10-2005

Expert Object Recognition in video

Matthew S. McEuen

Expert Object Recognition in Video

Matt McEuen

In partial fulfillment of the requirements
for the Degree of Master of Science

Department of Computer Science
Rochester Institute of Technology
Rochester, New York
October, 2005

Roger Gaborski

Professor Roger S. Gaborski, Chairperson

Carl Reynolds

Professor Carl Reynolds, Reader

Edith Hemaspaandra

Professor Edith Hemaspaandra, Observer

H. Bischof

Dr. Hans-Peter Bischof, Graduate Coordinator

Thesis/Dissertation Author Permission Statement

Title of thesis or dissertation: Expert Object Recognition in Video

Name of author: Matthew S. McEuen
Degree: MS in C.S.
Program: Comp. Sci.
College: Computing

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted:

I, Matthew S. McEuen, hereby **grant permission** to the Rochester Institute Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: Matthew S. McEuen Date: 10/7/05

Print Reproduction Permission Denied:

I, _____, hereby **deny permission** to the RIT Library of the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part.

Signature of Author: _____ Date: _____

Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive

I, Mathew S. McEuen, additionally grant to the Rochester Institute of Technology Digital Media Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: Matthew S. McEuen Date: 10/7/05

Abstract

A recent computer vision technique for object classification in still images is the biologically-inspired Expert Object Recognition (EOR). This thesis adapts and extends the EOR approach for use with segmented video data. Properties of this data, such as segmentation masks and the visibility of an object over multiple frames, are exploited to decrease human supervision and increase accuracy. Several types of runtime learning are facilitated: class-level learning in which object types that are not included in the training set are given artificial classes; viewpoint-level learning in which novel views of training objects are associated with existing classes; and instance-level learning of images that are somewhat similar to training images. The architecture of EOR, consisting of feature extraction, clustering, and cluster-specific principal component analysis, is retained. However, the K-means clustering algorithm used in EOR is replaced in this system by an augmented version of Fuzzy K-means. This algorithm is incrementally run over the lifetime of the system, and automatically determines an appropriate number of partitions based on the data in memory and on a system parameter. In addition, the edge and line-based feature extraction of EOR is replaced with a global application of the principal component analysis, which increases accuracy when used with segmented video data. Classification output for the system consists of a multi-class hypothesis for each tracked object, from which a single-class "hard" hypothesis may be determined. The system, named VEOR (video expert object recognition), is designed for and tested with noisy, automatically segmented real-world data, consisting of both videos and still images of vehicle (car, pickup truck, and van) profiles.

Acknowledgements

This thesis would not have been possible without the vast amounts of guidance, insight, and support that were freely offered by my chairperson, Dr. Roger Gaborski. I would also like to thank the other members of my committee, Dr. Carl Reynolds and Dr. Edith Hemaspaandra, for their feedback and support. I am grateful to Dr. Bruce Draper and Dr. Kyungim Baek for their helpful correspondence and assistance.

Soli Deo Gloria

Contents

List of Figures

Chapter 1

Introduction

Object recognition is a difficult and important problem in computer vision, and is integral to the understanding of video events by a computer. An encouraging fact, however, is that humans can perform this task with great speed, accuracy, and ease. The biologically inspired system proposed by Bruce Draper, Kyungim Baek, and Jeff Boody, called expert object recognition (EOR) [1, 2, 3], yields results that surpass those of the system's constituent algorithms. However, their approach is engineered for use with still images, and while potentially of great use to video analysis, has limitations which prevent it from being applied to that domain. An EOR-based approach called VEOR (video expert object recognition), which attempts to address these limitations, is proposed in this thesis. VEOR is specialized for the needs of video understanding and tuned for use with segmented video data. In addition, this thesis seeks to determine how well this approach ap-

plies to real world data, and is tested with a database of segmented cars, trucks, and vans.

The outline of this thesis is as follows. Chapter 2 gives background information on the EOR system and its incorporated algorithms. Chapter 3 presents problems that prevent the existing EOR implementation from being applied to video data, and describes some features that would be desirable in an object recognition system for video. The architecture of VEOR and the details of its subsystems and algorithms are found in chapter 4. Chapter 5 describes experiments with different data sets and their results. Chapter 6 presents conclusions.

Because continued research in this area is planned, implementation-specific notes that describe functions, their arguments and return values, system parameters, etc., are included. These notes are by no means an exhaustive guide to the VEOR implementation; they are merely meant as a starting point, so that code and comments may be put into context. The implementation consists of MATLAB code, and all system parameters are settable using the *parameters* function.

Chapter 2

The Expert Object Recognition approach

Expert object recognition is a specific kind of human object recognition that is concerned with fast recognition and discernment of familiar types of objects. Objects are recognized on both instance and category levels. For example, while a familiar face may be associated with a specific individual, an unfamiliar face would still be recognized as human.

In a series of two papers [3, 1] and one doctoral dissertation [2], Draper, Baek, and Boody proposed a biologically inspired approach to object recognition based on the human expert object recognition pathway, dubbed the expert object recognition (EOR) system. EOR has been actively developed, and the algorithmic choices and parameterization differ somewhat from paper to paper. In the most recent publication, Draper et al. find the results

of the system to be quite good, outperforming all of the constituent algorithms used [1]. Further research is being done in all of the modules of EOR, particularly in the areas of feature extraction and exemplar matching.

The expert object recognition pathway in the human brain, modeled by the EOR system, consists of three main stages which sequentially extract features from, categorize, and match viewed images to visual memories.

The EOR life cycle is divided into training and runtime phases, the latter of which can be used as a testing phase. The same general stages are found in both the training and testing phases: feature extraction, which computes feature vectors for image data; categorization, which partitions images into groups based on the similarity of their feature vectors; and exemplar matching, which is responsible for determining the best match for a runtime image from within its assigned partition. The flow of data among these stages can be seen at a high level in figure 2.1. The stages are described in detail in the following sections.

An implementation of EOR was developed as a starting point for VEOR, and so that the two approaches may be quantitatively compared. In this implementation, feature extraction is applied to each training image serially, generating a set of feature vectors. The categorization stage is applied to the feature vectors in a single step, and calculations necessary to allow exemplar matching are performed sequentially on each generated partition of training data. During testing, images are fully processed sequentially. Where the implementation described here differs from those found in the literature,

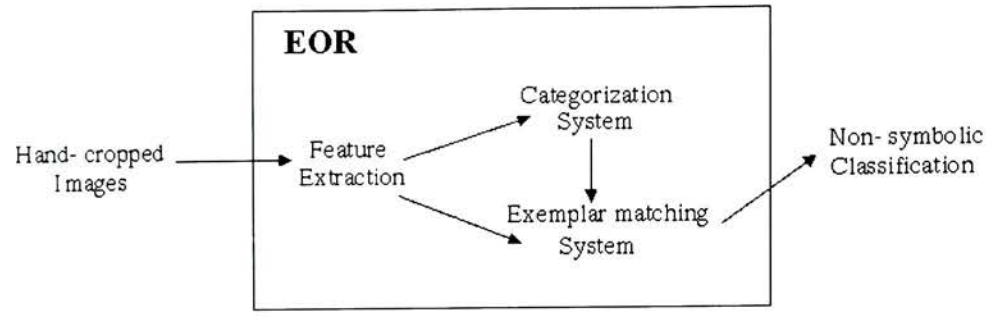

Figure 2.1: EOR system architecture.

special note will be made.

2.1 Edge Detection

The V1 area of the brain consists primarily of edge detecting simple cells, and of complex cells which sum the simple cell outputs [2]. Gabor functions, the products of Gaussian and sinusoidal functions, are the accepted model of simple cell receptive fields. Two types of Gabor filters, even and odd, are particularly sensitive to bars and to edges, respectively. The sinusoid used in the creation of even filters is a cosine function, while that used for odd filters is a sine function. Basic even and odd filters are generated in this implementation with a script from the VENUS [5] system. Those filters are then rotated to 0, 45, 90, and 135 degrees to detect edges and bars of different orientations; these are rescaled multiple times, producing filters

sensitive to edges and bars of various sizes. However, the performance of the implementation as a whole was slightly better when only the smallest filters (seven by seven pixels) were used, than when the three sizes (7, 15, and 31 square pixels [2]) found in the literature were used. Using only the smallest filter size also decreased processing time significantly. Recent correspondence with Dr. Draper confirms that the working version of the EOR system uses only the smallest filter size as well.

Complex V1 cells are modeled by summing the squared output from simple cells. This serves to create a rectified energy map denoting edge strength across the filtered image [2]. Some difference exists among the different EOR proposals as to whether the rectified energies of the individual orientations are to be summed into an average [1], or kept separate as input for the next stage of processing [2]. The approach used in this implementation was to sum the outputs and normalize the result to the range [0,1].

Two small algorithmic additions that increased performance a small amount were to smooth images using an averaging filter prior to Gabor filtering, and to normalize their values to [-1,1]. The latter step serves to remedy a shortcoming of applying Gabor filters to image data rather than to a contrast map (as in [5]). This problem is evidenced by the fact that a white object on a black background will generate a very different edge response than a black object on a white background. Normalizing the input to [-1,1], combined with the rectification step, produces identical edge responses for both. In fact, the results were qualitatively very similar to those achieved by apply-

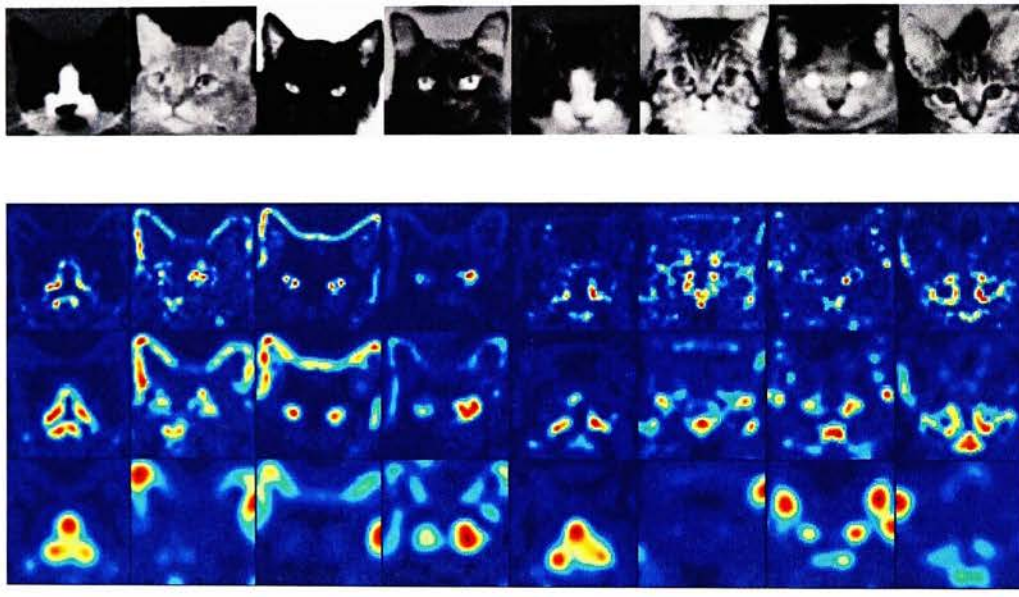

Figure 2.2: Gabor energies of cat faces. The top row is the unfiltered input. The second, third, and fourth rows are the faces filtered with even and odd Gabor filters of sizes 7, 15, and 31, respectively, in false color. The input images were first scaled to the range [-1,1] and smoothed.

ing Gabor filters to a contrast map. A contrast map is generated using a filter with a shape similar to that of a Gabor filter, such as a difference of Gaussians filter [5]. This similarity of results, however, calls into question the necessity of the extra filtering. The effects of Gabor filtering at different resolutions, after smoothing and normalizing, may be seen in figure 2.2.

In this implementation, the sizes of filters, and the number of filter sizes, are determined by the system parameter *gabor_filterSizes* (a vector of integers); the number of orientations by *gabor_orientCt*; and the number of phases (1 for odd filters, 2 for both even and odd filters) by *gabor_phaseCt*.

2.2 Line Detection

In the human brain, further feature extraction is performed by the lateral occipital cortex (LOC), the cells of which respond to structural, or edge-based, properties. Specifically, non-accidental properties, such as collinearity, parallelism, symmetry, and antisymmetry are detected. The EOR system performs a subset of the functions of the LOC by using the Hough transform to discover collinearity [1]. This algorithm finds, for every non-zero pixel in an image, along what lines that pixel may lie. Response values are collected in bins in "Hough space", a discrete two-dimensional space whose axes are the orientation and perpendicular distance from the origin of possible lines. Thus, bins represent individual lines across the input image, and the value of the bins represents the cumulative evidence for the existence of those lines. In

standard image processing, Hough space responses are thresholded to detect lines. However, the goal of this stage of the EOR system is to produce a signature that identifies an image by its non-accidental features, so all of the response data are kept.

Unfortunately, all attempts to gain performance in this implementation using the Hough transform were futile. Under any parameterization with Hough, performance was very significantly lower than without. This may be because of some incorrect assumption that is not specifically detailed in the literature, because, in the literature, Hough added greatly to performance [1]. Thus, under the default parameterization, this implementation follows the earliest specification of EOR [3], in which Hough was not used. The system flag *useHough* determines whether or not the transform is included in feature extraction. The implementation of the Hough transform used here is from [19].

