Library-based coding: a representation for efficient video compression and retrieval

Nuno Vasconcelos and Andrew Lippman
MIT Media Lab, {nuno,lip}@media.mit.edu

Abstract

The ubiquity of networking and computational capacity associated with the new communications media unveil a universe of new requirements for image representations. Among such requirements is the ability of the representation used for coding to support higher-level tasks such as content-based retrieval. In this paper, we explore the relationships between probabilistic modeling and data compression to introduce a representation - *library-based coding* - which, by enabling retrieval in the compressed domain, satisfies this requirement. Because it contains an embedded probabilistic description of the source, this new representation allows the construction of good inference models without compromise of compression efficiency, leads to very efficient procedures for query and retrieval, and provides a framework for higher level tasks such as the analysis and classification of video shots.

1 Introduction

The introduction of digital communications and inexpensive computation originated a shift from image representations based on very simple primitives (sinusoids) and operations (filtering and modulation) towards others based on more involved languages, capable of exploiting the statistical characteristics of video sources and the characteristics of the Human Visual System, or providing higher level descriptions of scene content. While this shift allowed more efficient use of the available bandwidth, it has also uncovered a universe of new requirements for image representations. Because digital decoding requires computational capacity at the receiving end of the channel, it leads to smart information and entertainment appliances capable of two way communication under the control of the user [8]. Digital decoders are intelligent, and capable of searching the network for the content that is "just right" for their users. Unfortunately, current digital representations, such as JPEG or MPEG, designed with the sole goal of achieving compression efficiency, are not helpful for this task.

Perhaps due to this, most of the recent effort in the area of content-based retrieval considers this task independently of the issue of bandwidth efficiency. Typical solutions consider a feature space for retrieval which does not overlap with the representation space used for compression, e.g. while compression is based on DCT [6] or wavelet basis functions [1], retrieval is based on color histograms [5] or texture features [7]. Such solutions imply that either the features are pre-computed and stored in addition to the compressed bitstreams - a process which is inefficient in terms of storage resources - or the bit-streams must be decoded and the features computed at the time of query - a procedure which is computationally very inefficient since all the work performed at the time of image encoding is useless for the task of retrieval.

Even when spaces used for retrieval and compression overlap and full image reconstruction is not required (e.g. when wavelet or DCT coefficients are used for retrieval), the process still suffers from inefficiencies: these coefficients are in general a sizeable portion of the bitstream and their decoding can, therefore, still be an expensive task, and it is generally not clear that the feature space associated with them is the most appropriate for statistical discrimination, or that it allows the construction of good models for statistical inference.

The fundamental goal of this work is to restore some of the production options or interactive potential of video by augmenting the representation used in coding and by exploiting the analysis used for compression as a retrieval aid. The notion is that the analysis used in making an efficient coder can potentially provide useful cues to the underlying action in the scene that may facilitate browsing, filtering, sorting or combining sets of moving picture sequences.

For this we introduce a representation, *library-based coding*, which contains an embedded description of the picture content and allows the construction of good inference models without compromise of compression efficiency. This embedded description is compact and can be decoded independently of the bulk of the digital bitstream, leading to very efficient procedures for query and retrieval. Furthermore, because the representation allows the construction of probabilistic models for statistical inference, it provides a framework for performing higher level tasks such as analyzing and classifying video shots. Finally, because it is close to the representations currently used for video compression, it requires only slight alterations to the existing standards.

2 Embedded probabilistic descriptions

A representation capable of supporting content-based queries without full decoding should include a compact, decodable on its own, description of the statistical properties of the image source. Ideally, one would want a complete probabilistic description, such as the probability density function (pdf) of the stochastic process from which the images were drawn, because that would allow retrieval to be based on statistical inferences. For example, given the probability densities associated with M sources $p(\mathbf{x}|\mathcal{S}_i)$, $i = 1, \ldots, M$, their relative probabilities of occurrence $P(\mathcal{S}_i)$, and a set of data $\mathbf{x}$ created from images to be classified as belonging to one of the classes, the task could be performed optimally by using an Maximum a Posteriori Probability (MAP) criterion and Bayes rule, i.e. pick the class i such that

$$i = \arg \max_{i \in \{1,\ldots,M\}} P(\mathcal{S}_i|\mathbf{x}) = \arg \max_{i \in \{1,\ldots,M\}} P(\mathbf{x}|\mathcal{S}_i)P(\mathcal{S}_i). \tag{1}$$

