, , 1–32 ()

Tools for Browsing a TV Situation Comedy Based on Content Specific Attributes

JOSHUA S. WACHMAN AND ROSALIND W. PICARD

wachman@media.mit.edu, picard@media.mit.edu

The Perceptual Computing Group of the MIT Media Laboratory
20 Ames Street Cambridge, MA 02139

Editor:

Abstract. This paper presents general purpose video analysis and annotation tools, which combine high-level and low-level information, and which learn through user interaction and feedback. The use of these tools is illustrated through the construction of two video browsers, which allow a user to fast forward (or rewind) to frames, shots, or scenes containing a particular character, characters, or other labeled content. The two browsers developed in this work are: (1) a basic video browser, which exploits relations between high-level scripting information and closed captions, and (2) an advanced video browser, which augments the basic browser with annotations gained from applying machine learning. The learner helps the system adapt to different peoples' labelings by accepting positive and negative examples of labeled content from a user, and relating these to low-level color and texture features extracted from the digitized video. This learning happens interactively, and is used to infer labels on data the user has not yet seen. The labeled data may then be browsed or retrieved from the database in real time. An evaluation of the learning performance shows that a combination of low-level color signal features outperforms several other combinations of signal features in learning character labels in an episode of the TV situation comedy, *Seinfeld*. We discuss several issues that arise in the combination of low-level and high-level information, and illustrate solutions to these issues within the context of browsing television sitcoms.

Keywords: computer assisted learning, video pattern recognition, video annotation, *Society of Models*, *FourEyes*, content-based retrieval

1. INTRODUCTION

Episodic television, with text captioning, scripts, clean video signals and recurring characters provides a unique opportunity to explore issues in video pattern recognition and video browsing. In this work we provide a set of tools and a strategy for integrating available high-level information, general purpose signal analysis techniques, and a computer assisted learning algorithm, to allow for intelligent browsing of video content. The performance of the strategy presented will be demonstrated through the construction of an intelligent video browser applied to data from a popular television situation comedy.

The organization of this paper is as follows: First, we outline briefly the context for the development of video browsers, and motivate the strategies used in this work. Then we give an overview of the system we built. This system is in two parts: a basic video browser that uses high-level information only, and an advanced video browser that augments the basic video browser with low-level video content analysis and the ability to learn. These systems are described and evaluated in the sections that follow. Several issues related to video content analysis and browsing are discussed in detail.

1.1. Background

Techniques for sifting through on-line text documents based on keywords, concepts, or measures of word proximity have spawned the on-line information retrieval business. Methods for searching large databases of still-images are being developed to assist stock photography houses, image archivists and World Wide Web search agents. The consumer of the future may soon have automatic tools to organize a *digital shoe-box* full of family photos based on an array of content specific characteristics (*ie.*, images captured in the forest, images *sans* mother-in-law, etc). The impending proliferation of digital video databases portends the need for semi-automatic tools which can intelligently browse, index, annotate and navigate through video data with the ease associated with traditional document retrieval. Nascent research in automatic retrieval of still-images has yielded promising results and is, in part, the basis for this work.

Methods which exploit the co-occurrence of text and images to index images have been explored previously [6], [3], [15]. These approaches parse the associated text for key words which may indicate some sort of geometric or temporal information about the imagery. In a similar way, the present work attempts to parse the closed caption information to establish relationships with the occurrence of characters in the accompanying video. However, by including computer learning and a human operator in an interactive feedback loop, the present system has the potential to get better results than methods which simply exploit the co-occurrence of text and image.

In the near future, the largest databases of image data may be composed of video. The simple sub-sampling methods utilized by commercial video players for video skimming tasks are inadequate for content based retrieval or browsing tasks. Future browsers may be expected to solicit users' queries on available databases and to display related sequences to a user for feedback. Furthermore, tools must be available to examine the underlying video content in order to provide the means for intelligent responses to database queries. The last is among the most provocative issues related to video browsers, and the one motivating this work.

There are several browsing methods [14], [19], [21], [9], [7], [2], which are useful in analyzing, labeling, correlating, integrating or reducing the search space associated with data in a video sequence. None of these is directed at identifying specific plot elements, such as which characters are in a particular shot. What distinguishes the present approach from those cited, is that it integrates low-level and high-level information together with knowledge that may be learned from interacting with a human operator. The tools presented in this work are not only able to perform low-level queries on visual similarity, but are also able to learn characteristics of plot elements which may then be browsed.

1.2. Using a Learning Approach

The issue of how to identify, label and extract the content of a video stream is the primary focus for this research. Specifically addressed is the problem of how to find, distinguish and *learn* the representation of a set of characters in a TV situation comedy. Our approach builds upon and extends earlier work with the still-image browser and learning system *FourEyes* (originally called "Photobook with Learning.") FourEyes, named to emphasize its combination of both computer vision and the input of a seeing user, was developed to help users annotate databases of still images [13]. The basic idea was that a user would label a small set of data, and FourEyes would use its *Society of Models* to infer how to label the rest of the data. The *Society of Models* allowed for many notions of similarity to co-exist in the same system. The most useful notions of similarity could depend on user subjectivity, which had to be learned from users' input. The best annotation and retrieval performance was obtained when the system *learned continuously*, that is in an ongoing way while interacting with a user who provided it with positive and negative examples of labeled images. This is in contrast to systems that learn offline or in a batch way, once for a problem, and then have to be re-trained if they are to learn to solve a new problem. Continuous learning is stronger than many traditional forms of relevance feedback, as the user feedback does not just influence the current query, but can influence the long-term representations used by the system as well. Minka and Picard later augmented FourEyes to have multiple continuous learning strategies [11] and [10].

In this work, we extend the FourEyes learning system to handle video. We also augment its low-level learning abilities with tools that handle high-level script and closed caption information. We hypothesize that the high-level information is not only useful for users who wish to browse the video at a high level, but it may also be useful for helping constrain the learning with low-level models in *FourEyes*. It is proposed that the combination of high-level contextual information with low-level video pattern recognition, can facilitate more effective video browsing and retrieval tools.

1.3. Why Work With Situation Comedies?

The structure of a television situation comedy is encapsulated in the script and echoed in the closed captions. From the script and closed captions, we can obtain setting (where), characters (who), actions (what), and order (when), which can be used to analyze and interpret the results of low-level image features. These high-level textual descriptions in combination with results from low-level image analyses can be used to parse television situation comedies into salient labeled regions.

Within the context of a TV situation comedy, characters have distinctive movements, clothing, expressions, speech patterns, vocabularies, behaviors, dietary habits, hair styles, sizes, shapes, gaits, etc. Although such personal attributes may be associated with all people, the characters in a TV situation comedy, are intended to be caricatures of real people. As caricatures, their mannerisms are exaggerated and usually more pronounced than those of the people they parody. Such hyperbole makes situation comedies, as a genre, a rich test bed for developing schemes for recognizing patterns attributable to specific individuals. Arguably, the most salient constituents of situation comedies are people. People appeared in 98% of the 182 shots of *Seinfeld's* episode *The Beard* analyzed for this work. Finding patterns that constitute people in a situation comedy is a reasonable starting point for annotating and perhaps even understanding, the video itself.

1.4. Overview: System Block Diagram

Figure 1 shows a block diagram of the browsers built in the course of this work. There are three main sources of input media: the script, closed captions and the broadcast video itself (*sans* commercials). The script may be considered to be an ordered list of what is being said as a function of who says it. It also contains information about scenes and their setting. The closed captions contain transcriptions for hearing-impaired viewers, such as what each character says and non-visible events such as a phone or doorbell ringing. The video at this level is parsed into shots using a shot detector (described below.)

The first stage of Figure 1 illustrates a Basic Video Browser which is constructed by extracting only high level features from the script, closed captions and video. Correlations among these features can be used to generate an ordered list of labeled shots. This is the Basic Video Browser. The technical tools involved in its construction are discussed in Section 2.

The second stage illustrates an Advanced Video Browser which incorporates the information in the Basic Video Browser as well as unsupervised selection of regions of interest and the basic components of FourEyes: low-level feature extraction and learning via the *Society of Models*, construction of similarity trees, and a classifier that learns by interacting with user input and ground truth. The Advanced Video Browser is considered in detail in Sections 3 and 4.

2. BASIC VIDEO BROWSER

How effective a video browser can be constructed without sophisticated computer vision and learning methods? The following data processing results in a Basic Video Browser which incorporates high-level textual features, a simple shot detection scheme and assumptions about the structure of television situation comedies.

