Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 56940, Pages 1–11
DOI 10.1155/ASP/2006/56940

MPEG-2 Compressed-Domain Algorithms for Video Analysis

Wolfgang Hesseler and Stefan Eickeler

Fraunhofer IMK, Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Received 1 September 2004; Revised 2 June 2005; Accepted 6 June 2005

This paper presents new algorithms for extracting metadata from video sequences in the MPEG-2 compressed domain. Three algorithms for efficient low-level metadata extraction in preprocessing stages are described. The first algorithm detects camera motion using the motion vector field of an MPEG-2 video. The second method extends the idea of motion detection to a limited region of interest, yielding an efficient algorithm to track objects inside video sequences. The third algorithm performs a cut detection using macroblock types and motion vectors.

1. INTRODUCTION

The demand for indexing techniques capable of handling increasing amounts of video at low cost can only be satisfied using automatic methods for metadata extraction. The semantics of metadata range from low level over mid-level to high level. Low-level metadata are information that can be extracted more or less directly from the video signal. Typical examples of low-level metadata are the color histogram of an image or spectrogram features of an audio waveform. In most cases, the low-level features are used for simple query by example applications or as a preprocessing step to generate mid- and high-level information. Mid-level metadata are more understandable for humans and can be derived from the images of a video sequence using pattern classification methods. Examples are the location of human faces in video sequences or the spoken words in the audio channel. High-level metadata give a comprehensive semantic description of the data. General methods to create high-level metadata are still under development, but techniques for automatic generation of high-level metadata are highly advanced for special cases like face recognition which gives the name of a person shown in the video sequence, or the topic recognition which gives the type of subject being discussed in the audio.

Usually the level of generated metadata increases within the processing chain, which starts with low-level preprocessing and ends with high-level methods. Generally, computational complexity increases with the level of metadata. It is essential to have efficient methods at all levels of processing.

MPEG-7 is a common standard to store the metadata information that is generated by many automatic video-analysis systems. A video-analysis system that automatically produces metadata conforming the MPEG-7 standard [1] is the iFinder, developed by the Fraunhofer IMK. The algorithms presented in this paper are included as a new module into the iFinder, extend its analytic capacity with additional metadata such as camera motion, and speed up the existing methods.

Most digital video in high quality, such as DVD or DVB (digital television), is encoded in the MPEG-2 compression standard. MPEG is an ISO standardized video- and audio-compression format. A central component of MPEG is motion compensation. With each pixel block (called a macroblock), the encoder looks for a matching pixel block in a previous or following frame and stores only its position using a motion vector. This search is computationally very expensive. Analysis software that generates metadata like camera motion or object tracking can achieve a large speed improvement by manipulating motion vectors in the compressed domain instead of performing a block matching in the uncompressed domain.

The algorithms presented in this paper all work in the compressed domain. Instead of transforming the compressed domain back into the raw pixel domain, the algorithms work in the compressed domain and extract information from the encoded video (e.g., from the DCT coefficients and motion vectors). In most cases, the camera motion detection algorithm is usually able to detect a camera motion with pixel precision even if the underlying motion vector field contains many outliers due to moving objects or due to large

low-textured areas in the video. In this paper, a camera movement is a generic name encompassing a pan or a tracking shot, where the camera changes its position. The object tracking algorithm is based on the camera motion detection algorithm. Objects like faces can be tracked over several minutes in the ideal case. Finally, we present a cut detection algorithm using macroblock types and motion vectors.

2. MPEG-2 VIDEO COMPRESSION

As mentioned above, MPEG-2 compression is the standard for digital video compression. MPEG-2 is downwardly compatible with MPEG-1, which means that the algorithms presented in this paper are also applicable to MPEG-1. The MPEG-2 algorithm uses two basic methods to compress digital video: frequency-domain compression for spatial redundancy (as with JPEG picture compression) and block-based motion compensation for the temporal redundancy. An MPEG-2 video consists of three frame types: I-, P-, and B-frames, which are grouped into GOPs (group of pictures). The I-frame is encoded similarly to a JPEG image. It does not make use of information from any other frame. P-frames are predicted from the previous I- or P-frame. B-frames use parts of both the previous and the following I- or P-frame. All three frame types use the DCT to transform the picture from the spatial domain to the frequency domain, which is better suited to exploit redundancy. In the I-frames, the DCT data is derived directly from the image; whereas in the P- and B-frames, the DCT data is derived from the residual error after prediction. Each frame is divided into 16×16 pixel blocks called macroblocks. In the motion estimation process, the encoder checks each macroblock of a P-frame if it can be predicted from a pixel block in the previous reference frame (I- or P-frame) with a possible offset to compensate for motion (the so-called motion vector) and a residual error which is DCT transformed. If the coefficients become too big, the encoder decides to intracode the macroblock. The encoding of B-frames works similarly. The difference is that B-frames can be predicted from the preceding or the following reference frame or from both frames.

The MPEG-2 standard only specifies the bit format and the decoding of the MPEG-2 stream, the algorithms and methods for the encoding, such as those needed for motion estimation, are not defined in the standard. For this reason, how well the motion vectors reflect the motion of the camera or an object depends on the encoder.

2.1. Related Work

Detecting camera motion in the compressed domain was investigated by Wang and Huang [2]. They use a recursive outlier-rejecting least-square algorithm. Smolic et al. [3] use the M-estimator method to underweight outlier motion vectors. Neither of these algorithms take into account motion vectors of low-textured macroblocks that do not reflect motion. Kuhn [4] uses the DCT coefficients to determine a list of macroblocks with high-texture information and performs an IDCT for them in order to calculate their motion. All these algorithms are not well suited for video sequences with large moving objects. Kobla et al. [5] distinguish eight motion directions and assume a camera motion if the direction receiving the highest number of vectors receives more than twice as many vectors as the second highest direction.

The object tracking algorithms that are commonly used in the uncompressed domain are the mean shift algorithm [6] and the Kalman filter [7]. Also, block-matching algorithms [8] are used where the object is divided into small blocks of constant size that are searched for in the next frame. In the compressed domain, Kobla et al. [9] present a method for determining the flow of a pixel block using motion vectors and frame types. However, this method is too imprecise to use for object tracking. In [10], Khan et al. present experiments with object tracking using compressed-domain MPEG data with P-frames only.

