Modeling the Temporal Components of Video Structure

by

Sailabala Challapalli

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 23, 2001

Author _____
Department of Electrical Engineering and Computer Science
May 23, 2001

Certified by _____
Andrew Lippman
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Modeling the Temporal Components of Video Structure
by
Sailabala Challapalli

Submitted to the
Department of Electrical Engineering and Computer Science

May 23, 2001

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Engineering and Electrical Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

This thesis presents a computational framework for parsing video based on the notion of "temporal units." Temporal units are formally defined as collections of frames under a common theme, e.g. a shot, a dialogue, or a football play. Computational procedures are devised to 1) identify the boundaries of certain types of temporal units and 2) cluster them into hierarchies that reflect the temporal structure of the content. It is shown that, in addition to enabling new modes of interaction with video, the resulting representation can significantly improve the efficiency of search and retrieval operations. Interesting applications of the new parsing technology are demonstrated on Minerva, a system that suggests cooking recipes and provides users with video and textual instructions about cooking a selected recipe. Minerva relies on object recognition technology to identify ingredients that users want to use and, based on their preferences, returns recipes that might interest them. Video parsing technology enables users to control and interact with the video while following the instructions.

Table of Contents

Chapter 1: Introduction

The media and networking group at the Media Laboratory and others construct audio and video systems that map the delivery of real-time entertainment information onto the Internet. The vision is to make television a global resource. By distributing programming to anywhere and at anytime, television can expand to become more than an interaction between one family and its television set.

An important problem that bears on this vision is the selection and searching mechanism. As more options become available, the selection of one that is of interest to a viewer or group of viewers becomes increasingly complicated. Various schemes have been attempted to provide a framework to facilitate the navigation through this space, including the use of autonomous agents (Maes, etc), and elaborate program guides (TiVo, etc). In this thesis, image similarity analysis is explored as an underlying technology to present options in an interactive, activity-based manner to iteratively guide a viewer through a visual programming space. Adding the notion of "temporal units" to existing search systems will facilitate the automatic parsing of moving images into frames, shots and scenes (defined more precisely below). Being able to identify these different components of video will allow viewers to search through video in an efficient and enjoyable manner. For example, viewers may be able to extract all the plays that a particular player is involved in a football game or find all the scenes that take place in the coffee shop of different "Friends" episodes.

1.1 Defining a Temporal Unit

A temporal unit consists of a frame or a set of frames that are linked by a common characteristic such as depicting a sequence, containing the same subject, or even illustrating a common theme. The basic algorithm used by this video parser first breaks a video down into its various temporal units and, using similarity matching, merges similar sections in a bottom up fashion. The goal is to identify similarities among the different components of the video and group similar parts together. Then, viewers can retrieve particular segments of a video that pertain to a particular subject rather than trying to search through the entire video themselves. These groups can then be compared and again clustered together to form a higher level grouping, thus creating a hierarchical structure for a video.

The most basic temporal unit is the individual frame. The next higher level is a shot composed of a sequence of frames from an individual camera without harsh transitions (such as switching to another camera). A scene is defined as a series of shots that pertain to the same theme. For example, if the video deals with a conversation between two people and involves switching cameras back and forth between the two people, a new shot begins every time there is a transition to a different camera but the whole exchange is classified as one scene. Although, these basic building blocks can be combined into higher-level temporal units, this thesis will only deal with clustering images at the shot and scene level. Solving this parsing problem involves three main steps: finding a compact frame representation, identifying the boundaries of each temporal unit, and using similarity functions to match units at the same level and to cluster them into higher level temporal units.

1.2 Modeling Temporal Units

The path chosen to solve this problem involves statistical modeling of the images by mapping their features into probability density functions. An image's features are measurements that allow a system to discriminate among other images with a different content. Probability densities can be compared using Bayesian techniques and maximum likelihood ratios are used to determine the likelihood that a particular set of features from a query image is similar to the set of features from a database of images. In this way, frames from the same temporal unit can be identified because their features will return a higher probability of being similar. Transition points can be recognized when consecutive frames are found to have a relatively small probability of being from the same temporal unit. Finally, to create representations for higher order temporal units, frames and their corresponding density functions from a lower level temporal unit can be clustered. This will result in a more general representation of the feature information stored in the densities of the lower level.

The Gaussian Mixture model has been shown to be an appropriate probabilistic representation for the feature space [1]. This model approximates the underlying feature density by creating a weighted sum of individual Gaussians. When dealing with a high dimensional space and multimodal densities, this model proves more effective than previously proposed solutions because it is less expensive computationally than the histogram and more accurate than the Gaussian.

To cluster frames at a given temporal level into higher order temporal units, it is necessary to calculate a representation, or probability density function, that describes the all the frames with enough granularity to explain their important features. In this thesis,

we analyze how the hierarchical mixture model, introduced by Vasconcelos can be used to perform such clustering of temporal units [2]. This model utilizes the Expectation-Maximization (EM) Algorithm and extends its capabilities to estimate the necessary parameters in the density function of the clustered features.

1.3 Applying the Concepts

The resulting higher order temporal units can be utilized in several different ways. Some applications include improvement in performance of image similarity matching and to facilitate navigation between different temporal units after parsing a video. These two applications can be seen in the implementation of the Minerva system.[1] The Minerva system is an innovative approach to help everyday households decide what's for dinner. Users place ingredients that they have, and feel like using for their meal, on a countertop monitored by the system. Minerva recognizes the ingredients given and suggests recipes that might be of interest to the user based on his stored profile. The user can then choose the most appealing recipe to view video instructions of how to prepare the dish. The hierarchical mixture models are utilized to improve the initial ingredient recognition. Use of the hierarchical mixture models allows the system to create a general representation for the various ingredients in its database. Therefore, it queries an unknown ingredient with these general densities rather than using previously proposed techniques. Furthermore, image similarity matching techniques are utilized to allow users to navigate through the video instruction in a "smart" manner. Shots of similar content can be identified and stored so that the video fast-forwards or rewinds to the next

[1] Application developed by Shyam Krishnamoorthy, Sailabala Challapalli and Bryan Ly in the Digital Life Group of the Media Laboratory of the Massachusetts Institute of Technology under the supervision of Professor Andrew Lippman.

shot rather than frame by frame. Also if the user pauses the video in the middle of a shot, the system resumes play at the beginning of the shot, thus providing some continuity to the presentation of the instruction.

Chapter 2: Image Representation

The first step to providing a framework to the search and selection mechanism is to characterize images comprising a video in a sufficient manner as to facilitate discrimination among them. Because it is impractical to simply compare all of the pixels in one image to another, it is necessary to transform the image pixel space to a feature space that allows more efficient characterization. Once the image is transformed into a feature space, a feature representation must be found that describes how each image populates that feature space. It is important to note that these representations must be compact because the complexity of evaluating image similarity directly impacts the viability of a retrieval system. Much work has been dedicated to finding such a representation and several researchers have shown that statistical modeling of an image's features is more effective than the straightforward comparison of pixels [1]. The basic algorithm involves mapping each image into a feature space and then finding a viable feature representation. The final step of the search and selection mechanism is then to find an appropriate retrieval measure that utilizes this compact feature representation to assess the similarity among different images.

2.1 Features

An important step in the overall process is the mapping of an image adequately from the pixel space into a feature space. Much research has been devoted to identifying correct features for various types of images such as texture databases, object databases and even databases used for face recognition. However, because a video parsing system should be able to deal with all kinds of images, a map that that produces generic rather

than domain specific features is required. Although it is impractical to hope for a perfect generic feature selection algorithm, one that provides a good approximation is sufficient when combined with the appropriate feature representation and retrieval measures. One such algorithm is to use the coefficients from the Discrete Cosine Transform (DCT) [1]. This resulting framework is general and applies to all types of images.

The choice of DCT features has several interesting properties. First, DCT has good decorrelating properties that result in a relatively small number of parameters required to estimate the feature representation and a reduction in the complexity of the similarity function. Second, the frequency decomposition of the coefficients gives a natural order from coarse to detailed information that can be used for implementing multistage retrieval strategies, again reducing the computational cost of retrieval. Finally, experiments with various databases have shown that when combined with the appropriate feature representation, the DCT features can perform as well as domain specific features, even in the domains for which these features were designed. [2]

2.2 Feature Representation

In the past, the two most common statistical feature representations have involved either characterizing features by their moments, such as their mean and covariance, or by computing image histograms. The first method has been used typically when dealing with texture recognition while histograms are more popular for object recognition and image retrieval. Analyzing relatively homogeneous texture patches, such as those found in the standard Brodatz[3] database, generally results in the unimodal distribution of

[2] For a more detailed discussion of the advantages associated with the choice of DCT coefficients as features, see [1].
[3] See Appendix A for sample images from Brodatz database.

features that is well approximated by a Gaussian density characterized by the feature mean and covariance. The Gaussian density is able to handle the high dimensional nature of the feature space necessary to capture the spatial dependencies that characterize texture. On the other hand, the histogram model fares better when dealing with the non-homogeneous images found in typical image retrieval or object recognition databases, such as the Columbia or Corel[4] databases [1]. The multimodal nature of these images cannot be described accurately by simply calculating a mean and covariance, and, therefore, it precludes the use of single Gaussians to describe such images' features. However, because the complexity of the histogram model increases exponentially with the dimension of the feature space, the histogram cannot be used to characterize all types of images. Although it performs well when dealing with low dimensional spaces, such as pixel colors of images found in object recognition databases, the histogram model fails when applied to the high dimensional spaces of images found in texture databases [3]. Figure 1 shows examples of two images that have the same histogram but are visually distinct[5]. The histogram fails because it cannot detect where the light and dark pixels are relative to each other, only that the sum of the light and dark pixels is the same for both images. A model that characterizes the higher dimensions of the feature space would be able to recognize these spatial densities.

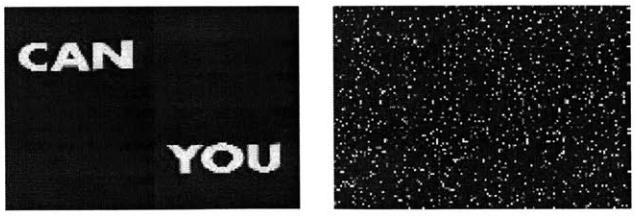


Figure 1: Two visually distinct images that have the same histogram.

[4] See Appendix A for sample images from Columbia and Corel databases.
[5] [3] Figure 2

The constraints on these two popular representations led to the continued effort in finding a better representation that can perform well for all types of images. In short, the new representation must capture the multimodal nature of generic feature distributions and also practically deal with high dimensional spaces. One model proposed by Nuno Vasconcelos is the Gaussian mixture model that is essentially a weighted sum of individual Gaussian densities. As shown in Section 3.2, the Gaussian mixture model provides a bridge between the standard Gaussian and the histogram models by satisfying both the conditions for a better representation [1].

2.2.1 Standard Gaussian Model, Histogram Model, and the Gaussian Mixture Model

To better understand why the Gaussian mixture model is a better representation than the standard Gaussian or the histogram model, a more rigorous analysis of the three representations is necessary. By examining the tradeoff between complexity and accuracy as applied to multimodal densities and high dimensional spaces, it becomes clear that the Gaussian mixture model offers the best solution.

