A Trainable View-Based Object Detection System
Thesis Proposal

Henry A. Rowley

Abstract

Object detection is a fundamental problem in computer vision. Techniques for addressing this problem range from those matching three-dimensional geometric models with images to using two-dimensional view-based representations. My previous work has shown that the latter technique can be effectively implemented using standard pattern recognition techniques (such as clustering or small neural networks). This approach has been successfully used to detect faces and eyes in cluttered backgrounds. In developing a view-based object detector, three main issues arise. First, to what extent should the problem be partitioned into sets of views or sets of features of the object, so that a simple pattern recognition algorithm will provide sufficient accuracy? Second, what type of pattern classifier should be used, and how should it be trained best detect an individual view or feature? Third, how should the outputs from multiple detectors be combined into a single decision about the presence of an object?

I propose to study the view-based approach to object detection. This will include experimental studies of how views and features should be selected, and how this selection effects detection performance. Some preliminary results in face and car detection show that partitioning the problem is effective, but more systematic study is needed. Other factors that effect detector accuracy include the classifier architecture itself (initially, I will use neural networks, but other tools are available), and on how the training examples are generated. Finally, I will study how the detections of individual views or features should be combined to give a final detection result. For view partitioning, I have found that simple arbitration strategies work well, while for feature partitioning, other researchers have used graph matching techniques. The system will be tested in at least the face detection domain, and in other domains to illustrate parts of the algorithm.

Thesis Committee:
Takeo Kanade, Chair
Shumeet Baluja
Dean Pomerleau
Manuela Veloso
Tomaso Poggio, MIT

1 Introduction

Object detection is a fundamental problem in computer vision, and there have been many attempts to solve of this problem. The techniques used range from matching three-dimensional geometric models of object with images, to using two-dimensional view-based representations. Previous work has shown that the latter technique can effectively detect faces and eyes in cluttered backgrounds [Rowley *et al.*, 1995, Sung and Poggio, 1994, Vaillant *et al.*, 1994]. In this thesis, I propose to study the view-based approach to object detection, and to evaluate this approach in a variety of domains. In developing a view-based object detector, three main issues arise. First, how should one partition the views of the object, so that a simple pattern recognition algorithm can detect the individual views? Similarly, how can sub-features of the object be separated and used to simplify the detection problem? Second, what is the best way to train the detector for an individual view or feature? This includes questions of generating training examples and determining the type of the classifier. Third, how should one combine the outputs from multiple detectors into a final decision about the presence of an object? Previous work suggests that graph matching or other geometric constraints can successfully merge individual feature detections [Leung *et al.*, 1995, Yow and Cipolla, 1996], while efficiently merging multiple view detectors requires further research.

The problems of object detection and object recognition are closely related. An object recognition system can be built out of a set of object detectors, each of which detects one object of interest. Similarly, an object detector can be built out of an object recognition system; this object recognizer would either need to be able to distinguish the desired object from all other objects that might appear in its context, or have an *unknown object* class. Thus the two problems are in a sense identical, although in practice object recognition systems are rarely tuned to deal with arbitrary backgrounds, and object detection systems are rarely trained on a sufficient variety of objects to build an interesting recognition system. The different focuses of these problems lead to different representations and algorithms. In this work, I will concentrate on object detection.

This proposal is based on my previous work in the frontal face detection domain, in which a view-based neural network approach proved successful. This work is described in Section 2, along with a discussion of how the problems of separating and recombining examples and training were addressed in this domain. Section 3 describes a much more challenging domain, that of detecting cars. Section 4 examines the differences between detection problems which have an effect on their difficulties. More details on my approach to view-based object detection are given in Section 5, and related work is given in Section 6. Finally, in Section 7 I list the expected contributions of this work, and give the time table in Section 8.

2 Case Study: Frontal Face Detection

In recent years, there have been several research efforts in the problem of detecting faces in arbitrary images. Some of them use matching of hand-coded templates [Yang and Huang, 1994, Seitz and Lang, 1992], others use geometric models [Leung *et al.*, 1995, Yow and Cipolla, 1996], while others use a view-based approach, treating the whole face as a pattern to be recognized [Rowley *et al.*, 1995, Sung and Poggio, 1994, Vaillant *et al.*, 1994, Lin *et al.*, 1996].

This section briefly presents my face detection system (from [Rowley *et al.*, 1995]), which locates frontal, upright faces at any location or scale in grayscale images. It is a view-based technique, implemented using neural networks.

Face detection has a wide variety of applications, such as in user interfaces where locating a user is important, and as a front-end for face recognition system. The system described in this section has already been incorporated into three projects. The first two are part of the Informedia project at CMU: using faces as one feature for selecting important frames in a video sequence [Smith and Kanade, 1996], and using located faces which are automatically matched with names news broadcasts [Sato, 1996]. Finally, the WebSeer project has used this face detector to allow searches for images containing faces on the World Wide Web [Frankel *et al.*, 1996].

2.1 The Detector

The detector is only concerned with frontal views of faces, and the face is not being split into sub-features. Thus there is no explicit partitioning of the detection problem into smaller pieces. Thus we will move directly to a description of the classifier itself, which consists of two main steps: extracting and preprocessing windows of the input image and applying a neural network to classify these windows.

2.1.1 Extracting and Preprocessing Windows

The first component of our system extracts 20 by 20 pixel regions from the image, for subsequent processing by a neural network. To detect faces anywhere in the input, the windows are extracted at every location in the image. To detect faces larger than the window size, the input image is repeatedly reduced in size (by subsampling), and the filter is applied at each size, as shown in Figure 1.

