VIDEO SEGMENTATION AND COMPRESSION USING HIERARCHIES OF GAUSSIAN MIXTURE MODELS

George Yazbek[1], *Chafic Mokbel*[2], *Gérard Chollet*[1]

[1]Ecole Nationale Supérieure des Télécommunications, Paris, France, [2]University of Balamand, Lebanon

ABSTRACT

We present a new method that permits arbitrary region description in video sequence using a reduced number of bits allowing promising results in video compression. A robust unsupervised 3D (2D + t) segmentation is used to detect arbitrary regions for motion compensated video coding. The video sequence is first modeled using a Gaussian mixture model where each pixel, defined by its spatiotemporal position and its color vector is supposed to be generated by one of the mixture components. This permits to segment the video sequence into objects each one modeled by one Gaussian distribution. Grouping the mixtures in a binary tree defines a hierarchical representation of the video objects and a gradual segmentation. This segmentation is then used for region description in a motion compensated video coder. This provides a large improvement in motion bits budget. When compared to a H.264 video coder, promising results were obtained.

Index Terms— video coding, video segmentation, stochastic models, Gaussian Mixture Models.

1. INTRODUCTION

The purpose of video segmentation is the extraction of static video background and independent objects from a video scene. It is considered to be a mid-level scene understanding in computer vision. Video segmentation has important applications in video indexing, searching, coding and motion estimation.

Megret and DeMenthon classify video segmentation techniques in three categories [1]: (1) segmentation with spatial priority where the pixels of each frame are grouped and then a trajectory is detected; (2) segmentation by trajectory grouping where the trajectory of each pixel across frames is detected and then the pixels are grouped; (3) spatiotemporal segmentation where the pixels from all the frames are grouped in space and time simultaneously. This last category is the most powerful as it does not give any preference to a given dimension. Moreover, it is known that human vision recognizes important structures simultaneously in space and time [2].

Salembier et al. use connected operators to segment images and video sequences and to build a tree representation of features. Events are then extracted from the compact binary tree model [3][4].

Greenspan et al. propose a probabilistic framework based on Gaussian Mixture Models (GMM) to describe video regions [5]. The GMM is used to model the pixels of a video scene where every pixel is represented by a vector including its spatiotemporal position and its color. Once the GMM model parameters determined, each pixel in the scene is associated with the most likely component of the GMM. Adjacent pixels associated with the same component define a spatio-temporal (or a 3-D) object of the scene.

In this paper, we propose to extend the use of GMM model by building a hierarchy of video regions based on an initial GMM. This allows the determination of the optimal number of regions. Actually, starting from an initially estimated model a hierarchy is built using simple model's components merging. This leads to a hierarchy of 3-D objects for video understanding and coding. This hierarchy of video objects allows a gradual segmentation of the video sequence into objects with variable level of precision.

The hierarchical video segmentation algorithm is applied in a video coding scheme. Hybrid video coders divide a frame into small rectangular blocks and encode the motion of each block. The residual error is then transformed, quantized and entropy coded. Major compression gains can be achieved by encoding larger arbitrary regions. The problem of this approach is the region description itself that would require a large number of bits. Therefore, a tradeoff must be found between the precision of the video segmentation and the complexity of the objects definition. The proposed GMM based segmentation allows arbitrary region description with a reasonable number of bits due to the parametric form of a GMM. This allows us to potentially achieve significant compression gains.

2. GAUSSIAN MIXTURE MODELS FOR VIDEO REPRESENTATION

Gaussian mixture models (GMM) are stochastic models defined as a mixture of components each of them being a Gaussian probability distribution function. A stochastic process following such models generates vectors in an observation space. GMMs have been used widely in different domains. In the following, the observation space on which GMM are applied, their training, and their use in video segmentation are presented.

2.1. Feature vector extraction

Each pixel is considered to be an element of a video region in both space and time. Its coordinates are its position in the video plane, its color and the frame time. The result is a 6 dimensional feature vector.

The color coordinates should be taken in a perceptually uniform color space, providing perceptual meaning to Euclidian distances. The possibilities are CIE L*a*b*, CIE L*u*v*, nonlinear RGB and Y'CbCr. We have tested CIE L*a*b* and Y'CbCr and both produced similar results.