The standard Gaussian model characterizes features according to their sample mean and covariance. Its feature probability density function takes the following form:

$$P(\mathbf{x}|S_i) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma^{-1}(\mathbf{x}-\mu_i)} \qquad (1)$$

where P($\mathbf{x}$| S_i) is the likelihood that the query $\mathbf{x}$ is from the source S_i, μ_i is the mean for class i, and Σ_i is the covariance for class i.[6]

[6] [1] Equation 9

The histogram model quantizes the feature space into hyperrectangular bins and then counts the number of feature points that fall into each of these bins. The feature probability density for a feature space with a dimension n and bins the size of $h_1 \times \ldots \times h_n$ takes on the following form[7]:

$$P(\mathbf{x}|S_i) = \sum_k \frac{f_k}{F} K(\mathbf{x} - \mathbf{C}_k) \tag{2}$$

where k is the bin, f_k is the number of features that fall on bin k, $\mathbf{C}_k$ is the central point of bin k, F is the total number of features, and $K(\mathrm{x})$ is a box function of the form:

$$K(\mathbf{x}) = \begin{cases} \dfrac{1}{h_1 \times \ldots \times h_n} & \text{if } |\mathbf{x}_1| < \dfrac{h_1}{2}, \ldots, |\mathbf{x}_n| < \dfrac{h_n}{2} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

For a constant h, the number of bins grows exponentially with the dimension of the feature space. Therefore, although it is computationally inexpensive when dealing with low dimensions, this model becomes impractical to implement at higher dimensions such as those required by images in texture databases.

The mixture model, on the other hand, introduces the concept of a weighted sum of class-conditional densities, weighted by the respective class densities. The feature probability density function is then[8]:

$$P(\mathbf{x}) = \sum_{i=1}^{C} P(\mathbf{x}|w_i) P(w_i) \tag{4}$$

where C is the number of classes, $\{P(\mathbf{x}|w_i)\}_{i=1}^{C}$ is the sequence of class-conditional densities, and $\{P(w_i)\}_{i=1}^{C}$ is the sequence of class probabilities [4]. It is apparent that the

[7] [1] Equation 7
[8] [4] Equation 5

standard Gaussian model is a particular case of the mixture model where C = 1 and $P(x|w_i)$ is the Gaussian density shown in Equation 1.

As described above, the standard Gaussian model was useful in analyzing texture databases, but ineffective in typical object retrieval problems because its simplistic nature failed to adequately model the multimodal nature of the feature spaces. Figure 2 offers a qualitative explanation of how the Gaussian model can lead to high classification error due to the great deal of overlap in class densities[9]. While the histogram performed better in the hypothetical object retrieval problem by returning a significantly better approximation to the class densities, the exponentially growing complexity with dimension does not allow its use when it is important to model spatial dependencies, as is the case with texture databases.

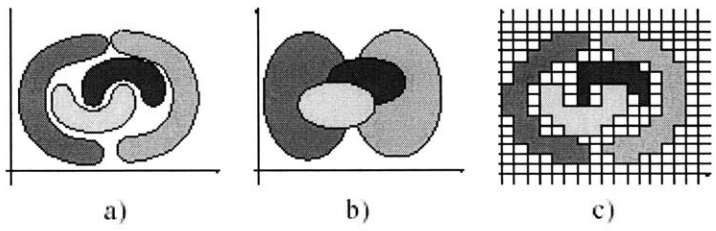


Figure 2: The results of a hypothetical two-dimensional retrieval problem with four image classes. Figure a) represents the four class densities, b) represents the best Gaussian fit to each class density and c) shows the histogram fits to each class density.

The best representation among these three options, therefore, proves to be the Gaussian mixture model that deals with the shortcomings of the standard Gaussian and the histogram model. It utilizes a weighted sum of the simpler Gaussian densities and takes the form[10]

---

[9] [3] Figure 1
[10] [1] Equation 8

$$P(\mathbf{x}|S_i) = \sum_{c=1}^{C} \pi_i G(\mathbf{x}, \mu_i, \Sigma_i) \tag{5}$$

where C represents the number of underlying classes in the data, π_i is the probability of class i, and $G(\mathbf{x}, \mu_i, \Sigma_i)$ is the probability density function for the Gaussian shown by Equation 1.[11] The weights allow the Gaussian mixture model to place densities in the populated regions of the feature space to create a more efficient and less complex model than the histogram model.

The Expectation-Maximization (EM) algorithm provides a standard method of computing the model parameters such as the weights, mean and covariance of each Gaussian [5]. This algorithm alternates between the E-step and the M-step. The E-step calculates the posterior probabilities of the mixture components given the data and current parameter estimates while the M-step updates these estimates. For Gaussian distributions where all the classes are all a priori equally likely:[12]

E-step:
$$h_i^m = P(\omega_i \mid \mathbf{x}^m) = \frac{P(\mathbf{x}^m \mid \omega_i) p_i}{\sum_{k=1}^{N} P(\mathbf{x}^m \mid \omega_k) p_k} \tag{6}$$

M-step:
$$\mu_i^{new} = \frac{\sum_m h_i^m \mathbf{x}^m}{\sum_m h_i^m} \tag{7}$$

$$\Sigma_i^{new} = \frac{\sum_m h_i^m \left(\mathbf{x}^m - \mu_i^{new}\right)\left(\mathbf{x}^m - \mu_i^{new}\right)^T}{\sum_m h_i^m}$$

$$p_i^{new} = \frac{1}{M} \sum_m h_i^m$$

[11] [1] Equation 9
[12] [5] Equations 4, 5

where h_i^m is the posterior probability that the m^{th} sample belongs to class i. In short, the Gaussian mixture model provides the ability to model multimodal densities for high dimensional feature spaces using much more efficient and less complex methods.

Chapter 3: Image Similarity

Given a suitable feature representation, the search and selection mechanism requires an appropriate function to judge similarity among images. In general, systems have used Euclidean distances or second order distance metrics, such as the Mahanolabis distance, to determine image similarity [1]. Deterministic similarity metrics are used because they seem to satisfy the criteria for a true metric: they are always positive, symmetric and satisfy the triangle inequality. However, recent work has been dedicated to discovering similarity functions that can fully exploit the accurate statistical descriptions discussed in the previous chapter. Like the feature representations, these functions are probabilistic in nature.

3.1 Using Bayesian Inference

One possible solution is to view the retrieval problem as one of Bayesian inference. The goal is to find an optimal map g from the set of feature vectors $\mathbf{x}$ to the image class y that minimizes the probability of error. This probability is defined as the probability that when faced with a query from class y, the system returns images from class $g(\mathbf{x})$ different from class y. From decision theory, it is well known that the optimal map is to select the class with the largest posterior probability:

$$g^*(\mathbf{x}) = \arg\max_i P(y = i \mid \mathbf{x}) \tag{8}$$

By applying Bayes rule:[13]

$$g^*(\mathbf{x}) = \arg\max_i P(\mathbf{x} \mid y = i)P(y = i) \tag{9}$$

[13] [4] Equations 1, 2

where P(x| $y = i$) is the feature representation of features from the class i and P($y = i$) is the prior probability for class i.

There are several advantages of using such a retrieval framework.[14] One interesting property is that the P($y = i$) term allows the system to incorporate prior knowledge about the relevance of a particular class to the query during the retrieval process. Therefore, if certain classes of object are more likely to be utilized then they can be weighted accordingly. For example, the Minerva system could be designed to incorporate users' preferences in its image recognition phase. If Minerva knows that user A hates tomatoes but loves apples, then when a user places an object that could be either a tomato or an orange, the system will return that the ingredient is an orange. However, in this thesis, all classes are considered to be a priori equally likely. As a result, Equation 9 can be simplified to the standard maximum-likelihood (ML) classifier[15]

$$g(\mathbf{x}) = \arg\max_i P(\mathbf{x} \mid y = i), \tag{10}$$

which, when querying N independent query features, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, reduces to[16]

$$g(\mathbf{X}) = \arg\max_i \frac{1}{N} \sum_{j=1}^{N} \log P(\mathbf{x}_j \mid y = i) \tag{11}$$

To understand the advantages of the ML classifier, we analyze the simple Gaussian model, for which Equation 11 reduces to[17]

$$g(\mathbf{X}) = \arg\min_i \log|\Sigma_i| + trace\left[\Sigma_i^{-1} \hat{\Sigma}_{\mathbf{X}}\right] + \left(\hat{\mathbf{x}} - \mu_i\right)^T \Sigma_i^{-1} \left(\hat{\mathbf{x}} - \mu_i\right)^T \tag{12}$$

[14] For a more detailed discussion of the advantages to using a Bayesian inference retrieval system see [1].
[15] [3] Equation 3
[16] [3] Equation 4
[17] [3] Equation 6

The variables $\hat{\mathbf{x}}$ and $\hat{\Sigma}_{\mathbf{X}}$ are the sample mean and covariance of $\mathbf{X}$, and μ_i and Σ_i represent the sample mean and covariance of $P(\mathbf{x} \mid y = i)$. This mapping essentially finds the Mahalanobis distance, the third term in Equation 12, augmented by the first two terms, and this distance is used to measure the differences between the density of the query feature space and the class feature space. As a result, the system determines the most probable match by finding the class that minimizes these augmented distances.

These terms have been shown to have crucial importance for handling changes in scale and orientation of the query density. The metric in Equation 12 is, therefore, much more robust than the Mahalanobis distance. Furthermore, the probability of error of Equation 12 tends to the probability of error of the Bayes classifier orders of magnitude faster than the density estimates tend to the accurate densities. This characteristic allows the use of coarser estimates than one would use with other algorithms, such as the histogram intersection method, to return just as accurate results. Because density estimation is so complex, using ML criteria can again lead to significant improvements in retrieval accuracy. Although the above equation applies to the use of a single Gaussian, it can be generalized to full mixtures of Gaussians.[18]

[18] [6] Definition 5.

Chapter 4: Temporal Units

Given a framework to represent images and determine their similarities, we next investigate how these tools can be applied to characterizing video. Video is obviously a collection of individual frames viewed in a consecutive manner. It contains varying levels of structure ranging from an entire clip being shot from the same angle to movies containing scenes of different format, length and content. Therefore, in order to provide a mechanism to navigate through video, these various temporal components must be identified and modeled.

4.1 Definitions

The first step to modeling temporal units involves defining some of the different levels that may exist. Although the idea of a temporal unit can vary greatly depending on the context, there are some basic terms that can be defined. The most basic element is the individual "frame." The next level, the "shot," consists of all the consecutive frames dealing with a particular subject filmed by the same camera. This may involve panning or zooming in and out, but the same camera is filming the subject continuously during this period. The subsequent higher level is defined as the "scene." A scene consists of a series of shots that pertain to the same theme, such as a conversation between two people. Because it is a series of shots, a scene may include cameras switching back and forth between different subjects as long as they obey some patterns and pertain to the same topic. These are some of the basic terms that can apply to all kinds of video, but different types of video can be characterized in other manners to better suit the content. For

example, it would be more useful to characterize a football game based on temporal units ranging from downs and possessions to quarters and halves rather than shots and scenes.