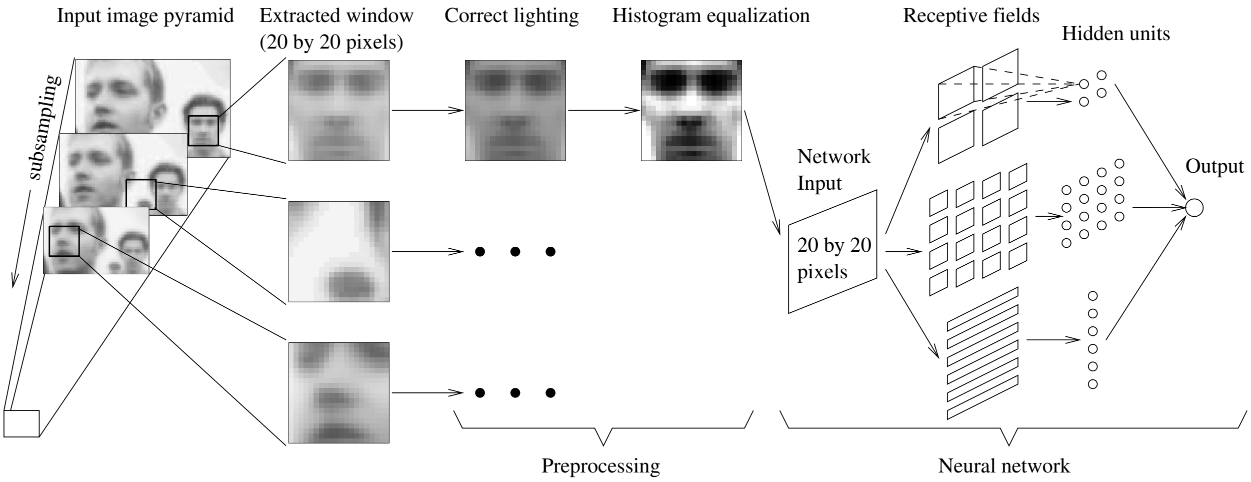

Figure 1: The basic algorithm used for face detection.

The window is then preprocessed, using a method developed for another face detection system [Sung and Poggio, 1994]. The preprocessing first attempts to equalize the intensity values in across the window. We fit a function which varies linearly across the window to the intensity values. The linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions. Then histogram equalization is performed, which nonlinearly maps the intensity values to expand the range of intensities in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases.

2.1.2 Classification with a Neural Network

The preprocessed window is then passed through a neural network. The network has retinal connections to its input layer; the receptive fields of hidden units are shown in Figure 1. There are three types of hidden units: 4 that look at 10x10 pixel subregions, 16 that look at 5x5 pixel subregions, and 6 that look at overlapping 20x5 pixel horizontal stripes. Each of these types was chosen to allow the hidden units to represent features that might be important for face detection. In particular, the horizontal stripes allow the hidden units to detect such features as mouths or pairs of eyes, while the hidden units with square receptive fields might detect features such as individual eyes, the nose, or corners of the mouth. Although the figure shows a single hidden unit for each subregion of the input, we found experimentally that two or three copies give the best performance. Similar input connection patterns are commonly used in speech and character recognition tasks [Waibel *et al.*, 1989, Le Cun *et al.*, 1989]. The network has a single, real-valued output, which indicates whether or not the window contains a face.

Examples of output from a single network are shown in Figure 2. In the figure, each box represents the position and size of a window to which the neural network gave a positive response. The network has some invariance to position and scale, which results in multiple boxes around some faces. All the faces are in face detected at least once. There are a number false detections; many of them will be eliminated by methods presented in next section.

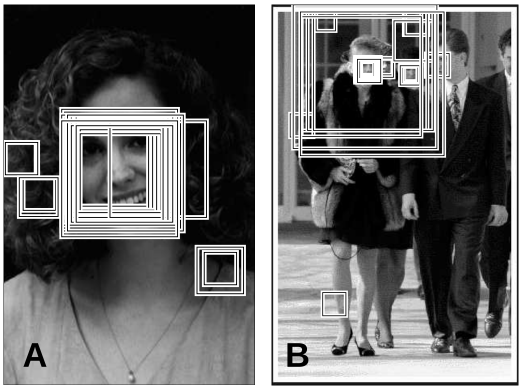

Figure 2: Images with all the above threshold detections indicated by boxes.

To train the neural network used to serve as an accurate classifier, a large number of face and non-face images are needed. Nearly 1050 face examples were gathered from face databases at CMU and Harvard[1]. The images contained faces of various sizes, orientations, positions, and intensities. The eyes and the center of the upper lip of each face were located manually, and these points were used to normalize each face to the same scale, orientation, and position. Although

[1]Dr. Woodward Yang at Harvard provided over 400 mug-shot images which we used for training.

the labelling process is quite tedious, it is important for the accuracy of the detector to reduce the variation in the training images caused by misalignments.

In the training set, fifteen face examples are generated from each original image, by randomly rotating the images (about their center points) up to $10°$, scaling between 90% and 110%, translating up to half a pixel, and mirroring. Each 20 by 20 pixel window in the set is then preprocessed (by applying lighting correction and histogram equalization). A few example images are shown in Figure 3. The randomization gives the filter invariance to translations of less than a pixel, rotations of less than $10°$, and scalings of 20%. The size of the windows must be large enough to show all the facial features (indeed, this resolution is large enough for a human to recognize familiar faces). However, making the window too large will increase the complexity of the neural network, making training more difficult and running the system more time consuming.