2.3 Categorization

While the next two stages in the biological expert object recognition pathway are not as well understood as those previously discussed, there is evidence that the fusiform gyrus and right inferior frontal gyrus perform categorization and exemplar matching, respectively [3]. Categorization in the EOR system serves to partition training images into groups according to the similarity of their feature vectors. The average values for these clusters, known as cluster

centers, centroids, or prototypes, are calculated as part of categorization. Runtime images are then assigned to clusters whose centroids are closest to their feature vectors. Exemplar matching for an image is performed using the subset of training data associated with its cluster.

Categorization in EOR is modeled by the K-means clustering algorithm. K-means is a simple iterative algorithm that can quickly categorize multi-dimensional input into clusters, while simultaneously calculating the mean value for each cluster [7]. Clustering does not improve the performance of the system much in terms of accuracy [1]; however, it allows visual memories to be stored at a high level of compression, as described in the next section. The number of clusters is determined before runtime by the parameter k.

K-means belongs to a category of clustering algorithms that seek to minimize a distance (or error) metric between data and their assigned cluster centroids. The objective function that the algorithm seeks to minimize is

$$J_{KM}(X, C) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \|x_i^j - c_j\|^2 \tag{2.1}$$

X is a vector of data to be partitioned; each datum $x \in X$ is itself a feature vector. $x_i^j$is read "the i^{th} datum in the j^{th} cluster", and n_j is the number of data in cluster j. C is a matrix of cluster centroids, and each centroid $c \in C$ has the same dimensionality (length) as the vectors in X. $\| * \|^2$ denotes a particular distance metric [7]. Squared Euclidean distance is the most common distance function, and is used here. Other error measures include

Algorithm 1 The *K-means* algorithm [8]

1. Initialize k (a parameter) cluster centroids in C to the values of k random data in X.
2. Assign membership of each datum to the cluster centroid to which it is closest.
3. Recompute the values of the centroids to be the mean values of their member data.
4. If a convergence criterion is not met, go to step 2.

"city block" and Mahalanobis distances. K-means is implemented using the technique of alternating optimization, in which optimal values for C and cluster membership are calculated in turn until some convergence criterion (or criteria) is met [8]. Common convergence strategies are to stop when no change in cluster membership occurs, or when total change of centroid locations is less than a threshold. The steps of K-means are as follows.

Some typical clusters generated when K-means is applied (following Gabor-based feature extraction) to the cat and dog database (see section 5.1) are shown in figure 2.3.

Draper and his colleagues found that performance was increased by over-estimating the value for k, possibly because the actual classes in the data do not reflect a Gaussian distribution [1]. The results of this implementation support this.

Although a new implementation of K-means was created for this implementation, its performance was not as consistent as that of the K-means

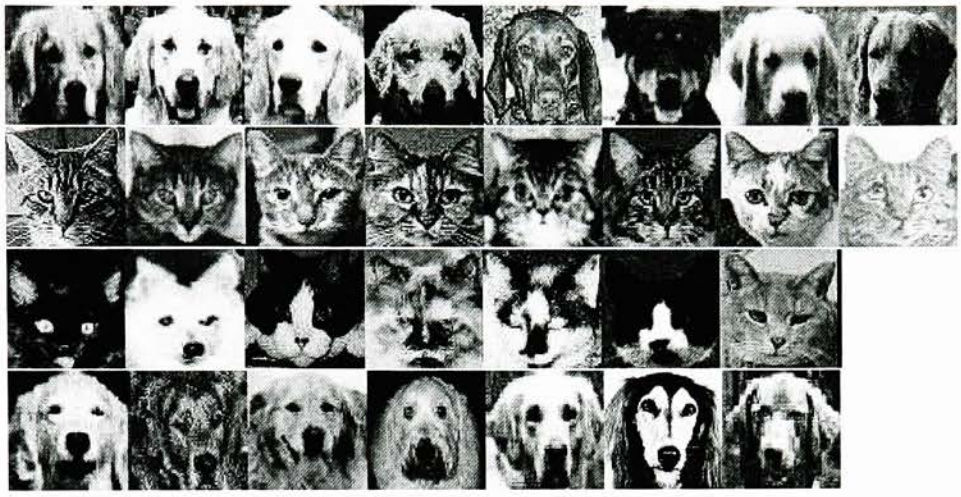

Figure 2.3: Clustered cat and dog faces. Each row represents a different cluster.

implementation found in the MATLAB statistical toolbox. The reason for this is unknown. The differing results may be due to the fact that the performance of K-means varies greatly with the choice of initial cluster centers. However, use of the statistical toolbox version did not improve exemplar matching performance of the system significantly when compared to the new K-means implementation. The statistical toolbox implementation may be enabled by setting the system flag *clustering_matlab_hkm*, and the new implementation by *clustering_mm_hkm*.

2.4 Exemplar matching

Once an image has been assigned to a cluster at runtime, exemplar matching finds its best match from within the contents of that cluster. Psychologists

have hypothesized that visual memories are retained in the brain as "compressed images", and some experimental data support this idea [1]. Compression in the EOR system is facilitated by the use of a subspace projection algorithm, the principal component analysis (PCA). PCA is a statistical technique that exploits similarities in data. It has proved useful in the field of computer vision, and has been used for years as a face matching algorithm [1].

PCA finds orthogonal vectors along which the maximum variance in a dataset lies, and captures that variance in new variables by making the vectors axes in a subspace. In this way, most of the variance of a dataset can be represented by a relatively small number of dimensions. Details of PCA are found in the following section. By forcing similar images into clusters, and generating a different subspace for each cluster, the EOR system allows PCA to very effectively compress a dataset consisting of dissimilar objects [2]. Another advantage to clustering and subspace matching is that, instead of finding the nearest match to an image from the entire training dataset in high-dimensional image space, EOR finds the nearest match to the image in a low-dimensional subspace from a clustered subset of the data.

In [3], PCA is applied to image data, while in [1], it is applied to the same feature vectors upon which K-means operates. Both strategies were tested in this implementation, with the former achieving better performance. The *pcaOnRawData* system flag determines which strategy is used.

The Principal Component Analysis

As mentioned previously, PCA projects data into an artificial subspace, and can function as a variable reduction procedure. PCA differs from subspace projection algorithms such as the Fourier transform in that the basis vectors for the subspace are derived from the data to be transformed. The projection matrix consists of basis vectors (subspace axes) that are orthonormal to one another [16]. Thus, the vectors are mutually uncorrelated [17].

When PCA is performed on a dataset, the mean value is subtracted from the data, and a covariance matrix is calculated from the result [18]. Information in data is usually encoded as variance, and the ability to capture most of that variance in a small number of variables is how PCA accomplishes variable reduction. PCA works best when there exists a high level of redundancy (covariance) among input variables [17]. When performed on raw image data, the redundancy exploited by PCA is that of pixels whose values vary together. A simplified example: if a particular region of pixels share the same graylevel value within each image in a dataset, but that value varies among the different images, then a single subspace variable may represent the intensity of all of those pixels. Likewise, a subspace variable may encode the presence and degree of a feature common to some, but not all, of the dataset. However, because PCA applied in this fashion is pixel-based, a spatial translation of a particular feature could not be so cleanly represented.

When calculating the projection matrix, a basis vector is found that accounts for the maximum amount of variance in the data. This is the first

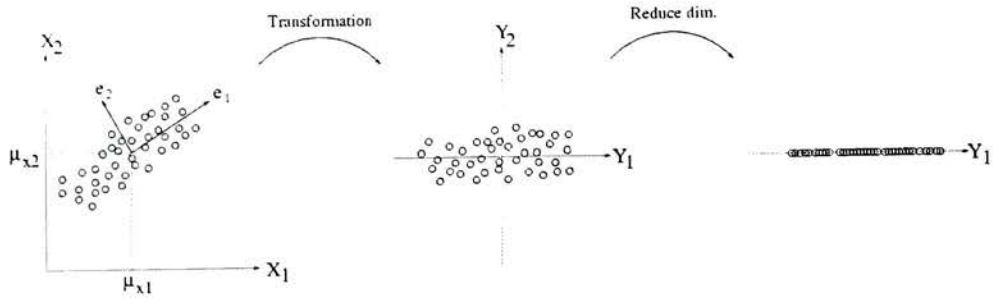

Figure 2.4: Two-dimensional illustration of the major steps of PCA, courtesy of [16].

principal component. Another vector is found, orthogonal to the first, which captures the maximum remaining variance in the data, and so on; each subsequent component accounts for the maximal amount of "residual" variance not captured by the previous components. The method by which these basis vectors are found is to calculate the eigenvectors and eigenvalues of the covariance matrix of the data [16]. The order of principal components is the same as the descending order of their corresponding eigenvalues; that is, the eigenvector with the largest eigenvalue is the first principal component [18].

As many principal components may be found for a dataset as exist variables in its input vectors. Each principal component represents an axis in the PCA subspace; therefore, when data are transformed by their "full" PCA projection matrix, the dimensionality (and size) of the output is identical to that of the input. However, because most of the information in the data is captured in the initial principal components, the latter components may be ignored, and only a small amount of information is lost. There are many methods that may be used to determine exactly how many and which com-

ponents to retain, such as the eigenvalue-one criterion, the scree test, and setting a threshold on the proportion of variance for which retained components must account [17]. These methods make their determination based on the dataset at hand. However, EOR takes a simpler approach, and retains a number of eigenvectors specified by a system parameter. This value was a variable in the experiments of Draper et al. [1].

Once the transformation matrix for a dataset is calculated, data may be projected through it [18]. The size of the projected data is proportional to the number of retained basis vectors. In addition, a reconstruction of the original dataset may be produced by projecting subspace data through the transpose of the transformation matrix. If as many principal components exist as variables in the input, then no information is lost, and a perfect reconstruction may be created. However, only an approximation can be recovered if some of the eigenvectors are ignored [18].

2.5 Summary

Now that the building blocks of EOR have been described, the end to end procedure will be shown for the sake of comprehension. The steps followed by EOR in training and in testing are as follows.

Algorithm 2 EOR, training phase

1. Extract features from training images using Gabor filters.
2. Cluster the feature vectors into k partitions using K-means.
3. For each cluster, create a transformation matrix by performing PCA on the images associated with that cluster. Project each datum into its cluster's subspace.

Algorithm 3 EOR, testing phase

1. Extract features from the testing image using Gabor filters.
2. Find the cluster centroid (as computed in training) closest to the feature vector.
3. Project the image into the PCA subspace for that cluster, and find its closest match from the projected cluster members. This is the exemplar match for the testing image.

Chapter 3

Considerations for video

Gaborski et al. from the RIT department of computer science have developed a novel video event recognition system, called VENUS [5], which is capable of detecting novelty in a video scene based on habituation to color and motion. However, the long-term goal for the system is to be able to describe what happens in a scene, and object recognition plays a key role in that regard. The prospect of applying the EOR approach in this way is appealing. EOR may be trained on any number of arbitrary classes; that is, it is a generic approach that can be specialized for various applications by choosing different training sets. In the area of video understanding, training data could be selected based on what objects are expected to appear in a scene, and also on those objects whose presence should be cause for alarm. In addition, EOR lends itself to use with video data. The attention window, a part of visual pathway lacking implementation in EOR, may be simulated through back-

ground detection-based object segmentation (see section 4.2). Also, while EOR associates images with previously seen images, an associative memory that would label those images (along with other reasoning-related tasks) is deemed to be outside the scope of EOR [2]. Plans to introduce a robust associative memory into VENUS could fill this role and achieve more reliable classification results.

However, significant issues stand in the way of providing an EOR implementation suitable for use in a video system, especially with regards to runtime learning. Learning would be a highly advantageous feature for a video-based system, especially one engineered for the detection of novelty. In a realistic setting, an object may look differently from one frame to the next, because of rotation, shadow, partial occlusion, or noise. Knowledge, gained through learning, that a class may appear in views different from those trained upon would aid further identification of instances of that class. Objects whose classes were not included in the training set may also be encountered, and should be classified as unknown objects. However, if two such objects appear in a video sequence, an intelligent response would be to classify the objects as the same type, even though the system does know a symbolic label for that type.

The current EOR system makes a strong distinction between training and testing phases, which precludes online learning. This distinction is complicated by the fact that two of the algorithms, K-means and PCA, would need to be applied to the entire dataset every time that an object is added to

memory. This high amount of processing may be redundant and unnecessary. Likewise, the number and selection of retained images is an important issue, as this determines how often learning must occur and in what way visual memory is increased. The K-means clustering algorithm is too inflexible of an approach to partitioning for this environment, because it requires a specification of k, the number of clusters. In a realistic setting, an appropriate number of clusters may not be determinable before runtime and may need to grow as a video sequence progresses and as images are added to memory.

Chapter 4

The VEOR system

The VEOR (Video Expert Object Recognition) system seeks to apply the EOR approach to video input, and addresses the concerns found in chapter 3 through algorithmic additions, substitutions, and modifications. Its responsibilities are a superset of those of EOR, because it performs object tracking and symbolic labeling in addition to visual matching. Runtime visual learning is facilitated in multiple ways. Instances of known classes, from both known (trained) and unknown (novel) perspectives, can be introduced into memory. Artificial classes are created for unclassifiable input. The time required to add an image to memory is kept to a minimum by reusing existing cluster information when repartitioning. This keeps cluster membership fairly stable, in addition to reducing clustering time. Thus, local PCA needs to be applied only to clusters whose membership has changed, and then only when an exemplar match into that cluster is required.