Digital video uses Y'CbCr natively. Therefore this space is chosen to represent the color vector to avoid extra transformations. Sub-sampled chroma values (in the standard 4:2:0 files) are oversampled before adding to the feature vector. Selecting the best color space to use depends on the final application. Finally, all the feature vector components are normalized to [0; 1] before further processing.

2.2. Model training

Given a feature vector x, the GMM defines its probability distribution function as follows:

$$\sum_{i=1}^{N} w_i \frac{1}{\sqrt{(2\pi)^d \left\|\underline{\underline{\Gamma}}_i\right\|}} \exp\left(-\frac{1}{2}\left(\underline{x}-\underline{\mu}_i\right)^T \underline{\underline{\Gamma}}_i^{-1}\left(\underline{x}-\underline{\mu}_i\right)\right) \qquad \textbf{(1)}$$

where N is the number of components, d is the dimension of the observation space, i.e. 6, and $\underline{\mu}$i and $\underline{\underline{\Gamma}}_i$ are the mean vector and covariance matrix of the ith component.

The realization of this distribution can be seen as the realizations of two successive processes; the mixture component selection and, the feature vector emission by the selected component. The unique observation of the feature vector provides incomplete data insufficient to allow analytic estimation, following the maximum likelihood criterion, of the model parameters, i.e. the Gaussian distributions weights, mean vectors and covariance matrices. The selected component is not observed. This is known as incomplete data estimation problem.

The Estimation Maximization (EM) algorithm offers a solution to the problem of incomplete data [6]. The EM algorithm is an iterative algorithm, an iteration being formed of two phases; the Estimation (E) phase and the Maximization (M) phase. The EM algorithm ensures a convergence towards a local optimum. This local optimum depends on the initial values given to the model parameters before starting the training. Thus, the initialization of the model parameters is a crucial step.

In this work the LBG algorithm is used for initialization [7]. In the video segmentation task, frames will be grouped in a sliding window, i.e. a GMM is trained for each window of frames. The model parameters for a group or window may be initialized as the parameters estimated on the previous group. This avoids the application of the LBG algorithm.

2.3. Number of components in the model

Selecting the optimal number of components in the model is a tradeoff between processing time, descriptive power and the quantity of training data. Storage of the GMM parameters is also important in video coding applications.

The approach used in [5] is to model a certain number of mixtures each with a different number of components k. The model that satisfies the Minimum Description Length (MDL) criterion is selected. If two models satisfy the criterion, the simpler one (the one with the least k) is selected. This approach is computationally intensive.

A more economical approach in terms of processing power is described in [8]. In this approach, starting from an initial quite developed model a tree is constructed and then pruned according to some criterion. The new leaves represent the optimal number of Gaussians. Another advantage of this approach is that it provides a hierarchical or multi-resolution representation of the video sequence. This is detailed in the section 3.

2.4. Segmentation

Sequence segmentation is done by hard decision. Each pixel is labeled with the Gaussian that provides the maximum likelihood.

$$s(\underline{x}) = \arg\max\left(w_i \frac{1}{\sqrt{(2\pi)^d \left\|\underline{\underline{\Gamma}}_i\right\|}} \exp\left(-\frac{1}{2}\left(\underline{x}-\underline{\mu}_i\right)^T \underline{\underline{\Gamma}}_i^{-1}\left(\underline{x}-\underline{\mu}_i\right)\right)\right) \qquad \textbf{(2)}$$

The result of the segmentation phase is a set of k video regions where k is the number of components in the GMM.

3. CONSTRUCTING A HIERARCHY OF GAUSSIAN COMPONENTS

Once the model is trained, the Gaussian distributions may be grouped in a binary tree. This tree is build iteratively by merging the most similar Gaussians, i.e. the closest in a distance measure sense. The tree is used to "guess" the optimal number of Gaussians by pruning the tree according to some criterion. By varying the criterion parameters, a multi-resolution representation is generated. This multi-resolution offers segmentation with variable degree of precision. It is straightforward that this can be further developed to have different levels of precision in different regions of the video sequence.

Distance calculation and Gaussian merging is done according to [8].

