Eigenfeatures coding of videoconferencing sequences

Marcia G. Ramos and Sheila S. Hemami

School of Electrical Engineering, Cornell University, Ithaca, NY 14853

phone: (607) 255-8836, fax: (607) 255-9072
e-mail: mramos/hemami@ee.cornell.edu

ABSTRACT

This paper presents a video coding technique that achieves high visual quality at very low bit rates. Each video frame is divided into two regions, consisting of a background area and a visually important feature to be coded at higher bit rates. The feature is tracked from frame to frame and is coded using a set of features that are extracted from a training set. The set of features, which will be referred to as eigenfeatures, is stored both at the encoder and decoder sites. The technique is based on the eigenfaces method proposed in [1], and achieves high visual quality at high feature compression ratios (around 200 for the salesman sequence and 1000 for the Miss America sequence) with considerably less computational complexity than the eigenfaces method. Using this technique for the feature together with H.261 for the background allows a reduction of up to 70% in the bit rate compared to using H.261 alone.

Keywords: Low bit-rate video, principal components, eigenfaces, eigenfeatures

1. INTRODUCTION

Video compression techniques to date have treated the entire image equally. However, in most videoconferencing sequences, the image consists of a background and a feature of interest, such as the speaker’s face. Consequently, to maximize the visual quality, more bits should be allocated to the feature of interest and fewer bits to the remainder of the scene. By tracking a specific feature such as the face of the speaker and applying a higher bit rate to describing that feature, a higher quality coded video sequence results.

One technique for compression is a method known as principal components in which a feature is coded based on a set of features that are extracted from a training set. The set of features, which will be referred to as eigenfeatures, based on the eigenfaces method proposed in [1], is stored both at the encoder and at the decoder sites. The primary difference between the eigenfeatures technique presented here and the eigenfaces method is that the eigenfeatures are constructed by a considerably simpler and more computationally efficient approach than the eigenfaces. The feature in the current frame is coded as a linear combination of the eigenfeatures so only the coefficients of the linear combination are transmitted to the decoder. The quality of the reconstructed frame is controlled by allowing an
update of the *eigenfeatures* set when necessary; for example, if the MSE of the coded feature is above a threshold, then the set is rebuilt and the feature is recoded. In this way, the quality of the coded feature is maintained.

This paper is organized as follows. Section 2 describes the tracking mechanism used to identify the features in each frame of the video sequence, Section 3 gives an overview of the *eigenfaces* method for recognition, and Section 4 presents the *eigenfeatures* technique for video coding. Finally, Section 5 presents the experimental results and Section 6 concludes the paper.

## 2. Feature Tracking

By devoting the majority of available bits to describing the feature of interest, the perceived quality of the videoconferencing sequence is increased. However, this benefit requires increased computation. Once the feature’s size and location are specified, the feature must be followed as it translates, rotates, and even deforms. Feature tracking is a well-known challenge in computer vision.

Tracking begins by examining a feature from the previous frame in an attempt to recognize it in a new frame. The feature in the previous frame is subtracted from a test location in the present frame to produce a numerical match value. By assuming that the feature of interest does not move more than *N* pixels between frames, only a subset of the current frame must be searched. This subset is called the search window. The coordinate with the highest correlation to the feature in a previous frame is designated the feature location within the present frame. The correlation $M(i,j)$ is defined as the L1 norm between the corresponding features of adjacent frames at location $(i,j)$.

An exhaustive search evaluates every location in the search window. The computational complexity is a function of both the size of the search window and the frame rate. A feature moving no more than 4 pixels horizontally or vertically translates into evaluating 81 possible correlating locations at the frame rate. To reduce the intensive amount of computations, a 2D logarithmic search procedure is used. The algorithm tracks the direction of minimum distortion, resulting in considerable computational reduction. The algorithm is described as follows:

1. The feature characterized by its upper left corner pixel $(i, j)$ is compared to the corresponding feature in the previous frame. If $M(0,0) < T$ where $T$ is a threshold, then the feature is classified as stationary and the search ends. Otherwise, go to step 2.

2. The next pixel positions searched are $(i-4, j)$, $(i, j+4)$, $(i+4, j)$, and $(i, j-4)$. If the minimum from these searches is greater than $M(0,0)$, go to step 4. If not, and a minimum is found for these pixel positions, then this value is compared against $T$. If the value is less than $T$, the search ends. Otherwise, go to step 3.

