A Time-Consistent Video Segmentation Algorithm designed for Real-Time Implementation

M. EL Hassani[1], S. Jehan-Besson[2], L. Brun[2]
M. Revenu[2], M. Duranton[3], D. Tschumperlé[2], D. Rivasseau[1]
.
[1] NXP Semiconductors, 2 rue de la girafe BP 5120 14079 Caen Cedex 5
[2] Laboratoire GREYC, 6 Boulevard du Maréchal Juin 14050 Caen, France
[3] NXP Semiconductors, 5656 AE Eindhoven - Eindhoven,Netherlands

***Abstract*— In this paper, we propose a time consistent video segmentation algorithm designed for real-time implementation. Our algorithm is based on a region merging process that combines both spatial and motion information. The spatial segmentation takes benefit of an adaptive decision rule and a specific order of merging. Our method has proven to be efficient for the segmentation of natural images with few parameters to be set. Temporal consistency of the segmentation is ensured by incorporating motion information through the use of an improved change detection mask. This mask is designed using both illumination differences between frames, and region segmentation of the previous frame. By considering both pixel and region levels, we obtain a particularly efficient algorithm at a low computational cost, allowing its implementation in real-time on the TriMedia processor for CIF image sequences.**

I. Introduction

The segmentation of each frame of a video into homogeneous regions is an important issue for many video applications such as region-based motion estimation, image enhancement (since different processing may be applied on different regions), 2D to 3D conversion. These applications require two main features from segmentation: accuracy of regions boundaries in the spatial segmentation and temporal stability of the segmentation from frame to frame.

As far as spatial segmentation is concerned, it can be classified into two main categories, namely contour-based and region-based methods. In the first category, edges are computed and connected components are extracted [1]. One of the drawback of such an approach is that the computation of the gradient is prone to large errors especially on noisy images. Moreover, the closure of the edges in order to create connected regions is a difficult task and an efficient resolution of such a problem may induce cumbersome computations. Finally, such an approach cannot take benefit of statistical properties of the considered image regions. The region based segmentation methods avoid these drawbacks by considering regions as basic elements. Among region-based segmentation methods [2], [3], [4], [5], [6], we are interested here in a bottom-up segmentation approach where regions are grown using a merging process. In such approaches, similar neighbouring regions are merged according to a decision rule [7], [8]. The initial regions can be the pixels or an over-segmentation of the image which can be obtained by a watershed algorithm [9], [10]. As mentioned by [11], bottom-up algorithms rely on three notions: a model for the description of a region, a merging predicate and a merging order. This gives rise to numerous heuristics according to the different choices performed on these three steps [12], [13], [7], [14], [4]. Compared to other classical approaches, e.g. [12], [13], [7], the authors of [4] have proposed recently an adaptive threshold justified by statistical inequalities. They obtain good results with few parameters to tune. However in the context of a real time implementation, their merging predicate still requires too many computations. Moreover, their algorithm is dedicated to the segmentation of still images and so, it does not take into account the temporal dimension of video sequences.

When dealing with video segmentation, various algorithms have been tested in the literature. The first class of approaches proposes to perform a 3D segmentation by considering the spatio-temporal data as a volume. We can cite the work of [15] that takes benefit of the 3D tensor of structures for segmentation. Some other recent works propose 3D approaches using a mean shift based analysis [16], [17]. Let us note that if each shot is segmented as a 3D volume the number of frames to store for each segmentation may be unbounded. On the other hand, if the number of stored frames is artificially limited by the available memory, some 3D regions may be artificially split on long shots. Therefore, 3D approaches require the storage of several frames in memory and necessitate a high bandwidth which is a drawback for the design of electronic devices.

The second class of methods concerns frame-by-frame algorithms. In these approaches, the spatial segmentation of the second frame is deduced from the spatial segmentation of the first frame using motion estimation [18], [13], [19], [20]. Regions from adjacent frames are then merged according to motion similarity, colour similarity or localisation similarity. In such approaches, a matching is performed between regions of the different frames. All the regions are then linked and video objects tracking algorithms [20] may then take benefit of such a correspondence between regions.

On the other hand some applications such as image enhancement or video compression, may need a coherent segmentation between frames without requiring an exact tracking of each region from frame to frame. In this paper, we propose a segmentation algorithm devoted to such applications. The first aim of our algorithm is thus not to match the regions of two consecutive frames but only to take benefit of the spatial segmentation of the first frame in order to construct a coherent

spatial segmentation of the second one.

Our contributions may be divided in three points :

- Spatial segmentation: Our spatial segmentation takes benefit of both an adaptive decision rule and an original order of merging. As in [4], the adaptive threshold is computed using a statistical modelisation of the region combined with the statistical inequality of McDiarmid [21]. However, in our approach, each pixel is modelled as a single random variable (in [4], the authors model each pixel as a sum of M random variables). This method gives a simpler predicate that is more adapted to real time implementation. Good results are obtained for spatial segmentation with few parameters to be set.
- Temporal consistency: Another contribution is the design of a region segmentation that does not encounter strong variations over time. We then propose to simply take benefit of scene change detection, that is widely used in video segmentation [22], [23], [24], rather than motion estimation that remains a real bottleneck for real-time implementation. We construct a coherent segmentation from frame to frame by combining both pixel and region information through the use of an improved Change Detection Mask (CDM) that takes benefit of the region segmentation of the previous frame. Experimental results conducted on real video sequences demonstrate a good temporal consistency.
- Hardware implementation: As far as the implementation is concerned, we exploit the Data Level Parallelism (DLP) by processing some basic treatments in parallel. Moreover, the classical UNION-FIND data structure [25] is improved by using local registers to reduce the access time of FIND operations. We obtain an efficient algorithm for video segmentation at a low computational cost. Our method runs in real-time on the TriMedia processor for CIF image sequences.

