Video Shot Characterization*

Mihai Osian[1], Luc Van Gool[2]

[1] Katholieke Universiteit Leuven, ESAT/PSI
Kasteelpark Arenberg 10, 3000 Leuven, Belgium

[2] Computer Vision Laboratory, ETH,
Gloriastrasse 35, CH-8092 Zurich, Switzerland

{mosian, vangool}@esat.kuleuven.ac.be

Abstract

This paper presents a practical approach to detecting shot cuts and extracting keyframes from video sequences. Shot cut detection has two stages - global motion compensation, followed by an adaptive thresholding algorithm. The motion information is further utilized to extract representative keyframes. Special consideration has been given to achieving real-time performance on a regular PC, which led to a motion estimation algorithm of linear complexity.

1 Introduction

Video sequences are one of the most difficult data types to handle and organize, due to the quantity and complexity of the information they contain. Manual annotation provides the best quality but is also a long and tedious process. The first step that has to be achieved by any automated indexing system is segmenting the video sequences into a collection of **video shots**. A video shot is defined as an uninterrupted sequence of frames, corresponding to the same scene or event. For each shot we can extract one or more **keyframes** representative for the entire shot.

The change of scene which appears at the boundary between two shots is called a **shot cut**. A shot cut can be **abrupt** - when it takes place between two consecutive frames, or **smooth** - when it takes place over a short period of time (special video effects like dissolves, fades, wipes or page turnings). Abrupt shot cuts are the easiest to detect. Smooth cuts require extensive analysis of several consecutive frames.

We will present in this paper an algorithm for detecting shot cuts, with emphasis on abrupt cuts. It includes a fast global motion compensation and

*Research supported by the EC project VIBES

an adaptive threshold module. The algorithm was intended to run close to real-time on a regular PC. We achieved linear complexity with less than 12 operations required per pixel, including camera motion compensation.

The paper is structured as follows: Section 2 presents the problems encountered in shot cut detection and how some of the existing techniques handle them. Section 3 describes the motion compensation module. Section 4 presents the adaptive thresholding mechanism. Section 5 introduces our approach to smooth cuts detection and Section 6 presents the keyframe extraction algorithm. The conclusions are drawn in Section 7.

2 Previous work

Most existing algorithms define a dissimilarity metric for comparing consecutive frames. The metric must be insensitive to motion, while responding well to scene cuts. Usual metrics include comparing histograms, blocks (cross-correlation), image features or pixel intensities. For MPEG video some authors use analysis in the compressed domain [10].

We start with a short review of these methods:

- Color histograms - are rather insensitive to motion, but they can miss shot cuts between frames with similar color distribution. To include some spatial information most authors prefer to split the image into blocks and compute local histograms, e.g.[3, 8].
- Joint Histograms [9] - try to improve the performance of regular color histograms by adding other image features like edge density, texturedness, gradient magnitude, etc.
- Feature based detection. Zabih [14] presents an algorithm based on analyzing entering/exiting edges between consecutive frames. During a cut many edges appear far from the location of old edges, while old edges disappear. His method also includes a global motion compensation.
- Average pixel difference [15]. Simply subtracting the pixel intensities in two consecutive frames is a useful measure. However, the generated difference depends very much on the motion magnitude.
- Block-based motion estimation. A more complex technique is used by Porter [11]. In her approach, rectangular blocks are matched in order to compensate for motion. Matching is done in the frequency domain. Blocks having only low-frequency components are not used.
- Block Likelihood Ratio [2]. Each frame is split into an array of 8x8 blocks of the same size. A two blocks thick border is left out in a frame and the 16 middle blocks are considered one by one. Each of the middle blocks in the current frame is compared to its 9 neighbors in the previous frame. The neighbor with the minimum distance will give the final value. If the average distance for the middle blocks is greater than a certain threshold, a shot change is declared.

Figure 1: *Two consecutive frames for which L takes on a small value. They are not detected as a shot cut by the Block Likelihood Ratio method.*

The likelihood ratio used is:

$$L = \frac{\frac{\sigma_k+\sigma_{k-1}}{2} + \left(\frac{m_k-m_{k-1}}{2}\right)^2}{\sigma_k \times \sigma_{k-1}}$$

where m_{k} and σ_k denote the mean and variance of intensity of a given region in frame k. The likelihood ratio is computed for the 3 color components (R,G,B) and the partial results are combined.