4.2 Parsing Video

Breaking a video into temporal units that can be retrieved as desired requires the application of the algorithms discussed in the previous two chapters. Each temporal segment must be transformed into appropriate representations that can be evaluated using similarity functions. These functions will then be used to identify the appropriate segments that a user is searching for. In addition to retrieving desired clips from movies, the similarity measures can also be used to identify segments that pertain to the same theme or are part of the same temporal unit by locating transitions to a new temporal unit. All of the segments in between transitions can then be clustered to form more general representations for higher-level temporal units using hierarchical statistical modeling. The building of models for the differing levels of temporal units is best achieved when proceeding in a bottom-up fashion. Starting with the bottom level, individual frames are statistically modeled using the Gaussian mixture densities described in Chapter 3. These densities can then be compared using the Bayesian inference algorithms discussed in Chapter 4 to determine which frames belong to the same shot or when a transition has occurred. Transition identification will be discussed in greater detail below in Section 5.1. The next step is to then cluster the frames from each particular shot and create a mixture model that describes the content contained in all the frames. This step can be performed using the Hierarchical mixture model described in Section 5.2. The video can be modeled at higher-level temporal units by repeating the steps dealing with identifying

transitions and clustering, although the implementation of these steps varies with each level.

4.3 Identifying Transitions

A transition to a new temporal unit depends primarily on the level of the temporal unit. This thesis focuses primarily on the identification of transitions between shots and addresses some of the issues that arise when trying to implement theory. The procedure for finding transitions between scenes will also be outlined. Once the demarcations between different temporal units have been determined, the video can then be indexed according to the different units it contains. This form of indexing will facilitate the retrieval of entire units and the clustering of different segments into higher order temporal units.

4.3.1 Shots

As stated above, a new shot occurs when there is a harsh transition between two consecutive frames, such as changing the camera being used to shoot a particular subject, changing the angle that it is shot or changing the subjects themselves. Therefore, determining when transitions occur between shots involves comparing the representations of consecutive frames. If the densities are similar within some threshold value, then it can be inferred that no harsh transition has occurred. Although consecutive frames will not be identical, these images have smooth changes in such characteristics as background, lighting, and orientation of objects. Therefore, the similarity measure will return a relatively low value. These frames can then be clustered together as part of the same

shot. Once the system detects a harsh transition in the feature space, a flag will indicate that a transition to a new shot has occurred.

The algorithm in determining shot transitions seems straightforward. However, choosing the right similarity function and threshold value proves challenging because of the many different kinds of transitions that can occur in a video. Frames can change quickly and harshly to indicate a change (Figure 3), they can fade from one shot into the new shot (Figure 4), or special graphics can be used that indicate a more gradual transition between shots (Figure 5).[19] These types of shot transitions occur in varying degrees depending on the type of video that is being processed. For example, a movie will contain more of the special effect transitions and fades than a video of a football game does. Finding a universal measure that robustly identifies the three types of transitions is a challenging question that is beyond the scope of this thesis.

Figure 3: Series of frames that represent a "harsh" transition.

Figure 4: Progression of frames that represents a "fade" transition.

[19] All frames from Figures 3-5 are captured from [7].

Figure 5: An example of using a special transition.

4.3.1.1 Identifying Parameters

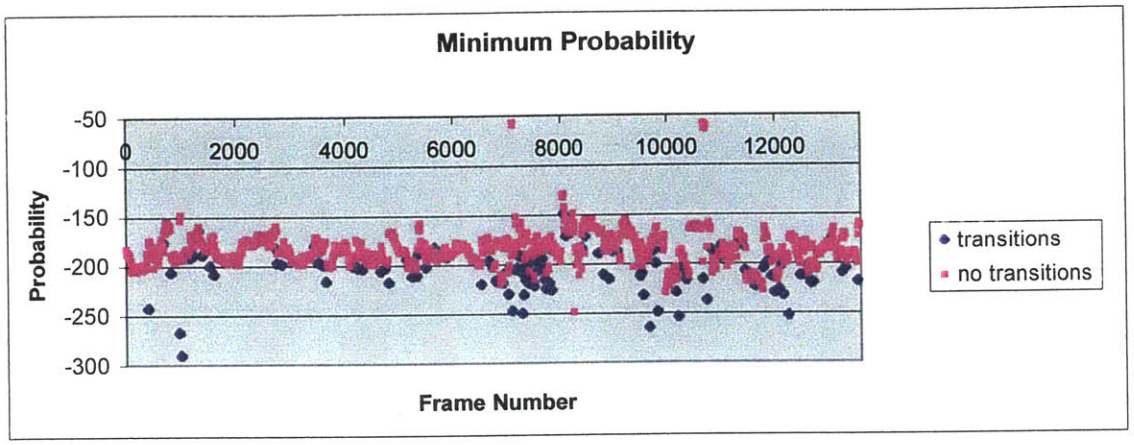
As a result, in the search for finding a useful similarity function between consecutive frames, several variables were experimented with.[20] The first decision focused on which function to use, where the choices included calculating the correlation between consecutive frames or finding the likelihood probability that consecutive frames were similar. The next decision involved determining the number of dimensions to look at when frames were compared. The final variable was the threshold value. If the function value returned when comparing consecutive frames was below this threshold value, then the second frame was considered a transition point and part of the next shot.

4.3.1.1.1 Finding a Similarity Function

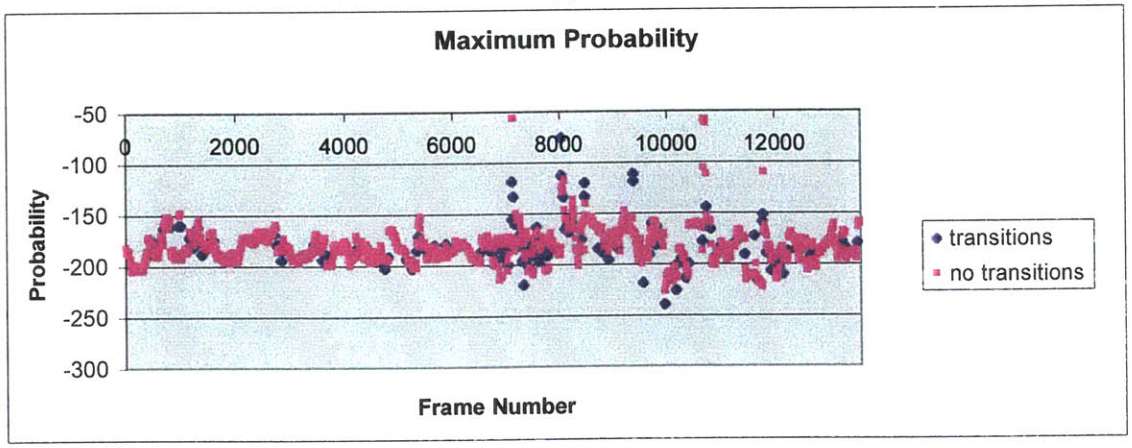
The likelihood function can be used as a measure of the similarity between consecutive frames in several different ways. This measure is used to return the likelihood that the mixture model of a "query" image is similar to the mixture model of a "database" or set of images that the system is trained on. Therefore, depending on which frame is classified as the "query" image and which image is classified as the "database" (or trained) image, there will be a different likelihood returned that the two images are

[20] All experiments were conducted on frames extracted from a cooking show clip that teaches a viewer how to make a bean casserole dish [7]. The original video is composed of on average 30 frames per second, and every 7th frame was extracted to use in the experiments. The resulting set of frames contains 4 frames per second of video, and adequately represents the entire video. It is important to note that when referring to

similar. Therefore, the function must be evaluated when the images take turns being the query image and the database image. At this point, either the minimum, maximum, or average likelihood probability returned when the two images are interchanged can be used as a measure. The following plots in Figure 6 show the results of using these functions to identify the shot transitions.

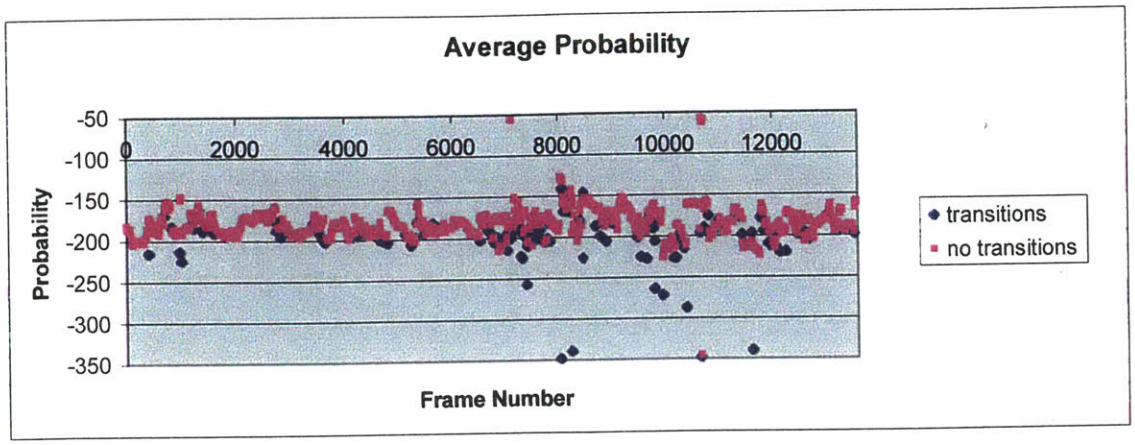


(a)



(b)

"consecutive frames," the frames being compared are in reality a quarter of a second apart in the original video.



(c)

Figure 6: These plots show the minimum, maximum and average likelihood probabilities returned for consecutive frames. The value for a frame number x is the likelihood probability that frame x and the previous frame (from the set of sampled frames) are similar. The likelihoods are plotted separately for frames where a transition should have been identified and where no transition actually occurs.

Using correlation to measure the similarity between feature representations is an alternate method. It is a symmetric measure and correlation coefficient between two vectors **X** and **Y** can be calculated by

$$\rho_{xy} = \frac{\mu_{XY} - \mu_X \mu_Y}{\sigma_X \sigma_Y} \qquad (13)$$

where σ_X and σ_Y are the standard deviations of **X** and **Y** respectively. For general image retrieval systems, similarity functions utilizing likelihood ratios as described in Chapter 4 to provide better performance when detecting image similarity. This class of similarity functions is more robust to changes in image characteristics such as scale and rotation. Furthermore, when images are mapped into feature spaces containing a high number of dimensions, the points in these higher dimensions are more sparsely located. The great deal of noise that can result in the higher dimensions require the use of a more robust search criteria such as the likelihoods in the Bayesian inference algorithm. Therefore, when comparing images of the same subject but with minor changes in these attributes,

the image retrieval systems return more accurate matches especially when dealing with mixtures containing higher dimensions. However, when evaluating consecutive frames in a video, alterations in the frame such as the scale of the objects and their rotation are precisely what the system is looking for. Therefore, using a measure such as correlation that is more sensitive to these changes might provide better performance in this application. Figure 7 shows the result of using the correlation measure on the same video clip as the one used with the likelihood function.