The paper is organised as follows. The spatial segmentation method is detailed in section II. The temporal consistency improvement is explained in section III. In section IV, we discuss the implementation of the algorithm. Experimental results and measures are given in section V.

II. SPATIAL SEGMENTATION

Let us consider an image I, the notation $|.|$ represents the cardinal and $I(p,n)$ the pixel intensity at position $p=(x,y)^T$ in the frame n.

A region-based segmentation problem aims at finding a relevant partition of the image domain in m regions $\{S_1, S_2, .., S_m\}$. We focus here on region merging algorithms where a decision criterion determines whether two regions must be merged or not. In this paper, we first introduce a statistical model for the regions. We then detail how these statistical tools are used for the computation of the merging predicate. We finally explain the whole merging algorithm and especially the order of merging.

A. Statistical model

Images are corrupted by noise which gives random values (r.v.) to pixel intensities. Due to this random part in image acquisition systems, an image I is classically considered to be an observation of a perfect statistical image I^*. The intensity $I(p)$ of a pixel $p=(x,y)^T$ is then modelled as the observation of a random vector X_i whose values belongs to the interval $[0,g]$ (e.g. $g=255$ for 8 bits images). An ideal region S^* is then represented by a vector of independent r.v $(X_1, X_2, .., X_n)$, where $n=|S^*|$. Let us denote by S the real region associated to S^*, i.e. composed of the same set of pixels than S^*. The intensity of the i^{th} pixel of S within I is then considered as an observation of the r.v. X_i. Following [4], we define a partition of I^* into homogeneous regions $\{S_1^*, \ldots, S_m^*\}$ by the following requirements:

1) All the pixels of any statistical region should have a same expectation:
$$\left.\begin{array}{lcl} \forall i & \in & \{1,\ldots,m\} \\ \forall (p,q) & \in & (S_i^*)^2, \end{array}\right\} E(I^*(p)) = E(I^*(q)) \quad (1)$$
2) Two adjacent pixels belonging to different statistical regions should have different expectations:
$$\left.\begin{array}{l} \forall (i,j) \in \{1,\ldots,m\}^2 \\ \forall (p,q) \in S_i^* S_j^* \\ p \text{ adjacent to } q \end{array}\right\} E(I^*(p)) \neq E(I^*(q)) \quad (2)$$

Such a definition may be easily extended to multi-channels images [4] by requiring that the pixel's expectations are equal on each channel within one region and that the expectation of at least one channel differs between pixels belonging to different regions.

Note that according to our definition, all the pixels of one region should have a same expectation. The regions extracted by a segmentation algorithm based on this definition should thus be composed of pixels with a nearly constant intensity (we thus assume an underlying flat facet model). This criterion may be justified by the reflective properties of surfaces. Indeed, the reflection of light under a surface is determined by a Lambertian and a Specular component [26]. The Specular component produces specular spikes often characterised by regions with a nearly maximal intensity. The specular component decreases abruptly and may be neglected, within a segmentation scheme, outside the specular spikes. The intensity of a Lambertian surface varies slowly according to its normals. A region of the image with a nearly constant value correspond thus either to a specular spike or to a Lambertian surface with an almost constant normal. Such a segmentation scheme provides thus a partition which resumes the main physical and geometrical properties of a 3D scene. Higher level processes such as the segmentation of the image into objects or the segmentation of textured objects [27] would require to input within the algorithm a priori knowledge about what are the expected objects of the scene or what a textured area is.

In order to be self content, let us now introduce the very useful statistical inequality proposed by [21] and introduced within the region segmentation framework by [4]. We take benefit of this inequality for the computation of the merging predicate.

Theorem 2.1 (McDiarmid's inequality): If $\{X_l\}$ are N independent random variables whose observations x_l take their

values in a measurable space A, and $f : A^N \longmapsto \mathbb{R}$ is a function that satisfies the following constraint for $1 \leq l \leq N$:

$$\sup |f(x_1,..,x_N) - f(x_1,..,x_{l-1},x'_l,x_{l+1},..,x_N)| < c_l$$

where x_l and x'_l are two different possibilities for the l^{th} component of an observation vector $(x_1,..,x_N) \in A^N$. Then for every $\epsilon > 0$,

$$P(|f(X_1,..,X_N) - E(f(X_1,..,X_N))| > \epsilon) \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{l=1}^{N} c_l^2}\right)$$

B. Merging predicate

In order to compute a merging predicate, we consider two regions S_1 and S_2 of a current partition. The associated vectors of r.v. in the ideal image I^* are respectively denoted by $\mathbf{Y}_1$ and $\mathbf{Y}_2$. The r.v. $\mu_1(\mathbf{Y}_1)$ and $\mu_2(\mathbf{Y}_2)$ denote respectively the means of $\mathbf{Y}_1$ and $\mathbf{Y}_2$. We suppose that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to a same homogeneous region of I^*. Our default decision rule consists thus to merge the two regions S_1 and S_2 respectively associated to $\mathbf{Y}_1$ and $\mathbf{Y}_2$. However, under the hypothesis that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are included in a same homogeneous region of I^*, the probability that $|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)|$ is greater than a given value is bounded by Theorem 2.1. If this probability falls under a given threshold we refuse the hypothesis and thus do not merge the two regions S_1 and S_2.