We implemented the block likelihood ratio method for comparison. We used about 15 minutes of various video sequences, including documentaries, sport sequences, news broadcasts and a movie fragment. Out of 176 cuts, 146 have been correctly detected, and there were 37 false alarms. This corresponds to recall/precision rates of $\frac{146}{176} = 82\%$ and $\frac{146}{146+37} = 79\%$, respectively. The recall measure indicates how many of the real cuts were detected. Precision says how many of the cuts found by the algorithm were valid.

During experiments, the value of L was of the order of hundreds or even thousands for shot cuts, and less than 10 in the other cases. Despite of this excellent contrast, there are many simple cases in which the algorithm fails to detect cuts (Fig 1). On the other hand, almost any large motion caused series of false detections.

3 Global motion compensation

Similar to the approach of Bouthemy (see Refs. [1]), we compute for each pair of frames an affine transformation which warps the first frame into the second. The two frames are compared only after they are aligned. The resulting difference is used as input to the adaptive thresholding algorithm.

Computation of the transformation is a two-step process. In the first stage, we split the image into 20×15 blocks. For each block we compute the best motion vector. In the second stage, we will try to find a transformation which approximates the entire set of motion vectors. The most complex part is computing the motion vectors. Basically, we want to find for each rectangular block

the position in the second image where the cross-correlation is maximum. To reduce the number of operations, a hierarchical search is utilized. We start from a very small scale version of the image, where each of the initial blocks becomes a single pixel. At each scale, we shift a block with one pixel in each direction. The best of the 9 displacements is chosen as starting position for the next scale. A displacement of one pixel at a certain scale corresponds to a displacement of 2^s pixels in the original image, where s is the current scale . The size of the blocks is doubled at each step, but the search space is always limited to one pixel away from the start positions. The total number of operations is less than 12N[1], where N is the number of pixels.

The horizontal and vertical components of the motion vectors are weighted separately. The weight along a direction is computed as $w = \frac{b-a}{b+1}$, where a is the score of the best matching position and b is the score of the second best position. Matching scores take values between 0 and 255. The weight is trying to model both the quality of the matching and the confidence level. A small a means that the match is good. If the second best match is good too, the block has a flat profile along the considered direction and the confidence of the motion vector drops. Compared to the simple difference $b-a$, the above formula increases a little the weight of a good match having a low confidence. This gave better results in our experiments.

To find an affine transformation which approximates all the motion vectors as good as possible, we start by writing a set of equations of the form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Each motion vector is defined by its start and end points - (p_i,p'_i), where p_i=(x_i,y_i,1) and p'_i=(x'_i,y'_i,1). The affine transformation is given by *(a,b,c,d,e,f)*. For the entire set of motion vectors, we have two matrix equations:

$$\begin{bmatrix} p_1 \\ p_2 \\ \dots \\ p_n \end{bmatrix} . \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x'_1 \\ x'_2 \\ \dots \\ x'_n \end{bmatrix} \Leftrightarrow \text{P.A=X', and}$$

$$\begin{bmatrix} p_1 \\ p_2 \\ \dots \\ p_n \end{bmatrix} . \begin{bmatrix} d \\ e \\ f \end{bmatrix} = \begin{bmatrix} y'_1 \\ y'_2 \\ \dots \\ y'_n \end{bmatrix} \Leftrightarrow \text{P.B=Y'}$$

with A and B as unknowns. Each line of the two matrix equations has an associated weight w_i - the confidence level for the corresponding motion vector. To find the best A=(a,b,c) coefficients, we have to minimize

$L = [W(PA - X')]^t [W(PA - X')]$

where W is a $N \times N$ matrix containing weights:

$$W_{i,j} = \begin{cases} w_i, i = j \\ 0, i \neq j \end{cases}$$

By differentiating with respect to A and taking into account that W is diagonal,

[1] $9N(1 + \frac{1}{4} + \frac{1}{16} + \ldots + \frac{1}{4^n}) < 9N\frac{4}{3}$