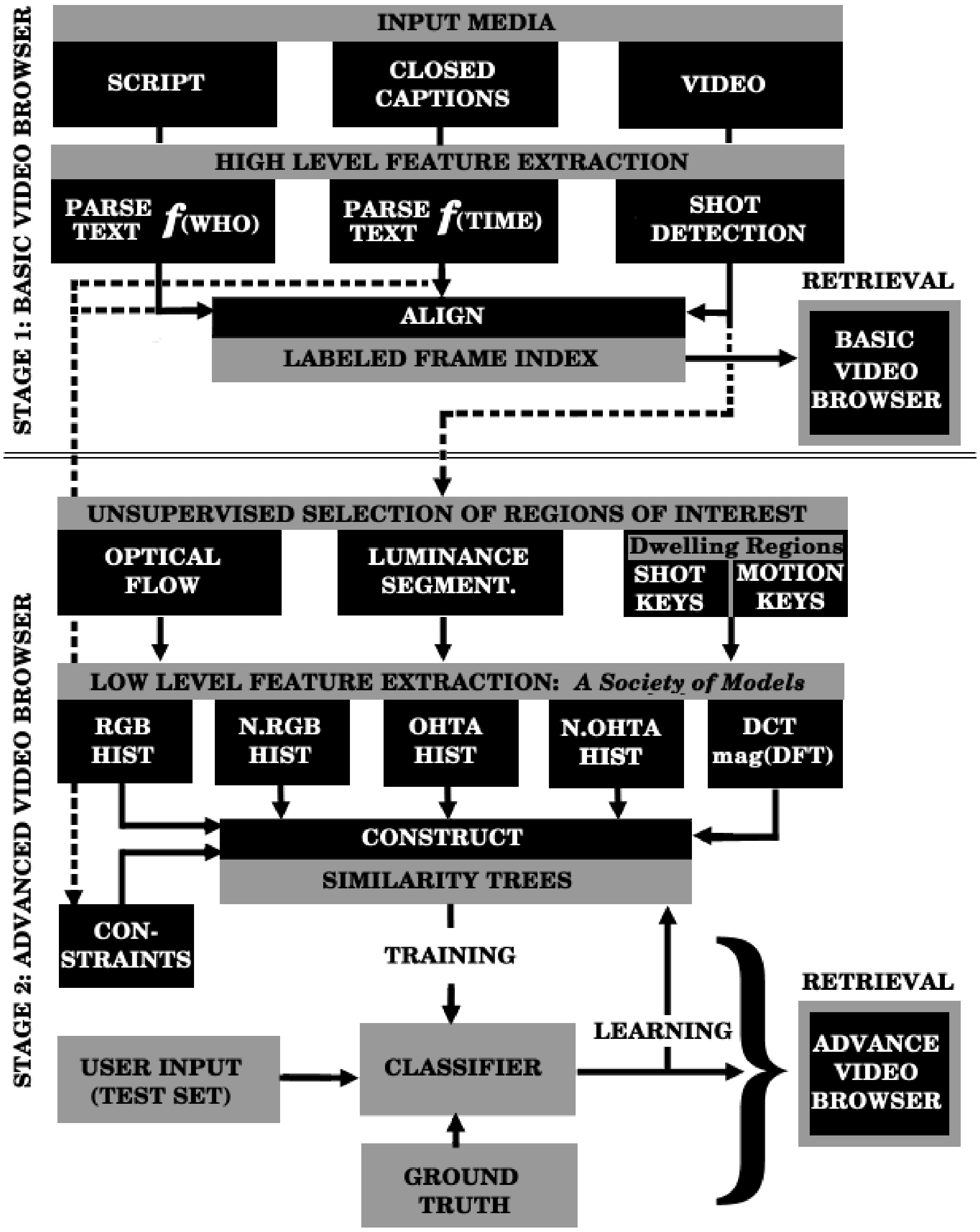


Figure 1. The system block diagram has two stages of analysis and results which echo the structure of this presentation.

2.1. High-Level Feature Extraction: Aligning Script and Captions

Within the scope of this presentation, high-level features contain primarily semantic information such as where a scene is shot, which actors appear in the scene, and what they supposedly said. The script and closed captions provides this textual information for the basic browser. A correlation and temporal alignment of these two textual representations may be produced with standard word parsing and matching techniques.

The onset and duration of any closed caption is easily correlated to the simultaneous broadcast of its associated video. While the dialogue's onset and duration may not be completely synchronous with the on-screen activity, they often appear during a relatively narrow sliding window of time in order for the viewer of the closed captions to make sense of the text with the on screen activity. One problem with closed captions is that since they are created by human operators and are edited in manually, their precision is never guaranteed. Therefore, the automatic alignment of script, closed captions and video yields an indefinite list of labels of who is in each sequence. Figure 2 illustrates the automatic alignment procedure tool.

To constrain the alignment, we made an assumption that the character speaking is visible in the shot in which his caption appears. This is a generalization which holds with few exceptions. One such exception occurs in in Shot 21, when George speaks on the intercom to Jerry. George does not appear in the scene until shot 32. The classic 'reaction shot' is a more common example.

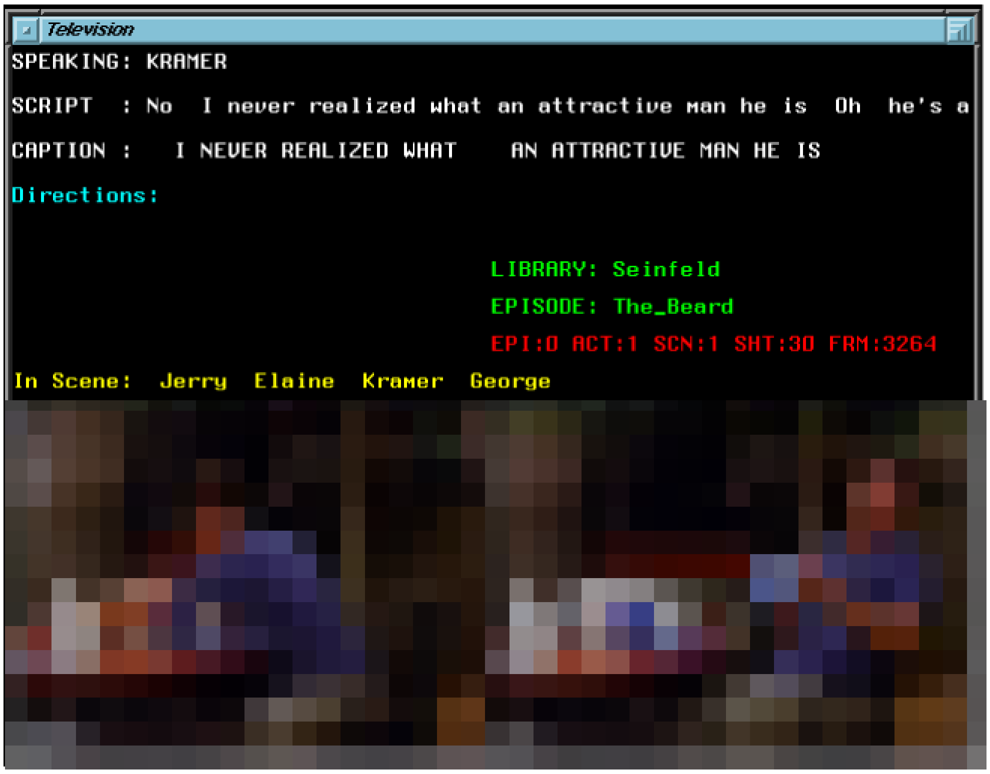
In practice the alignment process begins by siphoning the captions data off the broadcast video using a data recovery unit (EEG Model DE 152). This converts the closed captions into a serial stream of ASCII. Custom software was written to decode the stream into a file of captions. The actual script of *The Beard* was acquired from Castle Rock Entertainment and entered into the computer with all of the script formatting maintained. Since the format of the theatrical scripts is regular, it is straightforward to parse them into snippets of dialogue labeled with the name of the speaker.

2.2. Video Preprocessing

The Beard is the 22 minute (*sans* commercials) episode of *Seinfeld* used for the analysis performed (See Appendix A. for Episode Summary). The first ten commercial-free minutes of the episode were digitized, temporally smoothed and spatially subsampled. This produced 4 Giga Bytes (18,000 frames) of quarter resolution otherwise uncompressed video. Shot detection, script and closed caption analysis was performed on the entire 10 minutes of digitized video. Subsequent low-level image processing (for the Advanced Video Browser) was performed only on the first 5 minutes (82 shots).

2.3. High-Level Feature Extraction: Shot Detection

The video stream was partitioned into shots automatically by thresholding the absolute value of the difference between the luminance histogram of each pair of successive frames. Although more sophisticated methods exist e.g.[20], [1], a single threshold, which was found experimentally, was able to find 182 shots in the 10



[Images Courtesy of Castle Rock Entertainment]

Figure 2. A screen dump of the tool which correlates the script and closed captions and partitions the video into shots. The image in the bottom left is the fifth frame in the shot, the key frame. The image in the bottom right is a frame from the video as it plays. By parsing the script, membership of each frame of video into its parent act, scene, shot was automatically cataloged. In addition, a list of characters who appear in each scene as well as any relevant stage directions were determined.

minutes of video with acceptable accuracy. There were four false positives which occurred in two shots in which the camera was panning severely. There were no false negatives.