Figure 3: Example face images, randomly mirrored, rotated, translated, and scaled by small amounts.

Practically any image can serve as a non-face example because the space of non-face images is much larger than the space of face images. However, collecting a "representative" set of non-faces is difficult. Instead of collecting the images before training is started, the images are collected during training, in the following manner (a similar idea was used in [Sung and Poggio, 1994], although the details of the implementation are probably quite different):

1. Create an initial set of non-face images by generating 1000 images with random pixel intensities. Apply the preprocessing steps to each of these images.

2. Train a neural network to produce an output of 1 for the face examples, and -1 for the non-face examples. The training algorithm is standard error backpropogation. Before the first iteration of this loop, the network's weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point for training.

3. Run the system on an image of scenery *that contains no faces*. Collect subimages in which the network incorrectly identifies a face (an output activation greater than zero).

4. Select up to 250 of these subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

Some examples of non-faces that are collected during training are shown in Figure 4. We used 120 images of scenery for collecting negative examples in this bootstrap manner. A typical training run selects approximately 8000 non-face images from the 146,212,178 subimages that are available at all locations and scales in the training scenery images.

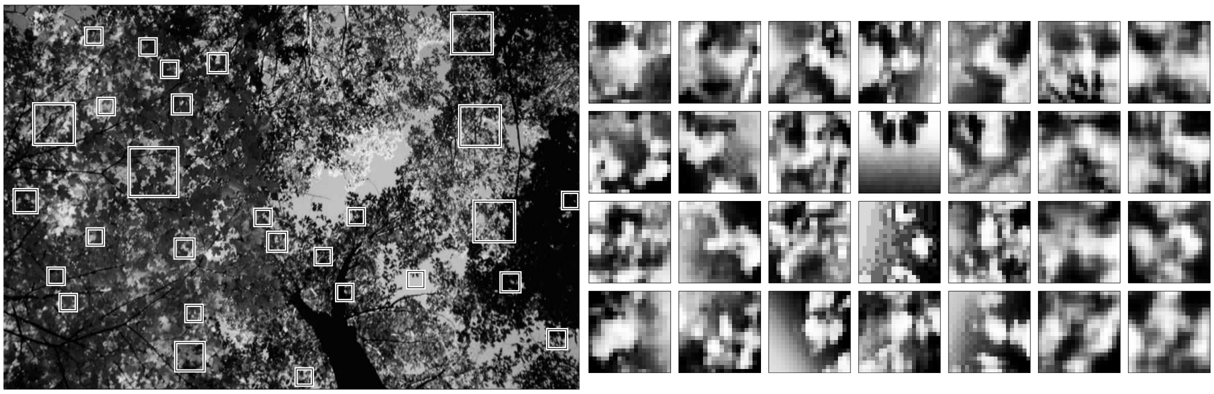

Figure 4: During training, the partially-trained system is applied to images of scenery which do not contain faces (like the one on the left). Any regions in the image detected as faces (which are expanded and shown on the right) are errors, which can be added into the set of negative training examples.

2.2 Problem Decomposition and Arbitration

The examples in Figure 2 showed that the system described so far cannot eliminate all false detections. The neural network has partitioned some portion of the set of negative examples from the set of positive examples, but allows some to slip through. To reduce the number of false positives, we can apply two or more networks and use arbitration to produce the final decision. Each network is trained by the procedure described above, but with different random initial weights, random initial non-face images, and random permutations of the order of presentation of the scenery images. The detection and false positive rates of the individual networks are quite close. However, because of different training conditions and because of self-selection of negative training examples, the networks will have different biases and will make different errors. Thus, the two networks will implicitly partitioned the problem into smaller pieces.

To combined the results from each network, we used a simple arbitration strategy. Each detection by a network at a particular position and scale is recorded in an image pyramid. We combine two such pyramids by ANDing. This strategy signals a detection only if both networks detect a face at precisely the same scale and position. This ensures that, if a particular false detection is made by only one network, the combined output will not have that error. The disadvantage is that if an actual face is detected by only one network, it will be lost in the combination. Similar heuristics, such as ORing the outputs or voting among three networks, were also tried. Variants of ANDing two networks and voting amount three networks gave the best performance.

Theoretically, it should be possible to train a single large neural network which performs the same computation as two smaller networks plus the fixed arbitration step. However, as the size of the neural network is increased, so does the difficulty in training it, in terms of computational efficiency, number of pattern presentations, and number of examples images required.

Further heuristics (applied either before or after the arbitration step) can be used to improve the performance of the system. Note in Figure 2 that most faces are detected at multiple nearby positions or scales, while false detections often occur at single locations. At each location in an image pyramid representing detections, the number of detections within a specified neighborhood of that location can be counted. If the number is above an empirically determined threshold, then that location is classified as a face. These detections are then collapsed down to a single point, located at their centroid. When this is done before arbitration, the centroid locations rather than the actual outputs from the networks are ANDed together.

If we further assume that a position is correctly identified as a face, then all other detections that overlap with it are likely to be errors, and can therefore be eliminated. There are relatively few cases in which this heuristic fails; however, one such case is illustrated in the left two faces in Figure 2(B), in which one face partially occludes another.