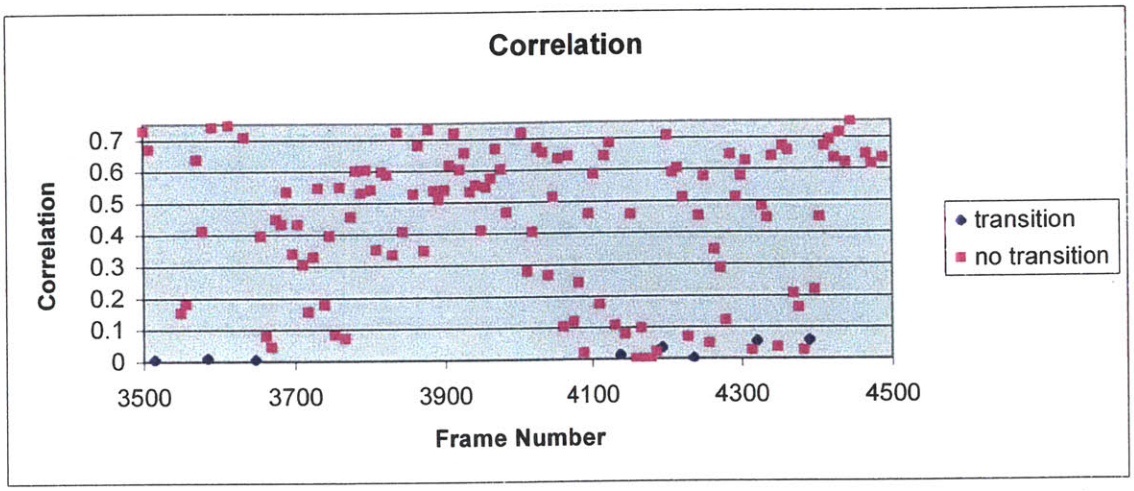


Figure 7: This plot shows the results of identifying shot transitions using correlation as a metric. Again, point (x,y) symbolizes the correlation between frame x and the previous frame. The "transition" plot shows the correlation calculated for the pairs where a transition actually occurred and the "no transition" plot shows the correlation calculated for all other pairs.

Upon examination of the transition probabilities returned when using the likelihood and correlation as similarity functions, correlation does prove to be a more reliable function. Figures 6 and 7 show the relationships between the two functions and frames examined. The plots show that there is a great deal of variance in both the functions. However, variation is more pronounced when likelihoods are used as the similarity function because these vary greatly when there is no transition and when there is a transition. Furthermore, because likelihood is a relative measure of how two frames

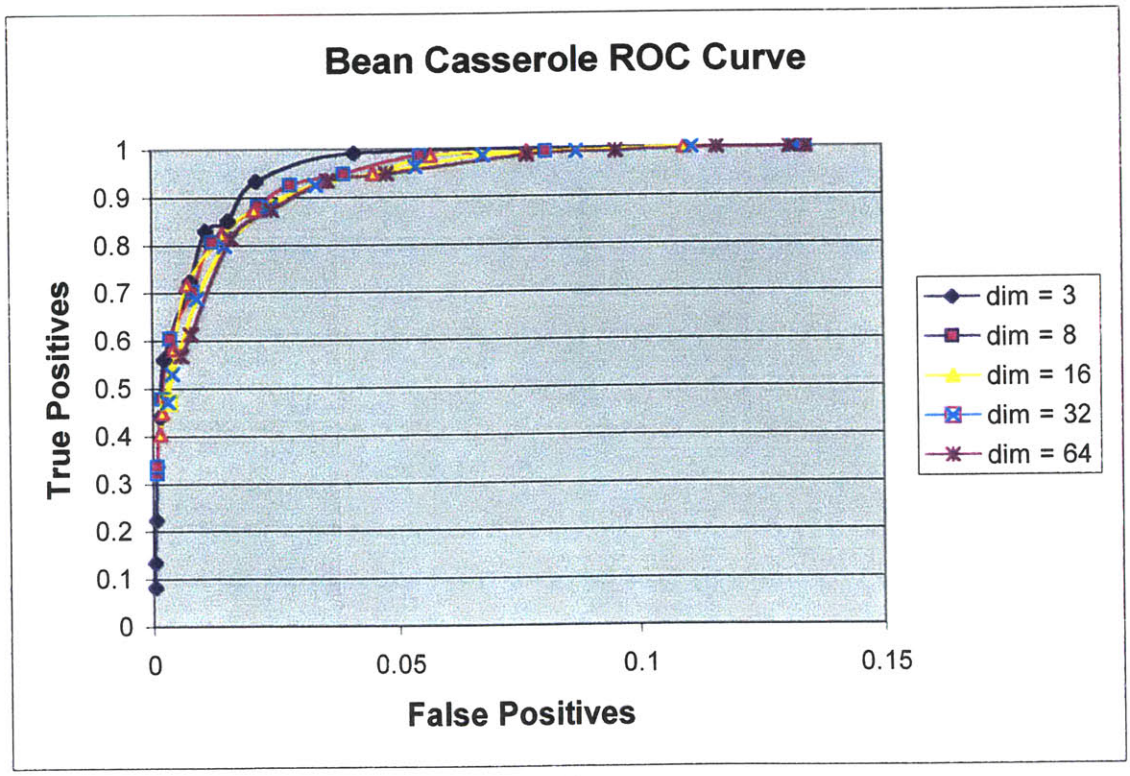
may be similar, it cannot be used on an absolute scale. As a result, it is difficult to find a general threshold value that would return consistent results. Correlation, on the other hand, is an absolute measure that is calculated by directly applying the formula to the two Gaussian mixture models rather than depending on Bayesian inference techniques. Although the correlation varies greatly when no transition occurs, the variation is relatively low for the frames where a transition does occur. Therefore, finding a general cutoff point seems more feasible when using the correlation as a similarity measure.

4.3.1.1.2 Determining Dimension of Matching and Threshold

The next step is to determine how many dimensions to include when performing the similarity matching. As described before, the number of dimensions is important in detecting the spatial dependencies existing between image features and creating robust search criteria that takes these characteristics into consideration. The search criteria developed by Vasconcelos typically uses the first 64 dimensions of each mixture model. Although individual mixture models contain many more dimensions than 64, the noise contained in the higher dimensions actually degrades the search performance rather than improve it. However, because of the nature of the similarity matching and the types of differences that the system is looking for even 64 dimensions may prove to be too many dimensions when applied to the task of shot detection. Similarly to why correlation proved to be a better similarity function, using fewer dimensions might detect the appropriate changes more effectively.

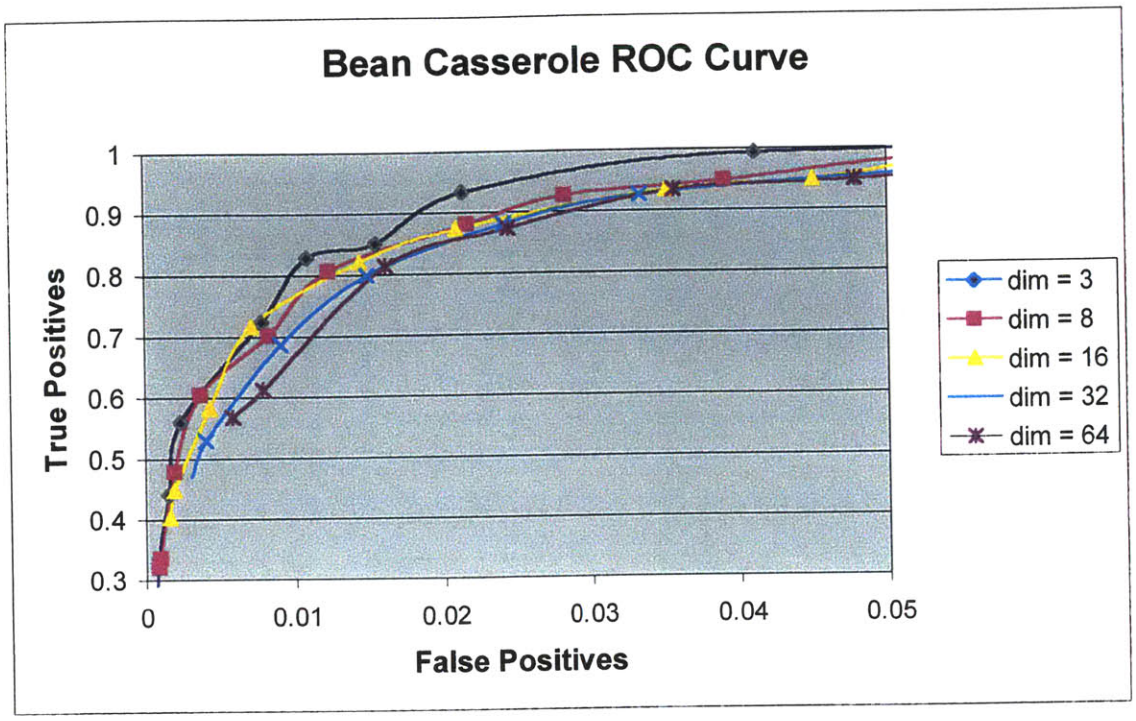
The last variable necessary to build a shot detection algorithm is the threshold at which a frame is classified a transition. If the correlation between two consecutive

frames is less than this threshold, then the system marks second frame of the pair as a transition point and the beginning of the next shot. Finding an appropriate correlation meant trading off between the number of false positives and the number of false negatives that the system would detect. The performance was tested on different dimensions with different threshold values. The number of dimensions tested range from 3 , analogous to using a simple color histogram, to 64, while the threshold values range from 0.005 to 1[21]. The results are shown in the Receiver Operating Characteristic (ROC) plot of Figure 8.



(a)

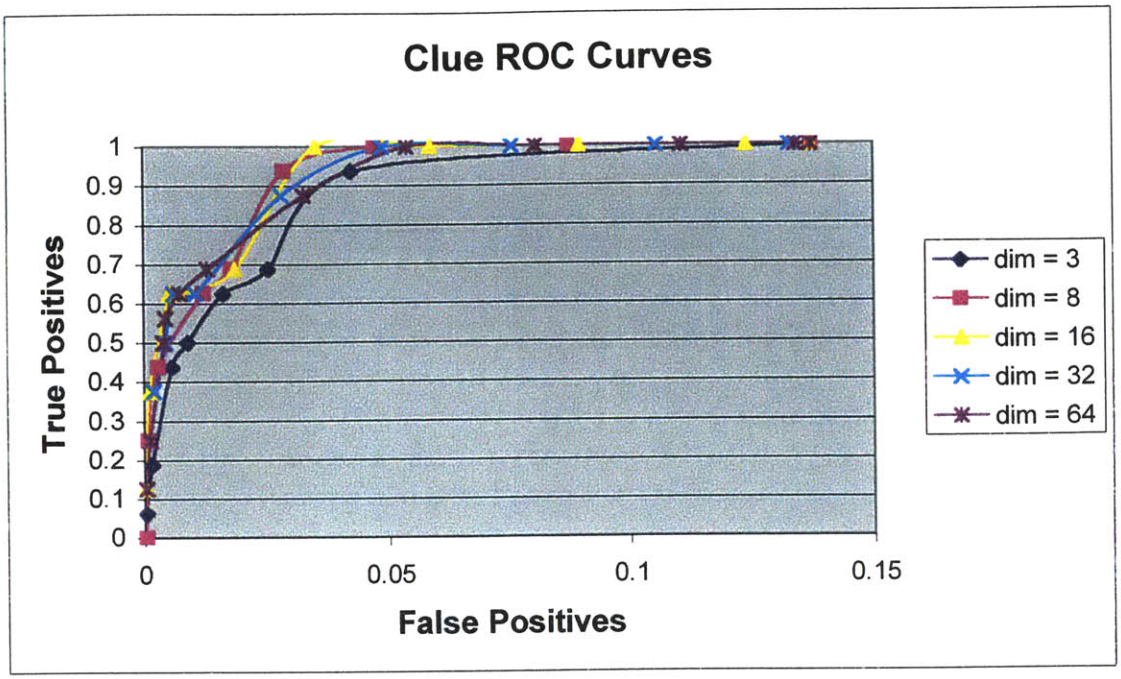
[21] The dimensions tested were 3, 8, 16, 32 and 64. The threshold values tested were 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0.