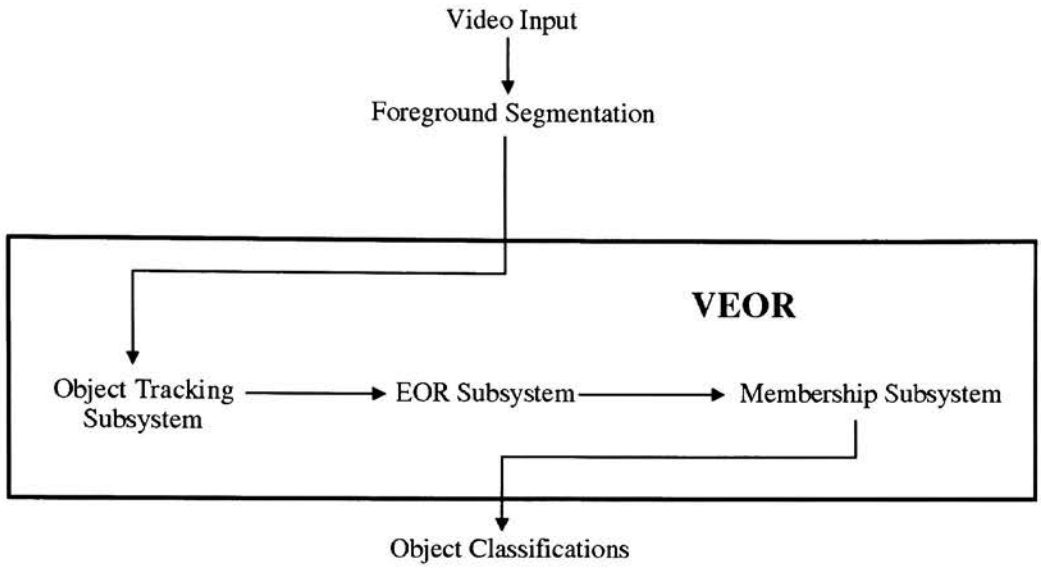

Figure 4.1: Architecture of the VEOR system. Arrows indicate the flow of information.

4.1 VEOR System Architecture

VEOR is divided into logical subsystems. Functionality modeled after EOR is found in the EOR subsystem. A separate Membership subsystem calculates class membership hypotheses for objects, while the Object Tracking subsystem is responsible for following objects through successive frames. Segmentation of video data is a necessary pre-processing step for VEOR input. The relationships between these modules are shown in figure 4.1.

The EOR subsystem in VEOR consists of the same stages as EOR: feature extraction, clustering, and exemplar matching, but substitutions and modifications have been made. The edge- and line-based feature extraction of EOR

has been replaced with a global application of PCA, and the K-means clustering algorithm has been replaced with a learning-enabled algorithm based on Fuzzy K-means [8]. Exemplar matching in VEOR happens in much the same way as in EOR.

The distinction between training and runtime (testing) phases has been carried over from EOR; however, this division may be somewhat misleading, because learning continues to occur during the runtime phase. Training data are extracted from segmented video footage, and consist of four parts: a video frame containing the training object, a corresponding segmentation frame that masks the object, the class of the object, and a coordinate pair specifying a point which lies within the object's segmentation mask. Because the training data are taken from real-world video footage, which is segmented based solely on motion, a segmentation mask used in training may find multiple objects within its corresponding video frame. This is why the coordinate pair is necessary: it specifies which masked region corresponds to the training object. Runtime input consists of a video and a corresponding segmentation video, each frame of which is a segmentation mask. Classes for runtime objects are unknown, and coordinate pairs are unnecessary because every segmented object (that meets certain constraints) is classified. Multiple objects within a single frame are classified simultaneously.

The major steps of the VEOR system during training and testing are described by algorithms 4 and 5.

In many applications, a particular viewpoint of an object looks very simi-

Figure 4.2: Example of VEOR training sample. The first two pieces of data associated with this sample are a video frame and its corresponding mask, shown above. The third and fourth are the class name "car" and the (row,column) coordinate (87,79), which specifies the location of the object of interest. The upper-left corner of the images are considered to be the coordinate origins, (1,1).

Algorithm 4 VEOR, training phase

1. Extract segmented image patches from training data.
2. Assign membership of all training images to their respective classes.
3. Extract coarse features from training images by performing PCA and projecting the data into the global PCA subspace. This generates feature vectors.
4. Initialize a small number of cluster centroids to the locations of random feature vectors.
5. Perform Fuzzy K-means. If any cluster does not meet certain "goodness" criteria, add a new centroid that is the mean of a random subset of the cluster, and repeat step 5.

Algorithm 5 VEOR, testing phase

For each tracked object in each frame:

1. Extract a segmented image patch from the frame.
2. Extract coarse features from the image patch by projecting it into the global PCA space.
3. Find the cluster centroid closest to the generated feature vector.
4. If the local PCA transformation matrix for that cluster is not up to date, compute it by performing PCA on the cluster's corresponding image data.
5. Project the image patch into the local PCA subspace for its cluster, and find the closest match from the projected cluster members.
6. If the distance (in global PCA space) from the testing image to its exemplar match is below a threshold, contribute the membership of the exemplar match to the membership of the test image, and skip steps 7-10.
7. Otherwise, introduce the testing image into the training set and perform steps 2-4 of the training sequence, with the exception that the initial cluster centroids in step 3 are set to the current cluster centroids.
8. Find the exemplar match for the testing image from within its new cluster by performing steps 4-5 of the testing sequence, while disallowing the exemplar match from being the testing image itself.
9. If the image is partitioned into a cluster by itself, then a new class is created for it, and membership is assigned to that class.
10. Otherwise, contribute the membership of the exemplar match to the membership of the test image.

lar to the mirror image (horizontal reflection) of another. For example, a car facing to the right probably looks like the mirror image of the same car facing to the left. Adding mirror images of the entire training database into memory would be redundant, so a different method is used. If mirroring is enabled, both an object's image and its reflection are exemplar matched (see section 4.6), and whichever match is "better", or closer, is used for classification (or is learned — see section 4.8.1).

Implementation notes

All persistent data are stored in a "VEOR data structure", which is a simple MATLAB structure. Examples of such data are cluster centroids, PCA transformation matrices, and membership information. This is done so that multiple "instances" of VEOR, trained separately and with different states, may be run side-by-side in an application. Several VEOR functions take one of the data structures as an argument.

The parameter flag *EOR_mirrorImages* determines whether image mirroring is used at runtime.

4.2 Foreground segmentation

Before an object can be classified, it must first be located; that is, pixels corresponding to objects must be distinguished from those corresponding to uninteresting background. In VEOR, this is done using Stauffer and Grimson's

mixture of Gaussians background subtraction technique [6], implemented by Jason Roberts. The input for this stage of processing is simple video data, and the output is a mask video. Each frame in the output video corresponds to a frame in the input, and the presence of an object in the input frame is noted by a connected group of white pixels in its mask. All other pixels in the mask video are black. Objects in a scene may then be separated from background by performing a logical "AND" operation between a video frame and its mask.

Segmentation is achieved through detection and modeling of background. When an object moves through a scene, it does not fit into the background model and is thus detected as foreground. The particular method described in [6] is noteworthy in that it adapts to a scene, is robust to lighting changes and multimodal backgrounds, and can assimilate new objects into the background if they become stationary for a period of time. Pixel color values are clustered into any number of Gaussian distributions (a parameter), and these distributions are ranked according to their likelihood of belonging to background. The heuristics used for this ranking are based on a Gaussian's variance (a moving object tends to cause more variation in a given pixel than does background) and persistence (the colors associated with a moving object tend to appear for a short amount of time, to be replaced by background). Once ranked, a certain number of Gaussians are accepted as the current background model, based on a threshold that defines the minimum number of recent data that should be considered background. Pixel values that fall

within those distributions (by default, within 2.5 standard deviations) are exclusively labeled as background.

The default number of Gaussians, three, allows for the segmentation of foreground from a bimodal background (the third Gaussian is used for foreground) under noisy conditions. When segmentation was performed for the testing data described in sections 5.2 and 5.3, however, only two Gaussians were used, because the short lengths of the involved video clips rendered multimodal background detection unneccesary. Multimodal background detection should be used when providing VEOR with longer input videos.

After detecting background, morphological processing is performed to remove from masks small patches of white pixels that represent noise or inconsequentially small moving objects. In addition, the morphological processing serves to smooth the outlines of objects, fill holes in masks, and connect foreground components that are close to one another. Components in such close proximity often belong to the same object. The results of foreground detection are shown in figure 4.2.

4.3 Object tracking

After foreground pixels have been identified, they may be interpreted as belonging to objects. Regions of connected foreground pixels are found, and region properties (centroid, area, and bounding box) are calculated. Regions that meet acceptance criteria are then tracked from one frame to the next.

In the current VEOR implementation, regions must meet two criteria to be tracked. First, any region that touches an edge of a frame is ignored. This is because EOR is an appearance-based technique, and the similarity that it would find between an object and a portion of that object decreases as less of the object is visible. For example, it is unlikely that an image of a car would be the exemplar match if one half of the same car image was used as input. Thus, an object is not tracked until it is fully within the field of view, and is no longer tracked when it begins to leave. Unfortunately, this same situation may occur if an object becomes occluded while fully within the frame, by background or another object, for instance, and this is a limitation of the current implementation. The second region acceptance criterion is that a tracked region must occupy at least a minimum number of pixels. One reason for this is that, although the majority of segmentation noise is removed through morphological processing, some is not. Any foreground region of substantial size cannot be considered noise, so only objects are tracked. A second reason is to eliminate the problem that far-away (low resolution) objects are difficult to distinguish from one another, and are therefore difficult to classify.

All regions meeting the acceptance criteria are tracked among frames as objects. The actual tracking mechanism currently used in VEOR is very limited, and is a stand-in for some more state-of-the-art object tracking algorithm. Implementation of a robust tracking scheme is beyond the scope of this thesis, and work on such a system is currently underway by individuals

in the RIT Department of Computer Science. The object tracking subsystem in VEOR considers foreground regions in two consecutive frames to be the same object if there is any overlap in their bounding boxes. Tracking behavior is undefined if there is a greater than one-to-one correspondence between such regions. Resulting limitations include the inability to track objects that disappear and reappear, objects that overlap one another, and objects that move fast enough relative to the frame rate of video input such that there is no overlap in regions between consecutive frames. In addition, if an object occupies more than one region, each constituent region will be tracked separately. For example, if a car drives behind a street sign, it will be tracked as two separate objects while visually divided. However, this form of object tracking serves as a proof of concept, and works well with real-world input that avoids these admittedly realistic situations.

Implementation notes

The object tracking subsystem is split into two separate functions: *findObjects*, which identifies foreground regions that meet the acceptance criteria, and *trackObjects*, which tracks objects among frames. *FindObjects* takes only a mask frame as a parameter, and returns all found objects as a MATLAB struct array. The *findObjects_minSize* system parameter sets the minimum number of pixels of which a region must consist for it to be considered an object; if unspecified, the parameter defaults to 40. *TrackObjects* takes two struct arrays, of the type returned by *findObjects*, as input: one for the

current frame and one for the previous frame. The function uses these to determine which of the found regions are new objects and which are objects that have persisted from the previous frame, and relays this information with three return arrays: one for objects that first appear in the new frame, one for objects that have been tracked from the previous frame, and one for objects that were in the previous frame but have disappeared. The structures used as input and output for these functions are identical to those returned by the *regionprops* MATLAB function, with the exception that they are assigned an additional *id* field which specifies a unique numerical identifier for each tracked object.

4.4 Feature extraction

Feature extraction in the EOR system consists of edge and line detection, implemented as Gabor filtering and the Hough transform, respectively. Although these algorithms are incorporated into VEOR, a different feature extraction pathway is used by default. Gabor filtering and the Hough transform may be enabled by setting system flags. In addition, processing is required to transform input data into masked image patches. This processing is not necessary in EOR, because image patches are provided as input.

Initial processing

Each tracked object within a frame of video must undergo several transformations before features may be extracted from it. Data provided to this stage of processing consist of the video frame and mask frame in which an object appears, as well as the coordinates of a point that lies within the object. At runtime, these coordinates are retrieved from the region properties found by the object tracking subsystem. During training, they are provided as input data, in part because it is impossible to track an object across a single frame.

An object in a video occupies occupies a region whose size in pixels depends on the resolution of the video and on the distance of the object from the camera. To compare pixel-based features among objects from different frames, however, these regions must be rescaled to a common image patch size, which in VEOR is square. Using the coordinate pair as a locator for the appropriate region, the center and bounding box of the object are calculated from the mask frame. All other foreground regions of the mask are zeroed out to ensure that masks corresponding to other objects do not affect the masking of the object at hand. Next, the major axis of the object's region, vertical or horizontal, is determined. The reason for this is that some objects (such as pedestrians) are taller than they are wide, and other objects (such as vehicles) are wider than tall. To most effectively scale the object, tall objects should occupy the full height of the final image patch, while wide objects should occupy the full width. The minor axis of the objects should be scaled by the same factor as the major axis. An alternate approach would

be to scale the axes separately, allowing all objects to occupy both the full width and height of the final image patch; however, in doing this, important implicit information regarding the aspect ratio of the object would be lost. If an object lies near the border of the frame in such a way that the patch to be extracted extends beyond the frame's borders, then the frame is padded symmetrically. Resizing, using bilinear interpolation, is then performed on the relevant part of the video frame. Nearest-neighbor interpolation is used when resizing the mask frame, because bilinear interpolation could cause inappropriate gray pixels to appear between the black and white regions in the mask.

If edge detection is enabled, then convolution with Gabor filters is applied. When resizing is performed, extra pixels in all directions are also resized to provide padding for the convolution, so that edge effects are avoided. The object itself is extracted from the padding after convolution occurs, if it occurs.