Taken together, the heuristics of merging detections and arbitrating among multiple networks can reduce the false detection rate significantly. Results of applying two networks to a set of images, and arbitrating their outputs using an AND operator are shown in Figure 5.

2.3 Analysis, or "Why Does It Work?"

The way in which the face detection problem is posed makes it sound difficult: we need to detect faces at unknown locations and scales in the image, and need to deal (to some extent) with variations in the pose of the face, lighting, and overall appearance. Given these challenges, the fact that the simple algorithm described above works well seems remarkable.

The reason that the neural network works so well is that it does not have to deal with most of the above difficulties. Instead, we use brute force search to deal with faces at different positions and scales, and the preprocessing algorithm removes much of the variation due to lighting and camera characteristics. The neural network must deal with overall appearance variation, but in frontal views of faces, there is simply not much variation. Thus a simple neural network is able to detect faces quite accurately.

In addition, the use of two or more neural networks implicitly breaks down the problem into smaller pieces: each network has the task of eliminating some portion of the set of non-face images. The outputs from multiple networks are then combined, to give more accurate results than are possible from a single network.

3 A Harder Problem: Car Detection

Recent research on automated driving and automated highway systems has spurred interest in processing images containing cars. Most algorithms in this domain make specific assumptions about the context in which they will be applied—for instance, that the camera is always pointed at a road, so any moving object will be a car [Dubuisson and Jain, 1995, Koller *et al.*, 1993]. The remaining problems are in tracking and motion segmentation, and removing moving artifacts such as shadows. The use of such cues is not restricted only to car detection. For instance, [Bobick and Pinhanez, 1995] used contextual information to guide vision algorithms used to control video cameras, in order to frame camera shots in an instructional video.

Figure 5: For each image, three numbers are shown: the number of faces in the image, the number of faces detected correctly, and the number of false detections. Some notes on specific images: False detections are present in A and J. Faces are missed in G (babies with fingers in their mouths are not well represented in the training set), I (one because of the lighting, causing one side of the face to contain no information, and one because the eyes are obscured), and J (removed because a false detect at the right location but wrong scale overlapped with it). Although the system was trained only on real faces, hand drawn faces are detected in D. Images A, I, and K were obtained from the World Wide Web, B was scanned from a photograph, C is a digitized television image, D, E, F, H, and J were provided by Sung and Poggio at MIT, G and L were scanned from newspapers, and M was scanned from a printed photograph.

The use of context does not eliminate the need for detection algorithms, so a car detector which does not rely exclusively on motion cues is of interest. Ideally, we would like to detect any car (broadly defined as a four-wheeled vehicle for transporting people), from any viewpoint, which appears in a single grayscale image against an arbitrary background.

One naive way to solve this problem is to try the same solution as was used for frontal faces. However, even for the simplified problem of detecting a car viewed from a single viewpoint, such as the side, the detector's performance was poor. The car detection problem has several sources of difficulty that are not present in the frontal face detector. In the face detector, the neural network had to deal with essentially one pattern, that of a frontal face, which does not vary much from person to person. For the car detector to work, the neural network would have to detect any car shape (of which there are a great variety), and to deal with appearance changes caused by specular reflections of the environment off of the car bodies. Furthermore, while detecting frontal faces is useful in itself, detection cars from only one viewpoint is less useful.

These challenges make car detection a much harder problem than frontal face detection. Separating views of the cars for individual detection does not provide sufficient simplification for a neural network based detector. One further simplification is to look for features which appear in all cars, but are more uniform than the image of the entire car. The most obvious such feature is the tires. Even the detection of tires presents difficulties that are not present in the face detector: the appearance of the hub-cap can vary dramatically, and the background the tire appears against also varies with the color of the car and brightness of the road.

We first attempted to detect tires from all viewpoints, using two identically trained networks, as was used for the face detector. The resulting system had much poorer accuracy than the face detector, so the viewpoints were split up among three sets of networks (viewing the tire from the front, side, and rear of the car). This strategy works well, as can be seen in Figure 6, although the performance is still not as accurate as the face detector. The remaining differences might be due to the relatively smaller training set used for the tire detector.

A rather different network architecture, which has shown good results for the tire detection domain, is based on the hierarchical neural-network model proposed in [Sajda *et al.*, 1996]. The idea is that some portion of the tire detection task can be solved easily using a lower resolution image, and the remaining distinctions can be made with a separate (and therefore simpler) decision process at a higher resolution. This was implemented in the tire detection domain as follows: We first train a network to perform the tire detection task with a low resolution image—half the resolution used by the original tire detector. However, this image has the same number of pixels, thus it looks at a larger area of the image. This detector has a rather high false detection rate, but rarely rejects real tires. A second network is trained, receiving higher resolution input windows. It has an additional input: the hidden unit activations from the low resolution network. These hidden units can be thought of as feature detectors, and thus their activations are features describing the context of the window (recall that the lower resolution network looks at a larger area of the image than the high resolution network). This second network is also trained to perform the tire detection task, while holding the the first network's weights constant. This approach works as well as the system splitting the tire into three viewpoints, but has the advantage that the example images do not need to be partitioned (which is currently a tedious manual process).

The hierarchical architecture can also be seen as an instance of partitioning the problem into smaller pieces—in this case, features of different resolutions—and then combining the results later.

Figure 6: Results of tire detection.