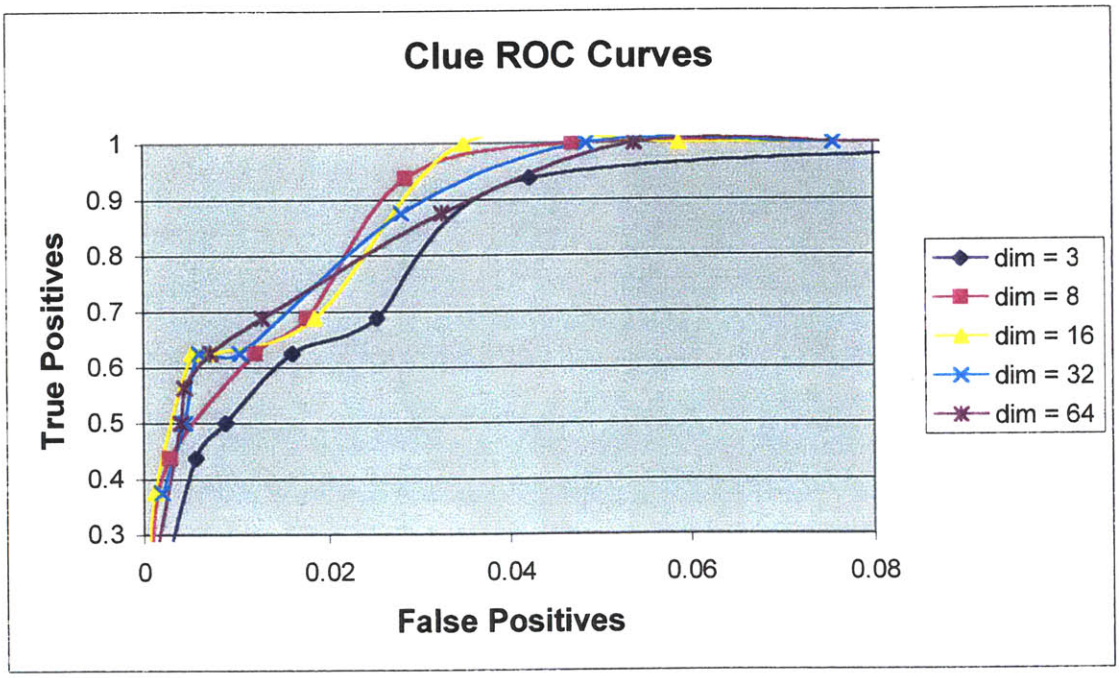
(b)

Figure 8: (a) shows the region of convergence plots for the bean casserole video clip. It contains the ROC curves for different dimensions so that the effect of dimension size can be observed. It is interesting to note the steep initial rise for all dimensions. (b) shows the interesting portion of the original ROC curve.

The above plots show that using 3, 8 and 16 dimensions result in a similar tradeoff between true positives and false positives. To analyze which dimension would return the best for all types of video, a video clip from the movie "Clue" was also explored. As stated above, different types of videos contain different types of transitions in varying amounts. The clip from the cooking show contains many shots that fade into each other interspersed with harsh transitions. The clip from "Clue," on the other hand, is characterized by harsh transitions. Therefore, it is expected for the system to perform better with this video clip. The ROC curve for "Clue" is shown below in Figure 9.



(a)



(b)

Figure 9: (a) shows the overall ROC plot while (b) again shows the initial rise of the curves pertaining to the clip from the movie "Clue."

It is interesting to note that for the "Clue" clip, using 3 dimensions actually resulted in the worst performance, whereas with the bean casserole clip it returned the

best results. Using 16 dimensions again performs fairly well, as does the higher dimensions. As a result, for these types of video it is best to use 16 dimensions when trying to identify shot transitions. The best threshold value for the two types of movies was 0.6. The results when using this value are 95% true positives and 4.5% false positives for the bean casserole clip and 100% true positives and 3.5% false positives for the "Clue" clip.

4.3.2 Scenes

Once the shot structure of the video has been identified, it is possible to progress to identifying the next higher-order temporal unit, the scene. Scenes are defined as a collection of shots pertaining to the same theme or subject. By using the hierarchical modeling described in the next section, it is possible to create representations for the entire shot, or collection of frames, rather than simply the individual frame. Then these representations can be compared to identify which shots belong to a particular scene and where transitions to new scenes may occur. Determining transitions between scenes requires the identification of this common theme between various shots that may not appear similar at all. For example, some videos contain scenes where the camera alternates back and forth between two or three different characters. A shot would consist of all the frames that focus on a particular character, and a transition would occur when the camera switched to someone different. The individual shots themselves would have similar feature representations because their content is the same. Therefore, to identify that this collection of shots constituted a scene, the system should recognize the pattern existing among alternating shots. Because consecutive shots are not likely to match using similarity functions, every other shot must be compared or every third shot and so on

within some predetermined limit. If a pattern is detected, then those shots will be clustered together into one scene. The transition to a new scene will be determined by the appearance of a shot that does not fit into the pattern previously identified by the system. This problem of detecting scene transitions is further complicated because different scenes contain different patterns. Even if the system isolates the pattern of shots composing one scene, the entire process must be repeated again for other scenes in the video.

4.4 Creating a Hierarchy of Images

When deconstructing video, it is useful to model various levels of temporal units (frames, shots, scenes, etc). This task is possible if after the video has been broken down to the frame level, the most basic temporal unit, it is then reconstructed in a bottom-up fashion by identifying all the components that belong to a single temporal unit and then clustering them to create a higher-level representation. This higher-level representation can then be used to compare and retrieve the entire temporal unit rather than simply its components. This higher-level representation must capture characteristics of the features of each subunit efficiently while minimizing the amount of information lost in the transformation. Because the chosen representation involves statistical modeling of features, to create a higher-level representation, it is necessary to create mixture hierarchies of densities where the densities of the lower temporal unit are combined into a more general probability density function.

To create the densities for higher-level temporal units, the hierarchical mixture density model is used [2]. This model extends the standard EM algorithm, but

generalizes it to compute hierarchical mixture models in a bottom-up manner. The data is modeled in the form[22]

$$P(\mathbf{X}) = \sum_{k=1}^{C^l} \pi_k^l \, p\left(\mathbf{X} \middle| z_k^l = 1, M_l\right) \tag{14}$$

where l is the hierarchy level, M_l is the mixture model at level l, C^l represents the number of mixture components, π^l_k is the prior probability of the kth component, and z^l_k is a binary variable that has a value of 1 when sample $\mathbf{X}$ is drawn from the kth component. The model is also subject to the following constraint, applied when node j of level $l+1$ is a child node of node I of level l:[23]

$$\pi_j^{l+1} = \pi_{j|k}^{l+1} \pi_k^l \tag{15}$$

The basic premise of this model is to compute mixture parameters at a given level, l, based on the knowledge of the parameters at the level, l+1, below it. One method involves running EM on a sample of mixtures from the lower level with the number of classes from the higher level to estimate the necessary parameters. However, this method does not guarantee that the resulting mixture hierarchy would have the desired structure by violating Equation 15 for certain classes. Also, this method is computationally expensive because a large sample would need to be drawn in order to relatively accurate models in the hierarchy. As a result, Vasconcelos' work develops a new method that finds the parameters for the mixture model for level l based on a virtual sample using EM and then determines a closed-form relationship between these new parameters and those from the model at level $l+1$. It assumes a sample $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_{C^{l+1}}\}$ from model M_{l+1}, where each $\mathbf{X}_i$ is a virtual sample from one of the C^{l+1} components from this model.

[22] [2] Equation 1

There are N total virtual points and each virtual sample has size $M_i = \pi_i^l N$. Using this virtual sample with Gaussian distributions, the update equations become[24]

$$\pi_j^l = \frac{\sum_i h_{ij}}{C^{l+1}} \tag{16}$$

$$\mu_j^l = \frac{\sum_i h_{ij} M_i \mu_i^{l+1}}{\sum_i h_{ij} M_i} \tag{17}$$

$$\Sigma_j^l = \frac{1}{\sum_i h_{ij} M_i}\left[\sum_i h_{ij} M_i \Sigma_i^{l+1} + \sum_i h_{ij} M_i \left(\mu_i^{l+1} - \mu_j^l\right)\left(\mu_i^{l+1} - \mu_j^l\right)^T\right] \tag{18}$$

where h_{ij} is calculated by

$$h_{ij} = \frac{\left[G\left(\mu_i^{l+1}, \mu_j^l, \Sigma_j^l\right) e^{-\frac{1}{2} trace\left\{\left(\Sigma_j^l\right)^{-1} \Sigma_i^{l+1}\right\}}\right]^{M_i} \pi_j^l}{\sum_k \left[G\left(\mu_i^{l+1}, \mu_k^l, \Sigma_k^l\right) e^{-\frac{1}{2} trace\left\{\left(\Sigma_k^l\right)^{-1} \Sigma_i^{l+1}\right\}}\right] \pi_k^l} \tag{19}$$

As is evident from these update equations, the parameters for M_l can be computed directly from the parameters of M_{l+1}. This algorithm proves to be much more efficient computationally than relying on real samples because the number of mixture components in M_{l+1} is usually less than the sample size at the bottom of the hierarchy.

The hierarchic mixture model based on virtual samples of mixture models from lower levels provides an efficient manner of creating a mixture hierarchy of densities in a bottom-up fashion. This approach allows images to be mixed into as many hierarchy levels as desired to structure video and model its temporal units as generally or as specifically as necessary.

---

[23] [2] Equation 2
[24] [2] Equations 12-14

4.4.1 An Alternate Application For the Hierarchical Mixture Model

In addition to creating representations for higher-level temporal units, the generalized representations can also be used to improve search performance at a given level. Images of a particular subject will have similar representations, but depending on scaling, size of the picture and any small variations in the position of the subject, the probability density function will vary. Therefore, the hierarchical mixture model can be used to create general representations that encompass these various properties. For example, the standardized databases such as Corel or Columbia contain dozens of pictures of different image classes such as horses or cans. Computing Gaussian mixture models for each image will result in a representation for that particular image only. Therefore, retrieval systems will be comparing the density of a query image to the densities of each individual image density to compute the likelihood of a match. Querying in such a manner requires O(n) computations, where n is the number of total images. However, if the densities for all the images within a particular image class can be clustered to create a generalized density representing that class, the retrieval system will then match the query's density to that one mixture model. Depending on how the hierarchical mixture models are created, this leads to improvements in the efficiency of the search by reducing the complexity from O(n) to O(log n). When dealing with image databases containing thousands of images for each image class, this can lead to a significant reduction in query time. Furthermore, because the query images are not likely to be exactly similar to the images in the database, computing a generalized representation may also lead to an improvement in the accuracy of the search. This generalized representation should encompass possible values for variables such as scale,

rotation of the objects, and lighting used in the picture. Therefore, a query image that differs from images contained in the database in some or all of these factors will still match to the correct image class.

To determine if the use of the hierarchical mixture model can improve image retrieval accuracy, experiments were conducted on the Columbia and Corel databases that tested the accuracy of determining image similarity when using databases built by using hierarchical mixture models with a varying number of mixture components. Using the Columbia and Corel[25] databases gives an idea of the effects of the hierarchical mixtures on standard object databases. Each database was divided into a set of training images and a set of query images. Hierarchical mixture models were created for each image class within the set of training images using the following number of mixture models: 8, 16, 32, 64, 128, 256, and 512. Increasing the number of mixture components in the hierarchical model creates a density that better approximates the lower level densities. The result is similar to fitting a polynomial curve to a set of data; using a higher degree usually results in a better fit to the data. However, there can be a problem of overfitting that actually results in a worse approximation. Overfitting results in a model that fits the set of training data extremely well, but fails to identify a generalized model that works for future data. If the same problem of overfitting occurs when the number of mixture components is increased in the hierarchical mixture model, there will be a break point at which using a higher number of mixture components actually results in a decline in accuracy. Figure 10 below shows the results from the experiment.