2.4. *Smart Fast Forward: Characters, Frames, Shots, and Scenes*

The video was displayed using a custom video browser written in C++ and SGI/GL. A screen dump of the browser is shown in Figure 3.

With the captions and script correlated and the video partitioned into shots, the video was indexed automatically by character, shot and scene. Using the Basic Video Browser, people are able to fast forward to the next frame ("¿"), shot ("¿¿"), or scene ("¿¿¿"), sampling the video in semantic chunks rather than simply skipping every N frames. By clicking on the buttons at the bottom of the browser, named for each character, the browser skips ahead to key frames in shots, or scenes that contain the selected character(s).

Shown in Figure 4 are the key frames retrieved given the query "Fast forward to shots with Kramer." For the first 82 shots analyzed, there were 27 in which Kramer had at least one caption. The Basic Video Browser was able to find 18 of these shots, for a performance of 67% accuracy. There were nine false negatives and six false positives retrieved.

The nine false negatives are defined as shots that should have been retrieved by the system, but were not found because of text parsing errors. The 6 false positives are shots retrieved that should have been skipped. Of the six shots falsely included, four were in fact shots in which Kramer appeared but did not have a caption so they could not have been identified by the Basic Video Browser. That is, they were fortuitous mistakes. In general though, these false positives violate the guiding assumption that the character speaking is in fact, in the shot.

There are two sources of error for the remaining false positives identified by this procedure:

1) Shots falsely labeled Kramer in which he speaks but the camera was trained on someone else (the classic reaction shot for example).

2) Shots which were improperly labeled due to false script/closed caption correlation. For instance, if there is a single word script snippet like "Yes" but the caption says "Yup", it is virtually impossible to match the two lines without incorporating a vernacular Thesaurus and sophisticated lexical parsing and timing criteria. These constitute alignment problems that could be fixed in subsequent implementations.

Note that shots in which Kramer is not in the key frame, but is in subsequent frames within the shot are counted as correct retrievals, not false positives. A more sophisticated browser should be able to detect this inconsistency and find a more appropriate key frame.

2.5. Towards a Smarter Browser

As demonstrated, the retrieval results of the Basic Video Browser provide greater functionality than what is commonly available. However, the potential for such a system to become more effective is limited, given the temporal imprecision of closed captions, and the inability to identify characters in shots in which they do not have spoken lines. If it were possible for a computer system to learn the visual appearance or pattern of Kramer in shots in which he speaks, then potentially when he appears in shots in which he does not speak, he could be recognized. More generally, if learning was robust, then the results of the Basic Video Browser could be used as input to an automatic training session. If this were possible, then the actor who plays Kramer might be able to be retrieved from video in which he appears even if there were neither a script nor captions available.

Ideally if the knowledge of who was in each shot was accurate, it could be used by *FourEyes* to bootstrap the learning of video patterns by providing positive training

[Image Courtesy of Castle Rock Entertainment]

Figure 3. Screen dump of the Basic Video Browser. Users can skip ahead to the next shot or scene that contains a given character or group of characters. Clicking on the button labeled “Kramer” and then clicking the button labeled “¿¿” causes the browser to fast forward to all shots with Kramer.

[Images Courtesy of Castle Rock Entertainment]

Figure 4. Key frames corresponding to shots in which Kramer has a closed caption attributed to him.

examples. This was our original intent. Unfortunately, the knowledge gathered in constructing the Basic Video Browser was too inaccurate to provide reliable positive examples for training FourEyes. In the Advanced Video Browser discussed below, this bootstrapping, or learning by example, is fostered by an interactive training session where a user provides the positive training examples. As will be discussed later, the auxiliary high-level, albeit imprecise knowledge, will be used in another form to expedite learning by constraining the possible classifications of characters in shots.

The Basic Video Browser was constructed using the automatic alignment of captions, script and shot boundaries. It would be possible to apply this technique to many styles of episodic captioned television for which a script exists. However, without the script or closed captions, the method fails. As mentioned, Kramer appears in many other shots in the episode and this approach will not find him in these other shots unless he has a caption during those shots. Under these circumstances what is needed is a means of identifying the visual pattern in the video which is Kramer. To do this, the system has to recognize components of the video signal which constitute Kramer's representation. This is a substantially more difficult problem than labeling sequences in which Kramer has text. It warrants a more sophisticated approach.

3. AN ADVANCED VIDEO BROWSER

In order to learn the patterns of a given character the system has to analyze constituents of shots smaller than a frame. Our approach is to have an unsupervised system reduce the data to a set of regions most likely to include characters of interest and then to present these regions to a user for an interactive labeling, training, and learning session. A discussion of methods for reducing the video data is considered in the remaining portion of this section. The learning procedure is discussed in more detail in the section that follows.

3.1. Assumptions Guiding the Unsupervised Region Selection

Reasonable assumptions about the way television situation comedies are filmed provide an implicit context which makes harvesting these regions relatively straightforward.

The first major assumption is that the character speaking in a shot is the one moving the most. This premise is guided by the observation that when characters speak, they move their heads, gesture and are the focus of the camera's attention. If there are other actors in the shot, they are generally less active as to not upstage the one speaking.

Another assumption is that, in a situation comedy, the most salient motion events are people oriented. One feature of most situation comedies is a style of cinematography where a set of cameras are allocated to several zones around the stage. Since the cameras are locked off (stationary), most of the motion that occurs in situation comedies, and in *Seinfeld*, in particular, is motion of people or parts of people. There is no hand held camera work and few zooms, tilts, trucks or other canonical camera motions. In fact in the first 82 shots of *The Beard* there is only one subtle zoom and approximately 15 pans of varying intensity. The remaining shots are from static cameras.

In the event of camera motion, it is the object that is *not* moving relative to the camera which is more likely to be of interest. Filtering for this object or set of objects is possible by various means e.g. [18]. The premise in this circumstance is that if the camera is moving, it is tracking a character, therefore the character is stationary relative to the frame. Except during the camera acceleration/deceleration or during the character's acceleration/deceleration, the magnitude of the motion should effectively distinguish the character from the background. Steam rising and doors opening were the only severe non-camera, non-human motion events which occurred during the first 82 shots of *The Beard*.

Another assumption which governs the processing described below is that in video, a change in image flow may indicate a significant change in content. These changes include shot boundaries, camera motions and object motions. Therefore, during this processing stage, changes in image flow above a certain threshold were acquired automatically by the database and tagged as potential regions of interest. This step also served to reduce the amount of data input to *FourEyes*.

3.2. Unsupervised Selection of the Regions of Interest

Unsupervised processing of the first 82 shots of the episode constituting the first 8200 frames (approx. 5 minutes of video) generated 15,052 irregularly shaped image segments or regions of interest. The rest of this section describes how these regions were identified and extracted.

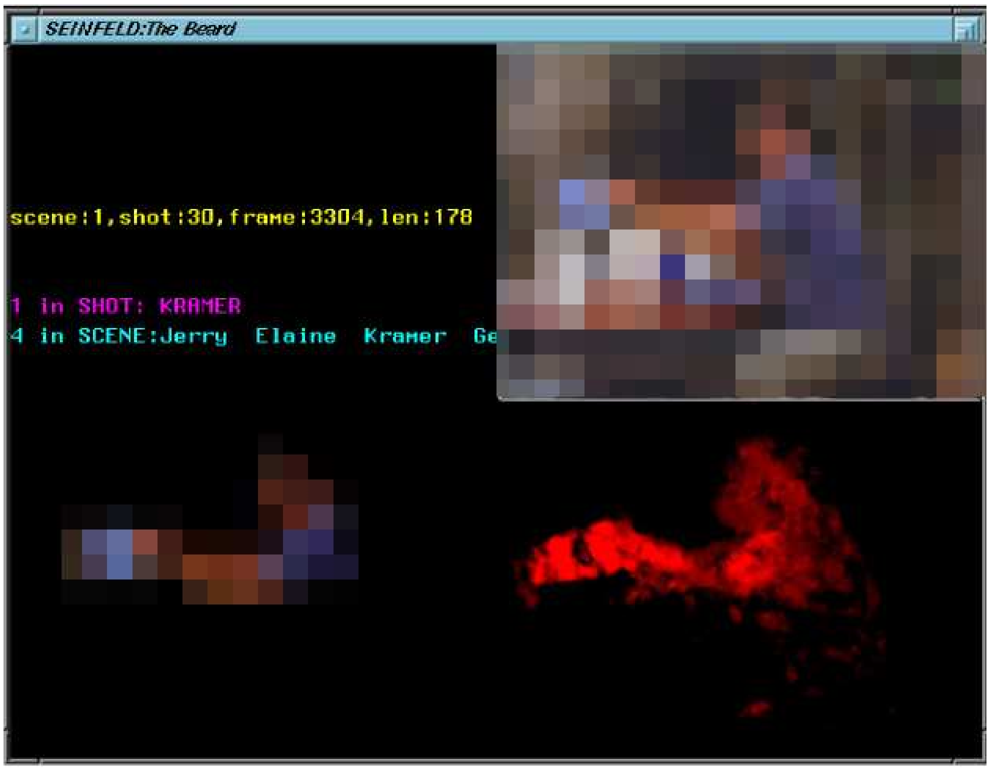
3.2.1. Optical Flow Analysis: Isolation of regions of substantial coherent motion or *activity* was performed by measuring the optical flow magnitude using three parameter estimation [8]. The optical flow is notoriously noisy; subsequently, we followed it with a five tap temporal filter and a 3x3 spatial median filter to reduce the noise. A bit map mask for each segment was generated and an oversized bounding box around each region was drawn using the maximum length and height. Figure 5 is a snapshot of the tool that performs this segmentation.