4 Challenges in Object Detection

In the two detection problems described above, the classification was performed by a neural network. The hidden units of a neural network divide the input image space into half spaces, and the output of the network is some boolean function of the position of the input with respect to each hidden unit's dividing hyperplane. While this characterization is not completely precise, it provides an intuitive understanding of what characteristics of the classification problem will cause difficulties. Partitioning views or sub-features of an object among multiple detectors can be seen as further subdividing the detection problem in image space into pieces that must be put back together later.

Formally, object detection is the problem of determining whether or not a sub-window of an image belongs to the set of images of an object of interest. Thus, anything that increases the complexity of the boundary of the set of images of the object will increase the difficulty of the problem, and possibly increase the number of errors the detector will make.

Increasing the amount of variability of the set of images of an object will not necessarily make the classification problem harder; it is possible that, by adding new images to the set of images of the object, the decision boundary becomes simpler and easier to learn. One way to imagine this happening is that the boundary of the set of object images is smoothed by adding more images into the set. However, the conservative assumption is that increasing the variability of the set will make the boundary more complex, and thus make the detection problem harder.

There are many sources of variability in the detection problem. These sources are outlined below, with references to how these problems have been addressed in previous work or suggestions of how they should be addressed in the future. Some of these sources of variation can be dealt with using preprocessing, before the window is classified. Others are handled by partitioning the input space among a number of detectors. Finally, some variation is left to the classifier itself, and may require changes type of pattern classifier used.

4.1 Variation in Image Plane

The simplest type of variability in images of an object can be expressed independently of the object itself, by rotating, translating, scaling, and mirroring the object's image. Each of these can be handled in the detector by a brute force approach. To detect objects at different locations, we examine sub-windows from every pixel location in the image. For objects of different sizes, we generate scaled versions of the image, and for objects of different in-plane rotations, we extract sub-windows of different orientations. As might be imagined, this approach leads to a rather slow detection system; some ideas for speeding up the system are briefly mentioned in Section 5.1.

In the face detection work, scale and position variation have been handled in this brute force manner. Rotation was not dealt with, because detection of only upright faces was thought to be sufficient.[2]

For robustness, the neural network was trained with rotations of the face examples of up to $10°$. Incorporating larger amounts of rotation into the detector increases the false detection rate, while lowering the amount will decrease the detection rate. Part of the proposed work of this thesis will be to examine these tradeoffs.

4.2 Object Pose

The previous subsection dealt with some aspects of object pose, those which could be explained by in-plane rotation and translation. Rotations of the object that are not in-plane can have a larger impact on its appearance. Another source of variation is the distance of the object from the camera, changes in which can result in perspective distortion.

In the face detector, small changes in head pose are accommodated by including rotated heads in the training set (this is also how a small amount of variation of in-plane rotation is handled). To handle larger amounts of out-of-plane rotation, we have begun experiments with multiple detectors, one for each orientation of the head. It may not make sense to detect all the viewpoints into a single detector, as it is no longer clear how to normalize the examples: how should one optimally place a profile and a frontal view of a face in the same 20 by 20 pixel window? The face detector assumed that faces were far enough from the camera that perspective distortion would not have an effect.

An approach that has proven useful in the face detection domain is to detect smaller features of the face, and then to match them using a statistical or graph model which accounts for changes in the relative positions of the features [Leung *et al.*, 1995, Yow and Cipolla, 1996]. However,

[2]Since implementing a real time demonstration of the face detector, this assumption has proven false. After a person's upright face has been detected, the first thing most people do is turn their heads $90°$ to the side, causing the demonstration to fail! However, adding brute force invariance to rotation would slow down the demonstration unacceptably.

this approach still requires the detection of the smaller features in a pose invariant manner, so the problem has not been solved. It may have been simplified, however, because the features are less complex.

In the tire detection domain, preliminary experiments showed that hierarchical detectors—those which examine the image at two different resolutions and thereby have access to more contextual information—may also help. Although the pose of the object may change the appearance of its sub-features dramatically, the appearance of the context of those features might change less. Detecting the context first helps the object detector work better.

4.3 Lighting and Texture Variation

Up to now, I have described variations due to the position and orientation of the object with respect to the camera. Now we come to variation caused by the object itself, specifically the object's surface properties and light sources.

In the face detector, two preprocessing steps are used to attempt to compensate for these appearance variations. The first fits a linear function to the intensity values in the window, and subtracts out that function. This compensates for small lighting variations, in which one region of the face appears slightly brighter than another. The second preprocessing step is histogram equalization, which ensures that all images will use the same range of intensity values and compensates for low contrast or low light conditions.

These corrections do not work perfectly; the classifier is left to deal with such situations as when most direct illumination is from above or behind the face, with only weak indirect illumination for the facial features, or when there is extreme lighting from one side causing unusual shadows to appear on the faces. Because the training set currently consists of photographs with good lighting, the current detector has trouble in poor lighting conditions. The preprocessing techniques also do not address individual variation in facial appearance, including expressions, skin color, facial hair, glasses, or makeup. There is no algorithm which can compensate for these variations; instead, the training examples for the classifier must be a sufficiently broad sample of faces. Based on experience with the face detector, it is probably not necessary to partition the frontal views of faces into subsets for detection.

A potentially useful technique for understanding the space of object appearances under different lighting conditions was presented in [Belhumeur and Kriegman, 1996]. This work showed that the shape of the set of images of an object which has a Lambertian reflectance model is a convex cone in image space. The convex cone has a fairly simple boundary, and is potentially easily represented by a neural network classifier. Thus, with a training set which covers these variations, there is some hope that the classifier can learn to detect objects under a wide variety of illumination conditions. In addition, this technique suggests ways to generate new training examples of objects under different lighting, and may lead to a way to directly compensate for lighting variation as a preprocessing step.