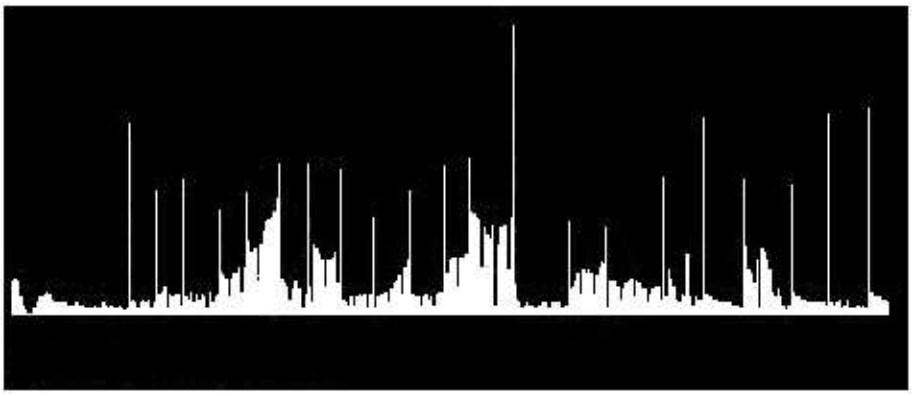

Figure 2: *Plot of the average pixel difference between consecutive frames. Spikes correspond to shot cuts.*

we get:

$(WP)^t[W(PA - X')] = P^tW^2PA - P^tW^2X' = 0 \Leftrightarrow$

$A = (P^tW^2P)^{-1}.(P^tW^2X')$, and similarly

$B = (P^tW^2P)^{-1}.(P^tW^2Y')$

The maximum motion the algorithm can handle is equal to twice the size of a block, minus 1 pixel. At the smallest scale each block is represented by a single pixel. Moving it one position in a direction corresponds to l pixels in the original image, where l is the original block size. In the next steps the maximum displacement will increase up to 2*l-1 pixels. A 20x15 blocks grid allows a maximum horizontal displacement equal to one tenth of the image length.

For color images having 352×240 pixels the processing speed is about 11 frames/second on a Pentium III at 700 MHz. In the absence of object motion the quality of the estimation is quite good - the accumulated error after 450 frames is still acceptable (Fig 7, to be discussed later, illustrates this).

4 Adaptive thresholding

We chose as similarity metric the average pixel difference between the motion compensated (i.e. affinely matched) images. Following the approach described by Yusoff [13], we tried to handle motion problems by using an adaptive threshold. In most cases the difference caused by motion changes little from one frame to the next. If we plot the difference between consecutive frames, shot cuts appear as sudden increases of the difference (Fig 2).

We analyze the value of the differences in a sliding window of 15-20 frames and compute several statistical parameters. This allows us to decide whether there is a scene cut between the two frames in the middle of the window. The algorithm has been refined through experiments and consists of checking the

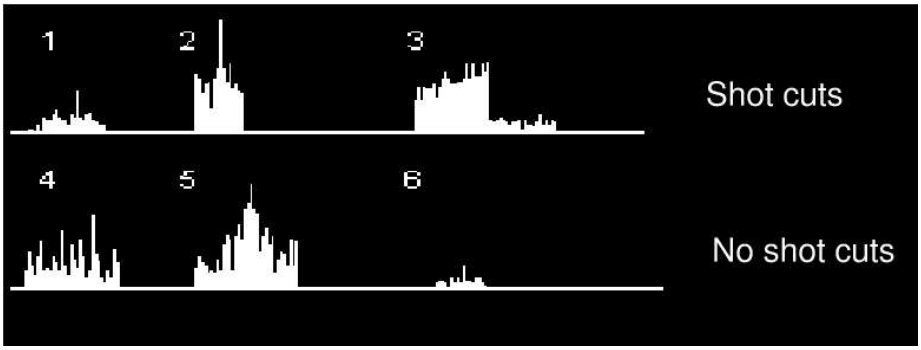

Figure 3: *On the upper row: cases in which a shot cut is found. Bottom row: no shot cut, because the variance is too high (4), the difference increases linearly (5) or the difference is smaller than the minimum threshold (6).*

following conditions:

- The difference corresponding to the middle frame pair is the largest within the considered window. This follows the assumption that the difference caused by the scene change is always bigger than the difference caused by motion. Two typical examples of shot cuts are cases 1 and 2 from Fig 3. Note that in the first case, the difference corresponding to the cut is smaller than almost any difference from case 2.