For the purpose of people detection, one may apply motion segmentation which may find objects that move coherently. Such coherent motion may be associated with hands waving, heads shaking, arms swinging, legs walking, etc. Although these activities may be facilitate isolating coherent regions in the optical flow, the detected objects may be insufficient identifiers of the active character.

3.2.2. Procedure for Luminance Segmentation: Walk a 3x3 kernel over the image. If the value of the peripheral pixels is close enough to the center pixel by some user prescribed threshold, then include the pixel in the present segment. Else assign that pixel to its own segment. Repeat until walk is done. Figure 6 illustrates the luminance segmentation performed.

3.2.3. Procedure for Acquiring Dwelling Regions: In addition to moving objects, static ones may also be of interest in a database query. An object which moves and then stops, called a *dwelling* object, would be invisible to the present system, if it were not for the following steps. A dwelling region may be identified through a sampling procedure that extracts full frames at points of significant change during the video.

Full image frames were extracted under two circumstances:



[Images Courtesy of Castle Rock Entertainment]

Figure 5. Screen dump of tool which segments optical flow and culls regions of interest based on the segmentation of the flow image. The bottom right is a still of the motion magnitude. The bottom left is the segmentation of the motion magnitude. The upper right is the bounding box of the segmentation superimposed over the source image.

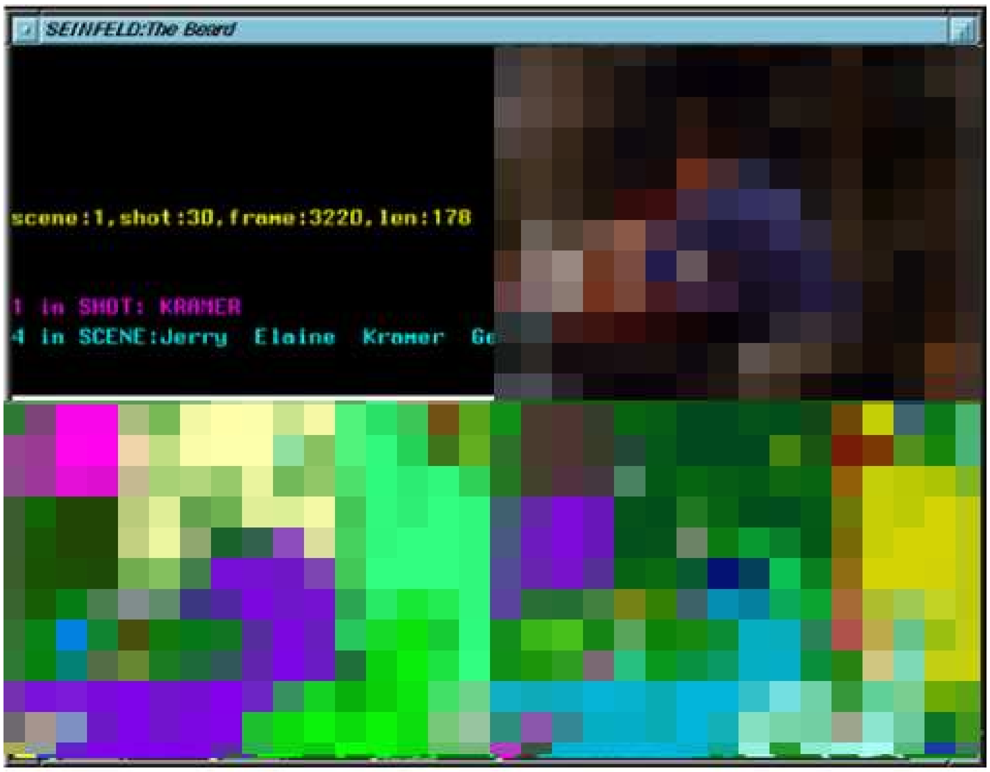
Shot key frames: Sample the fifth frame of each new shot.

Motion key frames: Extract new full frame luminance samples every 15 frames during instances of camera motion. These motion key frames capture the background imagery revealed during camera pans.

Detection of dwelling regions within shots relies on the fact that objects which stop moving persist. Once initialized, tracking systems (see [4] for survey) might be better at localizing dwelling regions within shots than our procedure which samples frames at shot boundaries and during camera motion.

However our learning system approach has the potential to identify objects that moved in one shot and appear static in others. Whereas a tracking system would have to be reinitialized.

Both shot and motion key frames were converted to luminance and then segmented as described. However, since there is no unique segmentation of an image,



[Images Courtesy of Castle Rock Entertainment]

Figure 6. Screen dump of tool which segments the luminance image of the key frames. Two of the four levels of the segmentation pyramid are shown in the bottom half.

segmentations were performed at four different thresholds. This pyramid scheme generated multiple segmentations for each image in an attempt to span the space of perceptually relevant segmentations. Regions below a certain minimum size were ignored. For motion key frames, a single threshold which was found empirically, tended to segment the images into perceptually salient regions.

To summarize, the preprocessing culled regions of significant and homogeneous change. Shots were detected by partitioning sequences along temporal discontinuities in the luminance image. Within each shot, regions were identified by segmenting the optical flow magnitude into coherent regions of motion. Refinements were made by segmenting these coherent motion regions into regions of coherent luminance. In addition, dwelling regions which are present during shot key frames and motion key frames, were acquired by luminance segmentation. The resulting regions became available to *FourEyes* for subsequent image analysis and labeling.

3.3. Output to FourEyes

Each image segment and bit map mask was read into *FourEyes* and then mapped to a coarse 16 x 16 grid of 20 x 15 pixel blocks. This permitted analysis of irregularly shaped regions in terms of individual blocks. It also introduced a measure of quantization error as regions were forced to adopt or abandon more of the background image than was originally extracted by the segmentation procedures. This is an artifact of implementation which should be addressed in future systems.

It should be noted that no explicit model of people as a class of objects was assumed in the work of this paper. The low-level analysis part of this work identifies regions of significant change and could be equally applied to video sequences that do not include people and are not scripted or captioned. The tools presented here are for general purpose video learning and retrieval, and are not contingent on a specific video compression scheme or subject matter. It is expected that the learning performance reported here on actors will not be as good as methods that incorporate explicit models using face detection, voice recognition or articulated geometric models.

A strength of the present general purpose approach is that it should be flexible enough to find other classes of objects within the database aside from actors. Unfortunately, within the data analyzed, there were an insignificant number of objects on which to search aside from the characters.

Intuition and experience gained from using *FourEyes* suggested which models might be the most effective. By observation, the wardrobes of the characters were quite distinguishable, which suggested that simple color and texture models might give the most robust characterizations of the actors. Since all of the characters in the episode have virtually identical skin and hair color, it was expected that abundant regions composed of hands and heads, would not be distinguishable by color or texture alone. While a face recognition system might be useful in this context, it would require additional schemes which would reduce the generality and increase the complexity of the system.

3.4. Low-Level Feature Extraction: Modeling the Regions of Interest

FourEyes is able to work with almost any models: color, shape, texture, position, and so forth. In developing a pattern recognition system, it is typical for a researcher to try many models, tuning each with different parameter settings. In FourEyes, we have automated this tuning process. The researcher "plugs in" as many models and potential parameter settings as he or she desires to investigate, and FourEyes does the work of comparing them all. The system not only can choose the best model, but it can also choose the best combination of multiple models. One of the advantages of FourEyes's structure is that it does not suffer from the exponential curse of dimensionality that strikes other systems which concatenate features into one giant feature vector. Hence, there is virtually no penalty associated with including redundant models or features which are only slight variants of each other. For this research, we supplied FourEyes with five models.

Model 1) RGB Color Histogram Each block was quantized into a set of 32 uniformly spaced bins for each of the three color channels. The three 32 level histograms were concatenated into a 96 unit vector which represented each block's distribution of color. A single vector for each segment was computed via block-wise averaging. The histogram intersection [16] was used as a similarity measure between regions. For probing associations among segments in a database browsing venue histogram intersection is advantageous. Consider, for example, two regions being compared: A to B. A is a full frame image. B is half the size of A by truncation. For real (non uniform) images, Euclidean distance metrics would find these histograms to be dissimilar but the histogram intersection of both finds them to be comparable. This metric is asymmetrical, and is beneficial when comparing images of different sizes. Figure 7 is an example of the benefit of using histogram intersection in this database. A query on Kramer's hands finds other hands, but also finds them *in situ*. The asymmetry is revealed in the opposite situation: given the query of the large frame, one would not expect the hand to be returned [17].