One possible way to achieve this would be to use a non-parametric description of the source, such as a histogram. However, non-parametric descriptions are not compact, implying a degradation of the efficiency of the representation in terms of compression, and it is not clear how they could be used as a part of the encoding procedure, i.e. not simply as overhead. Fortunately, the alternative of using parametric descriptors, in particular mixture densities, satisfies these two requisites, providing a much more efficient solution.

2.1 Parametric modeling, mixture densities, and the EM algorithm

A parametric probabilistic model capable of approximating any probability density is the class of *mixture densities* [9]. A mixture density has the form

$$P(\mathbf{x}) = \sum_{i=1}^{C} P(\mathbf{x}|\omega_i)P(\omega_i), \tag{2}$$

where C is the number of probability classes, $P(\mathbf{x}|\omega_i)$ are the *class-conditional densities*, and $P(\omega_i), i = 1,\ldots,C$ the *class probabilities* ($\sum_{i=1}^{C} P(\omega_i) = 1$). The class-conditional densities can be any valid probability density functions, even though they are in most of the applications (and in the rest of this paper) assumed to be Gaussians. In this case, the mixture density becomes

$$P(\mathbf{x}) = \sum_{i=1}^{C} p_i e^{-\frac{1}{2}(\mathbf{x}_j-\mu_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x}_j-\mu_i)}, \tag{3}$$

where $p_i = \frac{P(\omega_i)}{\sqrt{(2\pi)^n|\Sigma_i|}}$. The mixture is then completely characterized by the parameters $L^{(s)} = \{\mu_i^{(s)}, \mathbf{\Sigma}_i^{(s)}, p_i^{(s)}, i = 1,\ldots,C\}$.

The standard statistical tool for the estimation of mixture parameters is the *Expectation-Maximization* (EM) algorithm [3]. The EM algorithm treats the class assignments (i.e. which class is responsible for each sample) as hidden (non-observed) variables and, given a set of M independent and identically distributed samples $\mathbf{x}_i, i = 1,\ldots,M$, finds the mixture parameters that maximize the data likelihood by iterating between the following steps.

E-step:

$$h_i^m = P(\omega_i|\mathbf{x}^m) = \frac{P(\mathbf{x}^m|\omega_i)p_i}{\sum_{k=1}^{N} P(\mathbf{x}^m|\omega_k)p_k} \tag{4}$$

M-step:

$$\mu_i^{new} = \frac{\sum_m h_i^m \mathbf{x}^m}{\sum_m h_i^m}, \; \mathbf{\Sigma}_i^{new} = \frac{\sum_m h_i^m(\mathbf{x}^m - \mu_i^{new})(\mathbf{x}^m - \mu_i^{new})^T}{\sum_m h_i^m}, \; p_i^{new} = \frac{1}{M}\sum_m h_i^m, \tag{5}$$

where $m = 1, \ldots, M$ and $i = 1, \ldots, C$.

Given a image to compress, the parameter set $L^{(s)}$ can be estimated through the EM algorithm and included in the compressed bitstream leading to a compact, stand-alone description that can be used to perform statistical inferences such as those mentioned above. However, while this would increase the retrieval ability of the representation, it would also compromise its compression efficiency, as this description would amount to pure overhead. The interesting missing link is that mixture parameters are very closely related to the codebooks which form the basis of a representation which is known to be optimal from a compression standpoint: *vector quantization* (VQ) [4].

2.2 Vector quantization

A vector quantizer $\mathcal{Q}$ is a mapping from a K-dimensional vector space of input samples to a finite set of *reconstruction vectors*, usually known as *codevectors* or *codewords*. The set of reconstruction vectors is generally designated by *codebook*. The N-dimensional input vector space is partitioned into a set C of N K-dimensional *regions* $\mathcal{R}_i$, also known as *partitions* or *cells*, and a reconstruction vector $\mathbf{y}_i$ associated with each region.