Finally, masking occurs through a logical "OR" operation between the image and mask patches. If enabled, the Hough transform is applied to the resulting image patch. The resulting data, be they grayscale image, edge map, or hough space, are vectorized so that each pixel (or hough bin) is represented by a variable in a feature vector.

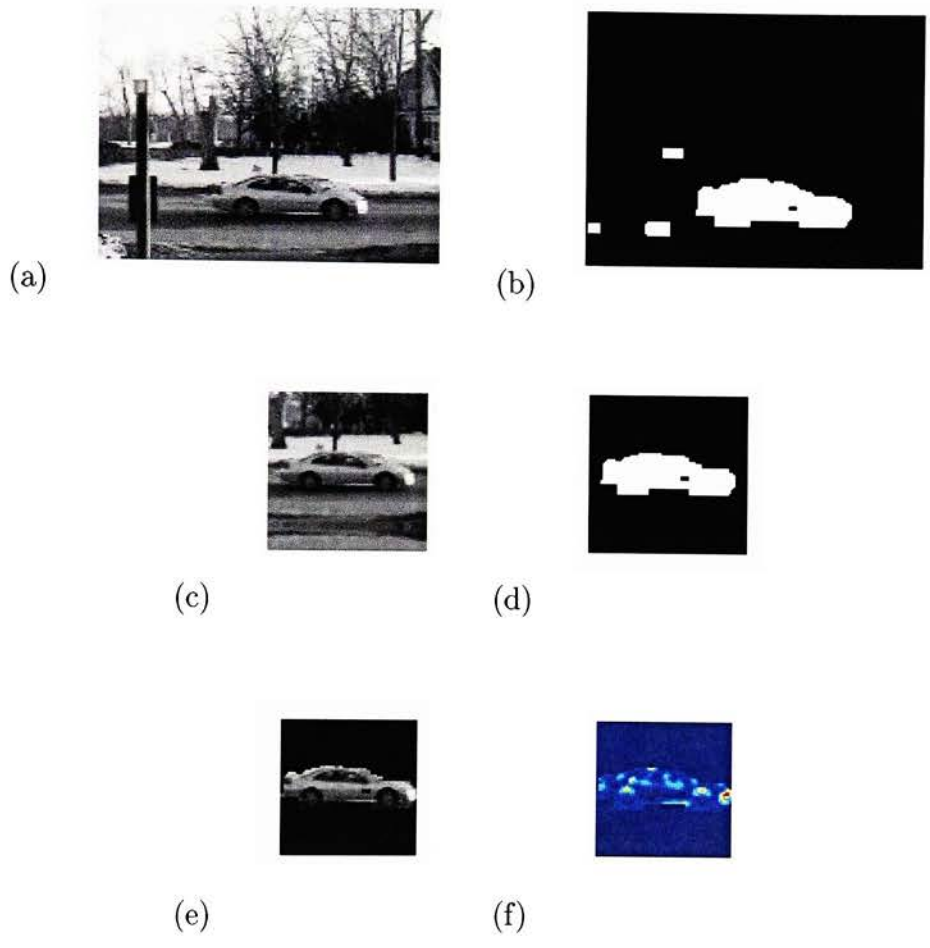

Figure 4.3: Stages of initial input processing on a sample. (a) and (b) are the sample's input video and mask frames, respectively. (c) and (d) are the extracted and resized video and mask patches. (e) is the masked object if Gabor filtering is not used, while (f), shown in false color, is the masked object if a 7×7 filter is applied.

Principal Component Analysis

PCA has long been used as a feature extraction tool in face recognition, in which the variations among many different faces may be well-represented within a PCA subspace [2]. In the EOR system, PCA is used in the exemplar matching phase to discriminate among what the clustering algorithm determines to be similar images. The creators of EOR note that PCA works best when applied to images drawn from a normal distribution [3], and while images of a certain type of object (such as the human face) may draw from a normal distribution, a dataset consisting of different objects does not. Thus, clustering allows local subspaces to be generated for the multiple normal distributions that are mixed in such a dataset. In [2], PCA is tested as single (global) classification step, and is found to be inferior to EOR when applied to the cat and dog database (see section 5.1). However, Draper et al. show that when EOR feature extraction is applied to aerial images of Fort Hood before PCA is applied, performance is as good as the entire EOR system, given a high enough number of retained dimensions [1]. Experiments with VEOR using segmented vehicle data, however, show that performance is significantly worse when clustering is left out, no matter how many subspaces are retained.

In the default VEOR configuration, a combination of these two approaches to PCA is used, and the algorithm is applied at both a global and local level. The global application of PCA functions as coarse feature extraction, results from which are used for clustering. The cluster-specific application performs

the same function as it does in EOR, which can be considered fine feature extraction. Thus PCA extracts the largest features that it can from a dataset: inter-class features from a multi-class mixture, and inter-instance features from a cluster. PCA is commonly used to reduce the dimensionality of data before clustering is performed. Ding and He [15] show that PCA itself performs data clustering according to K-means' objective function, improving the performance of clustering when the two techniques are used in sequence. Experiments with VEOR also show that the increase in execution time caused by the global PCA application is more than made up for by the decrease in clustering time, and overall execution time is significantly decreased.

Replacing edge-based feature extraction with PCA is not without caveats, however. As noted in [3], an advantage to using both Gabor filtering and PCA is that different properties of the data are exploited by the two techniques: boundary and intensity information, respectively. The negative effects of applying PCA to grayscale images at all levels is evidenced by the clusters shown in figure 4.4. Dark cars tend to be clustered together, as do grayish cars and whitish cars, whereas under edge-based feature extraction all such cars would tend to be clustered together. In fact, when using the cat and dog database (see section 5.1) as input, VEOR performs worse with PCA feature extraction than with Gabor filtering. Despite these issues, PCA-based feature extraction yields greater performance in terms of accuracy and speed when segmented vehicle data are used as input.

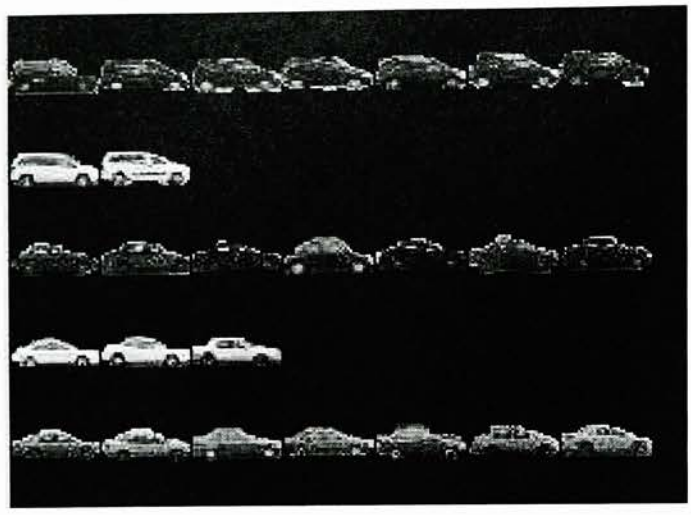

Figure 4.4: Some cars and vans clustered using global PCA for feature extraction. Rows represent clusters. Note that vehicles tend to be clustered both by their class and by their grayscale color.

Implementation notes

Image scaling and masking, as well as optional Gabor filtering and the Hough transform, is handled by the MATLAB function *EORPreProcessing*. The arguments to this function are a video frame, a mask frame, and a two-element location vector. An argument providing Gabor filters may be used if filtering is enabled, and is ignored otherwise. Two values are returned: an image patch of the masked object, and a feature vector. If neither Hough nor Gabor filtering are enabled, then that vector is simply the vectorized form of the returned masked object. The parameters *imgSizeC* and *imgSizeR* specify the size to which image patches should be rescaled, and the flags *useGabor* and *useHough* specify whether those particular technologies are enabled. Gabor

filtering is implemented in the functions *genGabor* and *applyGabor*, and the Hough transform in *hough*.

The global application of PCA is in the *EORClassify* function, which is also responsible for calling categorization and exemplar matching routines. The parameter *EOR_globalC* specifies the number of subspace dimensions to retain (a separate parameter specifies the number of local retained subspace dimensions). The principal component analysis itself is implemented in the *pca* function. A global PCA transformation matrix is recalculated every time that an image is introduced into memory through learning.

4.5 Clustering

4.5.1 Requirements

The major requirements for VEOR clustering, other than the need for high-quality partitioning, are firstly the ability to automatically determine, from input data, an appropriate number of clusters for those data; and secondly, to efficiently facilitate learning. The need for the former in a general-purpose, dynamic environment is immediately apparent. However, the latter requirement bears explanation. Every time the system learns, that is, when the system introduces new image data into memory, several time-consuming steps occur. If PCA is used for feature extraction, then a new transformation matrix must be calculated from the expanded data set. Clustering is then performed on the transformed data. If an image is later matched into the

cluster of the learned image, a cluster-specific transformation matrix is computed. These three steps are unavoidable. What we would like to avoid, however, are unnecessary changes in the membership of other clusters, because a change in any cluster requires that a new PCA transformation matrix be calculated for it when an image is matched to the cluster. So, the need to efficiently facilitate learning may be rephrased as an issue of the stability of clusters' membership between partitionings. Another way in which learning is facilitated is that, when a learned object is partitioned into its own cluster, an artificial class is created for it. This is described in detail in section 4.8.1.

4.5.2 Promoting stability of cluster membership

As stated previously, significant processing time may be saved if cluster memberships change as little as possible when adding new image data to VEOR memory. Unfortunately, the locally optimum partitioning reached by K-means is highly dependent upon the initial centroid values (seeds) chosen, and the most common methods for choosing seeds are by assigning them to the locations of random data members, or to random locations [7]. This means that in practice, clusters are rarely identical from one partitioning to the next when the natural groupings of the data are not well separated.

The K-Harmonic Means (KHM) algorithm [13] has been found to be largely insensitive to seed locations when compared to K-means [11], and was implemented in the hope of preventing unnecessary applications of PCA. Unfortunately, the partitions generated by KHM suffered from the conver-

gence of multiple centroids to the same locations, in effect preventing the data from being partitioned into more than a small number of clusters. This occurred when clustering both high-dimensional raw image data and lower-dimensional PCA subspace data, and under any parameterization of KHM.

A different approach to cluster membership stability is taken from Unsupervised Fuzzy Clustering — Optimal Number of Partitions (UFC-ONP) algorithm [10], which is described in further detail in section 4.5.4. Every clustering of UFC-ONP is seeded with the centroids from the previous clustering, which encourages the algorithm to converge to a local optimum similar to that of the previous clustering. In addition, because this approach uses the results of work that has already been done, it significantly reduces the amount of time necessary to converge to a local optimum. This method is used in VEOR, and the introduction of a new image into memory generally changes the membership of a small number of clusters, and often only of a single cluster.

4.5.3 Determining an appropriate number of clusters

For a clustering scheme to be useful in a real-world system, where the input data could vary significantly from one invocation to the next, and from one particular application of the system to another, it must not rely on an *a priori* specification of the number of clusters to be found. Users of clustering software often pick the value of a parameter, k, using knowledge of the input data and domain-specific intuition [12]. However, if the system is to learn from

new images extracted from a video stream, then it should be able to cope with the fact that the number of natural groups within the data might actually increase as time progresses. In addition, Draper et al. find that the number of actual classes in an image dataset does not necessarily correspond to the best value of k for the data, and get better results when overestimating the parameter [1]. They attribute this to the possibility that within each class, more cohesive logical groupings may exist. Thus, domain-specific knowledge regarding the number of classes among which discrimination must occur does not translate into knowledge of how many clusters should be found.

One method that may be used to estimate the relative goodness of different partitions is to apply a cluster validity index [8]. When applied to partitionings of the same data into different numbers of clusters, these indices may, in theory, show which number of clusters is more appropriate for the data. The Xie-Beni, Fukuyama-Sugeno [8], and Dunn [9] indices were implemented for this thesis. The Xie-Beni index of a partitioning is the ratio of total variation (the sum of the distances from centroids to their members) divided by the separation of the data (the smallest distance between two different data points). The Fukuyama-Sugeno index is a modification of objective function that Fuzzy K-means seeks to minimize. Both of these validity indices are expected to have smaller values for "good" partitionings, and higher values for "bad" partitionings. "Good" and "bad", of course, are relative terms that each index seeks to define differently. The Dunn index, on the other hand, is a ratio of smallest distance between two clusters and

the maximum cluster diameter; this is similar to a reciprocal Xie-Beni index (though the two indices differ on details), and as such, higher values indicate better clusterings.

Unfortunately, the optimal number of clusters found by these algorithms did not coincide well with our definition of a good partitioning; that is, a partition in which each cluster consists of a minimal number of classes, while avoiding needless fragmentation. The second criterion is important, because a trivial solution to the first criterion would be to cluster each datum individually — each cluster would then consist of a single class, but the partition would be useless. The ideal clustering, then, would have a number of clusters equal to the number of classes being clustered, and the members of each cluster would be of a single class.

Rather than rating a partitioning as a whole, as the discussed indices do, each cluster in a partitioning may instead be evaluated individually. This approach is utilized in the ISODATA clustering algorithm [7], which uses criteria such as a maximum and minimum number of data per cluster to determine if a cluster is valid, if it should be split, or if it should be merged with another cluster. For the purposes of learning-enabled object recognition in video, cluster splitting is more important than cluster merging, which was not included in VEOR. Given a good clustering of n data points into k clusters, we expect that a good clustering of the data after the introduction of the $n + 1^{st}$ datum would consist of either k or $k + 1$ clusters, because there should not be fewer natural groups in the data than existed previously.