Model 2) Normalized RGB Color Histogram The procedure is the same as described above except instead of computing the histogram intersection, the integral of each histogram was normalized to size 1. The Euclidean distance between vectors was used as a measure of similarity between segments. This normalization step was performed to facilitate color comparisons between segments of different sizes. Consider, for example, two regions being compared. One from a wide shot, the other from a close up of the same object. The histograms of these regions are similar, but without the normalization steps, the similarity between them would not be evident. In the database of *Seinfeld*, one would expect the histogram intersection to out perform the normalized histogram since the distribution of shots at different scales is fairly narrow. There are few zooms and the variance of scales is determined by the character's distance to the camera. This variance is small. This subjective judgment was based on observations of the cinematographic style used in the show.

Model 3) Ohta Color Histogram The Ohta color space is an expression of the RGB color cube in terms of *eigen*vectors calculated over real scenes [12]. Histogramming in this space can be expected to produce better segmentations then in the straight RGB space for some data, because it segments real world imagery with respect to its principal components. As above, each 15 x 20 pixel block was quantized into one of 32 uniformly spaced bins for each of the three color channels. The three, 32 level, histograms were concatenated into a 96 unit vector which represented each block. A single vector was computed for the segment via block-wise averaging. Histogram intersection was used as the measure of similarity.

Model 4) Normalized Ohta Color Histogram This model involves the same process as the Normalized RGB Color Histogram (model 2) only performed

[Images Courtesy of Castle Rock Entertainment]

Figure 7. Screen dump of *FourEyes* interface, the upper left is the test image, the others are the ranked responses in raster order. RGB histogram intersection gets Kramer's body from a sample of his hand that contains some of his shirt in the background.

in the Ohta color space. The block-wise average histogram was computed over the segment. Euclidean distance was used to measure similarity.

Model 5) DCT of the DFT Magnitudes This is a texture metric. Procedure: Tile the image into blocks of size 8x8 (size arbitrarily chosen). Take the magnitude of the DFT of each block in order to make comparisons shift invariant. Take the top 10 (number arbitrarily chosen) coefficients of the DCT of the result in order to express the texture discriminant in a compact form. Use the Mahalanobis distance between the vectors as a measure of similarity. Local covariance was used for segments of size greater than 10 blocks. For segments with fewer blocks, an identity matrix was used for the covariance.

Additional Models Other texture models were considered but were ultimately discarded: The strength of the Multi Resolution Simultaneous Auto-Regressive (MRSAR) model lies in its ability to characterize complex textures across scales. Its effectiveness on a database of natural scenes which might include foliage, water or bark has been shown [10]. In this database, composed of relatively textureless clothing (at television resolution) and man-made interiors, its effectiveness is limited by the lack of pattern variation, which can cause difficulties for the model parameter estimation. The Discrete Cosine Transform (DCT) is deterministic and works fine on smooth areas, but is sensitive to shifts in the placement of the analysis blocks during tiling. That is, the imagery varied enough that as each tile was analyzed its response was considerably different depending on how the tiles were mapped onto the image. Therefore comparisons across image regions were uninformative by this metric. Eigenvector analysis was rejected because comparison of irregularly shaped regions of such a variation in size is impractical.

3.5. Formation of Similarity Trees

Approximately 750,000 15 x 20 x 3 pixel tiles were analyzed by *FourEyes*. This is about 675 MB of image area or 35% of the 8200 frame sequence. After the metrics were run on each of these regions, a single-link hierarchical clustering was performed. A similarity tree was constructed for each metric using the *FourEyes* learning system. This bottom-up approach generated similarity trees like the following in Figure 8 where each node collects nodes of similar items. This is the initial training of the system.

The use of similarity trees is the key component that allows FourEyes to combine and explore different models without the exponential growth in dimensionality that accompanies systems that attempt to combine features in one feature vector.

Other kinds of hierarchical clustering for constructing the trees were evaluated but ultimately abandoned. Complete-link, unweighted average, weighted average, and Ward's method [5] all resulted in grossly inferior performance in terms of their ability to approximate the underlying feature space.

Once the trees are formed, classification can proceed rapidly, and features from the different models can be used jointly or individually to group similar objects, even those that span shots. Consider the case of a body moving. The optical flow segmentation may acquire a coherent region of the torso, head and arm. The luminance segmentation may segment those regions individually, depending on how the body is clad. FourEyes finds that with the histogram intersection method, the individual (luminance segmented) regions are clustered with the motion segmented collection of body parts. In general, the color features were found to be helpful in

[Images Courtesy of Castle Rock Entertainment]

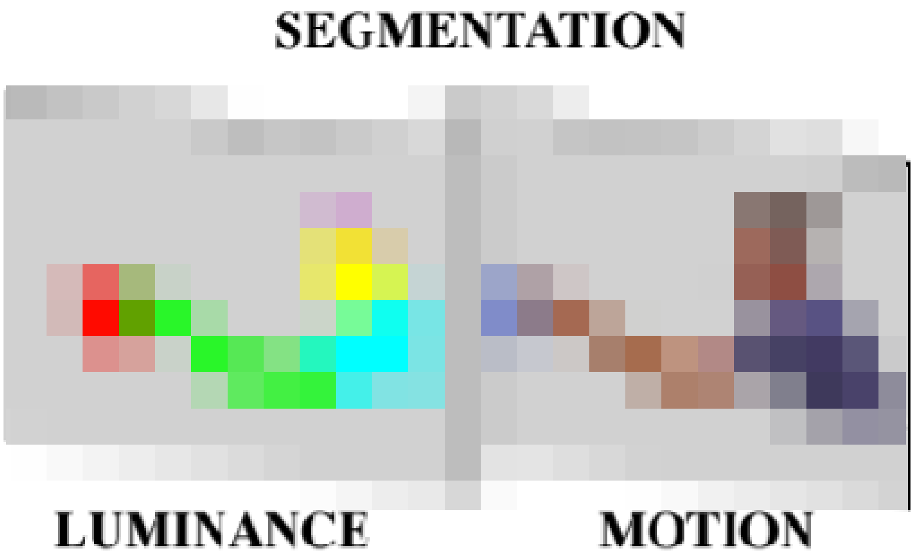
Figure 8. Samples of different branches of the same RGB histogram similarity tree. On the left are George segments; on the right are Kramer segments.

identifying objects that spanned shots where the motions of the individuals may be quite different. Figure 9 illustrates this idea graphically.

4. EVALUATING PERFORMANCE: LEARNING EXPERIMENTS

4.1. Determining Ground Truth Labeling for the Regions

In order to evaluate the performance of the preprocessing stage in culling regions related to the characters and the veracity of associations formed during training, the content of each region had to be determined objectively. To generate this classification, the set of 15,052 regions was first decimated into 10 subsets. Every tenth segment was presented to one of ten human subjects for manual labeling. The task was to label each of approx 1,505 segments with the name of one of 6 characters (Jerry, Elaine, George, Kramer, Man, Lou) or to leave the region "blank by default". The instructions specified that each segment was to get only one label (mutual exclusivity) and full frame segments were to be ignored. A segment could be labeled if more than half of it contained a portion of one of the characters.



[Images Courtesy of Castle Rock Entertainment]

Figure 9. Histogram intersection between segments derived from luminance and motion segmentations may increase the likelihood that an individual region in the luminance segmented image is associated with a region derived from a motion segmented image. The luminance segments illustrated above could be considered children of the parent motion segment of the same object. The segmentation of key frames in shots that had minimal motion could be used to find related objects in other shots in which objects with similar color histograms moved.

Aside from errors made by the human observers, there are potentially several sources of error in the design of gathering “ground truth” this way.

1) A viewer evaluated each segment within the context of the full source image. Therefore classification decisions were potentially influenced by surrounding image information. Had each segment been masked, the number of unlabeled regions likely, would be higher.

2) A single label may not be sufficient for a given box. As in the case of box number 11,612 (frame 6523) where Homeless Man and Kramer exchange a piece of Tupperware. The motion of both arms forms a single moving region. These ambiguous regions were left ’blank’. It is indeterminate whether such regions should be labeled ’Kramer’, ’Homeless Man’, ’blank’ or weather an alternative category should be provided, such as ’Chinese Food’? See Figure 10 for illustration.

3)Since the regions were viewed in the context of the full frame, and not as individually masked regions, the understanding of how people move from frame to frame may have influenced the labeling. Perhaps a most precise method would

[Image Courtesy of Castle Rock Entertainment]

Figure 10. Ambiguous labeling task. Is it Kramer, Homeless Man or Tupperware?

have the user outline objects themselves and then the area of overlap with available regions could be computed by the computer.