The non-linearity inherent to the operation of quantization makes it impossible to achieve a single, closed-form solution to the problem of optimal vector quantization. It is however possible to find two *necessary* conditions for optimality by decomposing the problem into two smaller ones: finding the optimal partition for a given codebook, and the optimal codebook for a given partition.

The optimal partition (encoder) for a fixed codebook (decoder) must satisfy the *nearest-neighbor condition*

$$\mathcal{R}_i \subset \{\mathbf{x} : d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \forall j \neq i\}, \tag{6}$$

while the optimal codebook for a given partition must satisfy the *generalized-centroid condition*

$$\mathcal{Q}(\mathbf{x}) = \min_{\mathbf{y}_i}\{E[d(\mathbf{x}, \mathbf{y}_i)|\mathbf{x} \in \mathcal{R}_i]\}. \tag{7}$$

The most popular algorithm for vector quantizer design - the *LBG algorithm* [4] - iterates between these two conditions, which, given a training set $\mathcal{T} = \mathbf{t}_1, \ldots, \mathbf{t}_M$, and assuming the mean squared error distortion metric become, respectively,

$$\mathcal{R}_i = \{\mathbf{t} \in \mathcal{T} : ||\mathbf{t} - \mathbf{y}_i|| < ||\mathbf{t} - \mathbf{y}_j||, \forall j \neq i\}, \tag{8}$$

and

$$\mathbf{Y}_i = E[\mathbf{x}|\mathbf{x} \in \mathcal{R}_i] = \frac{\sum_{j=1}^{M} \mathbf{t}_j S_i(\mathbf{t}_j)}{\sum_{j=1}^{M} S_i(\mathbf{t}_j)}, \tag{9}$$

where the $S_i(\mathbf{t}_j) = 1$ if $\mathbf{t}_j \in \mathcal{R}_i$, and $S_i(\mathbf{t}_j) = 0$ otherwise.

2.3 Relationship between mixture density estimation and vector quantization

To understand the relationship between vector quantization and estimation of mixture parameters we start by re-writing the EM equations for the simpler case of equally likely classes ($p_i = 1/C$), and identity covariances

E-step: **M-step:**

$$h_i^m = \frac{P(\mathbf{x}^m|\omega_i)}{\sum_{k=1}^N P(\mathbf{x}^m|\omega_k)} = \frac{e^{-\frac{1}{2}||\mathbf{x}^m-\mu_i||^2}}{\sum_{k=1}^N e^{-\frac{1}{2}||\mathbf{x}^m-\mu_k||^2}}, \quad (10) \qquad \mu_i^{new} = \frac{\sum_m h_i^m \mathbf{x}^m}{\sum_m h_i^m}. \quad (11)$$

The similarities with VQ design are made clear by the comparison of these expressions with equations 8 and 9. The only difference is that, in the VQ case, the h_i's are thresholded after the E-step so that

$$h_i' = \begin{cases} 1, & \text{if } h_i > h_j \forall j \neq i \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

i.e. the training sample is assigned to the mixture component that has maximum a posteriori probability of having generated it. This transforms equation 10 into equation 8[1], and consequently equation 11 into equation 9. Therefore, given a sample to encode, the optimal codeword to represent it is the mean of the mixture component which has maximum a posteriori probability of having generated the sample.

The results above can be extended to the generic Gaussian case with full covariances and class probabilities. In this case, the relationships between mixture density estimation and vector quantization are the same, but the VQ is optimal under the *Mahanolabis distance*, and has a constraint on its output entropy. I.e. given an input vector $\mathbf{t}$, the optimal partition is the one for which

$$\mathcal{R}_j = \{\mathbf{t} \in \mathcal{T} : ||\mathbf{t} - \mathbf{y}_j||^2_{\mathbf{\Sigma}_j} - \log p_j < ||\mathbf{t} - \mathbf{y}_i||^2_{\mathbf{\Sigma}_i} - \log p_i, \forall i\}, \quad (13)$$

where

$$||\mathbf{t} - \mathbf{y}_j||^2_{\mathbf{\Sigma}} = (\mathbf{t} - \mathbf{y}_j)^T \mathbf{\Sigma}^{-1} (\mathbf{t} - \mathbf{y}_j). \quad (14)$$

The main conclusion is thus that vector quantization is simply EM estimation with MAP class assignments and, in practice, this means that the codebooks originated by VQ are a good approximation to the parameters of the mixture density that best describes the data. Or, if one designs the codebook with the EM algorithm and uses MAP decisions only after the after training is completed, the codebook will also provide an optimal estimate of the mixture parameters.