Cut detection algorithms can be divided into compressed and uncompressed algorithms. Uncompressed-domain algorithms use pixelwise difference [11], histograms [12], edge tracking [13], and so forth. Such algorithms are, however, computationally expensive. Some compressed-domain methods use the increasing bit rate as an indicator [14] or approximated DC coefficients [15]. Kobla et al. [5], Calic and Izquierdo [16] use an approach similar to ours by checking the ratios of forward- and backward-predicted macroblocks. However, experiments show many erroneous detections in sections with no or little motion only.

3. DETECTING CAMERA MOTION

The encoder has already done the computationally most complex part of detecting camera motion by calculating a motion vector field. However, the vectors do not always correspond to true motion since the encoder optimizes the MPEG encoding for picture quality and not to reflect the exact motion of the objects covered by the macroblocks. For this reason, numerous outliers are to be expected. Moreover, moving objects will result in motion vectors that do not reflect the camera motion.

3.1. Algorithm

We only use P-frames for the analysis of camera movements since the distances between B-frames and reference frames vary. This means that the camera motion can only be determined for every second or third frame. However this restriction represents only a minor drawback, since every camera movement that does not last longer than several frames can rather be regarded as camera wobble.

A two-dimensional histogram of motion vectors is used and the peak is interpreted as reflecting camera motion. However, the motion vectors spread widely, especially with a fast movement of either camera or objects. For this reason, the histogram of the motion vectors will usually not result in a steep peak, but rather in a peak region. We solve this problem by using a pyramid filter that includes 2 pixels in each direction to detect the maximum. This filter is represented as

a weight matrix $\mathbf{W}$ with elements $w_{i,j}$:

$$\mathbf{W} = \frac{1}{3} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{1}$$

$h_{x,y}$ are the accumulated weights of the motion vectors (x, y). For each vector (x, y), the weighted frequency of the peak region called $H_{x,y}$ is calculated as follows:

$$H_{x,y} = \sum_{i=-2}^{2} \sum_{j=-2}^{2} w_{i+3,j+3} \cdot h_{x+i,y+j}. \tag{2}$$

The peak region around the vectors (x, y) is then given by the maximum $\max(H_{x,y})$. The $H_{x,y}$ values can be calculated quickly using the integral image algorithm [17].

Motion vectors are stored with half-pixel accuracy. Thus, the vector of the peak region has to be divided by two in order to obtain pixel accuracy. We chose a computationally less complex method and divided all motion vectors by two and built the histogram with these values. When interlaced motion estimation was used, the two motion vectors for each macroblock were averaged.

3.1.1. Low-textured areas

Large areas in the video that are low textured, that is, that do not have any edges, result in long chaotic motion vectors that do not reflect the camera motion. Since the macroblocks in such areas are similar to each other, the encoder is free to use an arbitrary pixel block as reference, which minimizes the error, but does not reflect the true motion. Figure 1 shows an example.

A motion vector can be regarded as usable for the detection of camera motion if its macroblock is textured, that is, if its brightness varies. This is reflected by the 63 AC coefficients of the four DCT blocks of which the macroblock consists. The DC coefficient only gives information on the average brightness of the DCT block, but does not indicate the texture.

Increased AC coefficients in the upper left area reflect variations in brightness of the 8×8 pixel block. Most of the other coefficients are zero or near zero. Experiments showed that it is sufficient to evaluate the nine AC coefficients in the top left triangle of the DCT block.

In order to be independent of the scanning pattern used, we select the nine AC coefficients after scanning the coefficients in the zigzag or alternate scan order. The absolute values of the nine AC coefficients of all four DCT blocks comprising the macroblock are added. The sum of these 36 coefficients for the macroblock at the macroblock position (x, y) reflects its texture and is denoted as $S_{x,y}$. This value is used for the weight function $s(S_{x,y})$. Its value range is between 0 and 1. Texture analysis is only performed on the luminance component since the encoder usually also uses only the luminance for motion estimation. When the motion vector histogram is built, the count for each element is not increased by one, but rather by the corresponding weight $s(S_{x,y})$ of the macroblock. In this way, the low-textured macroblocks are underweighted or excluded from the analysis.

Experiments have shown that a simple approach that classifies all macroblocks as either sufficiently or nonsufficiently textured yields adequate performance. So we get

$$s(S_{x,y}) = \begin{cases} 1 & \text{for } S_{x,y} \geq S_{\text{Threshold}}, \\ 0 & \text{for } S_{x,y} < S_{\text{Threshold}}. \end{cases} \tag{3}$$

These weights insure that only the motion vectors of macroblocks that are sufficiently textured are used in the analysis.

3.1.2. Approximation of the DCT coefficients

The DCT coefficients of I-frames and intracoded macroblocks of P-frames can be used directly from the MPEG-2 compressed-domain data after extraction. For non-intracoded macroblocks, only the motion vector and the difference with the referenced pixel block are stored as DCT coefficients. For this reason, the DCT coefficients for these macroblocks have to be calculated. There are computationally expensive algorithms that can do a precise calculation [18–20]. Since we only need information concerning whether the macroblock is sufficiently textured, an approximation is adequate. A DCT block of a P-frame can be built up by using up to four adjacent DCT blocks in a reference frame. If all these four DCT blocks are low textured and the error DCT coefficients are small, the composed DCT block will be low textured, too, assuming that there is no edge between the individual blocks in the spatial domain.

To approximate the DCT matrix of the current 8×8 pixel block B_{Cur}, we calculate the weighted sum of the DCT matrices of the pixel blocks $B_1 - B_4$ plus the error DCT as shown below:

$$\text{DCT}\left(B_{\text{Cur}}\right) = \sum_{i=1}^{4} w_i \cdot \text{DCT}\left(\text{B}_i\right) + \text{DCT}_{\text{Err}}. \tag{4}$$

The weights w_i are given by the fractions of the areas of the DCT block that overlap with the reference DCT blocks. Since we work with half-pixel accuracy, the weights are a multiple of 1/256. Figure 2 sketches the procedure.