[25] The experiments were conducted using all the image classes from Columbia and the following image classes from Corel: Arabian Horses, Auto Racing, Coasts, Diving, English Gardens, Fireworks, Glaciers Mountains, Mayan Ruins, Oil Paintings, Owls, Pyramids, Roses, Ski, Stained Glass, Tigers.

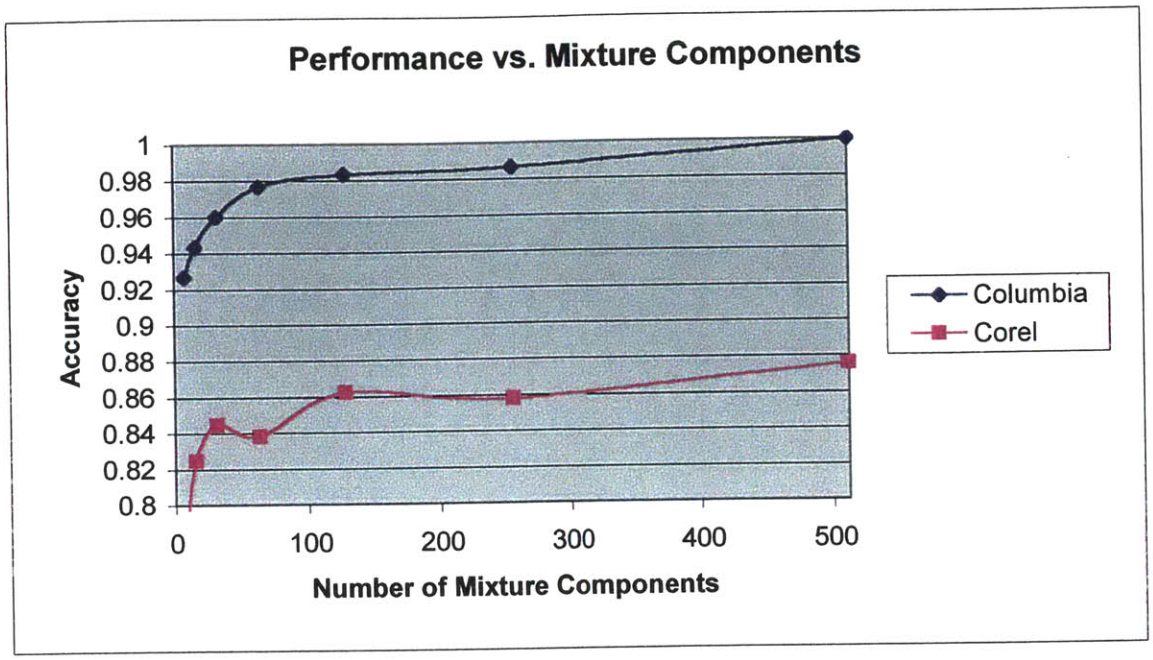


Figure 10: This shows the performance returned when using a varying number of mixture components to create the training databases. The accuracy is the ratio of correct to total matches.

From the results, it is apparent that increasing the number of mixture component does not result in a decline in accuracy. Further examination of how the hierarchical mixture model is computed offers an explanation as to why the problem of overfitting does not occur. This phenomenon would occur if as the number of mixture components increased, the resulting Gaussians contained in the hierarchical mixture model were characterized by progressively smaller variances. The mixture model would then be the weighted sum of many Gaussians with small variances to describe each of the points in the training data. The small variance of the Gaussians will result in gaps in the density space that should normally be covered but are not. Therefore, although the mixture model correctly identifies the data from the training set, a generalized representation has not been created. The constraints and the update equations used to create the hierarchical mixture model prevent the creation of Gaussians characterized by a smaller variance than the Gaussian densities from the lower level mixture model. In particular, Equation 18

that shows that the covariance of the higher level mixture model is actually a sum of covariances from the lower level mixture models.

Although the search accuracy improves as the number of mixture components is used, from Figure 10 it is apparent that the search accuracy is still relatively high when fewer mixture components are used. Therefore, the tradeoff of accuracy for a less computationally expensive model can lead to an overall improvement in performance.

Chapter 5: Applications

The applications for the algorithms described in the previous chapters can extend to all types of video. Retrieval systems can be created that parse video to varying levels of temporal units depending on the feasibility and the use for the structure. The Minerva project is one such application that encompasses both the video parsing technology and the use of hierarchical mixture models to improve search accuracy and efficiency. It deals only with images of cooking ingredients and video clips containing cooking show segments, but the technology can be expanded to deal with a variety of video ranging from other types of educational programming to movies to sports programming.

5.1 Minerva

The Minerva system is a video cooking assistant that helps users decide what to cook based on ingredients they have, preferences they specify and skills they exhibit. Once a recipe has been chosen, the system then proceeds to show video instructions of how to prepare the dish. This project is essentially an exploration into the expansion of computing into areas of people's lives that is typically devoid of computer interaction. It attempts to introduce computers in an unobtrusive manner into kitchens to simplify tasks that can benefit from the vast resources available through the use of technology. Like all complex systems, Minerva is a combination of the latest technologies in a variety of areas ranging from object recognition, context-aware computing, and responsive media.

5.1.1 System Overview

The physical components of Minerva include a countertop monitored by a small camera and a touch screen monitor used to interact with the user. The user first logs into the system, currently by selecting the appropriate login name on the touch screen.

Figure 11: The Minerva system's initial login screen.

Minerva then loads the user's preferences and begins by playing the user's favorite movie.

Figure 12: Minerva playing the user's favorite movie.

When the user is ready to proceed, he starts the system by pressing the button marked "Minerva." This takes him to a screen showing the camera view of the countertop. The user places the ingredients that he wants to use within the screen area and presses "Ok."

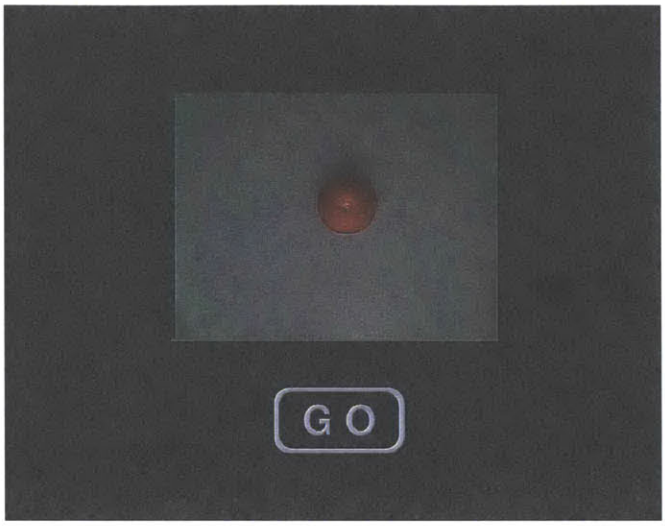


Figure 13: The countertop as Minerva sees it.

He can place multiple objects, either of the same ingredient or of multiple ingredients. The system then attempts to identify the ingredients by comparing them to the training database it contains. The next screen shows what the system thought the ingredients were and shows three recipes that are most likely to be relevant.

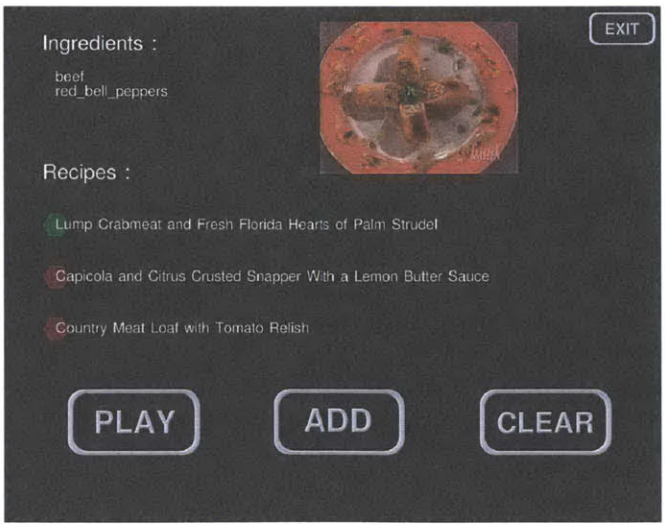


Figure 14: Minerva showing results of ingredient recognition. The screen shows the ingredients that were identified, the top three possible recipes and what the finished products for those recipes should look like.

The user can touch the recipe names to view how the finished product should look. If none of the recipes look appealing, he can restart the system with a different set of ingredients. Once he decides which recipe he is interested in, he touches the appropriate one and then pushes the "Play" button. A screen containing a text version of the recipe and the video segment of how to make the dish appears.

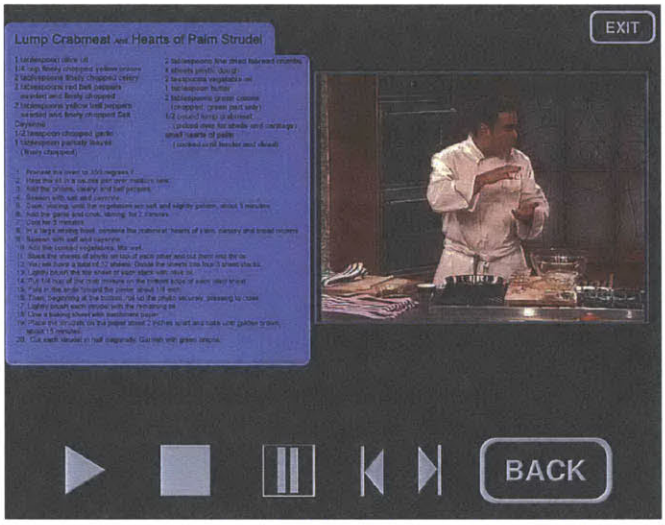


Figure 15: Final Minerva screen where text version of recipe is shown alongside the video instructions. The user can manipulate the playback of the video using the buttons at the bottom of the screen.

The user then proceeds to play the video and learns how to make the dish. He can jump around within the video and replay segments as needed. At any time, the user can restart the system by pushing the button marked "Exit" to return to the initial kitchen television mode.

5.1.2 System components

The Minerva system contains three main components: the user interface, image recognition technology, and a database containing information about recipes, links to video cooking show segments and information about user preferences. The user interface was created using the Isis technology developed by Stefan Agamanolis, which is a programming language designed to "support the development of demanding responsive media applications" [8]. The image recognition technology and the way the system plays the video cooking show segments use the algorithms discussed in this thesis.

5.1.2.1 User Interface

The user interface is a touch screen that simplifies the interaction between the user and the system by removing the need for keyboards or mice. The user only has to touch the appropriate button to perform the desired tasks. The interface is composed of the series of screens described in the overview of the system that show the normal kitchen television mode, the countertop, the recognized ingredients and possible recipes, and finally the text version of the chosen recipe and the video segment of how to cook it. The system takes the inputs given through the interface to feed the image recognition technology and then query the database for the appropriate recipes. Using Isis has several benefits including being able to handle complex two-dimensional compositions of visual geometry, images and video. A screen can be created that contains components for movies, images and other objects. Furthermore, movies can be converted to the Isis format to facilitate the manipulation of how the video is played.