More precisely, let us consider the vector

$$\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2) = (\underbrace{I^*(p_1),...,I^*(p_{|S_1|})}_{\mathbf{Y}_1}, \underbrace{I^*(p'_1),...,I^*(p'_{|S_2|})}_{\mathbf{Y}_2})$$

and the mean functions:

$$\mu_i(\mathbf{Y}_i) = \frac{1}{|S_i^*|} \sum_{k=1}^{k=|S_i|} I_i^*(p_k),\ i = 1,2$$

Our merging decision rule is based on the following theorem:

Theorem 2.2: Let us consider two vectors of r.v. $\mathbf{Y}_1$ and $\mathbf{Y}_2$ encoding the intensities of two connected regions of an ideal image I^*. Under the hypothesis that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are included into a same homogeneous region and using the previously defined notations we have:

$$P(|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > \epsilon) \leq 2exp\left(\frac{-2\epsilon^2|\mathbf{Y}_1||\mathbf{Y}_2|}{g^2(|\mathbf{Y}_1| + |\mathbf{Y}_2|)}\right)$$

where $(|\mathbf{Y}_j|)_{j\in\{1,2\}}$ denotes the size of vector $\mathbf{Y}_j$ (i.e. the cardinal of the associated region S_j).

Proof: Let us consider the vector $\mathbf{y} = (x_1,\ldots,x_N)$ in $[0,g]^N$. This vector may be considered as an outcome of the r.v. $\mathbf{Y}$. In order to apply the McDiarmid theorem we define the following function:

$$f(\mathbf{y}) = f(x_1,\ldots,x_N) = (\mu_1(\mathbf{y_1}) - \mu_2(\mathbf{y_2})) \qquad (3)$$

where $N = |\mathbf{Y}_1| + |\mathbf{Y}_2|$, $\mathbf{y_1} = (x_1,\ldots,x_{|\mathbf{Y}_1|})$ and $\mathbf{y_2} = (x_{|\mathbf{Y}_1|+1},\ldots,x_N)$.

Let us compute the variation of the function. If we make a variation on the intensity of one x_l with $l \leq |S_1|$. We have :

$$\sup |f(x_1,..,x_n) - f(x_1,..,x'_l,..,x_n)| \leq \frac{g}{|S_1|}$$

This gives us the value of the bounding coefficients $c_l = \frac{g}{|\mathbf{Y}_1|}$ for the $|\mathbf{Y}_1|$ first variables. Similarly, if we make a variation on the intensity of x_l, $l \in \{[\mathbf{Y}_1|+1,\ldots,N\}$, we obtain $c_l = \frac{g}{|\mathbf{Y}_2|}$. We then compute the sum over all the variables:

$$\sum_{l=1}^{N} c_l^2 = g^2\Big(\frac{1}{|\mathbf{Y}_1|} + \frac{1}{|\mathbf{Y}_2|}\Big) \qquad (4)$$

Moreover, according to our hypothesis, if $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to a same homogeneous region of I^*, all the pixels of $\mathbf{Y}_1$ and $\mathbf{Y}_2$ have the same expectation. We have thus, $E(f(\mathbf{Y})) = E(\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)) = 0$ and we obtain the expected result using conjointly Theorem 2.1 and equation 4. ■

Note that the bounds on the probability provided by Theorem 2.2 may be equivalently represented by:

$$P\big(|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > F^{-1}(\delta)\big) \leq \delta$$
$$\text{with} \quad \delta = F(\epsilon) = 2\,exp\left(\frac{-2\epsilon^2|\mathbf{Y}_1||\mathbf{Y}_2|}{g^2(|\mathbf{Y}_1| + |\mathbf{Y}_2|)}\right).$$

After some basic calculus we find that, under the assumption that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are included into a same homogeneous region of I^* we have with a probability at most δ:

$$|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > g\,Q\sqrt{\frac{|\mathbf{Y}_1| + |\mathbf{Y}_2|}{|\mathbf{Y}_1||\mathbf{Y}_2|}}$$

with $Q = \sqrt{\frac{1}{2}ln\left(\frac{2}{\delta}\right)}$

Below the probability δ, which is supposed to be low, we consider that the event $|\mu_1(\mathbf{Y}_1) - \mu_2(\mathbf{Y}_2)| > F^{-1}(\delta)$ is not probable. In this case, we refuse the initial hypothesis stating that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to a same homogeneous region of I^* and thus do not merge the two regions. Our merging predicate may thus be stated as follows:

$$P(S_1,S_2) = \begin{cases} true & \text{if} \quad |\mu_1 - \mu_2| \leq Q\,g\sqrt{\frac{|S_1|+|S_2|}{|S_1||S_2|}} \\ false & \text{otherwise} \end{cases} \qquad (5)$$

where μ_1 and μ_2 denote respectively the values of $\mu_1(\mathbf{Y}_1)$ and $\mu_2(\mathbf{Y}_2)$ for the observation I. These two terms represent the mean value of the two regions S_1 and S_2. The term g denotes the maximum level of I ($g = 255$ for gray-scale images).

Note that our merge criterion is equivalent to:

$$\frac{|S_1||S_2|}{|S_1| + |S_2|}(\mu_1 - \mu_2)^2 \leq (Qg)^2$$

The left member of this last equation corresponds to the difference between the squared error of $S_1 \cup S_2$ and the sum of the squared errors of S_1 and S_2 [28]. Our merge criterion may thus be also interpreted as a bound on the increase of the squared errors of the regions.

Our criterion may be adapted to multi-channels images as follows:

$$P(S_1,S_2) = \begin{cases} true & \text{if} \quad \max\limits_{c\in\{a,b,c\}} |\overline{c_1} - \overline{c_2}| \leq Q\,g_c\,\sqrt{\frac{|S_1|+|S_2|}{|S_1||S_2|}} \\ false & \text{otherwise} \end{cases}$$

where $\overline{c_i}$ represents the mean value of the region S_i for the channel c taken in the set of channels $\{a, b, c\}$ and g_c denotes the maximum value on channel c. We take the maximum of the values obtained for each channel as a criterion. Indeed if the predicate is true, it will be true for all the channels and so the merge hypothesis is accepted. In this paper we have chosen the YUV space which is the native colour space of video sequences.