The approximated DCT coefficients will be used for the calculation of the DCT coefficients of the subsequent P-frame. Errors will therefore propagate.

When interlaced motion compensation with two motion vectors is used, a special treatment is required. In the frame picture modus, the pixel rows of both 16×8 pixel blocks are interleaved. This is emulated in the DCT domain by averaging each coefficient of the upper and lower DCT blocks. In the field picture modus, the upper and lower 16×8 pixel blocks have their own motion vectors. In the DCT domain,

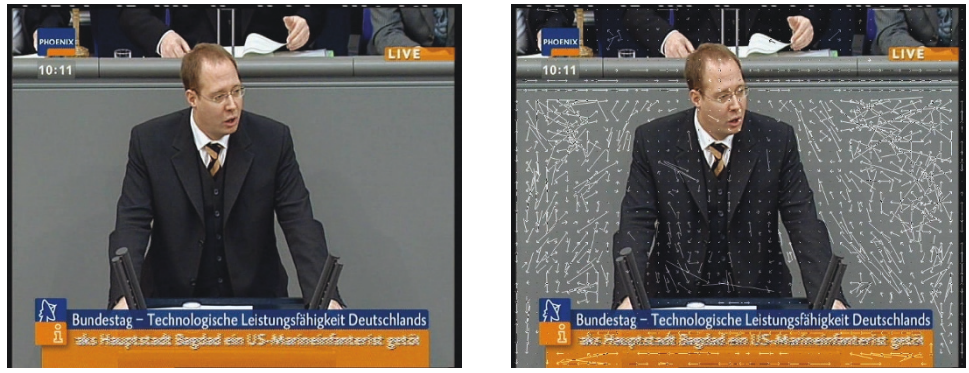

Figure 1: Motion vectors spread in low-textured areas.

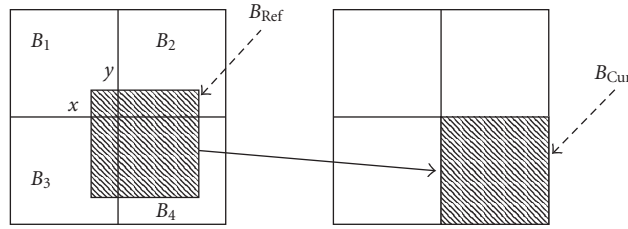

Figure 2: DCT approximation: B_{Cur} uses B_{Ref} as reference which is located inside $B_1 - B_4$. The motion vector is (x, y).

the upper two DCT blocks can be approximated using the upper motion vector and the lower two DCT blocks can be approximated using the lower motion vector.

3.2. Experiments

Comparisons of the approximated DCT with the forward DCT of the decoded data show higher differences especially if textured and low-textured areas are adjacent. However, an exact calculation is not important, but only the classification as sufficiently and nonsufficiently textured. Experiments with newscasts video material resulted in 94% correct classification of the macroblocks compared with the classification using the forward DCT, which is sufficient.

We used test sequences mainly from newscasts that were captured directly from the satellite in MPEG-2 format using a Linux system with DVB-S card. The camera motion was then determined by hand and compared with the computed results of our algorithm, the M-estimator approach, and with a simple average algorithm. The histogram algorithm with texture analysis was usually able to detect the camera motion precisely, whereas the M-estimator and the average algorithms' vectors were too small, especially with fast camera movements. The algorithm proved to be highly robust against objects moving in the foreground. An object did not have an influence on the peak region unless its size increased to nearly 50% of the frame size within the artificial test sequences. In this case, the object motion was interpreted as camera motion. Many smaller moving objects could wrongly lead to a detection of the 0-vector as camera motion.

3.3. Results

The approximated DCT using a simple weighted sum approach is adequate to be used to determine if the macroblock is sufficiently textured so that its motion vector will probably reflect the camera motion. The peak of a two-dimensional histogram using a 5×5 pyramid filter is suited to determine the camera motion.

4. OBJECT TRACKING USING MOTION VECTORS

Encouraged by the results of the camera motion detection, we performed tests involving the use of motion vectors to track objects. An object motion can be regarded as "camera" motion that is restricted to the object. By using the motion vector information, a speed increase could be achieved compared to pixel domain tracking algorithms.

4.1. Differences to camera motion detection

Compared to the detection of camera motion for the whole frame, there are additional challenges for the object tracking that is restricted to a subimage.

4.1.1. Problems using MPEG-2 compressed-domain data

One problem when using the MPEG-2 compressed-domain data is that the motion vectors are not available continuously

for all frames. The number of frames between the current frame and the reference frame varies. Moreover, there are no motion vectors for I-frames.

Another problem is that the macroblock grid does not match the object boundaries. An edge exact tracking is therefore not possible. Instead, a rectangle of constant size containing the object is tracked. The macroblocks and the referenced pixel blocks will partly contain background.

Object tracking using motion vectors has a few requirements concerning the object to be tracked:

(i) object must be textured but not periodically textured;
(ii) object must be of sufficient size;
(iii) object rectangle must contain no or small portions of background only;
(iv) object must not change its appearance abruptly, its size should remain constant.

If the object is not textured, the algorithm used to detect low-textured areas, as previously described, can be used. We primarily performed experiments with faces. If faces are sufficiently large, they meet the requirements on the object listed above and it is possible to track them using motion vectors.

4.1.2. The start object rectangle

The first step to find the object rectangle that should be tracked is best performed in the uncompressed domain. For this purpose, the video must be decoded at least for the first frame after a cut. The object rectangle to be tracked will either be determined by an automated system or by a human. After the initial determination of the rectangle, the object tracking algorithm only uses compressed-domain data. The algorithm is passed the coordinates of the top left corner and the size of the object rectangle. The size of the object will remain constant throughout the whole tracking process. Moreover, the edges of the object rectangle will always be located parallel to the image edges.

4.2. Algorithm

The tracking algorithm applies the histogram in a manner similar to the camera motion detection, plus some additional steps to increase the accuracy.

