FAST HYBRID BLOCK/REGION-BASED
ALGORITHM FOR OBJECT TRACKING

F. Jordan, M. Schütz, and M. Kunt

Signal Processing Laboratory,
Swiss Federal Institute of Technology,
CH-1015 Lausanne, Switzerland
Email: jordan@ltssg3.epfl.ch
Tel: ++41 21 693 70 88
Fax: ++41 21 693 70 90

ABSTRACT

This paper presents a technique to segment and track a region of interest in a video sequence. The presented technique starts with a spatio-temporal contour extraction. This contour extraction is performed by mixing information from a spatial gradient and a temporal differentiator. A block segmentation is then applied on the image, and within each block the contour part is approximated by a straight line segment. These contour approximations are then further quantized by a vector quantization technique, using a predefined codebook. In order to close and smooth the region of interest contour, an interpolation based on Hermite polynomials is performed on the set of discrete quantized contour parts. Finally, the tracking process is performed using an addition-removal of blocks containing contour parts by testing their possible belonging to the polynomial contour. This testing is done by using a blurred version of the polynomial contour in order to be able to track deformations in the object. The initial guess for tracking is given manually by an end-user.

Keywords: Contour extraction, quantization, and interpolation; block based; tracking; deformable objects; supervised segmentation; Hermite polynomials; shape coding; video processing.

1. INTRODUCTION

With the development of new multimedia applications and the future video standard MPEG-4, new image processing algorithms must be developed in order to reach high compression ratios as well as to provide various functionalities, such as the detection and manipulation of moving objects in video sequences. Classical block-based compression schemes which have been developed in the past can not easily provide the required functionalities, since this kind of segmentation has no way to detect or follow the real moving contours existing in the scene. Nevertheless, block-based approaches have widely proved their efficiency both in the rate/distortion trade-off [1] and reasonable computing time [2]. This paper presents a way to upgrade a block-based coding scheme into a region-based one. Similar approaches can be found in [3, 4].

In order to reduce the blocking artifacts introduced by a pure block-based approach and to improve the quality of the predicted image, several techniques to split blocks into two or more regions have been proposed in [5, 6, 7, 8]. These motion field refinement techniques define a new non-pure block-based segmentation. The block splitting is usually done using a predefined codebook which describes the
patterns in which a block can be sub-segmented. Such a codebook is a vector-quantized representation of all possible ways to split a block in several sub-regions. It has been shown in [5, 6] that the new segmentation obtained by such a block splitting technique is more correlated to the contours existing in natural images, the latter being a by-product of the motion field refinement which was originally designed for compression purposes by means of reduction of the displaced frame difference (DFD) entropy.

In this paper we propose a method which allows to define and track in time a closed region in a sequence. The closed region should correspond to a real moving object in the scene. To initialize the process, a manual selection of the region of interest is performed by the user. Then, the contours are extracted from this region of interest and approximated by a list of line segments. This list gives a rough approximation of the contour of the object we intend to segment and track. The tracking is performed by means of a matching process of line segments between the current frame and the previous one. Finally, the closed contour is computed using a Hermite polynomial.

The implementation of the presented segmentation and tracking technique has been made with real-time constraints in a software-based system on a massively parallel super-computer. For this reason the proposed segmentation technique has been designed with the following additional constraints:

- Low computation load
- Backward compatible with already existing block-based codecs.

This paper is organized as follows: Sec. 2 gives a brief overview of the complete technique and motivations for the performed investigations. The detailed description of the various tools used is given in later sections. First, Sec. 3 describes the spatio-temporal contour extraction scheme which has been developed. Then, Sec. 4 describes the technique used to quantize the contour parts. Sec. 5 explains how the quantized contour parts are interpolated in order to find smooth and closed contours. Finally, Sec. 6 deals with the developed object/contour tracking technique. Simulation results are shown and discussed in Sec. 7 and finally, Sec. 8 gives some concluding tracking remarks.

2. DESCRIPTION OF TECHNIQUE

The motion field refinement technique by means of block splitting proposed in [5] was based on properties of the DFD. This enabled to minimize the DFD energy, in other words to decrease its entropy, thus to increase the compression ratio for a given final quality of the decompressed sequence when compared with a pure block-based technique like MPEG-2 for example. It has been shown in [5, 6] that the segmentation refinement of the blocks can roughly be correlated with moving object boundaries. The major drawback of this segmentation technique is the heavy computational cost. Thus it can not directly be used for the real-time constrained implementation we have in mind.