The results of the seven alternative forced choice labeling constitute the priors and were used to measure learning performance. Ideally these data would be gathered over a library of episodes and would be normalized on an episode-by-episode basis given the tendency for some episodes to feature certain characters more than others.

4.2. Eliminating the "blank" Class

The six distinct character classes corresponded to regions which were hand labeled as positive examples of individual characters. The blank classes represented ambiguously segmented data. Segments in the 'blank' class could in fact be composed of portions of individual or groups of individual classes. Therefore, they were disregarded from the benchmark tests because of their ambiguity. In future work, the manual labeling could explicitly tag each blank class as a negative example of one of the other six classes, designating it as some other class (e.g., Chinese food, window, door, or some multiple groupings of existing classes such as the segment mentioned above which includes both a Kramer and Homeless Man).

4.3. The Symmetric Set Cover Algorithm

FourEyes is equipped with a reconfigurable bank of learning algorithms. Of these algorithms, set covering is the method which was used for the analysis reported. Given a set of data, the set cover method forms the smallest union of data which includes all of the positive examples and none of the negative examples. Everything else in the data set remains unclassified. Minka [10] implemented a symmetric set cover method for this work which is more optimistic than the standard set cover in *FourEyes*. The symmetric set cover (see Figure 11) finds the set of all positive examples similar to the instance, then finds all of the negative examples which are *not* examples of the instance. The assumption in the symmetric set cover is that false positives are common and their effect should be ignored. False positives refer to regions included in the set that should not be members. Given that the data set is composed of real world imagery and that each model does not well constrain its members by perceptual metrics, it is fair to expect spurious members. Specifically, it is likely that a given node in the similarity tree will group regions with a similar histogram together, but there is no guarantee that in culling through those nodes, that the leaves will be pointers to perceptually similar items. The symmetric set cover includes the false positives without penalty. In a browsing venue this should be acceptable, if not desirable.

4.4. Evaluating the Learning of Segment Labels

Given the ground truth labeling, several benchmark experiments were performed. The effectiveness of the tree clustering, and hence of the annotating and retrieval

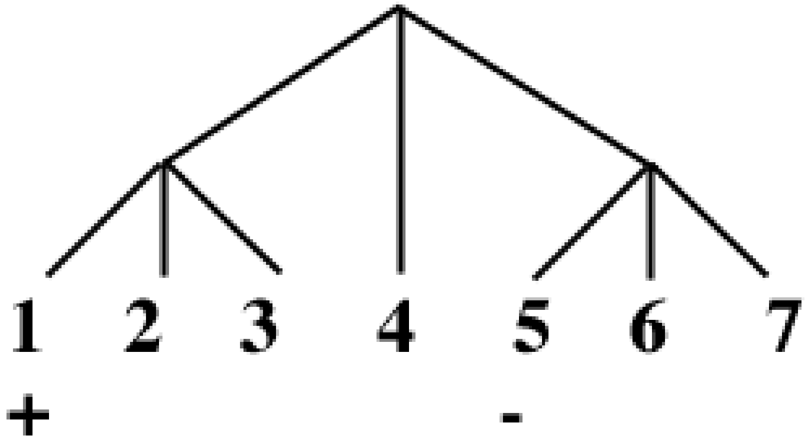


Figure 11. Example of Set Covering. Given that the user labeled segment 1 positive and segment 5 negative, the following illustrate the behavior of various set cover methods:

<u>Non-Symmetric Optimistic Set Cover</u>
Positive 1,2,3 Don't Know 4,6,7 Negative 5

<u>Non-Symmetric Pessimistic Set Cover</u>
Positive 1
Don't Know 2,3,4
Negative 5,6,7

<u>Symmetric Set Cover</u>
Positive 1,2,3
Negative 5,6,7
Don't Know 4

abilities of the system, can be evaluated in terms of a learning curve. The goal of the learning is to obtain the correct (ground truth) labels of all the data given the fewest number of labeled examples from a user. Hence, the learning curve indicates not only labeling accuracy, but also labeling efficiency (more efficient implies that the user needs to give fewer positive and negative examples before all the correct matches are found). The faster the curve drops, the better the performance. The procedure for generating the learning curve is described by [10] on page 26.

In Figure 12, the learning performance of the RGB, Ohta and DCT of the DFT magnitude models were plotted against a baseline classifier that learns nothing. The y-axis of the learning graph plots the number of errors during learning. The extent of the y-axis is bounded by the product of the number of possible segments (5382) and the number of wrong labels it can inherit minus a penalty for being wrong (2). For instance, a case in which the ground truth label for a patch is "Jerry," but the patch is labeled "Kramer" by one tree and "Elaine" by another tree, would be penalized with four errors. In Figure 21, the graph was cropped and scaled to increase readability so the limit of the y-axis is around 5500. The x-axis is simply the number of training examples (maximally 5382). The fact that the learning curve starts to be significant in the range of the number of segments means that the initial rate of learning is rapid.

Notice that of the individual models, color outperformed texture and Ohta performed best. The normalized Ohta and RGB were also evaluated. They each performed slightly worse then their unnormalized counterparts and were left off the plot to increase readability. The DCT of the DFT Magnitudes appears to be a poor model of this particular data-set. Its initial performance was worse than baseline. This can be explained by the fact that there is a double penalty for false positives. The combined models (RGB and Ohta) were also plotted and performed noticeably better than the individual models. Notice also, that the combined RGB and Ohta curves converge to zero after about 2000 training examples. This means that there is a grouping of 2000 training examples from which *FourEyes* can successfully extrapolate the labels to the remaining 5382 samples. Since the learning performance graph is generated by picking segments at random the performance will vary on each run depending on which segments are used for training. Nevertheless, these data appear representative of performance and it can be said that the combined RGB and Ohta models learned at a rate of 2.7:1. This measure was calculated by dividing the 5382 segments by 2000 training examples.

Since the system learns continuously, each interaction with a user results in more correctly labeled segments. Given enough time and consistency in the ground truth, all the segments will become labeled correctly, and the system will make zero errors in annotation and retrieval.

4.5. The Effect of High-level Information on Learning Performance.

From a perceptual and semantic standpoint, the appearance of characters change as the fictional day or occasion in the story changes. For instance, in *The Beard*, George sports a toupee. This is such a radical change in his appearance in the fiction of the episode that it becomes a major plot point. Whereas, when George takes off his jacket upon entering Jerry's apartment in Act I Scene A, it is an acceptable appearance change to the viewer and a negligible plot point. Although each character's wardrobe may change radically across scenes, observations made over a library of episodes should reveal that their wardrobes converge to a specific style. Viewers are not often confused by radical clothing changes because they monitor and recognize other features of people, including height, voice, behavior, gait and of course face. What is needed is a way of adjusting the set of features selected based on the high-level information available in the script.

The incorporation of high-level contextual information can be used to constrain the classes in which a given segment could be a member. Here are three examples of when this high-level information can be useful:

1) Location and Setting Information: The script may offer assistance in classifying with respect to wardrobe changes. In each scene header, the script indicates

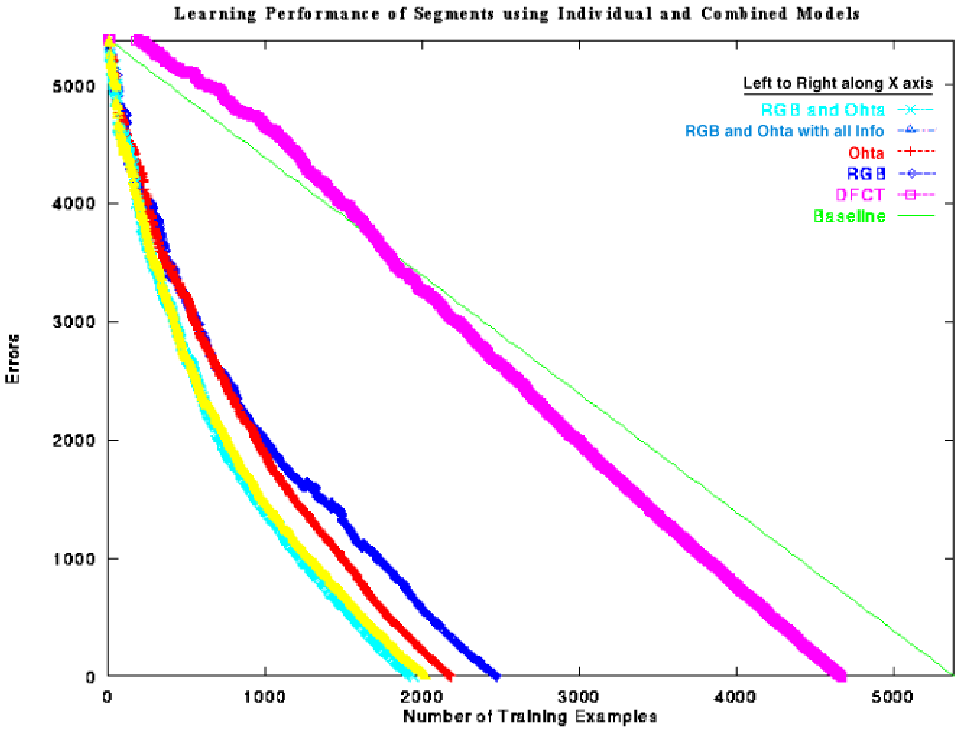


Figure 12. The faster the curve drops the better the performance. The graph illustrates the performance of individual and combined models against baseline. The graph also illustrates the negligible effect of high-level information on performance of the combined RGB and Otha model. [To identify the curves correctly, note that the key lists each curve as it intersect the X axis in left to right order.]