4.2.1. Histogram to determine the object's motion vector

Since the results of the object rectangle's calculations are used again with subsequent frames, a high accuracy is required. As seen with the camera motion detection, the histogram algorithm with texture analysis can usually determine the exact number of pixels. Since we assume that the object is sufficiently textured, texture analysis is not required. We rely on a two-dimensional histogram which includes only the motion vectors of macroblocks that are at least partly located inside the object rectangle. A slight and slow appearance change of the object like the mouth movement of a face does not usually influence the peak of the histogram. The majority of the motion vectors still reflect the object movement.

The object rectangle will in most cases not correspond to the macroblock grid so that many macroblocks or referenced pixel blocks will partly contain background. We therefore introduce a weight for each macroblock or referenced pixel block, respectively. If all 16×16 pixels are located inside the object rectangle, the weight is 1, otherwise, the weight is proportional to the number of overlapping pixels.

The analysis is done with half-pixel accuracy. This means that the histogram is built using half-pixel without rounding to pixels beforehand. The object rectangle's coordinates are also internally calculated with half-pixel accuracy. The two motion vectors of macroblocks using interlaced motion estimation are both used with half-weight when building the histogram.

$h_{x,y}$ are the accumulated weights of the motion vectors (x, y). For each vector (x, y) the weighted frequency of the peak region called $H_{x,y}$ is calculated. Because of the half-pixel accuracy, an extended range of three half-pixels (compared to two pixels with the detection of camera motion) is used to determine the peak region. The weight matrix $\mathbf{W}$ with elements $w_{i,j}$ is represented as

$$\mathbf{W} = \frac{1}{4} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 3 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{5}$$

and the peak region $H_{x,y}$ is calculated as follows:

$$H_{x,y} = \sum_{i=-3}^{3} \sum_{j=-3}^{3} w_{i+4,j+4} \cdot h_{x+i,y+j}. \tag{6}$$

The calculation of the object rectangle is dependent on the frame type. Intracoded macroblocks are always ignored.

The motion vectors of forward-predicted macroblocks specify the position in the frame from which the macroblock originates, whereas motion vectors of backward-predicted macroblocks specify to which position in the next reference frame the macroblock will move.

4.2.2. Backward search with P-frames

P-frames can only have motion vectors of forward-predicted macroblocks. They specify the position of the pixel block that is referred to in the previous reference frame. The number of pixels by which this pixel block and the object rectangle's overlap is calculated. The overlap specifies the weight of the motion vector used to build the motion vector histogram. The vector of the peak region is calculated by applying a method similar to that used in the camera motion approach. Adding this vector to the coordinates of the object rectangle in the reference frame gives the coordinates of the object rectangle in the current P-frame. This rectangle is used as the reference rectangle for the following frames.

4.2.3. Backward search and prediction with B-frames

The determination of the current object rectangle for B-frames is done in the same way that it was with P-frames, except that the forward portions of bidirectional motion vectors are included in the histogram as well. The reference rectangle is not updated since B-frames are never used as reference.

No motion vectors are available, either for I-frames or for P-frames, in which all macroblocks that are part of the object rectangle are intracoded. We therefore use the backward references of B-frames to make a prediction concerning where the object rectangle will move in the next reference frame. This is done again by using a histogram, but this time the histogram is built up using the motion vectors of backward-predicted macroblocks and the backward portions of bidirectional motion vectors. The sum of the peak-region vector and the object rectangle's coordinates of the B-frame gives the object rectangle's coordinates in the next reference frame.

Consequently, for a P-frame in the sequence B_1 B_2 P, usually three calculations of the object rectangle are available which are identical in the ideal case: the two predictions of the B-frames and the calculation for the P-frame. We use the latter one because it is most reliable since it is calculated in one step. In contrast, a prediction of a B-frame requires two steps, the B-frame calculation and the B-frame prediction. However, experiments showed that the predictions that were calculated with the B-frames are more precise than the P-frame calculations if the motion vectors spread, especially in the case of fast object movements. Therefore, an additional pseudomotion vector with weight 2 is added to the P-frames histogram. It is determined by the difference between the coordinates of the object rectangle in the reference frame and the prediction. This pseudomotion vector does not influence the maximum in cases in which no fast movement occurs.

4.2.4. Using predicted coordinates with I-frames

For I-frames, the prediction of the object rectangle derived from the previous B-frame is used. This prediction is also used for P-frames in which all macroblocks that are part of the object rectangle are intracoded. The reference rectangle is updated with these coordinates.

Figure 3 shows an example sequence with three consecutive frames with type P, B, and I. The object rectangle in the P-frame (depicted with solid lines in the left picture) is known when the B-frame is analyzed. Each macroblock in the B-frame is checked as to whether a pixel block of the object rectangle in the P-frame is used as reference. The macroblock depicted with dashed lines is located completely inside the object rectangle so that its weight is 1 whereas the weight of the macroblock depicted with broken lines is smaller since it is located partly inside the object rectangle. The object motion vector is calculated using the histogram. By subtracting the motion vector from the coordinates of the object rectangle in the reference frame, its position in the B-frame can be determined. To predict the object rectangle's position in the I-frame, a histogram of the motion vectors of backward predicted macroblocks and backward- portions of bidirectional motion vectors is built.

The algorithm does not support rotating objects. The object rectangle's orientation will always remain the same so that a 90-degree rotation will have the effect that the object rectangle contains background from that point on. This is also the case if the object size decreases. However, a slow increase of size will usually not result in problems. It is essential that the encoder uses B-frames without preferring the previous or the following reference frame.

4.3. Experiments

We primarily performed experiments with faces. For comparison, the ground-truth data of the position of the rectangle containing the face were determined using the Rowley face detection algorithm [21]. The rectangle position in the first frame of the sequence together with its size was used as start object rectangle for the tracking algorithm. Another face detection method, which could have been used to establish ground truth, is presented in [22].