Both our method and the one of Nock [4] are based on the McDiarmid's inequality. However, Nock models each pixel of the ideal image I^* as a sum of M random variables whereas our method only uses one r.v. per pixel. The approach proposed by Nock consists to fix the probability δ and to use M in order to vary the merge threshold. To our point of view, the probability δ below which we refuse the merge hypothesis has a more straightforward interpretation than the variable M. The resulting criteria are slightly different, our criterion differs by a factor $\frac{1}{\sqrt{M}}$ from the one first proposed by Nock. Our criterion is also significantly different from the second Nock's criterion which uses an estimate of the number of final regions whose cardinal is equal to a given value. However, both our criterion and the final Nock's criterion may be related, our one being more strict than the one of Nock [4] for a given probability δ.

Let us note finally, that the way we derived our criterion provides an alternative explanation to the eventual over-merging produced both by our algorithm and the one of Nock. Indeed, our basic hypothesis consists to suppose that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ belong to a same homogeneous region of I^*. As in a contrario approaches first introduced by [29], we refuse this hypothesis only when we observe an event which has a low probability (according to δ) to occur under this hypothesis. We may thus merge regions corresponding to different homogeneous regions of I^* if our observation does not contradict our hypothesis.

C. Merging order

An edge e denotes a couple of adjacent pixels (p, p') in a 4-connectivity scheme. The set of edges of an image is denoted by A_e and the number of edges by N_e. The order of merging is built on the edges weights as in [4], [12]. The idea behind this order of merging is to merge first similar regions rather than different ones. The similarity between pixels is measured by computing the distance between two pixel colours as follows:

$$w(p, p', n) = |I(p, n) - I(p', n)|. \tag{6}$$

For colour images, the edge weight becomes :

$$w(p, p', n) = \sqrt{\sum_{I \in \{a,b,c\}} (I(p, n) - I(p', n))^2}. \tag{7}$$

where a, b, c denote the three channels of a particular color space.

Note that alternative weight may be designed. For example, one may balance the distance along each axis of a color space by some weight (or equivalently scale each axis according to its weight). Numerous color space with different properties may be chosen in equation 7. For our algorithm, we consider the YUV colour space which is the native colour space of CIF sequences. The colour space $(L^*a^*b^*)$ provides partitions with a little greater subjective quality but with a higher computational cost.

The edges are sorted in increasing order of their weights and corresponding couples of pixels are processed in this order for merging. This sorting step only requires two traversals of the image: the first traversal allows to compute the histogram of edge weights. The second traversal stores each edge in an array associated to its weight. The amount of memory required for each array is deduced from the histogram of edge weights. This sorting step is similar to the one usually used within the watershed algorithm [9].

D. Merging algorithm

Our spatial segmentation could be divided in three steps. In the first one, we compute the weights of edges and their histogram. In the second step, we sort edges increasingly according to their weights. In the last step we merge pixels or regions connected by edges following their order. The algorithm 1 describes more particularly the merging loop:

Algorithm 1 Merging regions algorithm

```
for i := 1 to N_e do
    Read the i^th edge: (p_1, p_2)
    S_1 = FIND(p_1)
    S_2 = FIND(p_2)
    if P(S_1, S_2) = True then
        UNION(S_1, S_2)
    end if
end for
```

The term N_e represents the number of edges within the image I in the 4-connectivity. In the merging process, we use the UNION-FIND data structure [25]. The UNION function merges two disjoint regions into one region, and the FIND function identifies the region a certain pixel belongs to. Implementation details are given in section IV.

III. TIME CONSISTENCY IMPROVEMENT

In video segmentation, the quality of the spatial segmentation is not the only requirement, time consistency is also a very important one. If in two successive frames, one region is segmented very differently because of noise, occlusion or deocclusion, results of segmentation would be very difficult to exploit for any application like image enhancement, depth estimation and motion estimation. Many works, see for example [19], use motion estimation to improve time consistency in video segmentation. However, motion estimation [30] is a real bottleneck for real-time implementation and is even sometimes unreliable. In this paper, we combine an improved Change Detection Mask (CDM) with spatial segmentation in order to improve the temporal consistency of our segmentation.

A. Change Detection Mask

The CDM is designed using both illumination differences between frames and region segmentation of the previous

frame. We first detect changing pixels using the frame difference. Then, we take benefit of the region segmentation of the previous frame in order to classify the pixels not only at a pixel level but also at a region level.

Given the current frame $I(:,n)$ and the previous one $I(:,n-1)$, the frame difference FD is given by:

$$FD(n,p) = |I(p,n) - I(p,n-1)|\,.$$

Classically, FD is thresholded in order to distinguish changing pixels from noise. The pixel label is given by:

$$L(n,p) = \begin{cases} 0 & \text{if} \quad FD(n,p) \leq tr_1. \\ 1 & \text{otherwise.} \end{cases}$$

where tr_1 is a positive constant chosen according to the noise level of the image. This threshold may be set experimentally (section V) or estimated according to any measure of the image noise. A pixel p, with $L(n,p) = 1$ is considered as a changing pixel. We then use the previous segmentation in order to convert the CDM from the pixel level to a region level which is more reliable [23]. For each region S_i in the previous segmentation, we compute $N_{i,changing}$:

$$N_{i,changing} = |\{p \in S_i, L(n,p) = 1\}|$$

which denotes the number of changing pixels of the current image whose (x,y) coordinates belong to S_i in the previous segmentation. We then compute $\tau(S_i) = \frac{N_{i,changing}}{|S_i|}$ which represents the ratio of changing pixels between the previous and the current image in the region S_i. Pixels are then classified using three categories:

$$CDM(n,p) = \begin{cases} 0 & \text{if} \quad (\tau(S_i) \leq tr_2). \\ 1 & \text{if} \quad (\tau(S_i) > tr_2)\ and\ (L(n,p) = 0). \\ 2 & \text{if} \quad (\tau(S_i) > tr_2)\ and\ (L(n,p) = 1). \end{cases} \quad (8)$$

where tr_2 is a positive constant. In the experiments, we take $tr_2 = 0.01$ (i.e. a region is a changing region when it contains at least 1% of changing pixels). The value of the threshold is chosen so that we don't miss any changing region.
Every pixel of regions qualified as static is labelled using $CDM(n,p) = 0$. The two other labels concern pixels within changing regions. Depending on the value of the frame difference, the pixel is qualified as a changing one ($CDM(n,p) = 2$) or as a not changing one ($CDM(n,p) = 1$). Such a classification is then used to segment the current frame. An example of classification is given in Fig.1 for the video sequence "Table".

B. Merging process

The merging process is now divided in three main steps. Firstly, static regions are kept as they were segmented in the previous frame. Secondly we apply a connected component labelling (CCL) algorithm [31] to extract connected components of pixels with $CDM(n,p) = 1$. This second step builds seeds from the segmentation of the previous frame. These seeds link the current segmentation to the previous one in a time consistent way. Thirdly, we apply the spatial segmentation only on edges (p,p') connecting a changing pixel within a changing region ($CDM(n,p) = 2$) to a pixel belonging to

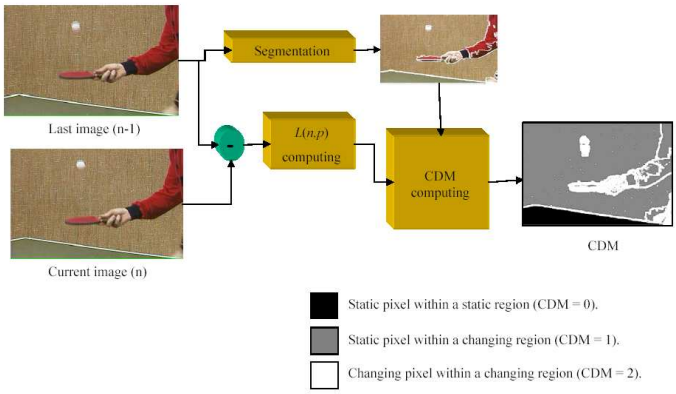


Fig. 1. Computation of the CDM using the difference between the current image and the previous one and the region segmentation of the previous frame.

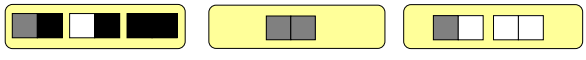
a changing region. This last pixel may be either changing or static($CDM(n,p) \in \{1,2\}$). Note that static pixels within changing regions have been connected in the second step by a CCL algorithm.

The whole process can be formalised as follows. Considering an edge (p_i, p_i') between two pixels, we define the following function:

$$\varphi(n,(p_i,p_i')) = \begin{cases} 0 & if\ CDM(n,p_i)CDM(n,p_i') = 0. \\ 1 & if\ CDM(n,p_i)CDM(n,p_i') = 1. \\ 2 & if\ CDM(n,p_i)CDM(n,p_i') \geq 2. \end{cases} \quad (9)$$

The φ function allows us to classify the edges in the following three categories (a brief summary is provided by Fig.2):

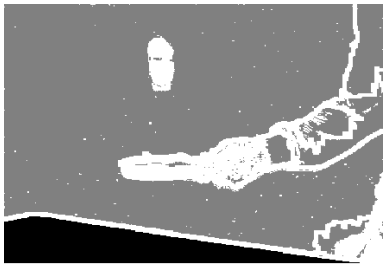
- The first category ($\varphi(n,a) = 0$) (Fig.2(a)) corresponds to the edges which have at least one pixel belonging to a static region. These edges are not considered for the segmentation of the current image n. Static regions are then segmented in the same way between two successive images $n-1$ et n.
- The second category ($\varphi(n,a) = 1$) (Fig. 2(b)) corresponds to the edges that connect two non changing pixels in changing regions. For these edges, we simply apply a connected component labelling (CCL) algorithm [31].
- The third category ($\varphi(n,a) \geq 2$) (Fig. 2(c)) corresponds to the edges which have at least one pixel that is considered as a changing one (i.e. $CDM(n,p_i) = 2$). These edges are processed using the merging order and the merging predicate defined in section II-B. Edges belonging to this category are denoted by A_u.



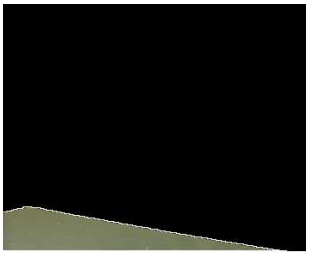
(a) $\varphi(n,a) = 0$ (b) $\varphi(n,a) = 1$ (c) $\varphi(n,a) = 2$

Fig. 2. The figure gives the different combinations of pixels available for each category. The pixels are designed as follows : black pixel ($CDM(n,p) = 0$), gray pixel ($CDM(n,p) = 1$), white pixel ($CDM(n,p) = 2$).

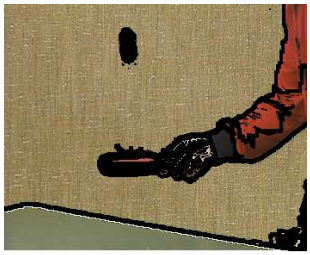
Fig. 3 describes the three steps corresponding to the process of the three categories of edges.