4.4 Background Variation

In his thesis, Sung suggested that with current pattern recognition techniques, the view-based approach to object detection is only applicable for objects that have "highly predictable image boundaries" [Sung, 1996]. When an object has a predictable boundary, it is possible to extract a window which contains only pixels within the boundary, and to ignore the background.

Preliminary experiments with tires and profile views of faces, both of which include portions of the background in their input windows, have shown that view-based detection is possible without completely predictable object boundaries. In particular, to extend the face detector to work with profile views of faces, one needs to take into account the boundary of the face, rather than ignoring it. The detector will thus see background pixels in its input, and because the shape of the boundary varies, it cannot simply ignore the background pixels. The training images must incorporate sufficient variation in the background portions of the images. Preliminary attempts to train profile face detectors indicate that, while it is a harder problem than frontal face detection (based on the higher false detection rates), it may be feasible.

4.5 Shape variation

A final source of variation is the shape of the object itself. This variation comes in two flavors. First, objects with rather generic definitions (like cars or chairs) may have a wide variety of shapes. Second, even a single object (like a person) may change shape over time as a result of articulated motions.

The car detection domain exhibits a great deal of shape variation. Cars can be loosely defined as any object with four wheels used for transporting a small number of people. As described earlier, naive attempts to detect whole cars failed, but detection of smaller, more uniform parts like the tires worked better. This indicates that while the overall shape may vary, smaller parts of it might be more consistent and easier to detect.

Articulated motions are a further source of variation. In cars, the articulated motions are quite limited—just opening and closing the car doors. But for people, the articulated motions are a much larger source of variation. Although this has not been tested, it may be possible to apply the same solution as proposed for car shape variation, namely that of detecting smaller pieces (such as the head and limbs) in a pose invariant manner, and then combining those detection results, although the limbs have fewer uniform distinguishing features than various parts of cars. This source of variability is probably beyond the scope of this thesis.

5 Proposed Approach

For my thesis, I plan to apply a view-based approach, implemented using neural networks, to a variety of object detection problems. The goal is to explore the capabilities and limitations of this technique, and to automate as much as possible the training algorithms used.

Specifically, I will address the following three subproblems: First, how should the range of views of an object be partitioned, so that a simple pattern recognition system will be effective? Second, the issues of how to generate training examples and how to train the system must be addressed. Third, how should the results from a set of feature or view detectors be combined? These points are discussed in more detail below.

5.1 Partitioning Views and Features

An important issue is how much variation in object pose a neural network based pattern recognition system can deal with. I propose a systematic study of how much translation, rotation, and scale invariance can be handled in a neural network, measuring the detection accuracy of the network as a function of the amount of variability in each of these dimensions. With this information, it should be straightforward to determine how the views of the object should be partitioned.

A related problem is whether smaller features of an object should be used for detection rather than the whole object, and if so how these sub-features should be selected. [Spence *et al.*, 1996] has some work on automatically making this selection, but it has not been demonstrated in a real domain. Some feature selection choices may be obvious—in the face detection domain, for example, the eyes, nose, and mouth are clearly useful. In fact, there is evidence that such simple features as corners may be sufficient, given intelligent methods for merging detections of those features [Yow and Cipolla, 1996].

In other domains, such as car detection, useful features may not be so obvious. Tires seem to be the most uniform features, but they are not visible from all viewpoints, and so therefore are not sufficient by themselves. Other features, such as bumpers or headlights, are less uniform, but may still be detectable. Features such as windows will change their appearance dramatically, depending on the environment, and are probably not reliably detectable using current techniques.

While selecting features automatically may prove difficult, it may be possible to develop a figure of merit for a set of examples of a specific feature. This measure would describe how much variability exists in a feature, and whether that amount of variability is small enough for the feature to be detected accurately.

In the face detector described in Section 2, multiple neural networks were trained with the same data, and they implicitly partitioned the problem so that each network ruled out some portion of the non-face images. The EM (Expectation-Maximization) statistical framework might provide a way to explicitly do this partitioning [Dempster *et al.*, 1977]. EM is a method for fitting a model to data, some of which may be unknown. In this case, the model is the set of neural networks, the known data are the positive and negative examples, and the unknown data are the assignments of the examples to each network.

A practical concern is getting the detector to run quickly. I have performed some preliminary experiments with the technique used by Umezaki for car license plate detection [Umezaki, 1995]. Instead of evaluating windows of the image sampled at each pixel location, the classifier is trained to have some invariance to translation, and the window is moved in larger steps over the image. The price is that the number of false detections increases, but this can be alleviated by using the non-translation-invariant detector to verify each candidate location. Experiments examining this tradeoff will guide the design of similar optimizations.

5.2 Training View and Feature Detectors

Closely related to the above question of feature selection is that of training example generation. In order to reduce the amount of variability in the positive training examples, it is necessary to align the examples. In work thus far, aligning the examples was a tedious process. A number of features were manually labelled for each training instance, and an ad hoc function of these feature coordinates was used to rotate, translate, and scale each image into alignment. It may be possible to define a uniform way of aligning the images—for instance, by labelling feature points and trying to bring each feature point as close as possible to the average feature positions over all the examples.