Figure 4 shows the tracking of a speaker in the German parliament. The object rectangle in the first frame was determined using the Rowley algorithm. It is the starting point of the tracking. The white-dotted rectangle shows the reference tracking by the Rowley algorithm, whereas the solid rectangle depicts our tracked object rectangle. The speaker's head moves several times quickly up and down and sidewards. The Rowley network was only trained on frontal faces and cannot therefore always find the face with sufficient confidence. Our tracking algorithm can link these face detections so that it is possible to automatically determine that it is the same face. The differences between the positions of the Rowley algorithm and ours are only a few pixels. Even if the head is slightly tilted, the face can still be tracked. The size of the rectangle that was determined using the Rowley algorithm varies, whereas the size of our rectangle is constant.

Experiments with low-quality video material, such as recordings from security cameras, gave poor results. The motion vectors do not reflect the object motion adequately. This is also the case if the object moves too fast or changes its appearance quickly. For this reason, the quality of the tracking is highly dependent on the quality of the motion vectors. A high-quality video with low noise is required as well as an encoder with a good motion estimation.

Table 1 shows the tracking results of five high-quality video sequences.

4.4. Results

The results of the presented object tracking using MPEG-2 motion vectors are very good if the video material is of high quality and the MPEG-2 encoders have a good motion estimation as they do with newscasts and parliament transmissions. Consequently, these are the fields of application of object tracking using motion vectors. If the video material is of low quality, especially noisy and blurry as with security cameras, the motion vectors do not reflect the object motion so that object tracking using motion vectors is not possible.

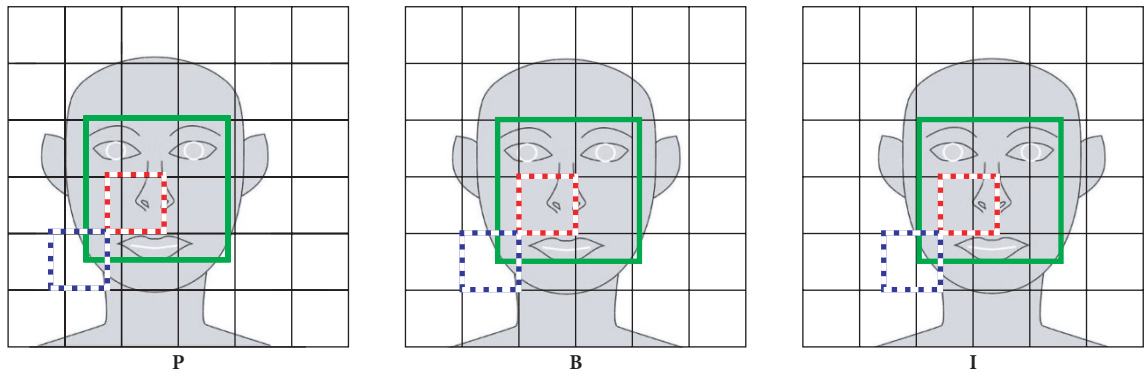

Figure 3: Example of object tracking: the object rectangle (with solid lines) is calculated for the B-frame shown in the middle. An object's motion vector is calculated relative to the object rectangle's position in the P-frame. After this, an object's motion vector is calculated to predict the object rectangle's position in the I-frame. The grid depicts the macroblock boundaries, the squares with dashed and broken squares depict two examples: macroblocks and the referenced pixel blocks, respectively.

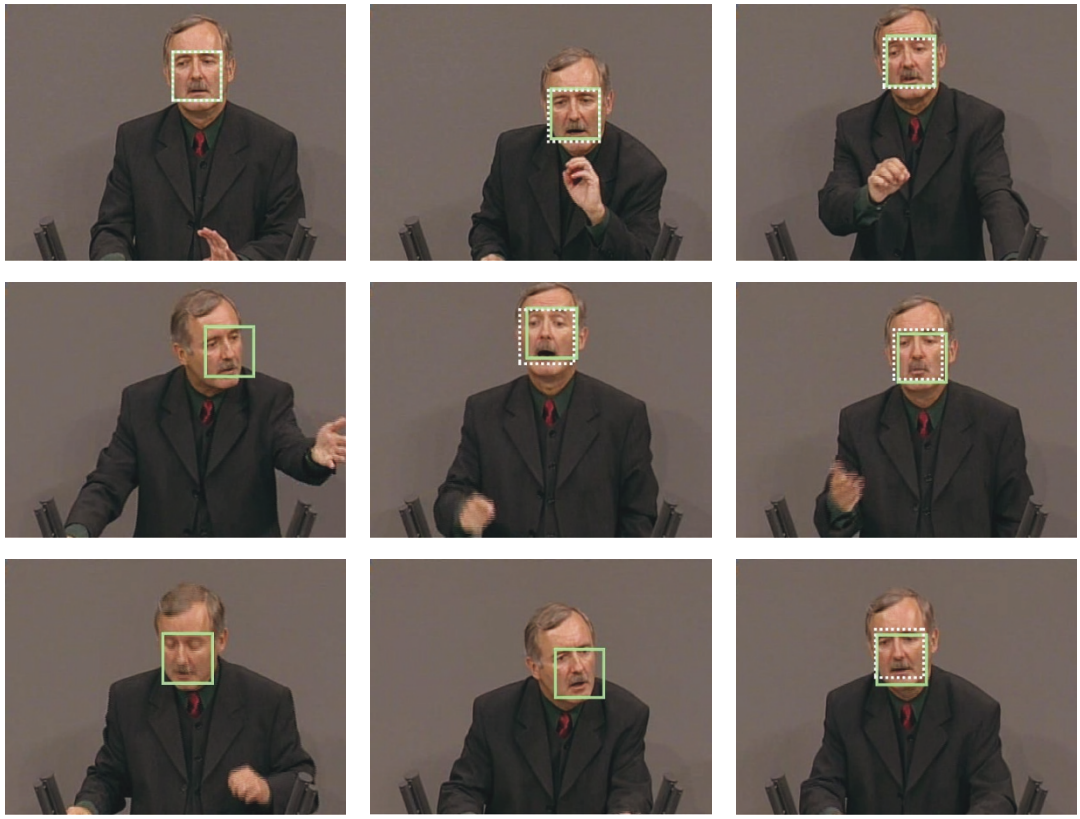

Figure 4: Face tracking for 53 seconds of a parliament speaker. The solid rectangle localizes the tracked object. The dotted rectangle reflects the reference data which were determined using the Rowley algorithm. In several cases, the reference algorithm failed to detect a face with sufficient confidence.