(a) The different values of CDM



(b) Segmentation of static regions ($\varphi(n,a)=0$)



(c) CCL ($\varphi(n,a)=1$) and static regions

(d) Segmentation of changing pixels

Fig. 3. Description of the three steps of the segmentation process for the video "Table". The figure a gives the different values of the CDM. The figures b,c,d describe the evolution of the process of the edges a with respectively $\varphi(n,a)=0$, $\varphi(n,a)=1$ and $\varphi(n,a)=2$. In these three last figures, black pixels are pixels that have not yet been classified, whereas white pixels correspond to region boundaries found at each step.

In section V, we propose the computation of an objective measure for temporal consistency. The measures obtained on real video sequences demonstrate a real improvement of temporal consistency. Moreover, the way we exploit the CDM decreases also the computational cost of the algorithm since the edges in static area are not reconsidered, and those linking the "no changing pixels" in changing area are simply processed by a CCL algorithm.

When successive images are not correlated (in the case of a scene cut for example), the set A_u contains most of the edges of the image which leads to a new spatial segmentation as shown in the example of a shot cut given in Fig.11. Our algorithm handles thus naturally the shot cuts and does not need to be combined with a shot cuts detection algorithm.

IV. Implementation considerations

In this section, we propose to describe optimisations that have been made to allow a real time treatment. The whole algorithm of video segmentation is summarised in Fig. 4.

Apart from the merging loop, all other functions access pixels data in a predictable way (for example from top to bottom left to right). The cache memory benefits from this regularity, since it exploits spatial and temporal locality of data, and consequently causes less cache misses. In the merging loop, the UNION-FIND data structure is unpredictable, and consequently causes an important data cache stalls. To reduce the data cache stalls cycles, we investigate some optimisations that are detailed in the following sections and we take benefit of the TriMedia processor to exploit the Data Level Parallelism (DLP) and Instruction Level Parallelism (ILP) of our algorithm.

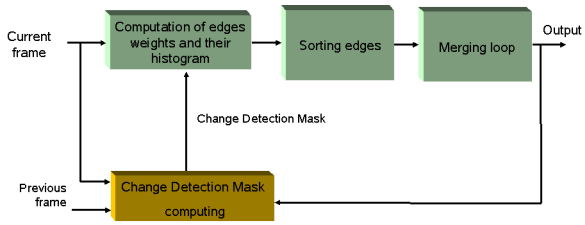


Fig. 4. The general diagram of video segmentation

A. Organisation of data

Our organisation of data should allow an efficient computation of both our merge criterion (equation 5) and our union and find operations. Let us recall that when using an union-find merging scheme each region of the image is encoded by a spanning tree whose vertices are the pixels of the region. These tree data structures are usually encoded by storing for each pixel the index of its parent within the spanning tree. The information about the region are associated to the root of the trees and both the roots and the region's information are updated during an union operation.

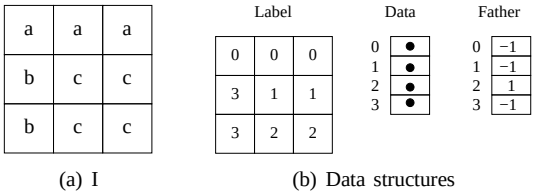
Since our merge criterion only uses the mean color $(\overline{y},\overline{u},\overline{v})$ and the cardinal $|S|$ of the regions, one simple organisation of our data would consist in associating each pixel p with the fields $(\overline{y},\overline{u},\overline{v},|S|,father)$, where $father$ denotes the father of p within the tree.

However grouping the region's data and the father's index would require to manipulate the whole vector $(\overline{y},\overline{u},\overline{v},|S|,father)$ within find operations. Since only the father field is required by the find operation such an organisation of the data would induce the storage of useless data within the cache memory.

We thus decided to store into two separate arrays the data required for the merge operations (namely the vector $(\overline{y},\overline{u},\overline{v},|S|)$ and the encoding of the trees. More precisely our organization of data is as follows:

1) One array $Data$ which stores for each created region its $(\overline{y},\overline{u},\overline{v},|S|)$ fields.
2) One array $Father$ which encodes our sequence of union operations.
3) One array $Label$ of size $|I[$ initialised to a special flag indicating that each pixel is initially its own father.

If a region is reduced to a single pixel p, $Label(p)$ is set to a special flag and the data of the region may be retrieve from the image I. We thus decide to create a new entry within the



(a) I (b) Data structures

Fig. 5. The data structures used to compute union-find operations and our merge criterion.

array $Data$ only if the associated region is composed of at least 2 pixels. More precisely, if a merge of two pixels p_1 and p_2 is decided by our merge criterion:

1) A new entry l is created within the array $Data$ and initialised according to $I(p_1)$ and $I(p_2)$.
2) $Label(p_1)$ and $Label(p_2)$ are set to l,
3) $Father(l)$ is set to a special flag indicating that l has yet no father.

Our data structure is further updated in the two following cases:

1) One pixel p is aggregated to an already created region labelled by l. In this case $Label(p)$ is set to l and $Data(l)$ is updated according to $I(p)$. The array $Father$ remains unchanged.
2) Two already created regions with respective labels l_1 and l_2 are merged. In this case, one of the label (say l_1) survives, $Data(l_1)$ is updated according to $Data(l_2)$ and $Father(l_2)$ is set to l_1.

Fig. 5(b) illustrates the state of our different data structures after the segmentation of Fig. 5(a). Two pixels in Fig. 5(a) are merged if they have a same label. In this example, we first considered horizontal edges between pixels and then vertical ones. Both horizontal and vertical edges have been considered using a scan line order. Note that the array Data is completely filled by the four regions created during the union operations. We only get three final regions as encoded by the array Father where all labels, except label 2 are their own father.