3 The library-based coder

The library-based coder builds on the similarities between EM and VQ design to obtain a representation that can be used to jointly address the issues of compression

[1] Notice that the denominator of equation 10 is simply a normalizing constant, equal for all h_i's.

and retrieval. For each frame, a codebook is designed and transmitted to the receiver. The frame is then encoded using VQ, and the quantization indexes transmitted as well. Because the codebook provides a probabilistic description of the source, it is all that needs to be decoded for the purposes of retrieval - the bulk of the data being decoded only when the frame is to be reconstructed. From the compression point of view, the scheme can be seen as a universal encoder, continuously adapted to the source probabilities.

While, in theory, VQ has long been known to be the optimal compression scheme; in practice, because optimality is only achieved with large vector sizes and encoding complexity grows exponentially with vector size, vector quantizers have fallen short of providing the theoretically attainable performance. In fact, if block sizes and encoding complexity are to remain compatible with those of the current standards, codebooks will be limited to relatively small sizes, leading to reduced rates and poor image quality.

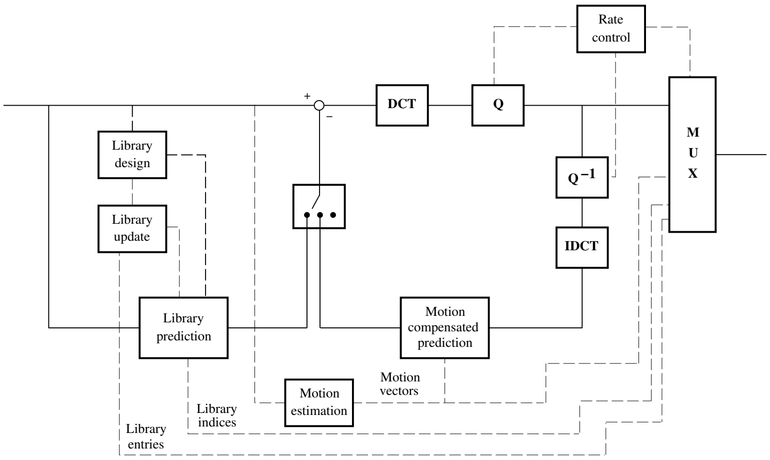

Figure 1: Block diagram of the library-based coder.

Due to this limitation, and the desire to keep the coding model as close to that of the current standards as possible, our implementation of the library-based coder is basically an extended version of MPEG, where the library is used as an additional predictor. In this setting, the library-based coder can be seen as *retrieval-enabled* MPEG, with the additional benefit of better prediction (through the library) during common events where block-matching fails.

A block diagram of the complete coder is presented in figure 1. Each input frame is segmented into square blocks, which are then processed to minimize both temporal and spatial correlation. Two different prediction structures are used for temporal processing: the library-based predictor discussed above, and a conventional motion-compensated predictor. By using the two prediction modes, it is possible to combine the higher efficiency of motion-compensated prediction in areas of translational or reduced amplitude motion, with the increased performance of library prediction in areas of non-translational motion, object occlusion, or where new objects are revealed. The encoding of the prediction error signal is similar to that used by MPEG-2 [6].

4 Content-based queries

Consider the task of finding the closest match in a database to a given a query image. This task can be solved in several ways if the library coded representation is used. The simplest of the solutions is probably to follow the route outlined in section 2.