5. CUT DETECTION

The first step of any video sequence analysis is usually the cut detection. A cut is defined as an abrupt scene change between two consecutive frames.

Cut detection algorithms using the ratio of forward- and backward-predicted macroblocks rely on the fact that B-frames are "average" frames between two reference frames. For this reason, the number of macroblocks using the preceding and the following reference frames should be roughly

Table 1: Tracking test results of five video sequences. Example frames of test sequence 3 are shown in Figure 4. Rectangle's average overlap means the average overlap of the tracked object rectangle and the face rectangle determined by the Rowley face detector. In test sequence 5 the face is lost when the speaker tilts his head too quickly. The rectangle overlap then falls below 50% so that the object can be regarded as lost.

Test sequence	Tracked object	Duration	Total frames	Rectangle's average overlap
1	Anchor person	50 s	1250	97.8%
2	Anchor person	52 s	1303	94.1%
3	Parliament speaker	53 s	1335	89%
4	Parliament speaker	140 s	3492	91%
5	Parliament speaker	Lost after 17 s	428	82.4%

identical. A significant change in the forward/backward ratio indicates a cut. Referential dependencies across cuts tend to lead to large motion vectors since frames on opposite sides of cuts are typically markedly different. These motion vectors can be expected to spread widely.

5.1. Algorithm

We use a combined evaluation of B-frame references and motion vectors. Figure 5 shows the three possible cut types. R_1 and R_2 stand for reference frames, in between are the B-frames B_1 and B_2. The arrows indicate the references of the majority of the macroblocks. The number of B-frames between the reference frames is arbitrary, but at least one. We use two B-frames in the example, which give a GOP such as IBBPBBPBBPBB.

(i) In the first case, most macroblocks from B_1 use pixel blocks from R_2 as reference. This indicates a cut between R_1 and B_1.
(ii) In the second and third cases, most macroblocks from B_1 use the reference frame R_1. At the moment of analyzing B_1, we can conclude whether the following cut occurs before or at R_2. It is necessary to check all of the following B-frames to determine if the macroblock-type ratio has inverted in order to localize the cut more exactly.
(iii) In the second case, the macroblock-type ratio has inverted in B_2. Therefore, a cut is detected at that frame.
(iv) If the ratio has not inverted by the time we reach R_2, the cut must be at R_2. This is shown in the 3rd case.
(v) If there was no significant ratio of forward/backward types, it is assumed that no cut occurred in within that sequence.

Note that after a cut has been detected at a B-frame, no further cut can be detected until the next reference frame.

5.1.1. Problem: little motion activity

There are general problems with cut detection using macroblock types if there is no or little motion as is the case with video depicting static graphics. The encoder does not need to use references to both reference frames, but uses a reference to the previous or following reference frame only; in which direction a reference is established depends on the encoder's preferences. For this reason, the cut detection method described above will always detect a cut. We therefore discard macroblocks whose motion vectors are 0. The idea is that after a cut, the position of the macroblock most probably differs from the position of the referenced pixel block. Bidirectional macroblocks are always used because they imply that no cut has occurred.

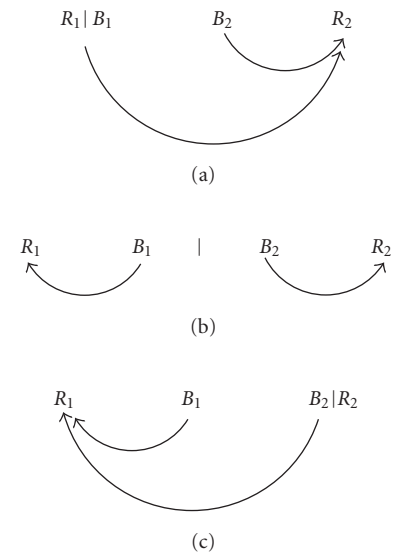

Figure 5: The three cut types. R_1 and R_2 are reference frames, B_1 and B_2 are the two B-frames, | indicates the cut position. The arrows indicate the references of the majority of the macroblocks.

Experiments showed that the results improve if all macroblocks with motion vectors whose absolute values are neither vertically nor horizontally greater than 1 are not used. Such vectors are half-pixel motion vectors that are primarily used because of coding efficiency. Effectively, we only consider the forward/backward predicted ratio of macroblocks that have moved or will move compared to the previous or following reference frame.

$B_{\text{For}}^{>1}$ is the number of forward-predicted macroblocks whose motion vectors are greater than 1 for at least one direction, $B_{\text{Back}}^{>1}$ is the analogous value for backward-predicted macroblocks. The number of bidirectional macroblocks is

denoted as B_{Bi} and the number of intracoded macroblocks as B_{Intra}.

The ratios A_{For} and A_{Back} are defined as

$$A_{\text{For}} = \frac{B_{\text{For}}^{>1} + B_{\text{Bi}} + R}{B_{\text{For}}^{>1} + B_{\text{Back}}^{>1} + B_{\text{Bi}} + R},$$
$$A_{\text{Back}} = \frac{B_{\text{Back}}^{>1} + B_{\text{Bi}} + R}{B_{\text{For}}^{>1} + B_{\text{Back}}^{>1} + B_{\text{Bi}} + R}, \tag{7}$$

where R is a value that compensates the noise in the distribution of macroblock types. It increases the ratio if the number of considered macroblocks is low. R depends on the total number of macroblocks B_{Total} and a constant r as follows:

$$R = B_{\text{Total}} \cdot r. \tag{8}$$

A cut of type 1 (see Figure 5) is assumed if

$$A_{\text{For}} < t_{\text{For}}, \tag{9}$$

where t_{For} is a constant. If the ratio is in an expanded range A_{For} with

$$A_{\text{For}} < t_{\text{For}}^{+}, \tag{10}$$

a cut hypothesis with lower probability is assumed. We therefore check the spread of the forward portion of motion vectors of the considered macroblocks using the average deviation from the mean of the motion vectors. This value is denoted as $S_{\text{For}\,XY}$ and is compared with the constant $S_{\text{Deviation}}$. A cut is assumed if

$$S_{\text{For}\,XY} > S_{\text{Deviation}}. \tag{11}$$

The idea is that in case the reference frame is located before the cut, the forward-predicted macroblocks will change their positions drastically without preferring one direction.