One major problem with the current face detector is its sensitivity to lighting conditions. The training examples are faces with good lighting, so training examples without this feature are desirable. Using ideas from [Belhumeur and Kriegman, 1996], it may be possible to synthesize training images of a face under a large variety of lighting conditions, given just three images of that face with different illumination conditions.

Thus far, most of my experiments have been done with neural networks having one hidden layer, which is sparsely connected to the input window. However, some preliminary experiments in the tire detection domain have shown that a hierarchical network, which receives a low and high resolution version of the input image, may provide higher accuracy. This and other variations on the network architecture will be evaluated, along with the possibility of using other pattern classification techniques (for instance, the Support Vector Network [Cortes and Vapnik, 1995]).

Related to the issue of classifier type is the speed of the classifier. For example, it is much faster to apply a linear classifier to an image than a neural network, but a linear classifier is not as flexible. [Sung and Poggio, 1994] used a linear filter as a preprocessing step, to eliminate a large number of candidate faces before applying the more accurate (and more time consuming) classifier. A similar technique could be employed in the current work.

5.3 Merging Views and Features

In my previous work, two types of partitioning have been used. In the face detector, the partitioning happened implicitly: the networks were trained in an identical manner, but each learned to reject a subset of all non-face images. Thus, simply ANDing the results of the networks worked well. In the tire detection domain, the tire images were separated into three viewpoints, each of which was detected independently. To determine whether a tire exists at a particular location, the outputs from each of these detectors were ORed at each location in the image.

One problem with the ORing approach is that each detector will have some (hopefully small) false detection rate, and thus the OR of the network outputs will add their false detections. An alternative, suggested by [Baluja, 1996], is to train a system that, given a window of an image containing an object, decides which viewpoint the object belongs to. This *router* can then be applied at every pixel location, to determine which viewpoint-specific network should be applied. Assuming the router is accurate, such a system will have the same number of false detections as a single classifier. The router can be implemented using a neural network, or using a nearest neighbor approach like the ones described in [Nayar *et al.*, 1996, Neiberg *et al.*, 1996].

Another challenge is how to merge the detection of sub-features into the detection of the entire object. Several approaches are available in the literature, and they will be examined [Leung *et al.*, 1995, Yow and Cipolla, 1996, Viola, 1996].

6 Related Work

Throughout this document, I have mentioned pieces of work related to the specific technique under discussion. This section will summarize some of these pieces of related work, in the framework of the approach described above.

Researchers have looked at the problem of how to represent many poses of an object in an efficient way. One useful approach has been to compute eigenfeatures from the set of images of the object under different poses, thus reducing the dimensionality of the space so that a nearest-neighbor search can be effective [Nayar *et al.*, 1996, Neiberg *et al.*, 1996]. These techniques have been applied to object and pose recognition rather than object detection; in many cases, the object is already extracted from the background. Also, these systems have no explicit model of variation other than that caused by changes in object pose.

An interesting question is whether sub-features for a parts-based detector can be selected completely automatically. Some preliminary ideas presented in [Spence *et al.*, 1996] sound promising, although they have not yet been completely implemented or applied to real domains. In recent work, [Viola, 1996] has developed a mechanism to find good sub-features of a set of example images. This technique was applied to select features in oriented energy images, but may have applicability in intensity images as well.

[Sung, 1996] provided some formalization of how a set of identically trained detectors can be used together to improve accuracy. He argued that if the errors made by a detector are independent, then by having a set of networks vote on the result, the number of overall errors will be reduced. [Baker and Nayar, 1996] used the converse idea, that of pattern rejectors, for recognition. Each classifier eliminates a set of potential classifications of an example, until only the example's class is left.

In training a detector, obtaining a sufficient number of examples is an important problem. One commonly used technique is that of virtual views, in which new example images are created from real images. In my work, this has taken the form of randomly rotating, translating, and scaling example images by small amounts. [Pomerleau, 1992] made extensive use of this in training a neural network for autonomous driving. The network learns from watching an experienced driver staying on the road, but needs examples of what to do should the vehicle start to leave the road. Examples for this latter case are generated synthetically.

The work of [Vetter *et al.*, 1992] suggests that for bilaterally symmetric objects (such as faces and cars), a large number of views of the object can be generated from a single view, given information about which points in the image are symmetric points on the object. This may provide a way to synthesize example images if real examples are scarce.

7 Expected Contributions

The tangible contributions of this thesis will be a small collection of object detectors. These will include a pose invariant face detector (for which many of the pieces already exist, but need further refinement and integration), and a pose invariant car detector, to the extent that this proves feasible in the course of the research. There are a number of other objects which might be useful to detect, and might help define parts of the detection system. For instance, text has been detected by searching for high horizontal spatial frequencies in the image [Smith and Kanade, 1996]. Such features should be straightforward for a neural network based detector to learn. Detection of fruit is an application important in the automated harvesting industry, and would provide an opportunity to explore the use of color in object detection. These or other object detection problems will be used throughout the research in designing and evaluating the detector.

An intangible contribution will be a methodology for building and training an object detector, beginning with suggestions for collecting and aligning training examples, through partitioning the examples into views and sub-features, to training the classifier itself. I will also provide heuristics for assigning the various sources of variability described in Section 4 to each of the parts of the detection system: view/feature partitioning, image preprocessing, and the classifier itself.