3. Assume that, from the previous step, the pixel position that had the minimum mismatch was at $(i-4, j)$. The next positions of search are $(i-4, j-4)$, and $(i-4, j+4)$, respectively. The test for a minimum and the threshold are performed again. If the test is true, the search ends. Otherwise, go to step 4.

4. Repeat set 2 for offsets of ±2 pixels.
5. Repeat step 3 for offsets of ±2 pixels.

6. Repeat steps 4 and 5 for offsets of ±1 pixel.

Note that this tracking approach is equivalent to motion estimation on the whole feature rather than on blocks as in MPEG.

Figures 1 and 2 show the results of this tracking scheme to track the head on the salesman sequence. To reduce the amount of computation required, it is assumed that the head does not move more than 4 pixels from frame to frame. This algorithm can also be used to track the mouth or the eyes of the speaker, or any other feature of interest.

Figure 1: Frame 0, salesman sequence, feature to be tracked

Figure 2: Frame 30, salesman sequence
3. EIGENFACES FOR RECOGNITION

Using principal components to code facial images originated in a technique developed by Sirovich and Kirby\(^3\) and Kirby and Sirovich\(^4\). Starting with an ensemble of original facial images, a “best” coordinate system for image compression is calculated, where each coordinate is actually an image termed an eigenpicture. In principle, any collection of face images can be approximately represented by a collection of weights for each face and a set of standard pictures (the eigenpictures). The weights associated with each face are found by projecting the face image onto each eigenpicture and the reconstructed face is formed as a linear combination of the eigenpictures, weighted appropriately.

Based on the eigenpictures work by Kirby and Sirovich, Pentland and Turk\(^1\) developed a face recognition system called eigenfaces for recognition. Since a multitude of face images can be represented by weighted sums of a small collection of characteristic features or eigenfaces, an efficient way to learn and recognize faces is to build up the characteristic features by experience over time, i.e., by training, and to recognize a particular face by comparing its weights (as found from projection onto the eigenfaces) with the weights associated with known individuals. Each individual, therefore, would be characterized by the small set of feature or eigenpicture weights needed to describe and reconstruct him/her - an extremely compact representation when compared to the images themselves. Pentland’s and Turk’s method to face recognition involves the following initial steps:

1. Acquire an initial set of \(M\) face images (the training set).
2. Calculate the eigenfaces from the training set, keeping only the \(N < M\) eigenfaces that correspond to the highest eigenvalues. These \(N\) eigenfaces define the face space. As new faces are experienced, the eigenfaces can be updated or recalculated.
3. Calculate the corresponding distribution in \(N\)-dimensional weight space by projecting their face images onto the face space.

The computation of the eigenfaces is based on principal component analysis (or Karhunen-Loeve expansion) which finds the optimal basis for representation of the training face space in the mean squared error sense. Each vector is of length \(WH\), describes a \(W \times H\) image, and is a linear combination of the original face images in the training set. The eigenfaces are calculated in the following way: Let the training set of face images be \(I_1, I_2, I_3, ..., I_M\). The average face of the set is defined by \(\Psi = \frac{1}{M} \sum_{n=1}^{M} I_n\). Each face differs from the average by the vector \(\Phi_i = I_i - \Psi\). This set of very large vectors is then subject to principal component analysis to identify a set of \(M\) orthonormal vectors, \(u_n\), which best describes the data.

The vectors \(u_n\) are the eigenvectors of the covariance matrix:

\[
\text{Cov} = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^t = AA^t
\]

where \(A = [\Phi_1 \Phi_2 ... \Phi_M]\). A computationally feasible method to compute these eigenvalues and eigenvectors is given in [1], by observing that \(u_k\) can be computed as

\[
u_l = \sum_{k=1}^{M} v_{lk} \Phi_k
\]
where \( l = 1, ..., M \) and \( v_k \) are the eigenvectors of the matrix \( L = A^tA \) which is a \( M \times M \) matrix. The *eigenfaces* are then chosen as the \( N \) vectors \( u_k, k = 1, ..., N \) that correspond to the largest \( N \) eigenvalues of the matrix \( L \).