In fact, this inductive cluster addition scheme can be used on training data as well, and an appropriate number of clusters for the data can be "grown" from a small number (the base case of the inductive algorithm). This is because, given a clustering of n data into k clusters, where k is less than the number of natural groups in the data, we expect that the introduction of the $k + 1^{st}$ cluster to result in a better clustering. This is under the provision, however, that the new centroid is added in such a way that it converges to a meaningful location; an added centroid that simply pulls off an outlier of a natural group, for example, does not contribute to a more meaningful clustering. This is a real problem for K-means and derivative algorithms, due to their convergence to a locally, rather than globally optimal partitioning, and much research has been done towards the goal of being able to determine initial centroid locations that result in meaningful clusterings [11].

This leads to a major benefit of per-cluster validity testing: it tells us not only *whether* a clustering is valid (that is, whether a centroid should be added), but also *where* a centroid should be added. In practice, when a centroid is added to a large cluster, the resulting partitioning is similar, except that the "invalid" cluster has been split. These newly formed clusters may also assume outliers of nearby clusters that are closer to the newly positioned centroids. The new centroid should be added "in the midst" of the cluster contents, to avoid the problem of pulling outliers from the cluster, and yet must not be the average of the data. That value would place it at the same location as the original cluster centroid, and the two would converge

to the same location. The method used in VEOR to determine a location for an added centroid, given a cluster to be split, is to take the average of a random subset of the cluster data, where the size of the subset is half that of the cluster itself. The new centroid can then compete with the original cluster centroid (which is initially assigned to the average location of *all* of the cluster data) for membership of said data.

The simplest method of validating that a cluster is not "too large" is to set a hard limit on the number of data that may be in a cluster, as is done by ISODATA [7]. However, this approach does not facilitate the scheme chosen to detect and learn the presence of a new object class in VEOR. New class detection is based upon the assumption that the first appearance of a novel class will be partitioned into a cluster consisting only of that image. If a cluster must reach a certain number of members before being split, and an instance of a new object happens to be closer to the centroid of a small cluster than to any other centroid, then it may be assigned to that cluster, despite being far from it. A similar approach to cluster validation that does a better job of handling this situation is to set a limit on total variance of a cluster, that is, on the sum of all centroid to datum distances within a cluster. This method will split both clusters that grow to have too many members and clusters that are not compact, and is the method used in VEOR.

4.5.4 Architectural considerations

Although not actually used in VEOR, the UFP-ONC (unsupervised fuzzy partition-optimal number of classes) algorithm [10] contributed architectural inspiration to the system. UFP-ONC is similar to VEOR clustering in that it is based on a K-means derivative and that it uses clustering validity measures to determine whether additional centroids should be added into a partitioning. However, the validity measures that it uses (fuzzy hypervolume and two types of average cluster density) are global in scope, and the method used for introducing a new centroid is likewise not cluster-specific. In addition, UFP-ONC has a secondary clustering phase, based on the Maximum Likelihood Estimation, whose centroids are seeded by the primary clustering phase. The secondary phase is adept at finding hyperellipsoidal clusters, given good cluster centroids found by the primary phase. It is unlikely that the ability to find such clusters is worth the added complexity within VEOR, because the goal of its clustering is not to group all members of a class together. Instead, it is to group similar-looking images together. Draper et al. speculate that non-hyperspherical clusters of images may be approximated with multiple hyperspherical clusters, such as those found by K-means [1].

Although UFP-ONC was not specifically engineered for video, it has a characteristic that lends itself to a learning-enabled video application: an incremental determination of an appropriate number of clusters, where the work of clustering data into k clusters is not repeated when clustering the data into $k+1$ clusters. This thriftiness is facilitated by using the centroids

found in one clustering as seeds for the next, as already discussed. UFP-ONC is meant to be applied to a static data set, and its incremental nature serves to determine the optimal number of clusters for the data. When this approach is used with a dynamically increasing data set, such as the memory of VEOR, then it also serves to dramatically decrease the time spent on clustering when learning is performed. Imagine a situation where n images are learned, one at a time, and that every time that learning occurs, all of VEOR memory must be reclustered. By using this clustering strategy, the processing necessary to cluster the data (including the nth image) "from scratch" may instead be spread out over the lifetime of the system, and the addition of nth image requires only one more "iteration" of the clustering algorithm (or two, in the case in which it is determined that a cluster is too large and must be split; see section 4.5.3).

4.5.5 Fuzzy K-means

Fuzzy K-means (FKM), Also known as Fuzzy c-Means, is a derivative of K-means that incorporates fuzzy set theory [14]. Rather than assigning hard membership to data in exactly one cluster, FKM gives data fuzzy membership (in the range [0,1]) in all clusters. Although (like K-means) FKM achieves a locally optimal solution to its objective function, it is better than K-means at avoiding relatively bad ones. Thus it is likely to arrive at a solution that is closer to the global optimum [7]. The objective function that FKM seeks to minimize is

$$J_{FKM}(X, C, U) = \sum_{j=1}^{k} \sum_{i=1}^{n} (u_{ij})^m \|x_i - c_j\|^2 \tag{4.1}$$

This is a simplified form of the FKM objective function that uses squared Euclidean distance as the error measure. Compare it to equation 2.1; the differences highlight the major variations between K-means and FKM. Where equation 2.1 sums, for each cluster, over only the data belonging to that cluster, equation 4.1 sums, for each cluster, over all data (because all data have membership in all clusters) and that summation is modulated by the fuzzy membership u_{ij} of each datum i in each cluster j. $\boldsymbol{U}$ is a fuzzy membership matrix of dimensions $n \times k$, where n is the number of data and k is the number of clusters.

The parameter m determines the "fuzziness" of the clustering: as m approaches the value one from above, cluster memberships approach those of "hard" K-means; that is, either one or zero. As m approaches positive infinity, membership of data becomes equally distributed among clusters, regardless of the distances between data and cluster centroids. m is constrained to the range (1,+Inf) [8]. Although Pal and Bezdek suggest that good values for m are generally in the range [1.5,2.5], such values proved to be inappropriate for clustering in VEOR. Values of m greater than approximately 1.3 cause multiple centroids to converge to the same locations when FKM is used in VEOR, a problem also encountered when using K-Harmonic Means (see section 4.5.2). Other authors have found 1.3 to be an optimal value for their

particular problem [11]. It is interesting to note that, according to [11], the membership functions of K-Harmonic Means and FKM are mathematically identical under certain configurations. Perhaps a single underlying property of VEOR data causes issues with both clustering algorithms. In any case, FKM, configured with m=1.3, allows VEOR to perform better than when "hard" K-means is used, despite the fact that the algorithm is not configured to be as "fuzzy" as when used in some other situations.

Like K-means, FKM uses alternating optimization to converge to a solution. The formulas used to optimize cluster membership and centroids may be derived from 4.1, and are, respectively,

$$u_{ij} = \left[\sum_{h=1}^{k} \left(\frac{\|x_i - c_j\|^2}{\|x_i - c_h\|^2} \right)^{2/(m-1)} \right]^{-1} \tag{4.2}$$

and

$$c_j = \frac{\sum_{i=1}^{n} (u_{ij})^m x_i}{\sum_{i=1}^{n} (u_{ij})^m} \tag{4.3}$$

These terms are alternately optimized in a loop that terminates when a convergence criterion is satisfied [8]. In the VEOR implementation of FKM, for each iteration, the difference between the membership matrix of that iteration and that of the previous iteration is calculated. If the Euclidean norm of that difference matrix is less than a threshold, the algorithm terminates.

Implementation notes

The VEOR Fuzzy K-means implementation can be found in the MATLAB function *fkm*. The algorithm takes as arguments a matrix of data to be clustered and a matrix of initial cluster centroids. In VEOR, the centroid argument consists of the centroids from the previous invocation of FKM, possibly along with a new centroid (see section 4.5.3). The parameter k is inferred from the size of the centroid matrix. To use the function in a situation where no prior centroids exist, the user must initialize the desired number of centroids by hand. It is recommended that the centroids be initialized to the values of random data (the Forgy method), as this technique is simple and works well [11]. The converged cluster centroids, as well as the final membership matrix, are returned by the function. Two algorithmic parameters may be set which affect FKM. The fuzziness coefficient, called m in equation 4.1, may be specified by the parameter *fkm_fuzziness*. It defaults to a value of 2.0. The parameter *fkm_eta* defines the membership difference norm threshold, and defaults to a value of 0.000001.

A singularity occurs in equation 4.3 when the distance between a centroid and a datum is zero. Pal and Bezdek [8] recommend splitting all membership of the datum exclusively among collocated centroids by any method; in this implementation, membership is split equally among such centroids.

4.5.6 Summary

For the sake of comprehension, the end to end clustering procedure will be reviewed now that its components have been described. When run for the first time, initial centroids are chosen to be the locations of a small number of random data. Otherwise, initial centroids are the converged centroids from the previous clustering. Fuzzy K-means is used to partition the data, and the membership of each resulting cluster is compared to the membership of the previous clusters to determine for what partitions membership has not changed. All clusters whose membership has changed are flagged as "dirty", signifying that their PCA transformation matrices must be recalculated before exemplar matching into those clusters can occur. Each resulting cluster is also checked for validity, and if the total variance of any cluster is above a threshold, then an additional centroid is inserted into that cluster, and the data are repartitioned.

4.6 Exemplar matching

Exemplar matching in VEOR is similar to that used in the EOR system as proposed in [3], which is different than the more recent proposal [1]. In both cases the principal component analysis is applied locally to clusters of images, but the difference lies in the representation of those images. In the latter proposal, PCA is applied to image data to which Gabor filtering and the Hough transform has been applied, and in the former, as well as in

VEOR, PCA is applied to raw image data. After being assigned to a cluster, an image is multiplied by the PCA transformation matrix for that cluster. If cluster membership has changed since the last time the matrix was calculated, then PCA is run on the cluster first, and a new matrix is generated. Within the subspace, the nearest neighbor is found for the image from the cluster members, using the squared Euclidean distance metric; this is the exemplar match. Whereas this match is the output of EOR, VEOR forwards the information to the Membership subsystem, so that an object's membership may be tracked throughout its lifetime. A distance metric is passed to the Membership subsystem, because such a measure is useful for determining how strong of a match has occurred, and how strong of a hypothesis should be drawn. However, the distance used to find the exemplar match in the local PCA subspace is not passed; this is because a distance in one subspace is not comparable to a distance in another subspace. To determine that one exemplar match is stronger than another, a global metric must be used. A simple approach would be to find the distance between input and exemplar in grayscale image space. However, feature extraction has already been done on both data, so the squared Euclidean distance is calculated based on their feature vectors. By default, then, distances are calculated in the global PCA subspace.

Implementation notes

Exemplar matching is performed within the *EORClassify* function, which in turn calls *pca*. The system parameter *EOR_localC* specifies the number of subspace dimensions that are retained when performing local PCA. The local PCA transformation matrices, as well as a vector of flags indicating the "dirty" status of those matrices, is stored from one invocation of *EORClassify* to the next in a VEOR data structure.

4.7 Membership

As noted by Baek, classification of images according to symbolic or logical labels is the responsibility of super-visual processes, and is therefore explicitly excluded from EOR [2]. The classification done by EOR is the matching of visual input to compressed memories, inspired by the apparent function of the right inferior frontal gyrus [1]. The output of the system during testing is the memory image that provides the best match for the input image. Symbolic labels associated with images (such as "cat" or "dog") are never provided to EOR, though they are necessarily used to evaluate performance. Because the goal of VEOR is to classify video input according to class labels, such super-visual processing was implemented. A true associative memory, as described by Draper et al., would receive input from higher-order reasoning, the visual pathway, and possibly from other sensors [3]. This more modest implementation is interfaced only by the visual pathway, and provides hypotheses

of class membership. These hypotheses are not probabilities (though they bear a passing resemblance); instead, they are scores in the range [0,1.0], for which evidence is accumulated over the lifetime of a tracked object, and which represent confidence that the object is a member of any number of exclusive classes. The hypotheses are "fuzzy" in that they simultaneously track membership in multiple classes, however, if a "hard" classification is desired, it may be obtained by considering only the class in which an object has maximum membership. Because the very images being classified in this manner can be introduced into memory, they may provide exemplar matches for future visual input. To facilitate this, assignment of membership is treated slightly differently depending on whether an exemplar match is a training image, a learned image of a different object, or a learned image of the object in question.

4.7.1 Matching to a training image

The simplest case of membership assignment is when the exemplar match is a training image. This is because we may be completely sure (a confidence of 1.0) that the image is of a certain class, assigned by a human before training; membership in all other classes is zero. The membership contributed by such a match should be dependent on the distance between the exemplar and the image being classified, because a smaller distance indicates a better match (and a higher confidence that the exemplar's class is also that of the input). The actual distance value is not calculated here, but is provided by the EOR

subsystem. Thus, the same metric is used, the squared Euclidean distance in the global PCA subspace (see section 4.4). The particular function used to determine a membership contribution should decrease monotonically as distance increases. The following contribution function was chosen:

$$contribution = \frac{1}{1 + \sqrt{distance}} \tag{4.4}$$

Note that the square root of the squared Euclidean distance between the images is simply the Euclidean distance. An experimentally determined stretching factor was added so that the curve of the function better fit the data. Although this did not affect classification performance, it was necessary for the learning of new classes. The modified function is then:

$$contribution = \frac{1}{1 + \frac{1}{10}\sqrt{distance}} \tag{4.5}$$

It may be desirable that a more suitable function (perhaps a Gaussian or a function that is automatically fit to the data at hand) be used; however, the above function achieves our goals. Each frame in which an object appears provides an image that contributes to the accumulated membership hypothesis for that object. To keep values in the unit range, an object's hypothesis is averaged over that number of frames.