3.1. Tree construction

An unbalanced binary tree is built by successive merging of the closest two nodes following the distance. The merged node replaces the two original nodes and the process is restarted until we reach the root of the tree. The following algorithm is used:

- Calculate the distances between all Gaussian pairs.
- Find the minimum distance.
- Merge the two closest Gaussians and replace them with the resulting node.
- Repeat until there is one Gaussian left, i.e. the root.

3.2. Tree pruning

The constructed tree can be pruned using multiple criteria. The tree is traversed from its root to its leaves. Depending on a criterion we select to stop the traversal and select the node as the leaf node or to continue. Two criteria have been tested: the first one is the sum of the weights of the branches going out of a node. This tends to eliminate small objects. The second criterion is the distance between the branches. This groups similar regions and thus is better at describing the sequence.

4. VIDEO CODING USING GMM

Using GMM to segment video sequences can be interpreted as a way to encode arbitrary regions with a limited number of parameters to define those regions. A single GMM can be used to segment a number of frames into coherent regions. Motion estimation is then applied on these regions to extract motion vectors. The estimated frame is then error coded.

4.1. Encoding algorithm

A *k*-component GMM is trained for each group of *n* frames. This generates *k* segments for each frame that can be further decomposed into dense sub-segments.

The motion vector can be estimated directly from the GMM parameters. However, to get a more precise estimation, for each segment, an exhaustive search in all the possible (v_x, v_y) motion vectors in a (*-M..M*, *-M..M*) square is performed. The size of the search space is a compromise between calculation time and the maximum real motion vector: A larger *M* allows more important motion at the expense of a quadratic increase in the search space.

In order to select the optimal motion vector, each vector is assigned a weight $W = \frac{SSD}{N^2}$ that is calculated using the following algorithm.

```
    • For each frame $n$
    • For each segment $k$
    • For each motion vector $(v_x, v_y)$
    • Loop through all pixels $P(x, y)$ of frame $n$.
    • If $P$ belongs to segment $k$
        o If $P'(x + v_x, y + v_y)$ belongs to the same segment $k$ then
            ▪ $N = N + 1$
            ▪ $SSD = SSD + (Y(P') - Y(P))^2$
```

N is the number of matching bits and *SSD* is the sum of the square of the differences in luminance between the two frames.

Finally, the difference between the predicted frame and frame *n*+1 is then encoded using DCT and entropy coding.

4.2. Decoding algorithm

In order to generate frame *n*+1 from frame *n*, having the GMM parameters, the motion vectors and the residual error, it is enough to:

- Segment frame *n* using the GMM
- Displace each segment using the stored motion vector
- Apply the residual error correction

5. EXPERIMENTAL RESULTS

5.1. Segmentation experiments

Experiments have been conducted on the Akiyo sequence using the BECARS toolkit [9]. The two features L*a*b* and Y'CbCr have been studied. The latter being the native color space of the sequence. A mixture of 16 Gaussians is used to model the video sequence of 5 consecutive frames. This mixture forms the starting point of our modeling algorithm. The mixture was first trained and the corresponding tree was then generated. Pruning the tree has led to 5 segments.

Regarding the two methods of pruning, they have been applied with L*a*b* features. The results show a clear advantage of using distance-based threshold as expected.

The feature extraction possibilities are studied. The obtained results are similar. Therefore, the Y'CbCr is more appropriate since it requires less computational effort.

5.2. Compression experiments

We have worked on the Foreman sequence in QCIF format (176x144). Each mixture consists of *k*=16 components. It is trained on a sequence of 5 frames. There is no overlap in the frames. We have executed the reconstruction algorithm and then calculated the peak signal to noise ratio (PSNR). Classical coders estimate the motion of each block in frame *n*+1 from frame *n*. Here we start with a block from frame *n* and estimate its motion. Some pixels in frame *n*+1 may not be targeted by any of the *k* Gaussians after motion compensation. These are not taken into account in the calculation of PSNR.