the day number, the time of day, and the location of the scene. For instance:

"SHOW OPEN STAND-UP #1 INT. COMEDY CLUB - NIGHT"
"ACT ONE SCENE A INT. JERRY'S APARTMENT - DAY (1)"
"ACT ONE SCENE B EXT. NEW YORK STREET - DAY (1)"
"ACT ONE SCENE C INT. POLICE STATION - DAY (1)"
"ACT ONE SCENE D EXT. NEW YORK STREET - DAY (1)"

Since the regions of interest have membership in a particular setting or location, the expectation of a particular labeled region can be modified according to this knowledge.

2) Scene Membership Information: Using the knowledge in the script of who is in each scene, the system can exclude those actors who are known *not* to be in the scene from the classification task.

3) Entrance/Exit Information: The information in the script on the exit or entry of a particular character can be used to further limit the classifications on a given region. Prior to a character's entrance, it is possible to exclude them from the classification task. As cited earlier, although George speaks on the intercom in Shot 21, he is not physically in the apartment until Shot 32. So none of the extracted regions until Shot 32 should be classified as George.

Figure 12 illustrates that the incorporation of high-level information did not increase learning performance on the combined RGB and Ohta model. The difference between learning performance with and without high-level information is small enough to be considered insignificant for the segment classification task on these data. In investigating why this was the case, we inspected the similarity trees and observed that segments tended to be clustered within shots. Since the high-level information available was relevant at the scene classification level, the performance on segment classification could not be expected to be enhanced.

4.6. Learning Shots with Particular Actors: Preliminary Results

We expect a common use of this system would be for browsing the video for particular characters. Therefore we undertook a preliminary set of experiments to evaluate how accurate the system was at this task. Each shot was classified by learning whether its member segments were regions of each character. If a shot contained one or more segments with a character's name it was classified as containing that character. Since a shot can contain segments from multiple characters, these were not mutually exclusive tests. For example, a shot with Kramer could not be used as a negative example of a shot for any other characters. The rate of learning these shot classifications was slower than for learning the classification of individual segments. This is because the rather optimistic symmetric set cover algorithm was penalized for having to learn the mutual exclusivity of segments in each shot. The FourEyes learning algorithms learn fastest when classes are mutually exclusive, i.e., when a positive example for one class is a negative example for all the other classes.

Nevertheless, these preliminary experiments indicated that the system still learned, and also that learning performance was marginally enhanced when high-level information was incorporated. Further these experiments indicated that the more high-level information present, the better the learning performance. Incorporating all three types of available information described above (Location and Setting, Scene Membership, Entrance/Exit) improved learning performance the most.

When a browser responds to a user's query, it is often instructive for the shots to be ranked in relevance order for retrieval. Although this is a subjective criterion in the classification of shots by character, the relative presence of the character may be used and can be evaluated by several means. A simple measure would determine the popularity of a given actor in a shot by normalizing the number of segments recognized as the character to the total number of segments in the shot, or to the total number of other segments of actors known to be in the shot. Another simple measure would determine whether the character had a caption during the shot and rank the retrieved shots according to the premise that characters that have captions are significant to the shot. A more complex measure might gather statistics on the temporal and spatial co-occurrence of segments. An hypothesis is that an actor is more significant to a shot if the segments associated with him clump together in time or space, than if the same number of segments were evenly distributed throughout the shot. For instance, if a Kramer segment shows up in an isolated frame his 1/30th sec appearance may not be significant to a viewer. Further research is necessary to explore these measures of proposed relevance.

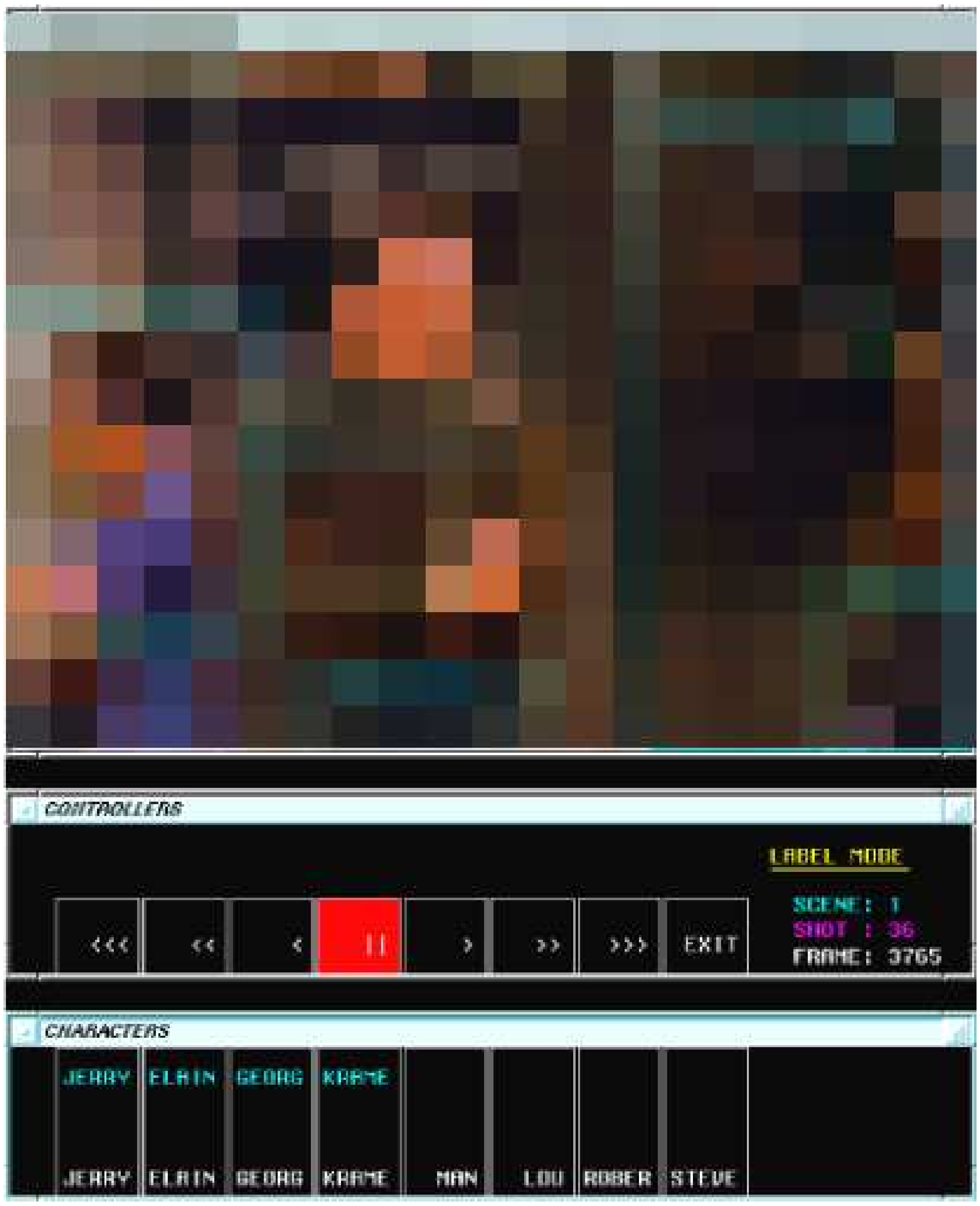
4.7. Discussion of the Classification Results

In a random database of still images, the ability to label a single image region and have the system learn relationships with other images in the 2.7:1 ratio achieved in this data-set would be a great starting point. This grouping across images means the similarity metrics are robust enough to find salient relationships among several image segments. However, in a video database, across image groupings are less significant unless the related images occur in shots, scenes, or episodes other than in the example shot. Since a shot is a coherent sequence composed of slowing changing images, it is important for the similarity metrics to be able to form groupings that span shots. However, for much of the data, across shot groupings of segments was poor. This means that 1) the similarity trees splintered along shot boundaries and the models did not characterize the data well, 2) ground truth was gathered imprecisely or inconsistently by the subjects, and/or 3) the quantization error in building segments added too much noise. There are three ways to improve the performance: 1) identify a better set of similarity measures for the data set 2) integrate more high-level attributes which coerce the low-level attributes to be grouped together, and 3) collect ground truth more precisely. All are equally valid directions for further development.