More formally, when an image A of an object J is matched to an training image B, the contribution of membership of A to J is calculated according to function 4.5, and the contribution is only of the class in which B has

membership. The accumulated membership values of J are averaged over the number of frames in which J appears.

4.7.2 Matching to a different object

An important distinction is that membership is assigned to tracked objects, not to their constituent images. Therefore, when a learned image is an exemplar match for another image, the membership of the former's object is taken into consideration. For example, a learned object may have been matched to car training images for several frames, and to van training images for several frames, and thus would have accumulated membership in both classes. If another object was later matched to any of these images, then membership in both classes would be contributed, regardless of whether the match was a "car-like" or "van-like" view. An approach could have been taken in which a "car-like" view would contribute a stronger car hypothesis; however, this does not make use of the certain knowledge (assuming perfect object tracking) that those images belong to the same object — and that by matching to this object, rather than to a car training image, the system determines that the new object looks more like the first than like the trained car. If uncertainty existed regarding the class membership of the first object, then it should be propagated to the second.

Because an exemplar may have membership in more than one class, the contribution function is modified in this way:

$$contribution = \frac{1}{1 + \frac{1}{10}\sqrt{distance}} \times exemplarMembership \tag{4.6}$$

Where *exemplarMembership* is a vector in which each value denotes the exemplar's membership in a different class. *contribution*, then, is likewise a vector. This function is actually a generalization of equation 4.5: when matching to a training image, *exemplarMembership* has a value of 1.0 for a single class, and zeros for all other classes, and this is reflected in the resulting contribution.

When an image A belonging to object J is matched to an image B belonging to an object K, where J is not K, then a portion of the membership of K is contributed by A to J. The size of the portion is dependent on the distance between A and B, using the same function as when matching to a training image. The contribution is averaged over the number of frames in which J appears.

4.7.3 Matching to the same object

When matching an image to another image belonging to the same object, that is, to a previously-seen view of the object being classified, then assignment of membership is treated in a different manner. This is because such an exemplar match does not provide any truly new information regarding class membership of the object. Whereas matching an object to a training image

of a car tells us (with a certain degree of confidence) that the object looks like that car, matching the same object, in the next frame, to the learned image of itself tells us only that it "looks like it did when it looked like a car". If such a case were treated in the same manner as a normal exemplar match, then the object would acquire a portion (dependent on the distance of the match — in this case, probably a small distance, so a large portion) of its own membership hypothesis. This gives no new information, since the individual class memberships of the objects would be affected proportionally. In fact, because contributions are averaged over the lifetime of objects, contributing a fraction of an object's own membership would cause its overall hypothesis to decrease over time. This would create a strange paradox where, the longer we see an object, the less confident we become in its classification!

On the other hand, neither can such matches be ignored, for they do indeed provide useful information. An object that looks "a little bit" like a car for many frames, and "a little bit" like a van for a few frames, should be classified, albeit weakly, as a car. To allow for both of these considerations, the following scheme is implemented: when an image A is matched to an image B of the same object J, the contribution of membership of A to J is the same as the earlier contribution from B to J (as opposed to a portion of the membership of J, as happens when matching to a different object). The accumulated membership is still averaged over the number of frames in which the object appears. Whereas matching an image to a different object or to a training image has the affect of *adding* evidence to the membership

hypothesis, this kind of matching *rebalances* the object's hypothesis among the classes in which it has membership, and *strengthens* the original contribution of the exemplar match image. The reason that the whole contribution of B, rather than a distance-related portion, is recontributed, is because we may be completely confident that A and B are from the same object. In addition, to recontribute only a fraction of B's contribution would have the effect (after averaging) of diminishing the hypothesis, rather than rebalancing it. That is, rather than strengthening the class memberships favored by B at the expense of memberships in other classes, recontributing a fraction would weaken membership in all classes (though not proportionally).

4.7.4 Implementation notes

Following the models of EOR and biology [2], visual functions (including exemplar matching) and classification according to labels are kept in separate subsystems. Membership functionality is implemented in a single function, called *membership*, which may be invoked with different parameters to perform different membership-related tasks. Membership setting modes include a training mode, in which training images are given full membership in a given class, and an exemplar matching mode, which is invoked in the same way for the three types of matches detailed above (the *membership* function determines which type is appropriate). In addition, *membership* has a mode in which a new class is created for an image, and the corresponding object is given membership in that class. This mode is used when the EOR subsys-

tem partitions an input image into its own cluster (see section 4.8.1). There are two membership retrieval modes: one for objects, and one for training samples (because training samples are not associated with objects). A final invocation mode returns the symbolic label associated with an integer class identifier; such integers are used as indices into the membership vectors returned by the retrieval modes. The length of membership vectors may be less than the total number of classes used in the system; however, a vector has an implied membership of zero in classes whose identifiers are greater than its length.

The persistent variables (i.e, memory) associated with *membership* are stored in the same VEOR data structure used by the EOR subsystem (though there is no intersection between the fields used by the two modules). When used in a "getting" fashion, *membership* takes this structure as a parameter, and when "setting", the structure is both a parameter and a return value of the function.

4.8 Learning

4.8.1 Types of learning

Two different types of learning are facilitated within VEOR. The first is the ability to learn new instances of familiar classes of objects, under familiar views. When an object image is extracted from video and classified by VEOR, a distance metric between the image and its exemplar match is

calculated. If that distance is above a threshold, the image is considered to be significantly different from the images in memory, and learning occurs. The image is added to the data in memory (consisting of both trained and learned images), which are then reclustered. A fresh exemplar match is found for the image, excluding the obviously bad choice of matching it to itself, and class membership is calculated. In effect, the image becomes part of the training set, with the following exception: while we can be certain of the class membership of a true training image, we cannot be certain of the membership of a learned image. This concept is discussed in more detail in section 4.7. If, on the other hand, the distance between an image and its exemplar match is below the threshold, then the image is classified, but not learned. The current implementation of VEOR does not actually make use of this instance-level learning, except in that it is necessary to facilitate the learning of new viewpoints and classes (described below). However, a future version of the system could use this knowledge, along with object tracking, to identify an object as being both a certain class and a certain instance of that class. For example, an object could be classified as both a car and as a certain model of car, with a separate level of confidence in both classifications. Because of this learning, a system would not need to be trained on all models of cars to be able to classify objects as those models. However, association of a model name with an instance learned at runtime is impossible, and an artificial label would be created. This label could later be replaced with a model name supplied by a user who is monitoring the system, and vehicles

classified on an instance level to that model would then be associated with the model label.

The second type of learning performed by VEOR is that of novel classes and viewpoints. As stated above, if an object is found to be significantly different from its exemplar match, it is introduced into memory as a learned image. Part of that process involves reclustering the data (see section 4.5). If, after this step, the new image is in a cluster by itself, then it is reasonable to say that it is extremely different from all of the images in memory (both trained and learned). Exemplar matching cannot even be performed, as there are no other images in the cluster to which an exemplar match can be made. In this case, a new (artificial) class is created, and the object is assigned membership in that class to a moderate degree. The image is not assigned certain membership in the new class, because it is unknown at that point whether it is actually a member of an unknown class, or if it is simply a novel viewpoint of a known class. If the latter is the case, and the view of the object later changes through rotation or other means, then the object may begin to resemble a trained class. Strong exemplar matches to training images will outweigh the membership in the artificial class, and the object (along with its constituent images) will be classified most strongly as the trained class. Future exemplar matches of new images to the novel viewpoint will classify the images as the trained class. Thus, the system will have learned, at runtime, to classify objects under viewpoints that were unfamiliar at the time of training. An example of this behavior, involving

classification of head-on views of cars using a system trained on car profiles, is included in section 5.3.

It should be noted that an image must be quite different from all images in memory for novel class detection to occur. If VEOR sees a profile of a van after being trained on profiles of cars and trucks, then the van will probably be clustered along with members of one (or both) of those classes. An exemplar match would then be made to one of the classes, though possibly with a low degree of certainty. If a pedestrian was classified by the system, on the other hand, he or she would almost certainly be found to be an instance of a new class. Another possible issue is that classification accuracy may diminish as learning occurs, because the likelihood of exemplar matching to a training image (with a known class) decreases, while the chances of matching to a learned image (with only a class hypothesis) increases. Whether performance degrades due to learning, and to what degree, has not been tested. It may be important to set learning-related parameters (see the next section) in such a way that learning occurs only when truly useful, and that "trivially novel" images are not learned.

4.8.2 Learning-related parameters

Several system parameters influence the behavior and performance of learning. One of them is the criterion by which clusters are split, the *EOR_maxClusterVariance* threshold (see section 4.5.3). If that parameter is set to too high of a value, then an image that is vastly different from training data may

be partitioned into a cluster along with the closest of those data, rather than into a cluster by itself. Because such individual clustering is used to detect a novel class or perspective, this type of learning would be prevented. If, on the other hand, *EOR_maxClusterVariance* is too low, then the clustering of VEOR memory may become overly fragmented, reducing performance. Setting the variable to a very low value might even cause images that ought to be classified as a known class to be misclassified as a novel class.

Another parameter to which class and perspective learning is sensitive is *membership_newClassMembership*. In fact, this variable determines the tendency of the system to prefer one of those types of learning over the other. Thus, if the learning of both new classes and new perspectives of existing classes is desired in the same application, care must be taken regarding this parameter. When an image is determined to be a member of a new class or perspective, it is given membership in a new class with a unique artificial label, such as "Unknown_1". The confidence of the object in this new class, in the range [0,1], is specified by *newClassMembership*. A high value for this parameter causes a high degree of membership of an object in its new class, and strong subsequent evidence is needed to associate the object with another class. This behavior favors the treatment of new images as novel classes rather than new perspectives. A low value for *newClassMembership* would cause membership in a new class to be overcome relatively easily by membership in an existing class. This would allow new perspectives to quickly be assimilated into their respective known classes. However, if an object of a

truly novel class, unseen under any perspective in training data, were viewed, any frame in which that object appeared to be similar to (i.e, in which a particular image of that object was exemplar matched to) training data would cause membership of that object to be strongest in the trained class.

Take for example the testing video shown in figure 5.5. In the first frame in which the object is tracked by the system, it clustered alone, and an artificial class is created. The images of that object for the next several frames are matched to that class. When the car begins to turn, it looks different enough from both the training images and the head-on images from the same object that a second artificial class is created. However, the few frames in which the object looks like this new class are not enough for membership in it to become higher than membership in the first, and the object is considered to be an instance of the first unknown class throughout. Further into the turn, the perspective of the car actually looks more to VEOR like a van than anything else. Here is where the choice of *newClassMembership* comes into play. A low value will cause membership in the two artificial classes to be instantly dwarfed by the membership contributed from the van training image. Once the car turns fully to profile view, the greater number of frames in which it looks like a car, along with the lower distance of the exemplar match to a car training image, cause the car hypothesis to become larger than the van hypothesis. On the other extreme, a very low value for *newClassMembership* would implement an assumption that an unclassifiable image, like the head-on car, represents a truly novel class, and would contribute such a high

confidence in that class that the contributions of the van and car matches are not enough to overcome it. This is especially true in this example, because the number of frames in which it looks like a car is actually smaller than those in which it looks like artificial classes. Obviously, the value for the parameter should be chosen based on the nature of the video input, and on the type of learning desired. By carefully choosing a value for the parameter, both class and perspective learning may be facilitated.

The *findObjects_minSize* parameter, which specifies how large an object must be in order to be tracked, plays a small role in learning. This value determines how large, in pixels, a segmented area in the video input must be for it to be tracked as an object (see section 4.3). If too small of a value is chosen, then distant objects occupying few pixels may be tracked, though their details have been obscured by the discrete nature of, and low number of, their pixels. Because of this low resolution, objects of similar grayscale intensity tend to look the same at far distances. Learning from such images does not provide a sound prototype to which to perform later exemplar matching. Indeed, even classifying such instances is unreliable, and should be avoided regardless of learning.

Chapter 5

Experiments

An estimation of the performance of VEOR on real-world video input is found through testing with two datasets: a large database of vehicle profile images, and a smaller set of videos. In section 5.2, the still images are used (in disjoint subsets) as both testing and training data, while in section 5.3, they are used for training while testing is performed on the videos. The former experiment serves to simulate video classification using a large number of image samples, while the latter tests whether these results reflect performance on real-world data.

5.1 The cat and dog database

In the literature [1, 2, 3], EOR implementations are tested in part with a database of cat and dog faces[1]. The database contains 100 grayscale images

[1]This database was graciously provided to the author by Dr. Bruce Draper.

Figure 5.1: Some examples from the cat and dog database, courtesy of [3].

from each of the two classes, and these images are 64 pixels in width and height. The images are cropped and scaled such that the eyes in each image are in roughly the same place, and the faces take up most of the image area. The faces are not segmented, and because VEOR requires segmented input, artificial segmentation masks were created which cause all of the pixels in the images to be considered foreground. Although this database does not represent the type of input for which VEOR is designed, its use here serves two purposes. First, it allows the effectiveness of VEOR, when used as a general-purpose object classification technique (that is, one which expects unsegmented still images as input), to be gauged. Second, it provides a point of comparison between parameterizations of VEOR and EOR.