Assuming that the blocks of the query image $\mathbf{x}_i, i = 1 \ldots N$, are independent samples of the same stochastic process, the images in the database are samples from M image sources $\mathcal{S}_i$, and using equation 1, the source with the highest probability of having generated the query image is

$$s = \arg \max_{k \in \{1,\ldots,M\}} \prod_{i=1}^{N} P(\mathbf{x}_i | \mathcal{S}_k) P(\mathcal{S}_k). \tag{15}$$

Given the library entries $\mathcal{L}^{(k)} = \{\mu_i^{(k)}, \mathbf{\Sigma}_i^{(k)}, p_i^{(k)}, i = 1...C\}$, the conditional probabilities of equation 15 are computed through equation 3. In the absence of any prior knowledge about the relative source likelihoods, the term $P(\mathcal{S}_k)$ can be disregarded. In a more complete setting, prior probabilities can, however, be used to constrain the search. If, for example, the images in the database are annotated with text and the user specifies a preference for pictures containing "people", the retrieval engine can assign a high prior to all the images annotated with the "people" keyword, and a low prior to the remaining images, increasing the posterior likelihood of the images in the desired category. The point is that, unlike other types of features, because libraries are probabilistic descriptions of the source, they allow statistical reasoning and the construction of powerful search paradigms in the compressed domain.

A practical limitation of a solution based on equation 15 is that computing all the N conditional probabilities can still be an expensive task, which grows proportionally to the image size and implies decoding the query image if this is also originally compressed. An alternative, less expensive, solution consists in substituting the product of conditional probabilities by a function which measures the similarity between the probability densities associated with the query and the database images

$$s = \arg \max_{k \in \{1,\ldots,M\}} \mathcal{D}[P(\mathbf{x}|\mathcal{S}_q),\ P(\mathbf{x}|\mathcal{S}_k)], \tag{16}$$

where $\mathcal{S}_q$ is the source of the query. Event though any of the traditional similarity metrics, such as the Kulback-Leibler distance [2], could theoretically be used in equation 16, these metrics typically do not lead to simple closed-form expressions for Gaussian mixtures. We have, therefore, considered the following simpler metric inspired by equation 15

$$s = \arg \max_{k \in \{1,\ldots,M\}} \prod_{i=1}^{C} P(\mu_i^{(q)} | \mathcal{S}_k), \tag{17}$$

where the image blocks $\mathbf{x}_i$ are replaced by the means of the Gaussians $\mu_i^{(q)}$ in the mixture associated with the query image. This reduces the number of conditional probabilities to be evaluated by approximately an order of magnitude, and allows searches that not even require full decoding of the query image.

The metric above can be further simplified by noticing that, for a given $\mathbf{x}$, the contribution of most of the Gaussians in equation 3 to $P(\mathbf{x})$ will be negligible. In the extreme case of negligible overlap between the Gaussians in the mixture, at most one of these Gaussians will be responsible for the bulk of the probability. One can, therefore, use the approximation

$$-\log P(\mu_i^{(q)}|\mathcal{S}_k) \approx \min_{*\in 1\ldots C}\{\frac{1}{2}||\mu_i^{(q)} - \mu_*^{(k)}||^2_{\mathbf{\Sigma}_*^{(k)}} - \log p_*^{(k)}\}, \tag{18}$$

obtaining

$$s = \arg \min_{k\in\{1,\ldots,M\}} \sum_{i=1}^{C} \min_{*\in 1\ldots C}\{\frac{1}{2}||\mu_i^{(q)} - \mu_*^{(k)}||^2_{\mathbf{\Sigma}_*^{(k)}} - \log p_*^{(k)}\}. \tag{19}$$

Comparing equation 18 with the vector quantization expression of 13 once again makes explicit the connection between VQ and MAP probability estimation using mixture densities. Thus, under the assumption of separated Gaussians and the approximation of equation 17, the closest image to that used as a query is the one whose library is closest to the library associated with the query image in the traditional VQ sense.

5 Simulation Results

In this section we analyze the performance of the library-based coder. Because the compression efficiency of the coder was already studied in [10, 11], we now concentrate on its retrieval capabilities. Figure 2 depicts the distances between a query image and the images in a database of 700 frames containing various scenes taken from trailers of the movies "Terminal velocity" and "East of Eden". Image 200 was used as a query example, and is presented inside a white frame.