A cut of type 2 or 3 (see Figure 5) is assumed if

$$A_{\text{Back}} < t_{\text{Back}}. \tag{12}$$

Again, if the ratio is in an extended range with

$$A_{\text{Back}} < t_{\text{Back}}^{+}, \tag{13}$$

we check the spread of the considered motion vectors, here the backward-predicted portions. Similarly, if

$$S_{\text{Back}\,XY} > S_{\text{Deviation}}, \tag{14}$$

a cut is assumed. However, in these cases we can only identify a cut up until the next reference frame. For this reason, each following B-frame is checked to see if the number of forward-predicted macroblocks is smaller than the number of backward-predicted macroblocks and a cut is assumed if this is the case. If no cut position was found by the next reference frame, this reference frame must be the cut position.

5.1.2. Cut validation using the following P-frame

To lower the number of false detections such as the ones occuring frequently with static computer-generated graphics, the P-frame after a detected cut can be used to validate the cut. The encoder will probably not be able to take over pixel blocks in the P-frame from the same position in the previous reference frame. The ratio of nonmoving motion vectors is calculated as

$$A_{\text{For}}^{0/1} = \frac{B_{\text{Total}} - B_{\text{For}}^{>1} - B_{\text{Bi}} - B_{\text{Intra}}}{B_{\text{Total}}} \tag{15}$$

and compared with a threshold constant t_{Valid}. A cut hypothesis occuring since the last reference frame can be assumed to be unlikely if

$$A_{\text{For}}^{0/1} > t_{\text{Valid}}, \tag{16}$$

and can therefore be rejected. The cuts can only be verified if the reference frame after cut is a P-frame. In case of an I-frame, no information can be obtained. Some encoders enforce an I-frame after a cut. This is, however, not a problem since only false cuts need to be identified.

5.2. Experiments

The tests were primarily done with newscast video material. They contain a broad test spectrum including, for example, speakers with static background, static computer-generated graphics, camera flashes, fast camera movement, reports with bad picture quality, and so forth. Table 2 shows the results of five test sequences. The cut positions were determined by hand.

The average percentage of detected cuts varies from 94.8% to 100.0% depending on the encoder and the video content. The average percentage of incorrect detected cuts (false alarms) varies from 0% to 23.0%. Up to 4.1% of the detected cuts differed from the actual positions by one frame. Parliament transmissions generally give better results than newscasts, since they do not contain typical problems.

The causes for nondetected cuts are, in descending order of importance as follows.

(i) The encoder did not use the characteristic distribution of macroblock types.
(ii) The frames before and after the cut are too similar.
(iii) The cut occurred between two half-pictures.

The causes for cuts detected where no cut actually occurred are in descending order of importance as follows:

(i) the consecutive frames are too similar, there are (almost) no changes so that it was sufficient for the encoder to use references to one reference frame only;
(ii) camera flashes;
(iii) the encoder did not use the characteristic distribution of macroblock types;
(iv) very quick camera motion.

No or little motion only is still a problem. This especially applies to computer-generated graphics since they do not

Table 2: Results of cut detection test for five videos. # reference cuts is the number of cuts in the ground truth. B-frames is the number of B-frames between two reference frames. Deletion = number of missed cuts/divided by number of reference cuts, insertion = number of falsely detected cuts/divided by number of detected cuts, and inaccuracy = $\sum$ displacement of cut/divided by number of detected cuts.

Video type	Duration	Total frames	# Ref. cuts	B-frames	Deletion	Insertion	Inaccuracy
Newscast	15 min	22500	132	1	1.5%	2.9%	0.0 frames
Newscast	15 min	22500	167	1	1.2%	2.3%	0.0 frames
Newscast	15 min	22500	96	2	5.2%	11.1%	0.009 frames
Newscast	15 min	22500	97	2	4.1%	23%	0.032 frames
Parliament	54.7 min	82054	81	2	0%	0%	0.0 frames

contain the noise that leads to small differences between the frames. In order to avoid the repeated detection of a cut, only the macroblocks with motion vectors of at least two half-pixels are considered. However, imposing this limitation can lead to a situation in which the number of remaining macroblocks is small, causing the result to be unreliable. In this case, both a nondetected cut and a erroneously detected cut are possible. Camera flashes lead to increased brightness in a single frame or a half-frame only. If a flash occurs in a reference frame that frame, will likely have less referential dependencies, which may lead to an erroneously detected cut. Cuts between two half-pictures can occur if the video material was produced with analog equipment. The first half-picture of a frame belongs to a scene and the second half-picture belongs to the next scene. Very quick camera motion can also lead to false detection of a cut since the motion vectors spread.

Generally, the number of nondetected cuts can be reduced by adjusting the parameters so that more cuts are detected, also resulting in more false alarms. All cut hypotheses could be checked using the DC coefficients of the approximated DCT as was shown with the camera motion detection. Such checking does not compromise speed improvement over uncompressed-domain techniques.

The performance of the algorithm presented here depends on the encoder. The parameters have to be adjusted to accommodate the particular encoder. It is necessary that the encoder uses B-frames and does not always prefer one reference frame like the next I-frame.

5.3. Results

The cut detection algorithm using compressed-domain data only can detect 94.8%–100% of all cuts in a video. The performance is dependent on the encoder and the video material. Only sequences with no or little motion remain problematic despite special treatment and in such cases, the detection is not reliable. Also, encoders do not always encode the video as presumed by the proposed algorithms but are rather programmed with the coding efficiency in mind.

A typical application of the cut detection method would be integration in distributed online television monitoring systems. The basic units for processing of video sequences are the shots. These shots are distributed to the different nodes of a computer cluster for a fast processing. The detection of the shots must occur before the distribution and is as such the bottleneck. For this reason, an algorithm which can detect shots faster than real time is highly desirable.