The testing procedure used here is similar to that described in [3]. 80% (160 images) of the database is used as training data, and the remaining 20% (40 images) as testing data. If the exemplar match found by VEOR for a testing datum is of the same class as that datum, then the match is considered a success. Because classification accuracy varies depending on

which data are included in the training and testing sets, classification of the testing set is repeated 25 times, and training on occurs a different random subset of the database each time. Therefore, a total of 1000 exemplar matches are performed for each configuration tested.

The default algorithmic configuration for VEOR is tested under different parametric configurations. Three variables (the fuzziness coefficient of Fuzzy K-means and the number of retained global and local PCA subspace dimensions) are varied independently while holding the others. The results may be seen in figure 5.2. For local PCA, little accuracy is gained by retaining greater than five subspaces (though in both cases, performance decreases if less than five are retained). This finding is similar to that found in the literature [2, 1]. Increasing the number of retained global subspaces does not show the same smooth increase in performance, but rather a fluctuation within a confined range. Performance increased significantly when a fuzziness coefficient of 1.3 was used, though values higher than 1.3 yielded empty clusters.

Several algorithmic configurations which resemble the EOR proposals [3, 2, 1] in various respects were tested as well. Gabor filter-based feature extraction (as opposed to global PCA) was enabled for all of these. The specifics of these configurations, as well as their classification accuracies, may be seen in table 5.1. In these tests, the Hough transform and the application of local PCA to clustered feature vectors (rather than to their corresponding image data) decreased performance, while the use of Fuzzy K-means and incremental clustering improved performance. While a fuzziness coefficient of

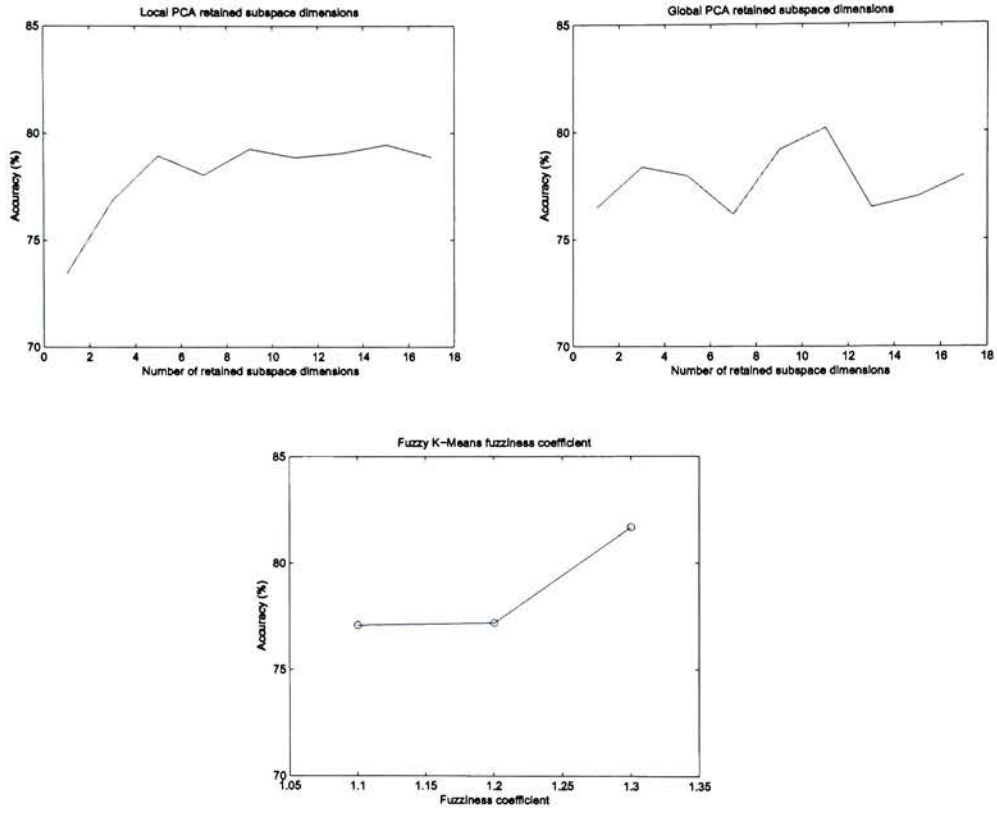

Figure 5.2: Cat and dog database classification performance using global PCA feature extraction. The number of local and global PCA subspace dimensions are varied (top left and top right, respectively), as well as Fuzzy K-means' fuzziness coefficient (bottom). The default values for the number of global and local subspaces is 5, and the default fuzziness is 1.3.

Configuration	1	2	3	4	5	6
Accuracy	67.0%	80.8%	84.6%	81.5%	81.7%	83.8%
Clustering algorithm	HKM	HKM	FKM	FKM	FKM	FKM
Fuzziness	n/a	n/a	1.1	1.1	1.1	1.3
PCA on raw images?	Yes	Yes	Yes	No	Yes	Yes
Hough transform?	Yes	No	No	No	No	No
Incremental clustering?	No	No	Yes	Yes	No	Yes

Table 5.1: Cat and dog classification using Gabor filtering for feature extraction. "PCA on raw image data" indicates whether PCA is applied directly to image data, or to a Gabor-filtered feature vector.

1.3 provided better results when global PCA is used for feature extraction, it does not when Gabor-based feature extraction is used on this data. The best performance was achieved with this dataset using Gabor-based feature extraction, along with incremental Fuzzy K-means.

5.2 The segmented vehicle profile database

The segmented vehicle profile database consists of 300 samples, each of which has four parts: a video frame, a mask frame, a coordinate pair, and a class label. The purpose of these components is described in section 4.1. Three classes (cars, vans, and pickup trucks) are represented by 100 samples each. The video frames were extracted from footage taken by the author, and were segmented using multi-Gaussian background modeling (see section 4.2). Frames were oriented such that for each, the vehicle identified by the corresponding coordinate pair faces from left to right. Vehicle types (such as the

station wagon) that are visually similar to more than one of the three represented classes were excluded, as were pickup trucks whfully ose appearance was significantly altered by scaffolding or cargo. Any vehicle whose classification was not instantly obvious to the author was also excluded, because VEOR is an appearance-based technique; if a human must use logic to deduce the classification of a vehicle, then a system such as VEOR should not be expected to classify it based solely on its appearance. Many variations do exist within each class represented by the database, however. For example, the van class also includes minivans. Videos were captured at a range of distances from objects, resulting in a variety of pixel resolutions.

Because the goal of this experimentation is to predict the effectiveness of VEOR on real-world video data, an attempt was made to keep testing data realistic. The video from which the data was extracted was taken at several locations, with both man-made and natural backgrounds. Seasonal and weather conditions vary from one frame to another, as does the time of day. Segmentation masks vary in their quality, and generally include some background or exclude part of a vehicle. Some even have holes; however, unconnected and very badly segmented objects were excluded.

The data are not completely realistic, however, because shadow removal was performed by hand on a majority of masks. This was necessary because many shadows significantly distort the appearance of their corresponding objects, and a bottom-up appearance-based approach such as VEOR would perform poorly when trained on such images. Although shadow removal

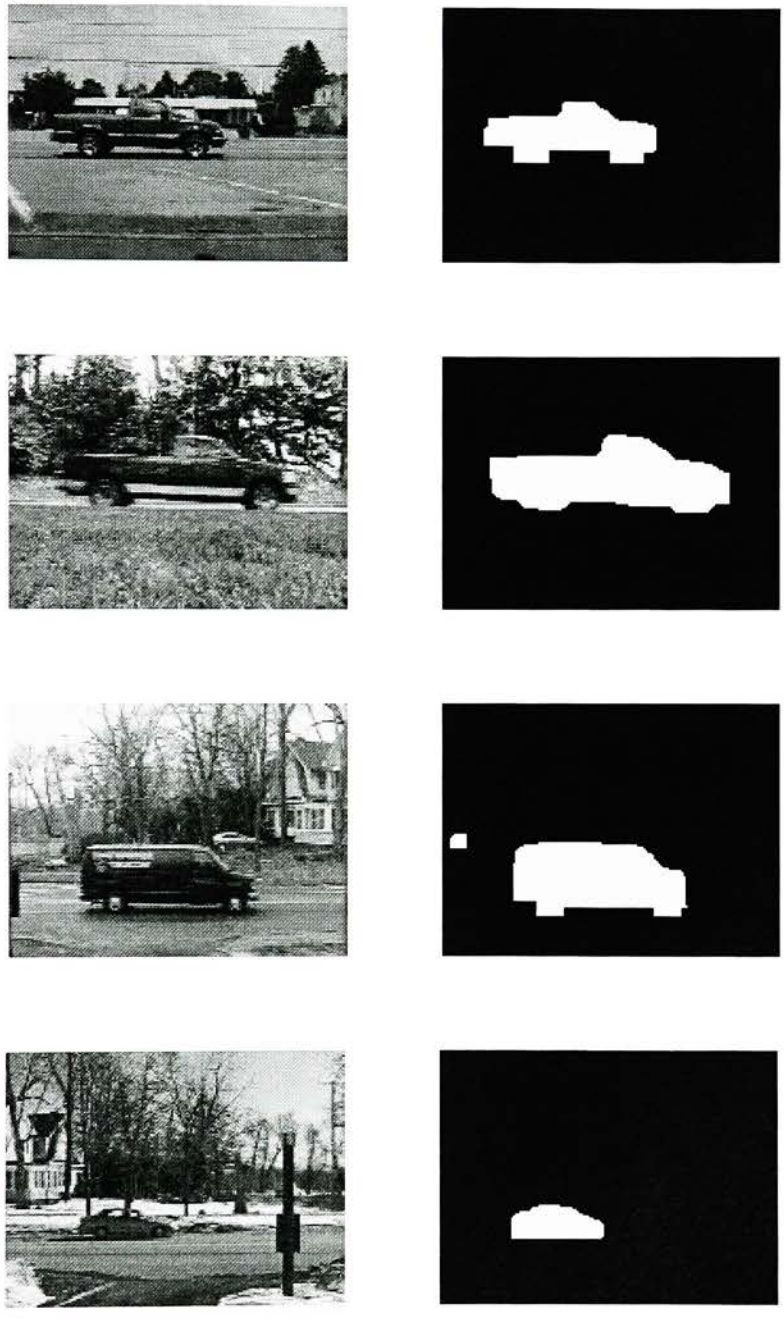

Figure 5.3: Examples of video frames from the vehicle database, and their corresponding masks.

makes the database less realistic, it is justified here because a robust shadow removal technique is being developed separately at RIT, and will be used in a future version of the foreground segmentation system. If such a system is used as a pre-processing step, then a deployed VEOR-based application would receive runtime input that is similar to this database. A "not too careful" approach was taken when removing shadows, corresponding to the imperfect segmentation of the objects themselves, and to the imperfect shadow removal that may be expected from any modern technique.

As with the testing performed on the cat and dog database, the number of local and global retained PCA subspaces and Fuzzy K-means' fuzziness coefficient were varied. Performance increased as local retained subspaces increased, tapering off after 5. Performance actually decreased slightly as more global subspace dimensions were added, however, the difference in accuracy over the tested domain was only approximately 2%. Accuracy was also not affected significantly by changing the value of the fuzziness coefficient. A final, major difference between the behavior of the vehicle database and the cat and dog database is that, although performance of the two is similar when Gabor-based feature extraction, global PCA feature extraction improves performance on segmented vehicle data, and decreases performance on unsegmented cat and dog faces. This finding shows that the algorithms and parameters that should be used for classification depend greatly on the image data to be classified.

Unfortunately, not all real-world vehicles face from left to right, as do all

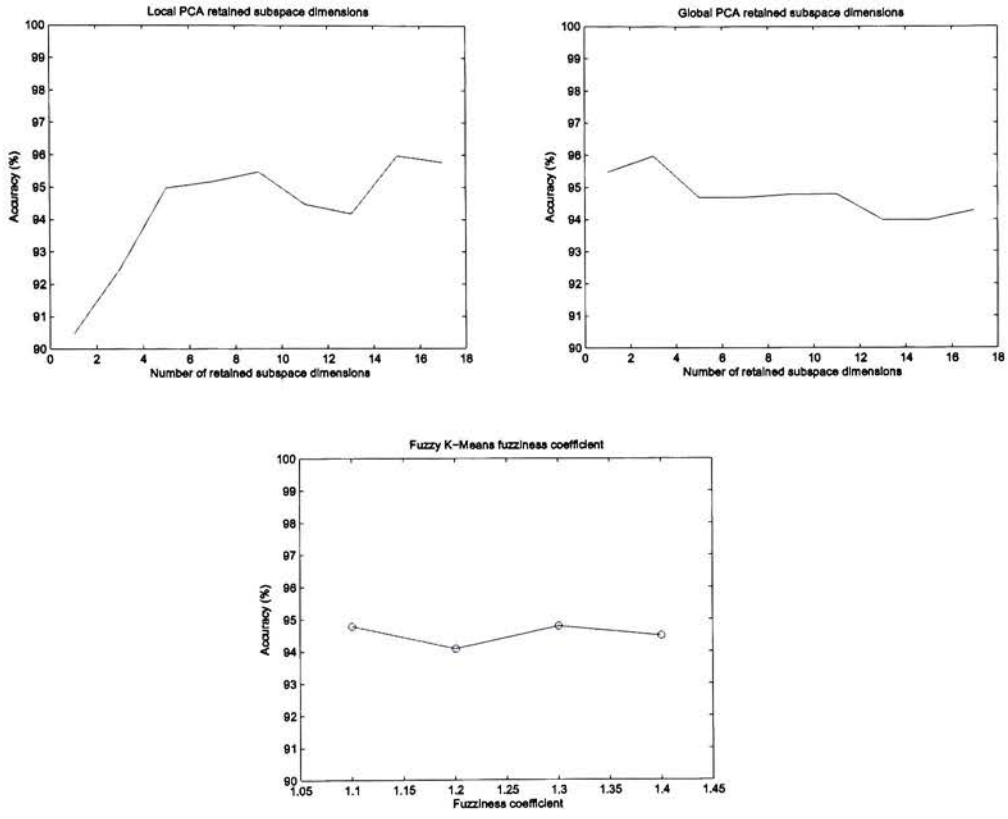

Figure 5.4: Vehicle database classification performance using global PCA feature extraction. The number of local and global PCA subspace dimensions are varied (the top left and top right graphs, respectively), as well as Fuzzy K-means' fuzziness coefficient (bottom).