Notice that the measured distances agree with what should be expected from the retrieval system. The query image obviously has a null distance to itself. Next, the closest images are those in the same video shot and, for these images, the query distance increases gradually with the temporal distance to the query image. The following closest images are those in scenes with content similar to that of the query image, namely people floating in mid air, and large areas of sky. Next come scenes which contain some degree of sky or water and finally, at a significant distance, are the artificially generated graphics. This indicates that, as a metric, the inter-library distance is capable of fine discrimination between different types of content.

Figure 3 shows examples of content-based retrieval on the same image database. It contains four images, each displaying the results of five queries. In all cases, each row corresponds to a query, with the query image shown on the right, followed by the four best matches (excluding itself) in the database which are presented from left to right according to their similarity rank - most similar on the left, less similar on the right. The query images were selected randomly from the 700 frames in the sequence. A total of 130 queries were performed.

The retrieval results were manually classified as good or bad matches. This is a relatively easy task in this setting, because it is easy to determine if the retrieved

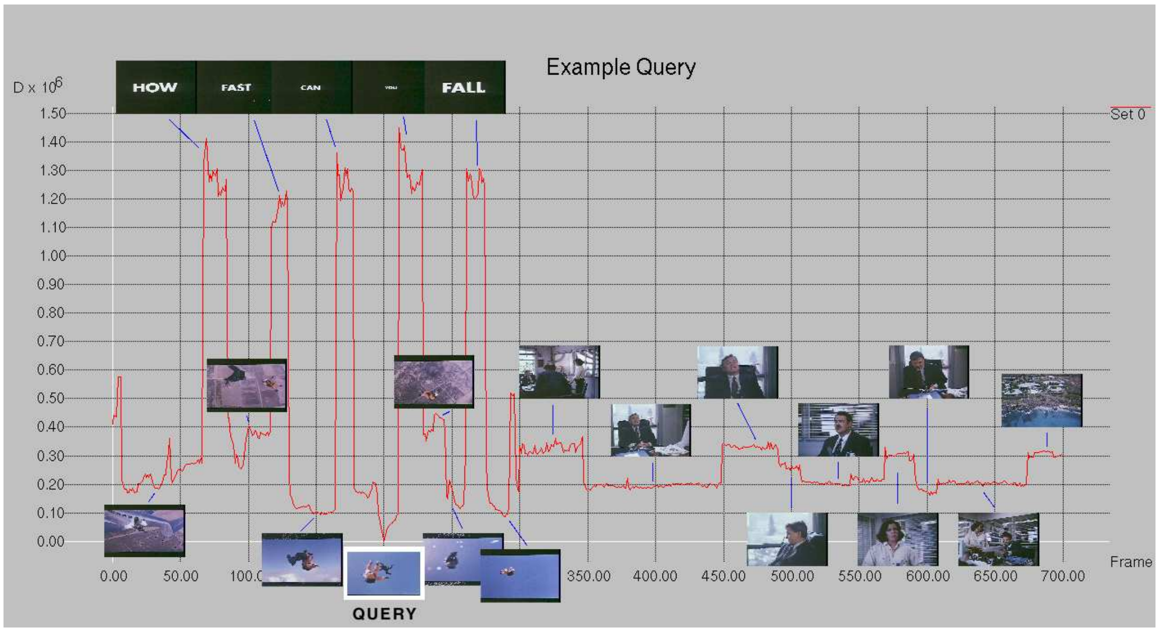

Figure 2: Example of the retrieval results achieved with the library-based coder. Graph shows the distance to the query frame (frame 200) versus frame number for the 700 images in the sequence. Key-frames are shown for most of the video shots. Query frame is presented with a white border.

image belongs to the same video shot as the query image. Table 1 presents the percentage of good matches, as a function of the rank of the retrieved image.

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image Coding Using Wavelet Transform. *IEEE Trans. on Image Processing*, Vol. 1, April 1992.

[2] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[3] A. Dempster, N. Laird, and D. Rubin. Maximum-likelihood from Incomplete Data via the EM Algorithm. *J. of the Royal Statistical Society*, B-39, 1977.