Since all regions encoded by the array $Data$ are composed of at least 2 pixels, the maximal number of entries within this array is equal to $\frac{|I|}{2}$. Moreover, the vertices of the trees encoded by the array $Father$ correspond to regions composed of at least 2 pixels. The maximal size of the array $Father$ is thus also equal to $\frac{|I|}{2}$. Note that this upper bound may be reached if we first decompose the image into regions made of 2 adjacent pixels and then order the merges in such a way that the tree encoding the union of all these elementary regions is linear.

Note that when using such an organisation of data all the required memory is allocated before union and find operations. We thus avoid the risk of a memory overflow.

B. TriMedia processor

We experimented this data organisation on the TriMedia processor [32]. The cache memory of this particular TriMedia is 128 $KByte$, 4 way associative, with block of 128 $Byte$. The replacement algorithm used is LRU.

In order to increase the computational efficiency, we propose to take benefit of the DLP (data level parallelism) provided by our algorithm (computation of edge's weight, frame difference, classification of pixels in CDM). This allows to increase the throughput (i.e. amount of pixels processed per unit time), by processing data in parallel when it is possible. The core of TriMedia is a VLIW architecture with 5 issues slots. Each slot has some functional unit, and each functional unit could process 4 bytes in parallel (SIMD mode). The ILP (Instruction Level Parallelism) is extracted by the compiler, while the DLP could be exploited through the use of custom operations, loop unrolling, and grafting. So we use these optimisations to exploit the DLP available in our algorithm.

V. Experimental Results

In this section we present experimental results of our algorithm run on TriMedia with many very known CIF video sequences.

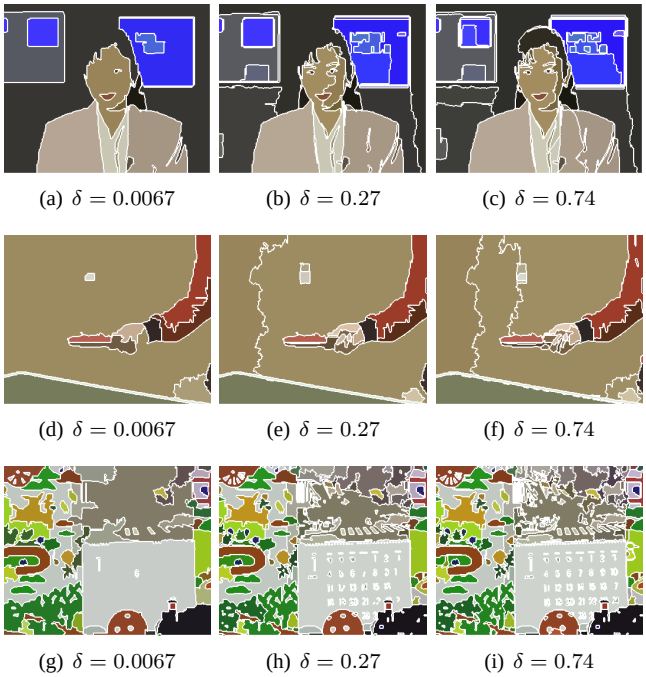
A. Spatial results

The probability δ tunes the coarseness of the segmentation. In Fig.6, we show the influence of this parameter on the level of details obtained. This parameter is highly correlated to the number of segmented regions. A value of this parameter around 0.74 provides a sufficient level of details for most of the video sequences we have considered. However, the chosen value and the associated level of details is highly dependent on the application. We can remark that this algorithm is able to segment very precisely small regions of interest such as the mouth or the eyes of "Akiyo". It can also segment the different numbers of the calendar in the sequence "Mobile". However, we can observe an over-segmentation of some textured regions such as the wall in the sequence "Table". This is mainly due to the fact that the assumption (1) is more adapted to the segmentation of flat regions. Our ongoing research is directed towards the design of a new merging criterion for the segmentation of textured regions.

As a comparison, we propose here some results obtained with two other well-known algorithms: the algorithm EGBIS [12] and the SRM (Statistical Region Merging) algorithm of Nock et al [4]. These two algorithms are based on region merging schemes with the same merging order than our method. The main difference between the three methods lies in the merging predicate. The results are displayed Fig.7. For each algorithm, we have tuned the parameters in order to reach a segmentation that allows a good subjective representation of the elements of the image (numbers of the calendar, eyes of the woman ...). We can see on these examples, that our real time algorithm gives comparable results than the two other algorithms. This last point has been confirmed by other experiments that are not reported here. Our real time implementation is thus achieved without detriment to the subjective quality of the results.

B. Spatio-temporal results

In the experiments, we take $tr_1 = 6$ and $tr_2 = 0.01$ (i.e. a region is a changing region when it contains at least 1% of