5.1.2.2 Object Recognition Technology

The image recognition technology is used to identify the ingredients that the user is interested cooking with. Vasconcelos developed the core components of the technology, and the system uses modified versions of these components to satisfy system requirements and constraints. The technology is built around a training database of various ingredients[26] consisting of a collection of images taken of each ingredient. Mixture models were computed for each image and were combined into a training database used as a reference for the images of the ingredients being queried. After the user places his ingredients on the countertop, the system takes a picture of the countertop. The system then splits the images into subimages, each containing one object.[27] These subimages are then queried with the training database to determine the identity of the object. The modifications to the original system designed by Vasconcelos will be discussed further in the next section.

5.1.2.3 The Database

The database contains two main parts: information pertaining to the user and information pertaining to the available recipes. All of the information is arranged in sets of tables that organize the data efficiently to allow easy access. The tables pertaining to the users contain basic information about the user, such as name, sex, age, and information about the user's experience level and preferences. The user's preferences include attributes such as "vegetarian" and "low-fat". The tables dealing with the recipes

[26] Ingredient classes used in training database include: beef, butter, eggs, garlic, green beans, macaroni, onions, oranges, pork, red bell peppers, tomatoes, and vegetable oil.
[27] Shyam Krishnamoorthy developed the splitting technique used. A description of how the image is split can be found in Appendix B.

include information about the origin of the recipes, ingredients needed, and information about how the recipe should be classified to match to possible user preferences. A detailed description of the structure of the database is shown in the Appendix C.

The database also contains links pointing to where the video segments containing instructions of how to create the dishes are located. The theories of temporal units and parsing video are utilized to efficiently annotate the various clips so that the system can play the video in a useful manner. By locating the transition points to different shots, the user can jump around to different parts of the video easily to replay segments that he may need to watch again or to skip ahead to other parts and bypass unnecessary instructions that they have already seen. How these videos were annotated and how the user can manipulate the playback is described further in the next section.

5.1.3 Modifications Implemented

Several modifications needed to be implemented to create a system that could practically be used. The setbacks in performance stemmed from having to deal with completely new query images every time the user wanted to find a recipe. He would put new ingredients on the table that varied in composition and orientation. Therefore, each new query image would have to be split into its sub-images and then mixture models would have to be computed for these sub-images. Because the creation of mixture models takes a significant amount of time, modifications to Vasconcelos' original system needed to be implemented that would improve the speed of object recognition, but maintain its accuracy as best as possible. The time involved in splitting the query image and to query the sub-images with the database is relatively insignificant, so much of the

effort in improving the speed of the object recognition was placed in computing mixture models faster.

There are several factors that could be adjusted to reduce the time needed to compute mixture models. The two main factors that were experimented with were size of the query image and the number of data points used when computing the Gaussian distributions. Each original image is 320 x 240 pixels, and to compute the mixture models it is first divided into smaller blocks of 8 x 8 pixels. Then points from each of these blocks are used as data points when computing the mixture models. In Vasconcclos' original system, he used a step size of four, so he extracted a point every 4 pixels. With these parameters, the creation of mixture models took approximately 30 seconds. When the image size was halved and the step size was doubled (halving the number of data points), the mixture models were created four times faster, in about 7 seconds. The question that remains is how these changes affect the accuracy of the query. By reducing the image size, the resolution decreases and the resulting image does not contain as sharp information as the original image. Reducing the number of data points that are used to compute the Gaussian mixture models also affects the accuracy slightly simply because there are not as many data points to fit the distributions to. However, using a step size of eight still results in a large enough sample size that the accuracy of the query should not be affected too much.

To test how these changes affected the accuracy, new training databases were created that also incorporated these changes. Mixture models were created for the original size training images using a step size of eight and a separate database was created using images half the size of the original images. These two types of databases

were then divided into four classes of databases: database using the Gaussian mixture model ("normal"), a database using the hierarchical mixture model ("hierarchical"), database created by splitting the original training images, using the same techniques that are used on the query images, and then using the Gaussian mixture model ("split normal"), and finally a database using the hierarchical mixture model on the split training images ("split hierarchical"). These databases were queried with a set of 24 query images from various ingredient classes.[28] The results from these experiments are shown below in Figure 16.

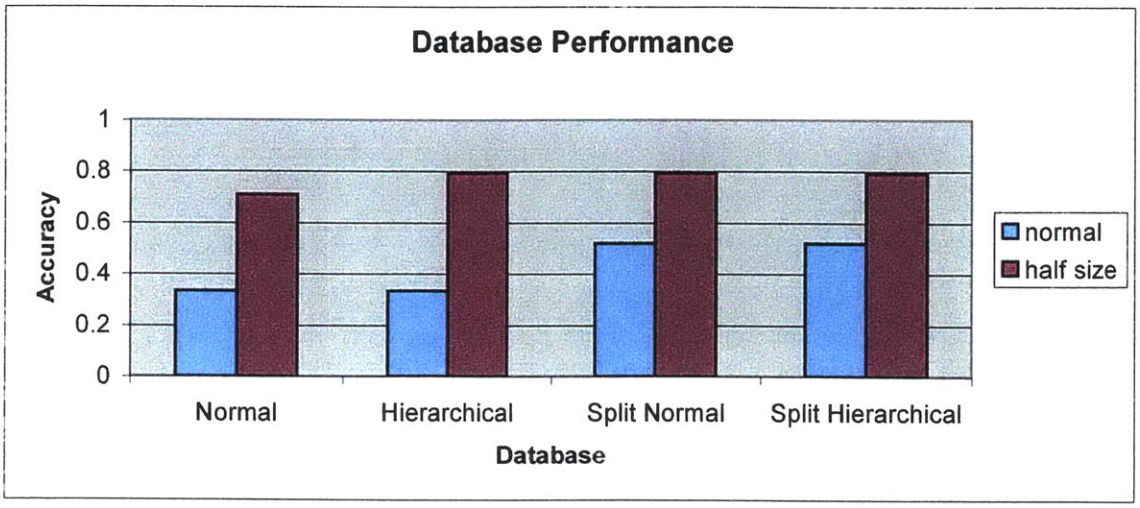


Figure 16: This plot shows the results when the different types of databases were queried with a set of query images from different ingredient classes. It shows that splitting the images and reducing the image size improves the accuracy of the image recognition.

Figure 16 shows that the database created using the half size images actually performed better than the original size images. Between the databases containing the split images and the original images, the split databases seemed to do better, at least with the original size images. Among the set of databases containing half size images, the hierarchical, split normal and split hierarchical all performed equally. This might have

[28] The ingredient classes tested were: beef, garlic, macaroni, oranges, pork, red bell peppers and tomatoes.

been due to the relatively small sample size of query images or the types of images contained in the training set. The final database chosen was the split hierarchical database with half size images. This database performed as well or better than the other databases, it incorporates the splitting technology to use images similar to the query images, and it reduces computational complexity by utilizing the hierarchical mixture model.

5.1.4 Use Of the Video Parsing Technology

Minerva uses the video parsing technology to play back the cooking show segments in such a way as to help the users learn how to prepare the dishes better. Very rarely can someone watch a cooking show straight through and follow precisely what the chefs are doing. Therefore, if the user can skip around in the video, to either play back certain portions or to move past a previously learned step, and pause the video so that play is resumed at an appropriate position, he will be able to better benefit from the video instructions. Being able to skip around the video implies that when the user selects the fast-forward or rewind option, he is fast-forwarding or rewinding directly to the next shot desired rather than the next frame in the video. Resuming play from the appropriate position means that it is resumed from the beginning of the shot during which it was paused. This is analogous to manipulating a person that is speaking. If the person is asked to move ahead or move back, he will move to the next sentence or complete thought rather than to the next word. Also, if a person is stopped in mid-sentence, he will start from the beginning of the sentence or thought when he resumes speaking, rather than simply from the precise point at which he stopped speaking. Playing the video in this manner allows the user to experience some continuity while watching the video.

Users will be able to access segments of the video at a time rather than trying to manipulate the video frame by frame. This functionality is particularly useful for viewers of cooking shows because if the user is inexperienced he will want to replay segments often and if the user is experienced, he will want to skip ahead to relevant segments.

These features for the video playback are implemented using the shot transition identification techniques described above in Section 5.1.1. This algorithm identifies where shot transitions occur within a particular video segment. These transitions represent the frame numbers of the first frame of each new shot. Shot transitions are identified for all the relevant video clips and stored in the database as simple text files containing the appropriate numbers. The system then accesses the corresponding shots file for a particular video after the user has chosen a recipe. To implement the playback features, the system simply keeps track of which frame it is currently playing and finds the frame number of the beginning of the next shot it needs to play by finding the closest frame in the stored file of frame numbers.

Chapter 6: Future Extensions

Minerva is a fully functional system that can take a given set of ingredients as inputs and output suggestions for recipes and video instructions about how to make a desired recipe. However, there are many improvements that can be made to the system to make it more practical and to give it more functionality. Furthermore, the topics discussed in this thesis can be extended into other areas outside of media.

6.1 Minerva

The types of extensions for the Minerva project fall into four main categories: the user interface, the object recognition system, the video playback system, and the database. It is important to remember that the overall goal of the system is to introduce computing in an unobtrusive manner into the kitchen.

In addition to simply upgrading the functionality, it is important to make the system more fun to entice users to employ it. Making it more "fun" can involve simply adding noises every time the user touches the screen, to playing catchy background music while the system is functioning, to randomly displaying advertisements for various products related to cooking while the system is querying or idle.

6.1.1 User Interface

The user interface can be changed in a variety of ways to make the system more enjoyable and user-friendly. The current login procedure consists of touching the button associated with a particular user's profile. This method is useful while the number of users is relatively low, as is the case with most households, but problems can arise as the

number of users grows to a more unmanageable level. There are many options for new login procedures ranging from voice recognition to face recognition to simply hand recognition (where the user passes his hand under the camera and the system recognizes whose hand it is). Furthermore, the initial screen that comes up currently plays a clip from the user's favorite movie. The system can be transformed to act more like a normal kitchen television if it tuned into the user's favorite television station. Or the system can transform itself into a kitchen radio and play some of the user's favorite songs.

6.1.2 Object Recognition Technology

Important extensions to the object recognition component of Minerva, besides simply improving the technology to return more accurate results, are to provide ways to deal with the incorrect identification of the ingredients and to "learn" about new ingredients that the system previously had no knowledge about. Currently, the system simply assumes that the ingredients it matched are correct and returns the most relevant dishes based on that assumption. If the user does not like the choices he is given, then he can start over with a new set of ingredients or try the query again by changing the way the ingredients are placed to see if the system returns better matches. However, if the system is consistently wrong in the recognition of ingredients, then the dishes it suggests to the user are useless. Therefore, there has to be a mechanism to correct the system. One possible mechanism is to again use voice recognition technology so that the user can say the name of the ingredient and the system will recognize what it is. The system should also provide a way for the user to teach it about new ingredients. This involves taking pictures of the new ingredient, creating the necessary mixture models and incorporating them into the existing library of mixture models. The tricky part lies in

how the pictures are taken because the user might not know how to teach the system. To overcome this problem, the system can give the user instructions about how they should place the object and change it to get a complete set of images that can be used to train the system.

