Comparison of Two Methods in Detecting Leno Talk Shows using Pattern Recognition

Joshua Burbrink
Rose-Hulman Institute of Technology

Justin Miller
Rose-Hulman Institute of Technology

Matthew Boutell, Ph.D.
Rose-Hulman Institute of Technology

Introduction

This paper presents two alternative approaches to detecting images taken from videos of Leno talk-shows: a Support Vector Machine (SVM) and an Eigen-classifier based on principal components analysis. On a testing set of 2952 images collected from 88 videos, the SVM approach produced an experimentally calculated 90.41% accuracy using color features. On the same set, the Eigen-classifier produced 97.37% accuracy employing thresholds derived from Eigen images. The paper describes strengths and weaknesses of both methods, as well as their potential use on the difficult problem of video copyright violation detection.

As internet based video websites, such as YouTube and Metacafe, continue to increase in popularity, the need for systems to detect and remove copyrighted material increases. Copyright owners often create their own websites to display advertisements along with their videos to earn money. Allowing the same material to remain on social video websites can reduce traffic to, and thus revenue earned from, the owner's website.

This paper focuses on one small example of copyrighted material: Jay Leno of the Tonight Show performing interviews. Due to the relative consistency, both in background and layout, of the images captured from Tonight Show videos, frames of these videos can be identified by both color analysis and principal components analysis. Using the ideas presented in this paper, it may be possible to develop a system that could detect almost any relatively static video clips, such as other talk shows, news programs, or other various shows with a consistent background.

Method

Our first method used spatial color moment features and a Support Vector Machine (SVM). We first split each image using an n x n grid into n^2 blocks of pixels; we used n=7, but the classifier is robust to changes in n. For each block, we calculate the first two moments (mean and variance) of each of the three color bands. This yields a feature vector with 7 x 7 x 3 x 2 = 294 dimensions. Intuitively, the means correspond to a low-resolution version of the image, and the variance to a coarse measure of texture. These features are normalized to the range [0,1] over the entire data set. These image features are then classified by a SVM. During training, SVMs use kernel functions to map each image's features to a higher-dimensional space to find a hyper

plane which will separate the images into classes. SVMs are designed for two-class problems, and output a real number for each image. If the output is thresholded at 0, the sign is the classification and the magnitude can be used as a loose measure of the confidence.

Our second method used Principal Components Analysis (PCA). PCA attempts to capture the directions of greatest variance in the data set. Projecting images onto the subspace spanned by these dimensions yields a lower-dimension representation of the data.

We represent each image in a training set of S images by $X_i = [x_{i.1}, x_{i,2}, \ldots x_{i,n}]$, $1 \leq i \leq S$, where *n* is the number of pixels in the image, which is typically very high. The training set is a matrix $X = [X_1, X_2, \ldots X_T]^T$. The mean image $m = [\mu_1, \mu_2, \ldots \mu_n]$ is computed, where $\mu_j = \Sigma_i x_{i,j}$. A matrix M is created, which has S copies of *m* as row vectors: [m; …; m]. Then the covariance matrix, C, of the data set is calculated as

$$C = (X-M)^T(X-M). \quad (1)$$

The eigenvectors of C form a basis for X, but those corresponding to the largest eigenvalues give the directions of greatest variability. This fact can be exploited to reduce the dimensionality of a data set, by keeping only those *d* ($d << n$) eigenvectors of highest variability, and projecting the data into the *d*-dimensional subspace (an "eigenspace") spanned by those eigenvectors. The eigenvectors are of dimension *n*, and can be represented as images: Figure 1 shows the three eigenvectors corresponding to the three largest eigenvalues of our training set.

Figure 1: First Three Principal Components from Leno Image Set

Because the eigenvectors form a basis, any image in the training set can be reconstructed as a linear combination of the mean image and the *n* eigenvectors. A linear combination of the mean and the top *d* eigenvectors gives a *projected-image*, an approximation of any image in the training set. Any image in the training set will be close to its projected image, but those *not* the training set will tend to lie further from their projected images, since they will vary in different ways than those in the training set.

The distance from an image to its projected image indicates how closely related the image is to the training set. We exploit this property to build a classifier to distinguish between two different types of images by simply thresholding the distance between a test image and its projected image; we dub this classifier an "eigen-classifier".