A new face \( I \) is coded as a vector of weights \( w \) that describes the contribution of each *eigenface* in representing the difference between the face \( I \) and the average face:

\[
w_k = u_k^t(I - \Psi)
\]

The vector is then used to reconstruct the face \( I \) by:

\[
I = \Psi + \sum_{i=1}^{N} w_i u_i
\]

where \( N \leq M \) is the number of *eigenfaces* desired to represent the training set.

### 4. EIGENFEATURES APPROACH

The *eigenfaces* method developed for face recognition can be used to compress features of videoconferencing sequences since the feature can be described as a small set of weights only. Welsh and Shash 4 used this method to code mouth images with reasonable quality at intraframe compression ratios of about 400.

A considerably simpler principal components technique is presented here, which avoids generating the average signal and forming the difference signals to be coded. Instead, the signal is directly coded. Considering the image to be a signal belonging to a signal class, the principal components of the signal class (the training set of images) are computed to develop the *eigenfeatures*.

The *eigenfeatures* themselves are determined by the dominant eigenvectors of the sample correlation matrix \( C \) of the signal class. Since the higher principal components, i.e., the eigenvectors corresponding to the larger eigenvalues of \( C \), contain more information about the signal class than the lower ones, compression is achieved by discarding the lowest principal components of the class. Since the matrix \( C \) is also \( WH \times WH \) and has a rank of at most \( M \), computing the eigenvectors of the \( C \) matrix can be reduced to finding a \( M \times M \) eigendecomposition by doing a simple QR factorization 6.

First define the matrix \( I = [I_1I_2I_3...I_M] \) where the columns are the vectorized feature images. The sample correlation matrix can then be expressed as \( C = II^t \) and using QR factorization, \( C \) is written as

\[
C = (QR)(QR)^t = QRR^tQ^t
\]

Now let \( K = RR^t \) and find the \( M \times M \) eigendecomposition \( K = UDV^t \). Therefore,

\[
C = QUU^tQ^t = (QU)D(QU)^t = VAV^t
\]

Since both \( Q \) and \( U \) are orthonormal, their product is also orthonormal. Since \( D \) is diagonal, and since a symmetric matrix has a unique diagonalization, we can see that \( VAV^t \) is no less than the
unique eigendecomposition of $C$, which we obtained by performing a smaller $M \times M$ eigendecomposition, a $QR$ factorization ($O(W^3H^3)$), and two matrix multiplications, resulting in an overall complexity of $\sim O(W^3H^3)$.

A new feature $I$ is coded as a vector of weights $w$ that represent the contribution of each eigenfeature in representing the new feature. The weights are simply the inner products between the new feature and each eigenfeature in the set. The eigenfeatures set can be updated to include the contribution of new features that differ largely from the features in the training set. In order to keep storage costs constant, the new feature is included in the training set by dropping the first feature in the training set. The new set of eigenfeatures is then recalculated. The training set can be selected in real-time as the initial frames of the videoconferencing sequence or it can be computed ahead of time for predetermined features of known speakers.

Complexity Analysis

Computing the $QR$ factorization corresponds to computational savings of $O(MWH)$ over the eigenfaces method because the difference and average features are not calculated. Furthermore, since the signal is coded directly instead of coding the difference signals, fewer principal components are needed to represent the eigenfeatures set than the eigenfaces set as the signal energy is more concentrated in fewer principal components.

Another advantage of coding the signal directly is reflected in the update of the eigenfeatures set, which is much more efficient than in the eigenfaces method. By constructing the training set as $I = [I_M \ldots I_2 I_1]$, including the new feature and dropping the first one corresponds to changing the last column of the matrix $I$. In this way, only the last column of $Q$ and the last element of $R$ will change if the modified Gram-Schmidt algorithm is used to compute the QR factorization. This results in only $O(2WH)$ operations. The eigenfaces method requires recomputing the matrix $L$ resulting in a total of $O(2WHM)$ operations. Therefore, considerably savings are achieved with the eigenfeatures approach.