In the proposed technique, a combined spatial and temporal contour extraction technique is chosen as a starting point. The obtained contours are then used to define the line segments. These line segments are taken from the predefined codebook. In more details, the result of the contour extractor is split up into blocks. Thus, we can have a small piece of contour in some of the blocks. These small pieces of contour are then approximated using the best fitting line segment that can be found in a given codebook.

These codebook-based approximated contour segments are then used as guide points and slopes for a Hermite polynomial interpolator, which is used in order to produce a smooth closed contour of a region of interest in the sequence. This region of interest is defined by the end-user, since he is the only one who is able to define what region he wants the system to extract from the sequence and track through time.

The region tracking process is performed by means of contour tracking. More precisely, the contour segments are matched with the polynomial contour. A decision process adds or removes some of the
control blocks where interesting vector quantized contour parts are located. This adding and removing process allows the presented technique to track deformable objects through time.

3. CONTOUR EXTRACTION

The contour extraction technique we used in this work is based on a combined spatial and temporal contour extractor. The idea is to use a combination of a gradient-based spatial contour extractor and a temporal frame differentiator in a weighted sum in order to get spatio-temporal contours.

The spatial contour $S_{i,j}$ of pixel at location $i,j$ is obtained by means of a gradient based contour extraction:

$$S_{i,j} = |I_{i-1,j} - I_{i+1,j}| + |I_{i,j-1} - I_{i,j+1}|.$$  \hspace{1cm} (1)

The temporal contour $T_{i,j}$ of pixel at location $i,j$ is computed as the difference between its value in the current frame and the one in the previous frame:

$$T_{i,j} = |I_{i,j} - I_{i,j}^-|.$$  \hspace{1cm} (2)

Finally the spatial $S_{i,j}$ and temporal $T_{i,j}$ components of our contour extractor are combined together in a weighted sum in order to get our spatio-temporal contour $C_{i,j}$:

$$C_{i,j} = \frac{\alpha \cdot S_{i,j} + \beta \cdot T_{i,j}}{\alpha + \beta}.$$  \hspace{1cm} (3)

The weights $\alpha$ and $\beta$ are a description of the tradeoff between spatial and temporal contours. Typically we used $\alpha = 8$ and $\beta = 2$.

4. VECTOR QUANTIZATION OF CONTOURS

The vector quantization of the contours obtained after contour extraction is done in two steps. First, the contour part contained in a block is approximated by a straight line segment. In a second step the codebook element which is geometrically the closest to the straight line segment is chosen.

4.1. Contour approximation with a line

To approximate the contour within a given block with a straight line, we look for two contour points on the boundaries of the block, as illustrated in Fig. 1. Contour points are simply points with high gradient values. Thus, the $C_{i,j}$ values (Eq. 3) located in the gray shaded area of Fig. 1 are scanned for maximum values. One maximum location is selected per block side. The two biggest of the four maximas are chosen in order to find the coordinates $(i_1, j_1)$ and $(i_2, j_2)$. These two coordinates give the two end points of the straight lines we intend to find. This technique ensures us not to take two end points on the same block side, which would lead to an unusable result.

4.2. Selection of the codebook element

In order to reduce the transmission cost for the segmentation parts, the straight lines approximating the contours are quantized using a vector quantization technique. In such a technique a codebook describing a sub-set of all possible block segmentations is used on the coder and the decoder side. Thus, for each block, a code describing its segmentation is transmitted; a zero code indicating that the block is not part of the contour. Fig. 2 gives an example of a codebook.
Figure 1: Contour approximation

Figure 2: Codebook elements used for segmentation (131 elements).
If $e_{k,l}$ denotes an element of the codebook for the block at coordinate $(k, l)$, and since the block splitter used to generate the codebook used straight lines to separate a block in two regions, $e_{k,l}$ can be described by the end points $(i, j)_{e1}$ and $(i, j)_{e2}$ of the straight line segment. In order to select the closest codebook element for a given line approximated contour, a minimization of a distance function is performed. The distance $d$ between two lines given by their end points is given by:

$$d = \sqrt{(i_{e1} - i_{k1})^2 + (j_{e1} - j_{k1})^2} + \sqrt{(i_{e2} - i_{k2})^2 + (j_{e2} - j_{k2})^2}.$$  \hfill (4)

For each block of the image which contains a contour part, the minimization process of done using all the codebook elements. The result of this process is a block-based description of the contours, as shown in Fig. 3.