The bits needed to encode a frame correspond to the Gaussians and the motion vectors. Without taking error encoding into account, we need: $2k \cdot n_d + f\left(k + kd + kd\frac{d+1}{2}\right)$ bits, with:

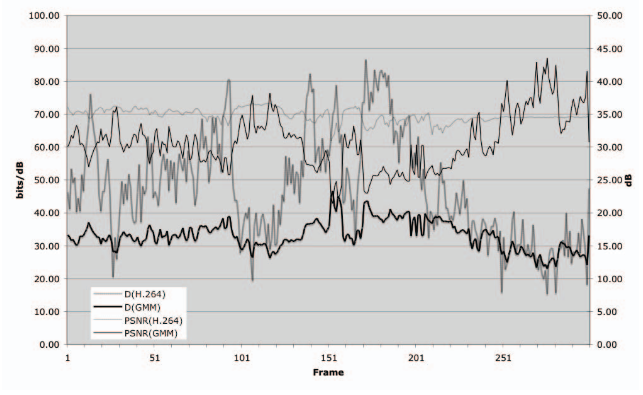
n_d = number of bits needed to encode motion (4 bits)

f = number of bits needed to encode a floating point number (32 bits)

d = dimension of the covariance matrix (6)

In our experiment, we need 2995 bits/frame. This information can be compressed using entropy coding. We have assumed that a factor of 3 can be easily achieved.

In order to compare our results to the H.264 reference encoder, we have modified it to extract motion compensated frames before residual error coding. We then calculated the resulting PSNR. We have also extracted the number of bits necessary for motion coding. This has allowed us to calculate $D = \frac{bits}{PSNR}$. This quantity is the number of bits necessary to gain 1 dB of PSNR. This measurement is useful to compare the performance of a GMM based coder to H.264. The D as well as the PSNR are measured on the Foreman sequence for both the H.264 and the GMM based coder. The results are plotted in the following figure. Looking to this figure, the GMM based coder seems to outperform the H.264 in the number of bits necessary to gain 1dB in PSNR.



6. CONCLUSION AND FUTURE WORK

In this work, we use a GMM to describe a video sequence. This model is then used to segment the sequence into 3D (2D + t) regions. A novel approach is proposed to determine the optimal number of Gaussian components. This number of Gaussian components corresponds to the number of independent objects in the video sequence. We have found that using a distance criterion, the number of components can be correctly identified.

Experiments we have conducted validate the idea that arbitrary region coding using GMMs is very efficient in terms of bits needed to encode a frame.

The achieved PSNR is far from that achieved by standard hybrid coders but the number of bits necessary to gain 1dB of PSNR is very promising. The number of Gaussians and thus the number of bits is constant. This means that our approach does not take into account the variations in the video sequence. To improve this, we need to select the number of Gaussians based on the PSNR.

Pixels that are not estimated in the target frame are another problem that must be dealt with. Bi-predictive frames could be the answer to this problem. Another improvement would be sub-pixel motion prediction.

In conclusion, we can say that automatic segmentation could enhance the efficiency of video coding. Work must be done to produce a full codec that improves on the state of the art.

REFERENCES

[1] R. Megret and D. DeMenthon, "A Survey of Spatio-Temporal Grouping Techniques", *Research Report CS-TR-4403*, LAMP, Univ. of Maryland, 2002.

[2] S. Gepshtein and M. Kubovy, "The emergence of visual objects in space-time," in *Proc. of the National Academy of Sciences*, USA, vol. 97, no. 14, pp. 8186-8191, 2000.

[3] P. Salembier and L. Garrido, Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval, IEEE Transactions on Image processing, vol. 9, no. 4, pp 561-576, 2000

[4] P. Salembier and F. Marqués, Region-based representations of image and video: Segmentation tools for multimedia services, IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 8, pp 1147-1167, 1999

[5] H. Greenspan, J. Goldberger, and A. Mayer, "Probabilistic Space-Time Video Modeling via Piecewise GMM", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 384-396, 2004

[6] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. of Royal Statistical Society*, Vol. 39, n° 1, pp. 1-22, 1977

[7] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, pp. 702-710, January 1980.

[8] C. Mokbel, "Online Adaptation of HMMs to Real-Life Conditions; A Unified Framework," *IEEE Transactions on Speech and Audio Processing*, vol. 9, pp 342-357, 2001

[9] R. Blouet, C. Mokbel, H. Mokbel, Sanchez Soto E., G. Chollet and H. Greige, "BECARS: a free software for speaker verification," ODYSSEY-2004, pp. 145-148, 2004