- The difference must be higher than a fixed minimum threshold.

- The current difference is larger than a multiple of the average difference (computed over the entire window). The multiplication coefficient is proportional to the variance within the window. This criterion is related to the observation that moving persons or objects close to the camera generate an irregular profile of the difference. An example of such a profile is illustrated in case 4 from Fig 3.

- An exception from the previous criterion is if the average difference of the previous frames is very high and the average difference of the next frames is very low or vice versa, because the current difference delimits a high activity shot from a low activity one - case 3 from Fig 3.

- If we reached this point, we look for a linear increasing tendency of the difference in the first half and a decreasing tendency in the second half (a triangular shape). We extrapolate the evolution of the difference from the beginning/end of the window towards the middle. If the maximum of the two predicted differences is comparable to the actual difference in the middle, then the cut is rejected - case 5 from Fig 3. This is a situation encountered when there is a fast object passing in front of the camera.

On the 15 minutes data set used for comparison, our algorithm detected 169 out of 176 cuts, with 7 false alarms (96% recall and 96% precision). These results compare favorably with those of the Block Likelihood Ratio method (see Section 2).

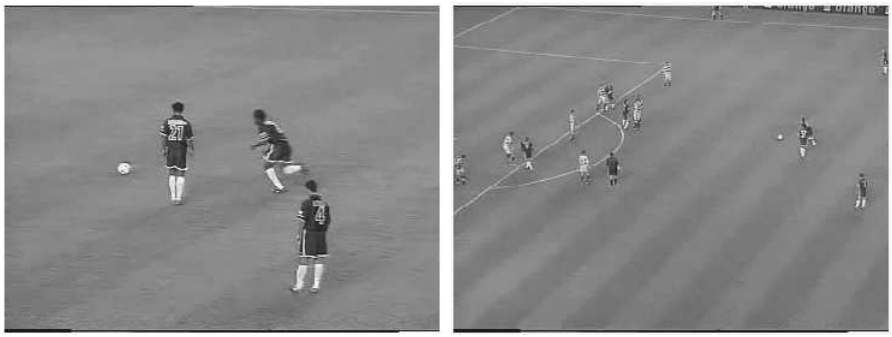

Figure 4: *Two consecutive frames which are difficult to find as a shot cut. Both the color components and the shot activity are very similar.*

A larger data set of 40 short video sequences, having about 35 minutes, produced the following results: 450 out of 464 cuts were detected, with 35 false alarms. This corresponds to 96% recall and 92% precision rate. The content of the second data set ranged from news broadcasts and commercial presentations to film fragments and sport sequences. After the tests we tried to classify the situations where the algorithm failed. Sudden object or camera motion triggered almost all the false alarms. Continuous motion had as effect a raise in the value of the threshold, which made the algorithm miss a few true shot cuts. Other missed cuts appeared when the two images had moderate motion but were very similar in content (Fig 4).

For a third data set, consisting of two TV news sequences (one hour in total), the detection rates were particularly good. Out of 601 cuts, 596 were correctly detected. There were 17 false alarms, 14 caused by camera flashes at press conferences. This corresponds to 99% recall and 97% precision.

5 Detecting smooth cuts

For video effects like dissolves or fades we can not confine the analysis to pairs of consecutive frames. For example, the frames in Fig 5 are taken during such a transition. The difference caused by the dissolve is negligible compared to the difference caused by motion.

A good smooth cut detector must be able to track features along 2-3 seconds. If too many of these features are lost during that period and new ones appear, we conclude that there has been a smooth cut. The easiest solution would be to compute the optical flow for several frames and check the evolution of each individual pixel. The drawback of this approach is the computational complexity - several seconds are needed for each pair of frames. This translates to minutes of computation for a single second of video data.