From a human perceptual standpoint characters do not vary their appearance radically between scenes, let alone between shots. George is still George even if he changes his pose or clothes. Humans have robust cognitive models of what it means to appear like a given person. The computer's representation is much more brittle. A more explicit model might also improve performance, but would sacrifice generality.

4.8. A Smarter Fast Forward: Advanced Video Browser

Since individual regions can be learned through human-computer interaction, a video browser can be constructed that indexes shots by actors even when the actors do not have a caption. Figure 13 is a screen dump of the Advanced Video Browser which illustrates what a user would see when teaching the computer by example. Figure 14 shows key frames the computer retrieved on shots, given that it learned the classification of all character segments. The key frames returned represent a substantial improvement over the Basic Video Browser because the shots shown do



[Image Courtesy of Castle Rock Entertainment]

Figure 13. Screen dump of Advanced Video Browser demonstrating the ability to find shots in which Kramer did not have a caption, but in which he appears. This browser is more powerful than the Basic Video Browser because regions have been learned by *FourEyes*. Note that the names of the four characters known to appear in the scene are highlighted on the Character selection interface.

not require a script or caption in order to be retrieved. In addition, the system, guided by human interaction, has learned the correct representation.

[Images Courtesy of Castle Rock Entertainment]

Figure 14. Key frames returned on a query "Find Shots with Kramer." Using the Advanced Video Browser shots are retrieved in which Kramer appears but does not necessarily have a caption. The last image, set off by itself, is a shot in which Kramer appears (if you look very carefully you can see Kramer's back in the bottom left side of the picture), but could not be retrieved by the system. Since during the preprocessing stage, no luminance or motion segments isolated Kramer in this shot he is effectively invisible to the system.

5. SUMMARY AND FUTURE WORK

5.1. Summary Conclusion

We have presented a general purpose video browser that learns about its content by integrating high-level narrative elements with low-level signal features and user interaction. The system helps people browse video in real time by enabling them to fast-forward (or rewind) to the next frame, shot, or scene, with constraints on who is present in the selected footage. The system learns to improve its performance over time as it interacts with people.

In preliminary experiments learning performance was enhanced, albeit slightly, by the integrated knowledge provided by the high-level script and closed caption information. A strength of the work is its ability to configure itself in an unsupervised manner by preselecting regions of significant change. A further contribution of the work is the human-computer interaction and feedback which makes he system responsive to perceptual similarities even when they do not correspond to statistical similarities.

At the outset of this research effort it was unclear, what if any influence high-level script and closed caption information could have on learning performance.

The performance of the Basic Video Browser demonstrated that a useful browser could be constructed, relying almost exclusively on commonly available high-level information. Although preliminary experiments suggest that the incorporation of low-level information could increase accuracy and functionality of the Advanced Video Browser, more experiments are needed to state conclusively how much the performance increases by these means. Another contribution of this work therefore, is the definition of a role which high-level information could play in video annotation and computer assisted learning.

In addition, this preliminary work in interactive learning as a methodology to assist video pattern recognition tasks has highlighted the need for better image feature analyzers and improved labeling schemes to train the computer. In short the problem domain is a fertile area of research. Computer assisted learning of video patterns shows promise, and more research is needed to realize its potential.

5.2. Future Work

Among other avenues, future work could explore whether finer grained or multiply labeled segments improves learning performance. Perhaps hierarchical labels such as "George' jacket", or "Homeless man's Chinese food." could be generated for each segment. Then similarity measures which corral all of the jackets or Chinese food boxes together could be identified. In this regard, it is instructive to think of the representation of a person as a collection of dissimilar objects or patterns. Hands, faces, sweaters, may individually be identifiable as relatively homogeneous patterns, but collectively the distribution of these dissimilar patterns is a meta pattern which may be recognizable as the presence of a specific person. Consider the representation of the similarity tree where each branch is a set of similar stuff. One could imagine constructing a *forest* of similarity trees where a meta pattern unique to an individual is formed. The goal would be to compare a "Jerry forest" to an "Elaine forest" in an attempt to isolate characters in unlabeled video.

It is not difficult to envision future computer systems watching television and automatically constructing associations between visual, textual and auditory patterns. In this scenario, patterns of video information could be automatically associated with textual concepts, and computers would be empowered to learn about the world as represented by Television. Since vast archives of data already exist in the scripted and closed caption format and new episodes are generated weekly, closed captioned television may offer an untapped resource for automatic computer learning.

Acknowledgments

This research was originally reported in the form of a MIT Masters Thesis entitled *A Video Browser That Learns by Example* by Joshua Wachman, advised by Prof. Rosalind Picard. (Also available as *MIT Media Laboratory Technical Report #383*). The design and evaluation of benchmarked learning tests with *FourEyes* was a collaborative effort between Thomas P. Minka and the authors.

Minka adapted the *FourEyes* system to communicate with the video browser and also made positive contributions to the research in style and substance. The authors gratefully acknowledge Minka's substantial contributions. Additional thanks go to Dr. HongJiang Zhang and Dr. Andrew Lippman who read and commented on early versions of the related thesis. This work was supported in part by IBM and British Telecom.

Appendix Episode Summary *Seinfeld The Beard*

Excerpt from *Seinfeld The Beard*

As broadcast Feb. 9, 1995 Written by Carol Leifer, Directed by Andy Ackerman #04-0615

The following episode summary is from the *Seinfeld* home page:
http://www.spe.sony.com/Pictures/tv/Seinfeld/Seinfeld.html

"While posing as a beard for a gay acquaintance needing an escort, Elaine becomes infatuated with the handsome man and sets out to convert him to heterosexuality. Meanwhile, George–now sporting a toupee–lands a date with a beautiful friend of Kramer only to find that she's covering up a cosmetic problem too. Also, Jerry frets when a female police officer he likes wants to give him a lie-detector test that could reveal his secret TV viewing habits."

References

1. B. Astle. Video database indexing and method of presenting video database index to a user. US Patent Office 5,485,611, January 1996. Assigned to Intel Corporation of Santa Clara CA, Filed Dec. 30, 1994.
2. S. Chang, J. Smith, and H. Wang. Automatic feature extraction and indexing for content-based visual query. Technical Report 408-95-14, Columbia University, New York, NY, 1991.
3. R. K. Srihari et. al. Use of collateral text in image interpretation. In *CEDAR Proceedings of The Image Understanding Workshop, Vol II*, pages p. 897–907, Monterey, CA., Nov 13-16 1994. ARPA Software and Intelligent Systems Technology Office.
4. S.S. Intille and A.F. Bobick. Visual tracking using closed-worlds. Technical Report 294, MIT Media Laboratory Perceptual Computing, 20 Ames Street, Cambridge, MA 02139, 1994.
5. A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.
6. V. M. Bove Jr. Personalcasting: Interactive local augmentation of television programming. Master's thesis, MIT, 1983.
7. K. Karahalios. Salient movies. Master's thesis, MIT, Cambridge, MA 02139, 1995.
8. B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages p121–130, April 1981.
9. J. Meng and S. F. Chang. Tools for compressed-domain video indexing and editing. *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology — Storage & Retrieval for Image and Video Databases IV*, 2670, February 1996.
10. T. P. Minka. An image database browser that learns from user interaction. Master's thesis, MIT, Cambridge, MA 02139, February 1996. Also appears as MIT Media Lab Perceptual Computing Section Technical Report #365.
11. T. P. Minka and R. W. Picard. Interactive learning using a 'society of models'. *Special Issue of Pattern Recognition on Image Databases Classification and Retrieval*, 1995. Also appears as MIT Media Lab Perceptual Computing Section Technical Report #349.

12. Y.I. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13:222–241, 1980.
13. R. W. Picard and T. P. Minka. Vision texture for annotation. *ACM/Springer-Verlag Journal of Multimedia Systems*, 3:pp 3–15, 1995. Also appears as MIT Media Laboratory Perceptual Computing Section Technical Report #302.
14. B. Salt. *Film Style and Technology History and Analysis.* Starwood, London, 1983.
15. R. K. Srihari. Linguistic context in vision. In *Proceedings*, pages p78–88, MIT, Nov 10-12 1995. MIT AAA-I Fall Symposium Series Computational Models for Integrating Language and Vision.
16. M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
17. A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.
18. J. Y. A. Wang and E.A. Adelson. Layered representation for motion analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, June 1993. Also appears as MIT Media Lab Perceptual Computing Section Technical Report #221.
19. M. M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *Proceedings IEEE International Conference on Image Processing Vol 1.*, pages p338–341, Washington, D.C., October 23-26 1995. Princeton University.
20. H. Zhang, A. Kankanhalli, and S. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1:10-28, 1993. Springer-Verlag.
21. H. Zhang, C. Y. Low, and S. Smoliar. Video parsing and browsing using compressed data. *Journal of Multimedia Tools and Applications*, 1(1):89–111, March 1995.