[4] A. Gersho and R. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press, 1992.

[5] Y. Gong, H. Zhang, H. Chuan, and M. Sakauchi. An Image Database System with Content Capturing and Fast Image Indexing Abilities. In *Proc. Int. Conf. on Multimedia Computing and Systems*, May 1994, Boston, USA.

[6] ISO-IEC/JTC1/SC29/WG11. *MPEG Test Model*, MPEG93/457.

[7] F. Liu and R. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. Technical Report 320, MIT Media Laboratory Perceptual Computing Section, 1995.

[8] N. Negroponte. *Being Digital.* Alfred A. Knopf, Inc, 1995.

[9] D. Titterington, A. Smith, and U. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley, 1985.

[10] N. Vasconcelos. Library-based Image Coding using Vector Quantization of the Prediction Space. Master's thesis, Massachusetts Institute of Technology, 1993.

[11] N. Vasconcelos and A. Lippman. Libray-based Image Coding. In *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Adelaide, Australia, 1994.

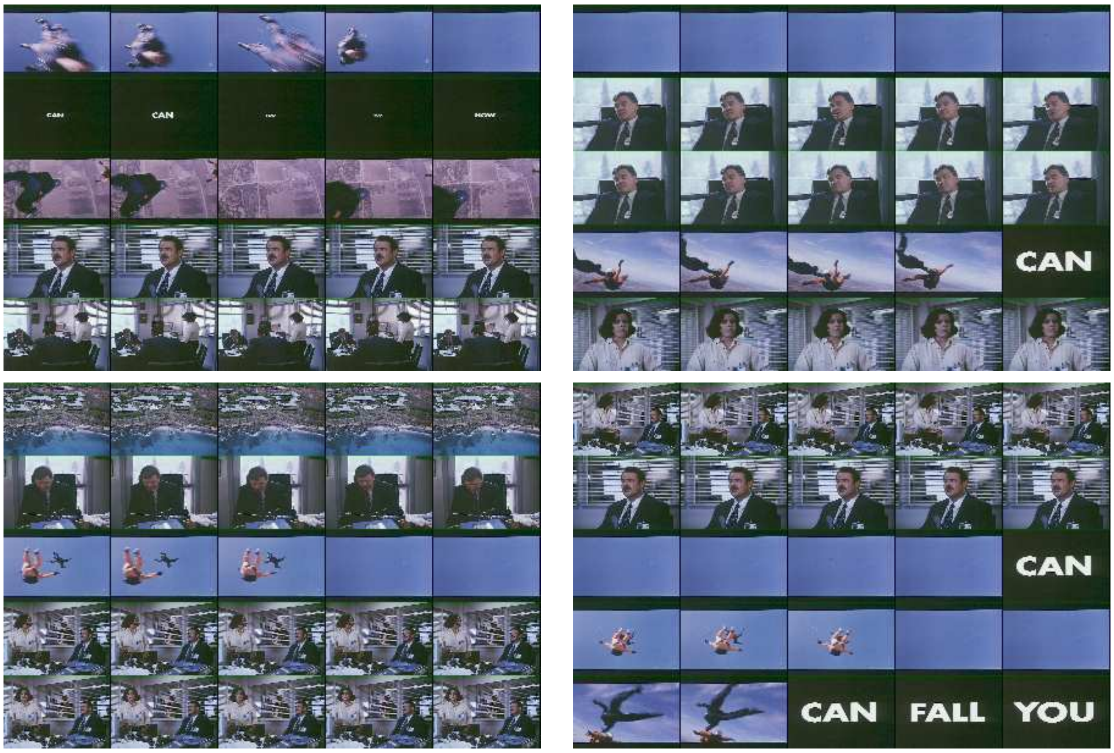

Figure 3: 20 examples of retrieval from a database of 700 images. Each row contains the results of a retrieval, query image shown on the left followed by the closest match, second best match, etc.

Rank	1	2	3	4
Good matches (%)	99.2	93.1	90.7	87.7

Table 1: Percentage of good matches versus similarity rank for 130 random retrievals from a sequence of 700 frames.