Related work

Support vector machines using spatial color moments have been used for sunset detection[2], and later extended to various types of outdoor scenes[1]. They are robust to small changes in

background, which is desirable for talk shows, since the camera angle often changes slightly. Classifiers based on creating images of the principle components of a distribution have long been used for face recognition; both to distinguish between known faces and between faces and non-faces[4]. This method was effective at classification, even when few faces were used in training; however, the data sets contained only mug shots, and thus were highly constrained. Our work differs in that there are several types of images (e.g., monolog, band shots, and interviews); however, within each type, camera angle and background are relatively static, so individual classifiers can be created for each type. More recently, PCA has been applied to sets of outdoor images taken from webcams over several months[3]. While the content is relatively static, the analysis reveals variations due to the time of day, season, and weather. Our application is applied to video, where the content is moving, but the length of time between images is much shorter.

Experimental setup

Because the focus on this project is detection of copyrighted material, we collected videos from two social video websites: YouTube (85 videos) and Metacafe (3 videos), of the types and numbers shown in Table 1. All the talk show videos, regardless of the host, were collected by performing keyword searches; additionally, Leno videos were required to be of an interview with a guest. The non-talk show videos were collected by downloading the most recent videos from each genre catalogued by YouTube to ensure the data sample was diverse.

Individual frames were extracted from the videos at a rate of one frame per three seconds. This provided enough training data, while having much lower correlation between successive frames than would be the case had we sampled at higher frame rates. The frames were inspected to verify the process was successful: any frame from a Leno video that did not actually show the host behind his desk, the guest sitting, or the host and the guest sitting were removed the list of Leno frames. Frames that showed a transition, videos of movies, and other broadcasts were also removed. Table 1 shows the number of frames of each type in the training and testing sets. Since the purpose of our classifier was to detect Leno interviews, we marked those as positive examples, and all others, including other talk shows, as negative examples.

Table 1: Data Set

	Video Count	**Training**	**Testing**	**Total:**
Leno Interview (positive)	18	1099	645	1744
Other Talk Shows (negative)	77	448	904	1352
Other Videos (negative)	33	694	1385	2079
Total:	88	2241	2934	5175

We extracted spatial color moments from each of the frames in the data set. We then trained an SVM as discussed above, using a radial basis function kernel with a width of 12.33. The resulting SVM had 118 support vectors (5.3% of the frames in the training set), 47 of which corresponded to Leno interviews.

We used two separate eigen-classifiers in our work. One was trained on frames that depict a straight shot of the guest only on a Leno show, and one was trained on frames containing both the guest and Leno. We used d = 100 for each. We chose the thresholds for each by using

validation sets; we calculated the distance between each frame in the validation set to its projected-frame and chose a threshold that maximized the true positive rate and minimized the false positive rate on the validation set. This yielded thresholds of 80,000 and 90,000 for the two eigen-classifiers, respectively. We combined the results of the two classifiers simply by calling a test frame a Leno interview if either of the eigen-classifiers classified it as such.

Results and discussion

We classified each of the 2934 frames in the test set as Leno interview or non-Leno-interview as discussed above. We set the threshold on the SVM output to be 0, as discussed earlier. Table 2 compares results from both the SVM and eigen-classifiers, using four measures: accuracy (percentage correct), true positive rate (percentage of Leno interviews detected), false positive rate (percentage of non-Leno interviews incorrectly detected), and precision (percentage of those frames detected as Leno interviews that actually are Leno interviews).

Table 2: Results for the SVM and Eigen-classifiers

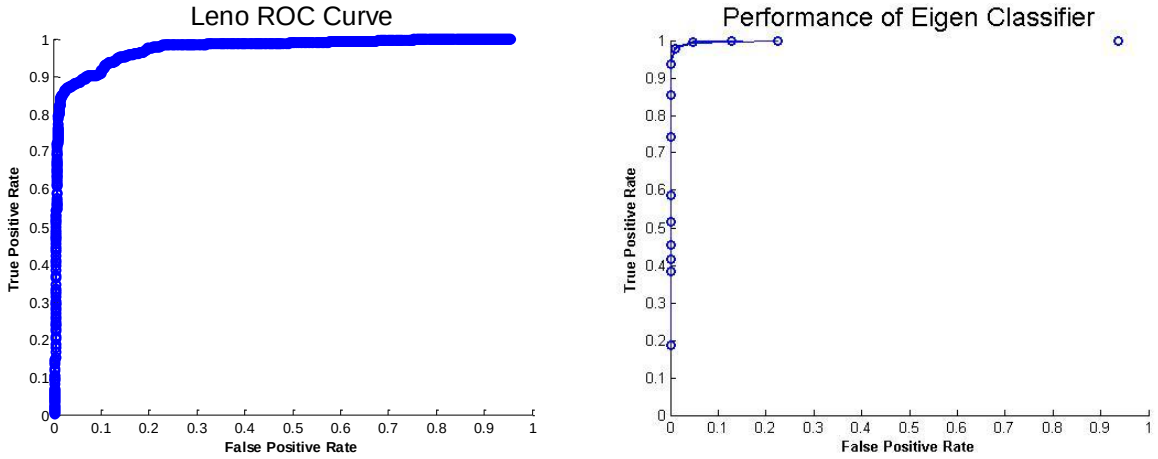
Statistic	SVM	Eigen-classifier
Accuracy:	90.41%	97.37%
True Positive Rate:	90.85%	99.28%
False Positive Rate:	9.71%	3.21%
Precision:	72.35%	90.33%

