop*, 1996.

[Rao and Ballard1995] R.P. Rao and D. Ballard. Object indexing using and iconic sparse distributed memory. Technical report, University of Rochester, 1995.

[Ratan and Dang1996] A. Ratan and D. Phi Bang Dang. Wavelets for indexing into databases: A study. Class project for 9.520, 1996.

[Rosch1978] E. Rosch. Principles of categorization. *Cognition and Categorization*, 1978.

[Rowley *et al.*1998] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Rotation invariant neural network-based face detection. In *Computer Vision and Pattern Recognition*, 1998.

[Rubner *et al.*1998] Y. Rubner, C. Tomasi, and L. Guibas. A metric for distributions with applications to image databases. In *Proceedings of IEEE Int. Conf. on Computer Vision*, 1998.

[Rucklidge1994] W. J. Rucklidge. Locating objects using the hausdorff distance. In *International conference of computer vision*, 1994.

[Sakoe and Chiba] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *IEEE Transactions Acoustics, Speech and Signal Processing.*

[Schneiderman and Kanade1998] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Computer Vision and Pattern Recognition*, 1998.

[Shashua and Ullman1988] A. Shashua and S. Ullman. Structural saliency:the detection of globally salient structures using a locally connected network. In *International Conference on Computer Vision*, 1988.

[Shi and Malik1997] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Computer Vision and Pattern Recognition*, 1997.

[Sinha1994] P. Sinha. Image invariants for object recognition. In *Investigative Opthamology and Visual Science*, 1994.

[Smith and Chang1996] J. Smith and S. Chang. Visualseek: a fully automated content-based image query system. In *Proceedings of the ACM International Conference on Multimedia*. Morgan Kaufmann, 1996.

[Stricker and Dimai1996] M. Stricker and A. Dimai. Color indexing with weak spatial contstraints. In *SPIE Proceedings*, 1996.

[Sung and Poggio1994] K. Sung and T. Poggio. Example based learning for view-based human face detection. Technical report, MIT, 1994.

[Swain and Ballard1991] M.J. Swain and D. H.. Ballard. Color indexing. *International Journal of Computer Vision*, 1991.

[Turk and Pentland1991] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 1991.

[Ullman and Basri1991] S. Ullman and R. Basri. Recognition by linear combination of models. *Pattern Analysis and Machine Intelligence*, 1991.

[Vetter and Poggio1996] T. Vetter and T. Poggio. Image synthesis from a single example image. In *European Conference on Computer Vision*, 1996.

[Viola and Wells1995] P. Viola and W. Wells. Alignment by maximization of mutual information. In *International Conference of Computer Vision*, 1995.

[Viola1996] P. Viola. Complex feature recognition: A bayesian approach for learning to recognize objects. Technical report, MIT, 1996.

[Viterbi1967] A.J. Viterbi. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Trans.on Information Theory*, 1967.

[W.Forstner1994] W.Forstner. A framework for low level feature extraction. In *Proc. European Conf. of Computer Vision*, 1994.

[Yang1998] Cheng Yang. Image database retrieval with multiple instance learning techniques. Master's thesis, MIT, 1998.

[Zahn1971] C.T. Zahn. Graph-theoretic methods for detecting and describibg gestalt clusters. *IEEE Transactions Computer*, 1971.

[Z.Wu and R.Leahy1993] Z.Wu and R.Leahy. An optimal graph theoretic approach to data clustering: Theory and its applications to image segmentation. *Pattern Analysis and Machine Intelligence*, 1993.