5. EXPERIMENTAL RESULTS

This method was applied to the salesman (65 frames) and Miss America (51 frames) sequences. For the salesman sequence, the first 30 head features that were tracked with the object tracker described in section 2 were used as the training set, i.e., the signal class, and for the Miss America sequence the first 20 head features were used. The frame and feature sizes for the two sequences are given in Table 1.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Frame Size</th>
<th>Feature Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>salesman</td>
<td>$360 \times 288$</td>
<td>$69 \times 97$</td>
</tr>
<tr>
<td>Miss America</td>
<td>$352 \times 288$</td>
<td>$186 \times 212$</td>
</tr>
</tbody>
</table>

Table 1: Frame and Feature Sizes

To evaluate the required number of principal components to represent the signal class, the MSE was plotted for each basis vector used. The results are shown in Figures 3 and 4. Experiments show that
about 5-10 eigenfeatures are enough to represent the 30 salesman head features in the MSE sense and about 6-9 eigenfeatures to represent Miss America head features. Subjective tests have also shown that using these eigenfeatures result in high visual quality coded feature images. The graphs also show that the maximum MSE for the salesman sequence is about twice the maximum MSE for the Miss America sequence. This is due to the fact that the feature in the salesman sequence is shown against a more textured background than in the Miss America sequence, which has a black and flat background. As the feature in the salesman sequence moves, the objects in the background will be different thus increasing the MSE.

It is desirable to select the number of eigenfeatures below the “knee” of the MSE curves, where the MSE starts to grow exponentially. As the size of the eigenfeatures set is increased, fewer updates will be necessary since more features will be represented in the set. However, this will increase the storage requirements to keep all the eigenfeatures. The visual quality of the video features can be improved at the expense of the number of computations required to build and update the eigenfeatures set and the storage required to keep the set on both encoder and decoder sites.

The next 35 head features of the salesman sequence were then coded using 9 principal components and the next 31 head features of the Miss America sequence were coded using 8 principal components. When a new frame arrives to be coded, the feature is tracked and then the eigenfeatures weights are determined. The quality of the reconstructed features can be controlled by using the MSE as a threshold. If the MSE of the coded feature is above the threshold, then the training set is updated to incorporate the new feature that arrived. The threshold is set just around the “knee” of the MSE curves. A higher threshold leads to coded features with higher MSE but fewer updates to the training set are required. To keep the size of the training set constant when an update is necessary, the first feature used in the training set is dropped and the new one is included. A new set of principal components is then built as described in Section 4, and the subsequent features are then reconstructed based on this new set.

Figures 4 and 5 show an original feature of the salesman sequence and the coded feature with this scheme. Figure 6 and 7 show the original and coded features for the Miss America sequence.
pression ratio was 185.9 for the head feature in the salesman sequence and 1232.2 for the head feature in the Miss America sequence. The compression ratio is calculated by dividing the original number of bits required for each feature with the number of bits used to code the feature (i.e., the coefficient precision multiplied by the number of eigenfeatures). The compression ratio for the Miss America sequence is much higher than for the salesman sequence since the coded feature in the Miss America sequence is about 5.8 times bigger than the coded feature in the salesman sequence.

Figure 5: Frame 46, salesman sequence, original frame

6. CONCLUSIONS
This paper presents a video coding technique based on the *eigenfaces* method proposed in [1] that achieves high visual quality at very low bit rates with considerably less computational complexity than the *eigenfaces* method. The technique proposed achieves high compression ratios for the feature of interest and is targeted for low bit-rate head-and-shoulder videoconferencing sequences. Using the *eigenfeatures* technique to code a particular feature and H.261 to code the rest of the frame results in a reduction of up to 70% as compared to coding the whole frame with H.261. The training set can be selected in real-time as the initial frames of the videoconferencing sequence or it can be computed ahead of time for predetermined features of known speakers. In the first option, H.261 can be used to code the whole frames while the training set is being generated. The best compression/visual quality trade-off is obtained for close-up images in which the size of the feature is only a fraction smaller than the entire frame.

As network speeds increase, higher visual quality than H.261 becomes feasible. H.261-based video codecs operating at 64 Kbits/sec are likely to produce low-quality images; the frame rate is usually low, and perceptually noticeable degradations occur when the motion is large. The high-coding efficiency of the *eigenfeatures* technique presented here permits more frequent transmission of facial features without introducing a large data overhead.

References


Figure 7: Frame 43, Miss America sequence, original frame


Figure 8: Frame 43, Miss America sequence, coded frame