Some experiments were carried out using the global motion compensation described above. Starting from the current frame, we compose the transfor-

Figure 5: *A smooth cut. Upper row: starting and ending frames. Lower row: two consecutive frames in the middle of the cut. Both scenes are highly textured (dense vegetation) and the camera is also moving (see Refs. [17])*

Figure 6: *Results of subtracting frame i from frame i+20, after motion compensation. In the first frames there is large object motion. The three "humps" correspond to smooth cuts*

mations between consecutive frames, so that we obtain a transformation which aligns frame 1 to frame N:

$T(1,N) = T(1,2) \circ T(2,3) \circ ... \circ T(N-1,N)$

Figure 6 shows the evolution of the difference, when aligning and subtracting images 20 frames apart. We distinguish two cases with large difference:

a) The difference is small, with sudden increases. These cases correspond to smooth cuts.

b) The difference is constantly big. This means that there is significant object motion, which can not be compensated. The algorithm will fail to detect smooth cuts, but the video fragment can be analyzed by a more complex algorithm, at the cost of more computations (see Refs. [7, 5]).

The accuracy of our algorithm depends drastically on the type of the analyzed sequence. On a documentary containing long shots of landscapes, with camera motion only (see Refs. [17]), out of 154 smooth cuts, 149 were detected and there were 10 false alarms. This corresponds to a recall rate of $149/154 = 96\%$ and a precision of $149/159 = 93\%$. On the other hand it performed poorly for a football match: the recall rate dropped to 30% and the precision to 60%.

6 Keyframe extraction

Key frames should be a very brief description of the video content, preserving at the same time the most interesting characteristics. It is rather difficult to define which frames are "interesting" and which are not. Existing approaches can be classified in:

- extracting the first, middle or last frame of every shot,
- motion analysis by using optical flow. The frames at the local minima of motion are chosen as keyframes.
- clustering frames based on their similarity. The similarity is computed by looking at the image histograms or other cues [16, 4].
- mosaic images (see [12])

A detailed review of these methods can be found in [6]. Our method uses informations about the shot boundaries and the motion within each shot. The

Figure 7: *Panoramic image obtained from 450 successive frames. The camera started in the upper-right part of the image, panned to the left and then zoomed out.*

first frame of each shot is chosen as keyframe. Depending on the magnitude of motion, we may need to extract additional keyframes. We chain the transformations between consecutive frames and we compute the overlapping surface of each frame, relative to the last keyframe. If the overlapping surface becomes too small, we pick a representative keyframe for this segment. The keyframe must be close to the current position (the last 2 seconds) and have minimum local motion. The keyframe must be located at the end of the current segment, otherwise it can be too similar to the previous keyframe. Small local motion gives a better chance of picking a relevant frame and ensures less motion blur.

A popular method of synthesizing the video content are mosaic images (e.g.[12]). They are suited when the video sequence presents landscapes or other mostly static scenes. By cutting stripes from each frame and pasting them together, we can obtain a single panoramic image. The technique is the same as for motion compensation: successive frames are aligned by a geometric transformation. At each step, the mosaic is extended with the parts of the new frame which fell outside the existing image. To illustrate the accuracy of the motion estimation, we present such a mosaic image in Fig 7. The keyframes of the corresponding video sequence are shown in Fig 8.

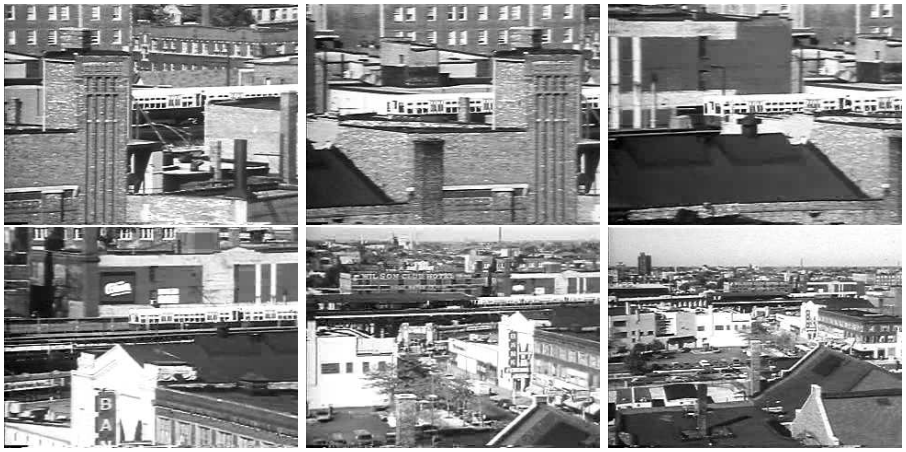

Figure 8: *Frames corresponding to the mosaic image from Fig7. The first frame corresponds to the upper-right corner of the mosaic image. The camera is panning to the left and then zooms out.*