6.1.3 Video Playback System

The video playback component can be improved upon and enhanced with new technologies. In addition to being able to jump around to different shots within the video, the concept of scenes can be implemented to give the user further control of how to play the video. Also, irrelevant portions of the video clips, such as commercials, can be removed from the clip altogether, so that the user does not have to worry about fast-forwarding past them. Another interesting functionality that could be added is the presence of multiple versions of a video that would depend on the user preferences such as his experience level or whether he wants commercials or not. For example, there could be two different versions of a video recipe that either gives very detailed instructions or shows only the basic steps. These various streams would be useful to users with different levels of experience since advanced cooks would not want to watch the clip with detailed instructions every time and have to fast-forward through all of the tedious sections. Having these options would also be useful if users are trying to make the same recipe again because users will not want to watch the detailed video the second or third time that they make the dish. An extremely useful feature would be the ability to move back and forth between these various streams. This way the user could watch the detailed instructions for the parts of the video that he needs more help on and watch the basic video for the rest of the preparation of the dish. Viper is a system that allows the

manipulation of multiple video streams and audio streams and "providing primitives for assembling complex edits and transitions, such as L-cuts, audio and video inserts, dissolves, graphic overlays, slow motion, and so on, and for finely adjusting these elements in response to real-time response activity" [9][29] Although the technology might not be in place to realize all of these visions, using Viper and building on its technology is a step towards the right direction.

6.1.4 The Database

Extensions to the system's database depend heavily on the types of changes implemented in the other components. The purpose of the database is basically to store all the information that the other components need to access and to return the appropriate information efficiently. Therefore, any additions to the knowledge base the system requires would need to be incorporated by the database. If the interface is converted to be a kitchen television or radio then the database must store information about the users' favorite television channels or radio stations. If the system is capable of learning about new ingredients, then it needs to be able to return recipes that contain them. Furthermore, the database can include information about whether the user has made a certain recipe and how many times if so. With this information, a less detailed stream could automatically be played if the user has experience with a particular dish.

6.2 Applications In Other Fields

Thus far, this thesis has only discussed the application of the search and selection mechanisms examined to media. There are other far-reaching uses for these algorithms

[29] [9] p. 81.

ranging from law enforcement to medicine. Any areas that rely on imaging and must search through databases of images can utilize the improved techniques for searching and selection.

Law enforcement uses pictures and video to identify criminals, observe trends in victims, and characterize crime scenes. Using the object recognition technology to retrieve images that relate to a particular individual or to a particular scene would give officials greater resources to work with. One important tool that the police use to solve crimes is looking at surveillance videos. If they can parse the video and extract all the shots containing the suspect, they can compare those shots to previous videos to extract more clues.

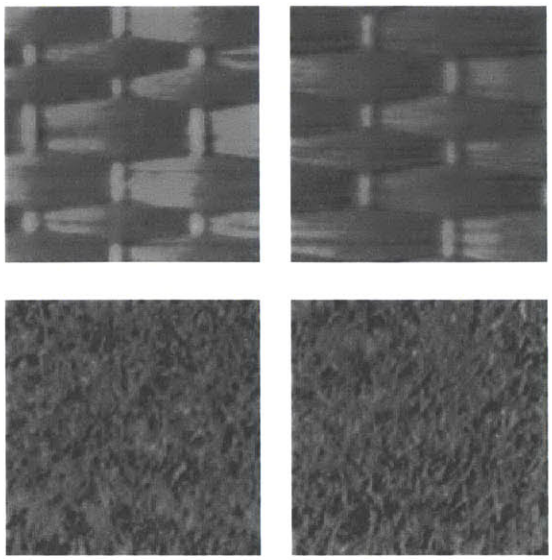
Physicians also rely heavily on imaging to make diagnoses and observe a patients progress. Using the search and selection mechanism to retrieve records for other cases with similar ailments might help make the doctor to make a better diagnosis or devise a better treatment plan. For example, if an X-ray reveals a mass in a patient's lungs, the doctor could search for other patients with a similar mass. He can then examine everyone's records to come up with the best possible plan of attack.

Implementing these algorithms in other fields would be an exciting step toward the future of image processing. However, many advances would need to occur in order for such steps to be taken. The many components of the overall process from feature selection to finding a compact feature representation to developing suitable similarity functions would need to be examined further and optimized. Current systems retrieve images with a great deal of accuracy but to apply these methods to threatening situations such as those in law enforcement or to life and death situations found in medicine, the

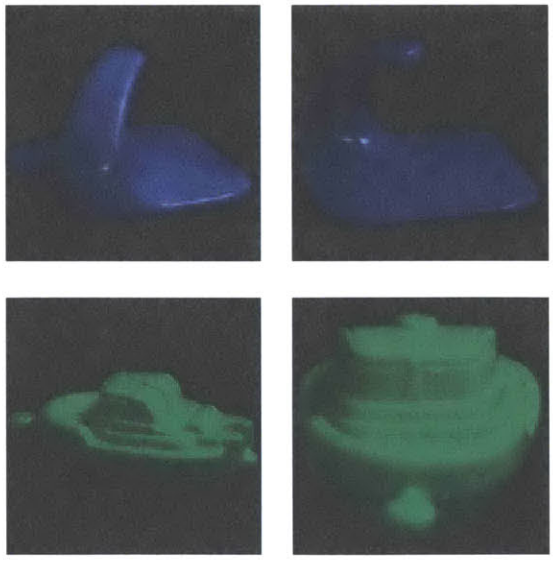
accuracy and efficiency of the systems need to be improved by orders of magnitude. Nevertheless, each step brings us closer to achieving these previously inconceivable notions.

Appendix A: Examples from Standardized Databases

Images from Brodatz Database from two images classes:[30]



Images from Columbia Database from two image classes:[31]



[30] The image classes are (in order) D55 and D8.
[31] The image classes are (in order) D35 and D77.

Images from Corel Database from 3 image classes:[32]

[32] The three image classes (in order) Arabian Horses, Fireworks, and Pyramids.

Appendix B: Splitting an Image

The first step to splitting an image is to subtract the background from the original image and then using flood-fill techniques the individual objects are extracted from the resulting picture. The system goes through each pixel recursively to find a pixel that is not part of the background and then finding dimensions of the corresponding object by locating all of the neighboring pixels that are also part of the object. It then extracts this object, fills the extracted region of the original picture with the background and begins the procedure again with the first pixel that doesn't match to the background to find the rest of the objects. The following diagram shows how a query image is split using the method implemented by Krishnamoorthy [10].

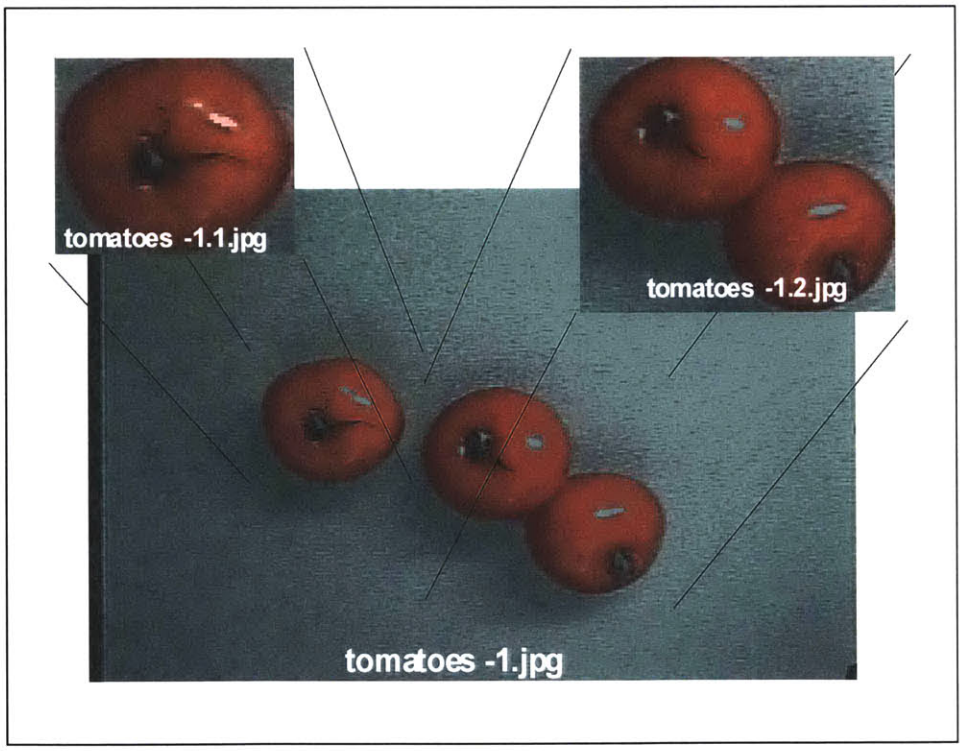


Figure 17: The original query image is shown by tomatoes-1.jpg. After the splitting technique is applied the two other images, tomatoes-1.1.jpg and tomatoes-1.2.jpg, are created.

Appendix C: Structure Of miniSQL Database

The database component of Minerva was implemented using mini SQL 2.0 [11]. It consists of a series of tables connected by the primary keys of each table. The following is a description of its table structure:

```
shows:              {showID (int, show_name (char), chef (char)}

episodes:           {episodeID (int), showID (int), episode_date (date),
                     episode_title (char), notes (char)}

ingredients:        {recipeID (int), ingredient (char), ingredient_wt
                     (int)}

recipes:            {recipeID (int), episodeID (int), recipe_name (char),
                     filename (char), recipe_type (char),
                     ingredient_count (int), difficulty (int)}

recipe_categories:{recipeID (int), category (char)}

user_info:          {userID (int), login (char), first_name (char),
                     last_name (char), age (int), experience (int)}

user_preferences: {userID (int), category (char), difficulty (int)}
```

References

[1] Vasconcelos, Nuno and Andrew Lippman. "Embedded Mixture Modeling for Efficient Probabilistic Content-Based Indexing and Retrieval", SPIE Multimedia Storage and Archiving Systems III, Boston, 1998. p. 5.

[2] Vasconcelos, Nuno and Andrew Lippman. "Learning Mixture Hierarchies", NIPS'98, Denver, Colorado, 1998.

[3] Vasconcelos, Nuno and Andrew Lippman. "A Probabilistic Architecture for Content-based Image Retrieval", a short version appears in CVPR'00, South Carolina, 2000, © IEEE.

[4] Vasconcelos, Nuno and Andrew Lippman. "Feature Representations for Image Retrieval: Beyond the Color Histogram." ICME '00, p. 2.

[5] Vasconcelos, Nuno, Andrew Lippman. "Library-based Coding: a Representation for Efficient Video Compression and Retrieval", DCC'97, Snowbird, Utah, 1997, © IEEE..

[6] Vasconcelos, Nuno. "Bayesian Models for Visual Information Retrieval." Ph.D. Thesis, MIT, June 2000.

[7] Florence, Tyler. "Food 911", The Food Network. Episode title "Moist Meatloaf in Henderson Nevada." Green Bean and Pearl Onion Casserole dish. February 4, 2001.

[8] Agamanolis, Stefan. "Isis: A Programming Language for Responsive Media." [http://isis.www.media.mit.edu/] MIT, Media Lab.

[9] Agamanolis, Stefan. "Isis, Cabbage, and Viper: New tools and strategies for designing responsive media." Ph.D. Thesis, MIT, June 2001.

[10] Ly, Bryan. "Minerva: A Smart Video Assistant For the Kitchen." Advanced Undergraduate Project, MIT, June 2001.

[11] miniSQL version 2.0. [http://www.Hughes.com.au]