8 Approximate Time Line

Month	Year	Activity
December	1996	Proposal
January	1997	Example alignment, lighting variation
February	1997	Data collection, implement parallel training/testing
March	1997	Industrial internship
April	1997	
May	1997	
June	1997	
July	1997	Experiments with object pose variation
August	1997	Evaluation of different classifier types
September	1997	Speed up techniques
October	1997	Feature/view selection and detection
November	1997	
December	1997	Merging and arbitration heuristics
January	1998	
February	1998	Writing
March	1998	
April	1998	
May	1998	Defense

References

[Baker and Nayar, 1996] S. Baker and S. K. Nayar. A theory of pattern rejection. In *ARPA Image Understanding Workshop*, Palm Springs, CA, February 1996.

[Baluja, 1996] Shumeet Baluja. Personal communication, 1996.

[Belhumeur and Kriegman, 1996] P. N. Belhumeur and D. J. Kriegman. What is the set of images of an object under all possible lighting conditions? In *Computer Vision and Pattern Recognition*, page 270, San Francisco, CA, 1996.

[Bobick and Pinhanez, 1995] A. Bobick and C. Pinhanez. Divide and conquer: Using approximate models to control view-based algorithms. Perceptual Computing Technical Report 357, MIT, October 1995.

[Cortes and Vapnik, 1995] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[Dempster *et al.*, 1977] A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B39:1–38, 1977.

[Dubuisson and Jain, 1995] Marie-Pierre Dubuisson and Anil K. Jain. Contour extraction of moving objects in complex outdoor scenes. *International Journal of Computer Vision*, 14:83–105, 1995.

[Frankel *et al.*, 1996] Charles Frankel, Michael J. Swain, and Vassilis Athitsos. WebSeer: An image search engine for the world wide web. TR-96-14, University of Chicago, August 1996.

[Koller *et al.*, 1993] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. CSD-93-780, University of California, Berkeley, October 1993.

[Le Cun *et al.*, 1989] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropogation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[Leung *et al.*, 1995] T. K. Leung, M. C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. In *Fifth International Conference on Computer Vision*, pages 637–644, Cambridge, Massachusetts, June 1995. IEEE Computer Society Press.

[Lin *et al.*, 1996] Shang-Hung Lin, S. Y. Kung, and Long-Ji Lin. Face recognition/detection by probabilistic decision-based neural network. Submitted to *IEEE Transactions on Neural Networks*, 1996.

[Nayar *et al.*, 1996] Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase. Real-time 100 object recognition system. In *ARPA Image Understanding Workshop*, pages 1223–1227, Palm Springs, CA, February 1996.

[Neiberg *et al.*, 1996] Leonard Neiberg, David Casasent, Robert Fontana, and Jeffrey E. Cade. Feature space trajectory neural net classifier: 8-class distortion-invarient tests. In *SPIE Applications and Science of Artificial Neural Networks*, volume 2760, pages 540–555, Orlando, FL, April 1996.

[Pomerleau, 1992] Dean Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, Feburary 1992. Available as CS Technical Report CMU-CS-92-115.

[Rowley *et al.*, 1995] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. CMU-CS-95-158R, Carnegie Mellon University, November 1995. Also available at http://www.cs.cmu.edu/˜har/faces.html.

[Sajda *et al.*, 1996] Paul Sajda, Clay D. Spence, and John C. Pearson. Learning image context for improced computer-aided diagnosis. In *ARPA Image Understanding Workshop*, pages 1375–1380, Palm Springs, CA, February 1996.

[Sato, 1996] Shin'ichi Sato. Personal communication, 1996.

[Seitz and Lang, 1992] P. Seitz and G. K. Lang. Pattern matching and object recognition employing local orientation and hierarchical spatial feature matching. In *First Swiss Symposium on Pattern Recognition and Computer Vision*, January 1992.

[Smith and Kanade, 1996] M. Smith and T. Kanade. Skimming for quick browsing based on audio and image characterization. CMU-CS-96-186R, Carnegie Mellon University, May 1996.

[Spence *et al.*, 1996] Clay Spence, John C. Pearson, and Paul Sajda. Learning hierarchical representations of objects. In *ARPA Image Understanding Workshop*, pages 1415–1427, Palm Springs, CA, February 1996.

[Sung and Poggio, 1994] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. A.I. Memo 1521, CBCL Paper 112, MIT, December 1994.

[Sung, 1996] Kah-Kay Sung. *Learning and Example Selection for Object nad Pattern Detection*. PhD thesis, MIT AI Lab, January 1996. Available as AI Technical Report 1572.

[Umezaki, 1995] Tazio Umezaki. Personal communication, 1995.

[Vaillant *et al.*, 1994] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings on Vision, Image, and Signal Processing*, 141(4), August 1994.

[Vetter *et al.*, 1992] T. Vetter, T. Poggio, and H. Bülthoff. 3D object recognition: Symmetry and virtual views. A.I. Memo 1409, MIT, December 1992.

[Viola, 1996] Paul A. Viola. Complex feature recognition: A bayesian approach for learning to recognize objects. A.I. Memo No. 1591, MIT, November 1996.

[Waibel *et al.*, 1989] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *Readings in Speech Recognition*, pages 393–404, 1989.

[Yang and Huang, 1994] Gaungzheng Yang and Thomas S. Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53–63, 1994.

[Yow and Cipolla, 1996] Kin Choong Yow and Roberto Cipolla. Feature-based human face detection. CUED/F-INFENG/TR 249, Department of Engineering, University of Cambridge, England, 1996.