We see that both methods were successful, obtaining accuracies of over 90%, likely because our assumptions of a static background held true with very few exceptions, like when the show would change backgrounds (such as Christmas episodes and seasonal differences). Slight variations due to the operator bobbling the camera slightly did not affect the classifiers. The results are also inflated due to some cases in which the same guest appeared in the training and the test set. (While identical frames were not used in both sets, some guests did appear in both sets.)

However, there are two limitations of the current system. One is the lack of frames containing a Jay Leno alone; ironically, this causes the current system to fail to detect Leno himself! (Our test sets do not include these types of frames.) The other is the inability to handle significant changes of background or new camera angles not encountered in training. Advertisements of future shows, weather alerts, and other insertions by video editors also contribute to misclassification.

We can obtain a more comprehensive understanding of the performance of each classifier by looking at its ROC curve (Figure 2). This curve is obtained by repeatedly shifting the thresholds in either direction, to increase (or decrease) true positive rate while increasing (or decreasing) the false positive rate.

Figure 2: ROC Curves of Both Approaches



While both classifiers were successful, we see that the eigen-classifier had a higher true positive rate than the SVM classifier for any given false positive rate, and so appears to be better suited for identifying Leno images than the SVM classifier.

Finally, we observed that the eigen-classifier was much more efficient, classifying frames more quickly than the SVM.

Conclusion and future work

We have demonstrated that both the SVM approach and the eigen-classifier approach are successful at classifying individual frames of videos as Leno or non-Leno, with accuracies of 90.4% and 97.4%, respectively.

This work could be extended in many ways. First would be to implement the eigen-classifiers in a real world application. Because of its relatively high accuracy and true positive rate, it could be included into a web crawler that searches video websites for copies of specific shows. Of course, false positives must be minimized in such a system, as they would be very inconvenient. A simple way to do this is to classify each frame in a video, and only classify the video as a Leno show if a relatively-high percentage of frames were detected as Leno interviews. (If a lower, but still significant percentage of frames were detected, then it could be presented to the operator of the system as a potential infringement of copyright, and then manually checked. This is still much less tedious than checking every video manually.) This could be made more robust by computing the average distance of all the frames from the thresholds as well. Using a collection of these crawlers, a company could easily alert video websites to the copyright violations and have the videos promptly removed. The company could also detect potential copyright violations at upload time.

Second, we hypothesize that similar systems, trained appropriately, could detect videos of other copyrighted, static television programs, but would like to demonstrate it experimentally. We would need to increase the types of videos used to train the system to see how it scales.

References

1. Boutell, M; Choudhury, A; Luo, J; and Brown, C. "Using semantic features for scene classification: How good do they need to be?" In *Proc. IEEE International Conference on Multimedia and Expo*, Toronto, ON, July 2006.

2. Boutell, M; Luo, J.; and Gray, R.T. "Sunset scene classification using simulated image recomposition," In *Proc. IEEE International Conference on Multimedia and Expo*, Baltimore, MB, 2003.

3. Jacobs, N.; Roman, N.; and Pless, R. "Consistent temporal variations in many outdoor scenes. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, June 2007.

4. Turk, M. and Pentland, A. "Eigen faces for Recognition. Journal of Cognitive Neuroscience", 3(1), pp. 71-86, 1991.

Joshua Burbrink (burbrijw@rose-hulman.edu) is a third-year Computer Science major at Rose-Hulman Institute of Technology in Terre Haute, Indiana.

Justin Miller (millerj6@rose-hulman.edu) is a third-year Computer Science and Mathematics double major at Rose-Hulman Institute of Technology in Terre Haute, Indiana.

Matthew Boutell, Ph.D. (boutell@rose-hulman.edu) is Assistant Professor of Computer Science and Software Engineering at Rose-Hulman Institute of Technology in Terre Haute, Indiana. His research interests include image understanding, machine learning, probabilistic modeling, and computer science education.