![Figure 3: Contours approximated by codebook elements.](image)

### 5. CONTOUR INTERPOLATION

The segments which have been computed in Sec. 4 represent a rough definition of the object shapes as can be seen in Fig. 3. In order to define a closed region, it is necessary to "connect" the segments. One way to achieve such a task consists in defining the contour as a polynomial $p(s)$.

The proposed method approximates contours using multiple Hermite polynomials. The curve is drawn between pairs $(m, m + 1)$ of consecutive blocks which belong to the contour and that we call "control blocks". Each of these control blocks uses the line segment orientation of the codebook element to evaluate the derivative of the polynomial, as described in Fig. 4. The center $(i, j)_m$ of a segment $m$ is called the "control point" of the polynomial and defined as:

$$(i, j)_m = \left(\frac{i_{p1} + i_{p2}}{2}, \frac{j_{p1} + j_{p2}}{2}\right).$$  \hfill (5)

The Hermite polynomials are computed for each pair of blocks $(m, m + 1)$ using two constraints which are the derivatives and the control points. The two derivatives are both multiplied by the distance $D$ between the two control points. This allows to adapt the derivative constraint when two control points are not equally spaced. Thus, when control points are distant, we can impose a hard constraint on the derivative. When the control points are close, better results are obtained when the derivative constraint is lowered. Actually this gives a rough control on the maximum curvature of the contour. The derivative
of a polynomial can now be written as:

\[
\frac{d(i, j)_m}{ds} = D \cdot \left( \frac{i_{p1} - i_{p2}}{l}, \frac{j_{p1} - j_{p2}}{l} \right),
\]  

(6)

where \( l \), the segment length is defined as:

\[
l = \sqrt{(i_{p1} - i_{p2})^2 + (j_{p1} - j_{p2})^2}.
\]  

(7)

Each of the \((i, j)(s)\) coordinates can now be written as a Hermite polynomial:

\[
p(s) = as^3 + bs^2 + cs + d.
\]  

(8)

The coefficients \(a, b, c\) and \(d\) of the Hermite polynomial can be found for \(i(s)\) and \(j(s)\) using the following constraints:

\[
\begin{align*}
p(0) &= (i, j)_{i_1}, \\
p(1) &= (i, j)_{i_{p2}}, \\
p'(0) &= \frac{d(i, j)_m}{ds}, \\
p'(1) &= \frac{d(i, j)_{m+1}}{ds}.
\end{align*}
\]  

(9)

Fig. 5 shows an example of a region closed using such a polynomial after line and codebook approximation of contours. It may be noticed that this technique can also be used for shape coding, as shown in [9].

6. TRACKING

The object at time \(t\) is defined with a list \(L(t)\) of codebook elements and the corresponding \((k, l)\) block coordinates:

\[
L(t) = \{(k, l)_r\}(t).
\]  

(10)

The tracking process is initialized with a manual drawing of the region of interest, as illustrated in Fig. 6. The shape of the region of interest is updated from frame to frame using two processes: a removal of segments from the list according to a motion criteria and an addition of new segments to the list. Fig. 7 shows the processes.
Figure 5: Example of a region closed by Hermite polynomial.

Figure 6: Manual definition of the region of interest.

Figure 7: System diagram.
6.1. Element removals

From frame to frame, and for each block defining the contour, the distance \( d \) (defined in Eq. 4) between two codebook elements \( e_{k,l} \) belonging to this block at time \( t-1 \) and time \( t \) is computed. A threshold \( T \) is applied to this distance to decide whether the element should be removed or kept in the new list \( L \):

\[
d(e_{k,l}(t), e_{k,l}(t+1)) < T, \quad \forall (k, l) \in \{(k, l)_r\}.
\]

The criteria then enables the tracking of objects whose motion is small (lower than \( T \)). Since it is applied to each element of \( L \), it allows the tracking of deformable object through time. This process creates a new list \( L'(t+1) \), which has always less elements than the previous list \( L(t) \). With this removal technique, noisy elements or elements which move out of the current block are removed from the list:

\[
L'(t+1) = L(t) - \{(k, l)_r | d \geq T\}.
\]

6.2. Element additions

In order to not completely destroy the list of elements, an “additive” process is needed, which will add new segments to the new list \( L(t+1) \). These elements will be created either from the motion of an element from another block or if a new contour appears.