7 Conclusions

We have presented a global motion compensation and adaptive thresholding algorithm for detecting shot cuts. The algorithm has linear complexity, which allows it to run in real-time. For abrupt cuts, it achieves consistently detection rates of more than 90% for most types of video data. Its most interesting feature is the motion compensation module, which is used both to improve detection of video shot cuts and as decision criterion for extracting keyframes. Due to the fact that it can not track individual objects, it has limited capabilities for detecting smooth cuts. It can, however, focus the attention of more complex algorithms to the segments where these cuts may appear.

References

[1] P. Bouthemy, Y. Dufournaud, R. Fablet, R. Mohr, S.Peleg, A. Zomet (1999) Video Hyper-links Creation for Content-based Browsing and Navigation. In: Proc. Workshop on Content-Based Multimedia Indexing, Toulouse, France

[2] Rakesh Dugad, K. Ratakonda, and N. Ahuja (1998) Robust Video Shot Change Detection. In: IEEE Workshop on Multimedia Signal Processing, Redondo Beach, California, pp. 376-381

[3] Yihong Gong, and Xin Liu (2000) Video Shot Segmentation and Classification. In: International Conference on Pattern Recognition (ICPR'00), Volume 1, pp 860-863, Barcelona, Spain

[4] R. Hammoud, R. Mohr (2000) A probabilistic Framework of Selecting Effective Key-Frames for Video Browsing and Indexing. In: International workshop on Real-Time Image Sequence Analysis, Oulu, Finland

[5] A. Hampapur, R. C. Jain, T. Weymouth (1995) Production Model Based Digital Video Segmentation. In: Journal of Multimedia Tools and Applications, Vol. 1, No. 1, pp. 9-46

[6] Y. Li, T. Zhang and D. Tretter (2001) An overview of video abstraction techniques. In: HP Laboratory Technical Report, HPL-2001-191

[7] Rainer Lienhart (2001) Reliable Dissolve Detection. In: Storage and Retrieval for Media Databases, Proc. SPIE 4315, pp. 219-230

[8] A. Nagasaka, Y. Tanaka (1992) Automatic Video Indexing and Full-Video Search for Object Appearances. In: IFIP Proc. of Visual Database Systems, pp. 113-127

[9] Greg Pass, Ramin Zabih (1999) Comparing Images Using Joint Histograms. In: ACM Journal of Multimedia Systems, May 1999, Volume 7 issue 3, pp.234-240

[10] N.V. Patel, I.K. Sethi (1996) Compressed Video Processing for Cut Detection. In: IEEE Proceedings: Vision, Image and Signal Processing, Vol. 143, pp. 315-323

[11] S. Porter, M. Mirmehdi, B.Thomas (2001) Detection and classification of shot transitions. In: Proceedings of the 12th British Machine Vision Conference, BMVA Press, pages 73-82

[12] B. Rousso, S. Peleg, I. Finci (1997) Mosaicing with Generalized Strips. In: DARPA Image Understanding Workshop, pp. 255-260, New Orleans, USA

[13] Y. Yusoff, W.J. Christmas, J.V. Kittler (2000) Video Shot Cut Detection Using Adaptive Thresholding. In: British Machine Video Conference, Bristol

[14] R. Zabih, J. Miller, K. Mai (1995) A Feature-Based Algorithm for Detecting and Classifying Scene Breaks. In: Proceedings of the third ACM International Conference on Multimedia, pp. 189-200

[15] H.J. Zhang, A. Kankanhalli, S.W. Smoliar (1993) Automatic Partitioning of Full-Motion Video. In: Multimedia Systems, Vol. 1, No. 1, pp. 10-28

[16] Y. Zhuang, Y. Rui, T. Huang and S. Mehrotra (1998) Adaptive Key Frame Extraction Using Unsupervised Clustering. In: Proc. of IEEE International Conference on Image Processing, Chicago, IL, p.866-870

[17] Video sequences: “New indians”, ”A new horizon” – http://www.open-video.org/; Canal+ TV Belgium, evening news, 30/11/2002, 12/09/2002