Configuration	1	2	3	4	5
Accuracy	53.8%	81.2%	85.0%	82.3%	82.3%
Clustering algorithm	HKM	HKM	FKM	FKM	FKM
Fuzziness	n/a	n/a	1.1	1.1	1.1
PCA on raw images?	Yes	Yes	Yes	No	Yes
Hough transform?	Yes	No	No	No	No
Incremental clustering?	No	No	Yes	Yes	No

Table 5.2: Vehicle database classification using Gabor filtering.

Configuration	1	2	3	4	5	6
Accuracy	77.3%	88.5%	78.8%	83.1%	73.5%	80.5%
Global dims	10	50	10	50	10	50
Metric	sq.euclid.	sq.euclid.	city bl.	city bl.	mahal.	mahal.

Table 5.3: Configurations using image mirroring. The number of local retained PCA subspace dimensions in all cases is 10, and the fuzziness coefficient is 1.3. The number of retained global subspace dimensions is either 10 or 50, and the subspace distance metrics tested are squared Euclidean, city block, and Mahalanobis.

of the samples in the vehicle database. Therefore, additional tests were performed using image mirroring (see section 4.1), which reduced the accuracy of classification. In an attempt to improve the global subspace distance metric, upon which exemplar matching under image mirroring depends heavily, the city block and Mahalanobis metrics were employed, as suggested in [4]. Neither metric proved superior to the squared Euclidean distance in this case. However, accuracy was improved by increasing the number of global subspace dimensions, which presumably made the distance metric more meaningful.

5.3 Experiments with video data

5.3.1 Vehicle profile videos

To determine whether the results found in section 5.2 are representative of the performance of VEOR when video data is used as input, experimentation was performed using 45 video clips as testing data (15 each of cars, pickup trucks, and vans). Each clip shows a vehicle driving in profile from one side of the camera's field of vision to the other. The classification of the object in each frame contributes to its overall membership hypothesis. If the final hypothesis for an object attributes more membership to the object's actual class than to all other classes (remember that a membership hypothesis specifies some level of membership in all classes), then that object is considered to have been correctly classified. Image mirroring was necessarily enabled, because vehicles drive from right to left in many of these videos. No shadow removal is performed on these clips. However, VEOR is given an advantage in the classification of these vehicles — evidence is gathered over the period of time that the vehicles are visible, which should produce more reliable classifications. Results of these tests may be seen in table 5.4.

Accuracy was lower when this data was classified than when mirrored still image data was used (see table 5.3). Relatively bad segmentation, due to lack of shadow removal, is probably the main cause for this. Among the three tested vehicle classes (cars, pickup trucks, and vans), pickup trucks, due to their high clearance, were affected the most by the included shadows. This

Configuration	1	2	3
Accuracy	64.4%	82.2%	82.2%
Accuracy (excluding pickups)	83.3%	100%	96.6%
Global subspace dimensions	10	50	50
Local subspace dimensions	10	10	50

Table 5.4: Classification of vehicle profile videos.

fact is reflected in the results — most of the incorrect classifications in table 5.4 were pickups; in fact, when pickups were removed from testing data (yet still included in the training set), accuracy was nearly perfect. This fact is encouraging, and the combination of an automatic shadow removal technique with the ability to accumulate evidence from multiple exemplar matches may yield results superior to those achieved with the still image database.

5.3.2 Learning-related experiments

Testing was also performed to demonstrate the ability of VEOR to learn novel views and classes at runtime. To test the former, video footage was taken in which cars followed a road containing a right-angle turn. The cars drive towards the camera, make the turn, and continue on for a short distance in profile. The test video shown in figure 5.5 consists of three sequential segmented video clips, each showing a different car driving in this manner. The challenge of this test is that the system is trained using only vehicle profiles, and must associate the previously unseen view of the front of a car with the car class. This is done so that future appearances of the front of

cars will be classified as cars.

As expected, VEOR initially classifies the first car as an instance of an unknown object; that is, an artificial class is created for it. This unknown classification persists until the car is in mid-turn, where VEOR briefly finds that it most resembles a van. When the car finally turns fully to a profile view for a short period of time, strong matches to car training samples outweigh the classifications of van and unknown object. When the second car appears in head-on view, it is exemplar matched to the newly learned images from the previous object, which the system remembers as a car. Thus, the second car is consistently classified as a car while it is in view. The same is true for the third car. It is not particularly interesting that the system changes its hypothesis about the first car based on its different views. What is more interesting is that after seeing only one such unknown object "morph" into a known one, further instances are classified correctly — whether or not those instances make the right-angle turn.

To test the bounds of this type of learning, another video was constructed consisting of the first car from the previous test, followed by two other cars whose grayscale values are significantly darker than that the first, but similar to one another. VEOR treats the first car in the same manner as in the previous test. In this case, however, the front view of the second car was not matched to that of the first, and it is initially classified as unknown. The third car benefits from the views learned from the second, and is classified as a car throughout. This demonstrates a limitation in using global PCA-based

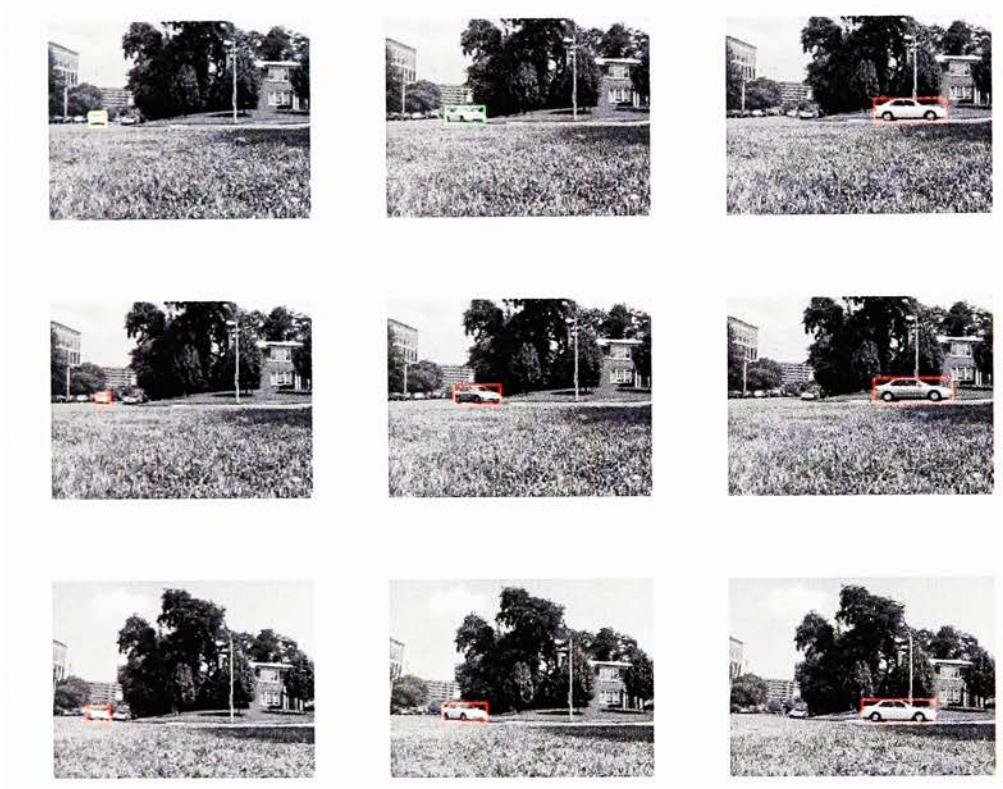

Figure 5.5: Frames from the classified output video. Each row shows a different car approaching the camera (column one), turning (column two), and continuing in profile (column three). A red bounding box indicates a "car" classification, green indicates a "van" classification, and yellow an unknown classification.

feature extraction.

To demonstrate the learning of new classes, a video was filmed in which a pedestrian walk by, followed by a car, followed by another pedestrian. VEOR was trained solely on vehicle data. As expected, VEOR classifies the first pedestrian as a member of a new (unknown) class, classifies the car correctly, and classifies the second pedestrian as a member of the same unknown class as the first.

Chapter 6

Conclusions

The EOR system is noteworthy not for the novelty of its component algorithms, but for the careful selection and arrangement of those algorithms, which yields greater performance than that achieved by the components individually. In the same fashion, VEOR approaches the problem of object classification in segmented video by incorporating algorithms suited to that data and connecting them in ways complimentary to the progression of a video sequence. In addition, the VEOR architecture allows new viewpoints and classes to be learned at runtime.

The results in chapter 5 demonstrate that different types of image data are best classified using different algorithmic variations. And, while VEOR achieves additional flexibility by removing the parameter k, it adds parameters of its own, the values of which influence one another. For instance, an increase in the number of retained global PCA subspace dimensions should

be accompanied by an increase in the thresholds that act on distances in that subspace. If techniques were found that would automatically determine good values for parameters such as these, based on the needs and properties of different applications, then the ease with which VEOR could be ported to those applications would be increased. Currently, some trial-and-error is necessary to determine appropriate values. However, VEOR provides an architecture upon which further refinements and additions can be made.

The addition of automatic shadow removal and robust object tracking may increase performance on real-world video data, as may integration into a system with a full associative memory and higher-order reasoning abilities. In the same way that algorithms may be tied together into techniques such as VEOR, these kinds of technologies should be combined in ways that exploit their individual strengths, leading to systems that are greater than the sum of their parts.

Bibliography

[1] B. Draper, K. Baek and J. Boody. "Implementing the Expert Object Recognition Pathway," *Machine Vision and Applications*, 16(1):27-32, 2004.

[2] Kyungim Baek. *An Algorithmic Implementation of Expert Object Recognition in the Ventral Visual Pathway.* Ph.D. thesis, Colorado State University, Fall 2002.

[3] B. Draper, K. Baek and J. Boody. "A Biologically Plausible Approach to Cat and Dog Discrimination," *Joint AIPR International Workshops on Structural, Syntactic and Statistical Pattern Recognition*, Windsor, Ont. Aug. 6-9, 2002. p. 779-788.

[4] B. Draper, K. Baek, M. S. Bartlett, and J. R. Beveridge. "Recognizing faces with PCA and ICA", *Computer Vision and Image Understanding*, Volume 91 , Issue 1-2, July, 2003, p. 115-137

[5] R. Gaborski, A. Vaingankar, V. Chaoji, and A. Teredesai. "VENUS: A System for Novelty Detection in Video Streams with Learning," *FLAIRS*, May, 2004.

[6] C. Stauffer and W. E. L. Grimson. "Adaptive Background Mixture Models for Real-Time Tracking", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Volume 2, No. 2, p. 2246, 1999.

[7] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data Clustering: A review", *ACM Comput. Surveys, 31* (1999), p. 264-323.

[8] N. R. Pal, J. C. Bezdek. "On cluster validity for the fuzzy c-means model", *Fuzzy Systems, IEEE Transactions on*, Volume 3, Issue 3, Aug. 1995. p. 370-379.

[9] J. C. Bezdek and N. R. Pal. "Some new indexes of cluster validity", *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, Volume 28, Issue 3, June 1998. p. 301-315.

[10] I. Gath and A. B. Geva. "Unsupervised Optimal Fuzzy Clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 11, Issue 7, July 1989. pp. 773-780.

[11] G. Hamerly and C. Elkan. "Alternatives to the k-means algorithm that find better clusterings", *Proceedings of the Eleventh International*

Conference on Information and Knowledge Management, McLean, VA, ACM Press. 2002. pp. 600-607.

[12] G. Hamerly and C. Elkan. "Learning the k in k-means", Technical Report CS2002-0716, University of California, San Diego, 2002.

[13] Bin Zhang. *Generalized K-Harmonic Means — Dynamic Weighting of Data in Unsupervised Learning*, Technical Report HPL-2000-137, Hewlett-Packard Labs, 2000.

[14] Isabelle Bloch. "Fuzzy sets in image processing", *Proceedings of the 1994 ACM symposium on Applied Computing*, p.175-179, 1994.

[15] C. Ding and X. He. "K-means Clustering via Principal Component Analysis", *Proceedings of the 21st International Conference on Machine Learning,* ICML, 2004.

[16] T. B. Moeslund. *Principal Component Analysis - An Introduction*, Technical Report CVMT 01-02, ISSN 0906-6233, 2001.

[17] Larry Hatcher. *A Step-by-Step Approach to Using SAS for Factor Analysis and Structural Equation Modeling*, SAS Press, 1994.

[18] L. I. Smith. *A tutorial on Principal Component Analysis*, Term Project, University of Otago. February, 2002.

[19] R. Gonzalez, R. Woods, and S. Eddins. *Digital Image Processing Using Matlab*, Prentice-Hall, 2003.