6. CONCLUSIONS

In this paper, three different algorithms for low-level metadata extraction were presented. All three algorithms work in the MPEG-2 compressed domain, and are therefore efficient, saving time for the further processing steps that create metadata on a higher semantic level. The algorithms operate in realtime on a computer with an 866 MHz Pentium III processor under Linux.

Detection of cuts and camera motion can be performed immediately after parsing a frame in bitstream order without having to wait for the data of the following frames. This procedure allows on-the-fly cut and camera motion detection, for example, with live videos from a satellite. The object tracking algorithm requires the frames to be in display order, which gives a delay of up to three frames (less than 100 milliseconds).

Considering that the encoder's task is to encode the best picture quality with a given bit rate and not with the camera or object motion in mind, the results of the compressed-domain algorithms are convincing. The results show that it is not necessary to expend computing power on decoding in order to perform low-level metadata extraction in the pixel domain.

The algorithms presented in this paper are not fundamentally limited to MPEG-2. They can be extended to all video compression methods that use motion compensation and bidirectional frames, for example, MPEG-4 simple and advanced profile (also known as DivX).

REFERENCES

[1] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*, John Wiley & Sons, New York, NY, USA, 2002.

[2] R. Wang and T. Huang, "Fast camera motion analysis in MPEG domain," in *Proceedings of IEEE International Conference on Image Processing (ICIP '99)*, vol. 3, pp. 691–694, Kobe, Japan, October 1999.

[3] A. Smolic, M. Hoeynck, and J.-R. Ohm, "Low-complexity global motion estimation from P-frame motion vectors for MPEG-7 applications," in *Proceedings of IEEE International Conference on Image Processing (ICIP '00)*, vol. 2, pp. 271–274, Vancouver, BC, Canada, September 2000.

[4] P. M. Kuhn, "Camera motion estimation using feature points in MPEG compressed domain," in *Proceedings of IEEE International Conference on Image Processing (ICIP '00)*, vol. 3, pp. 596–599, Vancouver, BC, Canada, September 2000.

[5] V. Kobla, D. Doermann, and A. Rosenfeld, "Compressed domain video segmentation," Tech. Rep. CAR-TR-839 (CS-TR-3688), University of Maryland, College Park, Md, USA, 1996.

[6] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of 7th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 2, pp. 1197–1203, Kerkyra, Greece, September 1999.

[7] B. Danette Allen and G. Bishop, "Tracking: Beyond 15 Minutes of Thought," in *SIGGRAPGH 2001 Course 11 Booklet*, Los Angeles, Calif, USA, August 2001.

[8] A. Gyaourova, C. Kamath, and S.-C. Cheung, "Block matching for object tracking," Tech. Rep. UCRL-TR-200271, Lawrence Livermore National Laboratory, Livermore, Calif, USA, 2003.

[9] V. Kobla, D. Doermann, K.-I. Lin, and C. Faloutsos, "Feature normalization for video indexing and retrieval," Tech. Rep. CAR-TR-847 (CS-TR-3732), University of Maryland, College Park, Md, USA, 1996.

[10] J. I. Khan, Z. Guo, and W. Oh, "Motion based object tracking in MPEG-2 stream for perceptual region discriminating rate transcoding," in *Proceedings of 9th ACM International Conference on Multimedia (ACM Multimedia '01)*, pp. 572–576, Ottawa, Ontario, Canada, September–October 2001.

[11] H. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," *ACM Multimedia Systems*, vol. 1, no. 1, pp. 10–28, 1993.

[12] A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-video search for object appearances," in *Proceedings of IFIP 2nd Working Conference on Visual Database Systems*, pp. 113–127, North-Holland, September–October 1991.

[13] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying scene breaks," in *Proceedings of 3rd ACM International Conference on Multimedia (ACM Multimedia '95)*, pp. 189–200, San Francisco, Calif, USA, November 1995.

[14] G. Bozdagi and H. T. Sencar, "Preprocessing tool for compressed video editing," in *Proceedings of IEEE 3rd Workshop on Multimedia Signal Processing*, pp. 283–288, Copenhagen, Denmark, September 1999.

[15] A. Bovik, Ed., *Handbook of Image and Video Processing*, Academic Press, New York, NY, USA, 2000.

[16] J. Calic and E. Izquierdo, "Towards real-time shot detection in the MPEG-compressed domain," in *Proceedings of Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '01)*, pp. 95–100, Tampere, Finland, May 2001.

[17] P. Viola and M. J. Jones, "Robust real-time object detection," Tech. Rep. CRL 2001/01, Cambridge Research Laboratory, Cambridge, Mass, USA, 2001.

[18] S.-F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 1–11, 1995.

[19] K. Shen and E. J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proceedings of IEEE International Conference on Image Processing (ICIP '95)*, vol. 2, pp. 252–255, Washington, DC, USA, October 1995.

[20] B.-L. Yeo and B. Liu, "On the extraction of DC sequence from MPEG compressed video," in *Proceedings of IEEE International Conference on Image Processing (ICIP '95)*, vol. 2, pp. 260–263, Washington, DC, USA, October 1995.

[21] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.

[22] H. Wang and S.-F. Chang, "A highly efficient system for automatic face region detection in MPEG video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 615–628, 1997.

Wolfgang Hesseler received his Diploma in computer science from the University of Bonn, Germany. The algorithms presented in this paper were developed during his work at the Fraunhofer IMK in Birlinghoven, Germany, where he wrote his Diploma thesis about using MPEG-2 compressed-domain data for computer vision. His areas of interest are video indexing and video analysis.

Stefan Eickeler received his Dipl.-Ing. degree and Dr.-Ing. degree in electrical engineering from the University of Duisburg, Germany, in 1996 and 2001, respectively. He joined the Fraunhofer Institute for Media Communication (Fraunhofer IMK) in the year 2001. Since 2003 he has taught as a lecturer at the Bonn-Aachen International Centre for Information Technology. His research interests are statistical pattern recognition and signal processing with the applications face and gesture recognition and automatic media and document analysis.