To select new elements, an assumption on the smoothness of the object’s shape is done: if the smoothness is sufficient, the polynomial \( p(s) \) created with \( L'(t+1) \) should approximate correctly the shape. For this reason, we use the polynomial \( p(s) \) to select new blocks \( (k, l) \) where an element to be added to the list \( L'(t+1) \) can be found. The blocks \( (k, l) \) whose codebook elements \( e_{k,l} \) “belong” to the polynomial \( p(s) \) constructed with \( L(t) \) are added to the new list to create:

\[
L(t+1) = L'(t+1) + \{(k, l)_r | e_{k,l} \in p(s)\}.
\]

To give the “belonging” property better chances than a trivial dual choice, it must be defined such that it also works for contour approximations and not only perfect matches. The first process in the “belonging” computation is to get a blurred version of the polynomial contour in a block. First the polynomial contour is drawn in white on a black background, as shown in Fig. 5. We denote with \( P' \) the black and white image of the polynomial \( p(s) \) in one block, which is defined by the following rule:

\[
\forall m, \quad P' = \begin{cases} 
255, & \exists s \in [0,1] \mid (i, j) = (i(s), j(s))_m \\
0, & \text{else}.
\end{cases}
\]

The blurred version of the polynomial image in one block, denoted by \( P' \), is obtained by applying a \( 5 \times 5 \) kernel operator.

For the same block as for \( P' \) and \( P \), we denote by \( E \) the black and white image of the codebook element line \( e_{k,l} \) defined for the block \( r \). We can now define two values \( a \) and \( b \) defining the sums of matched and unmatched contour pixels within the block \( r \):

\[
a = \sum_{(i,j) \in r} P_{i,j} + E_{i,j}, \\
b = \sum_{(i,j) \in r} |P_{i,j} - E_{i,j}|.
\]
It is now possible to define an estimation of the “belonging” or matching of codebook element $e_{k,l}$ with the polynomial $p(s)$ within a given block with a normalized value $c$:

$$c = \frac{a - b}{a},$$

(16)

where $c$ simply expresses how many pixels have the same location between a codebook element line and the polynomial part within a given block. Fig. 8 shows an example of the superposition of a blurred polynomial contour and a codebook element line segment in a block.

![Blurred polynomial contour and codebook element line](image)

Figure 8: Codebook/polynomial superposition used for matching process in a 12x12 block.

**7. SIMULATION RESULTS**

The simulations have been made with a 352x288 8 bits “LTS” video sequence, with 12x12 blocks. A region is selected at frame 200. Fig. 9 shows the extracted region “girl” for frame 201 and frame 225 of the test sequence “LTS”. The frame 225 shows the result after recursive tracking of 25 frames. Fig. 10 shows the extracted region “hand” for frames 204, 213 and 225 of the test sequence “LTS”. The list $L(t)$ has been updated 25 times between these two frames, using the above described process.

Fig. 11 shows in details the updating of the list $L(t)$. During the tracking, it is possible to notice that the number of elements which define the shape does not vary very much, which indicates a good equilibrium between removed and added elements. The number of removed and added elements is irregular, which shows that the updating process is working correctly.

**8. CONCLUSION**

The segmentation technique described in this paper allows to obtain a joint detection and segmentation of objects on the basis of a block-based segmentation coding scheme. Since each stage of the algorithm works only at block level (instead of the usual pixel level), the computational load remains very low. In practice, it requires about 0.2 second per image to obtain the segmentation and tracking on a “Impact” Silicon Graphics workstation.

The presented technique allows a smooth transition from a block-based to an object-based video processing approach, thus achieving the backward compatibility constraint. The quantization of extracted contours, combined with the contour reconstruction by means of Hermite polynomials allows a low cost
Figure 9: Extracted regions from the sequence “LTS” at frame: a) 201, b) 225.

Figure 10: Hand tracked in the sequence “LTS” at frame: a) 204, b) 213, c) 225.
representation of the contours in terms of data transmission. Furthermore, the contour parts updating scheme allows tracking of moving and deformable objects in video sequences.

9. REFERENCES