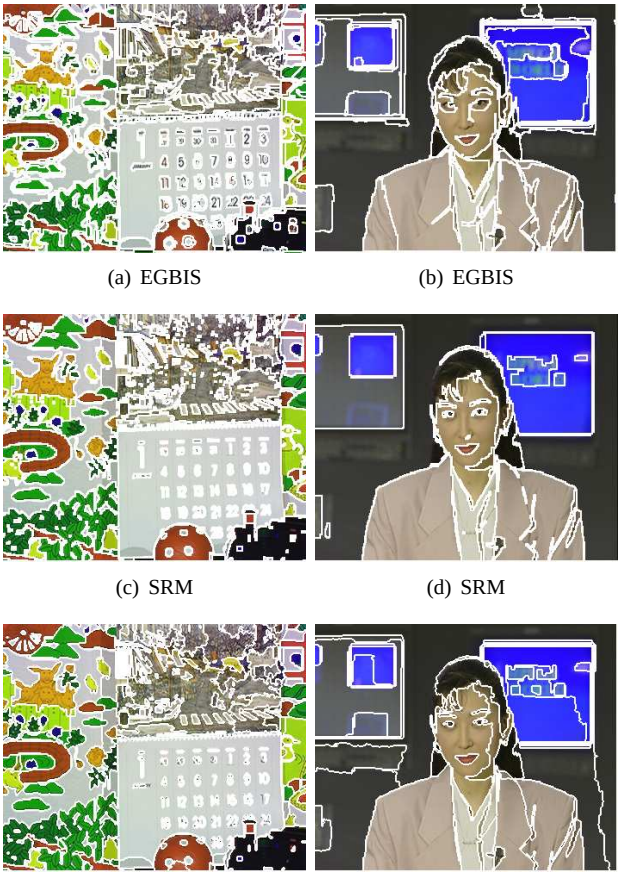


(a) $\delta = 0.0067$ (b) $\delta = 0.27$ (c) $\delta = 0.74$

(d) $\delta = 0.0067$ (e) $\delta = 0.27$ (f) $\delta = 0.74$

(g) $\delta = 0.0067$ (h) $\delta = 0.27$ (i) $\delta = 0.74$

Fig. 6. Segmentation of one frame of the video sequences "Akiyo", "Table", "Mobile" with $\delta = 0.0067$, $\delta = 0.27$, $\delta = 0.74$.



(a) EGBIS (b) EGBIS

(c) SRM (d) SRM

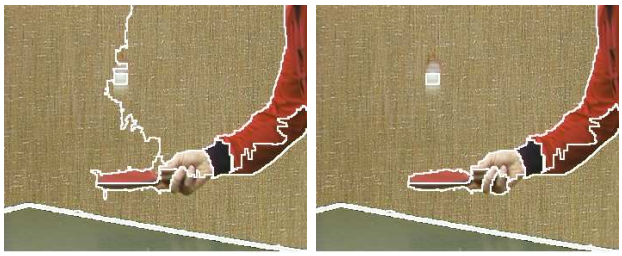
(e) Our algorithm (f) Our algorithm

Fig. 7. Comparison of our segmentation results with those obtained using the algorithms EGBIS [12] and SRM [4].

changing pixels). The value of these thresholds are the same for all the video sequences.

In order to see the influence of our temporal process, we show here an example of segmentation results with and without time consistency in Fig.8(c) and 8(b). We can see that the segmentation of the wall is the same for the two frames 1 and 9 of the video sequence "Table" when we use the time consistency improvement.

(a) Segmentation of frame 1



(b) Segmentation of frame 9 without time consistency (c) Segmentation of frame 9 with time consistency

Fig. 8. Comparison of the segmentation results obtained with and without time consistency on the video sequence "Table".

We then propose to display the segmentation results along the video sequence "Akiyo" in Fig.9 and the video sequence "Paris" in Fig.10. We can observe that the method gives satisfying and stable results for these sequences.

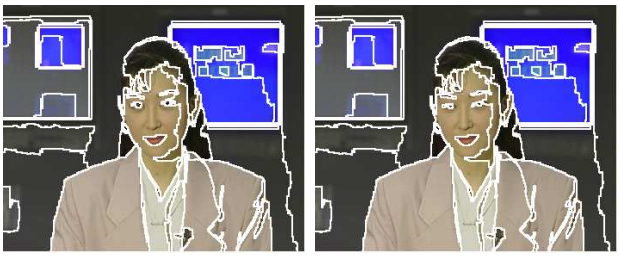
We have also tested the robustness of our method in the case of a shot cut. The video sequence "Football" is followed by the video "BBC Disc". Experimental results are given in Fig. 11. We can observe that the spatial segmentation of the first frame of the video "BBCDisc" is not influenced by the spatial segmentation of the previous frame that belongs to the video "Football". Indeed, in this case, most of edges will belong to the third category of edges ($\varphi(n, a) = 2$) where the predicate is re-computed.

C. Evaluation of time consistency

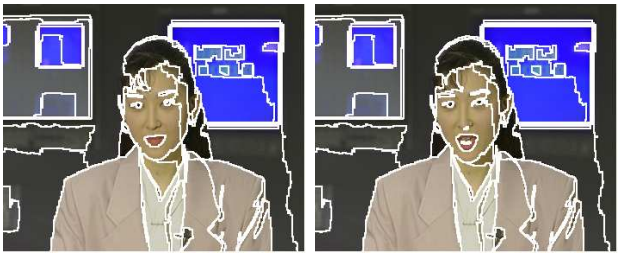
We use a classical measure to evaluate time consistency. Given the segmentation of the previous frame $SEG(n-1)$ and the segmentation of the current one $SEG(n)$, we find a correspondence between regions in $SEG(n-1)$ and $SEG(n)$. For each region $S_{i,n-1} \in SEG(n-1)$, we choose the region $S_{j,n} \in SEG(n)$ that produces the most overlapping area:

$$Overlap(i, n-1) = \max_j |S_{i,n-1} \cap S_{j,n}|.$$

We then sum the overlap's measures for all the regions in $SEG(n-1)$. The consistency measure is the percentage of this



(a) n=1 (b) n=5



(c) n=10 (d) n=15

(e) n=20 (f) n=25

Fig. 9. Results for the spatio-temporal segmentation of two video sequences "Akiyo"($\delta = 0.81$).

(a) n=1 (b) n=5

(c) n=10 (d) n=15

Fig. 10. Results for the spatio-temporal segmentation of two video sequences "Paris"($\delta = 0.81$).

(a) (b)

Fig. 11. Experimental results in the presence of a video scene cut. Figure a : Segmentation of the last frame of the video "Football". Figure b : Segmentation of the first frame of the first image of the video "BBCDisc".

number to the size of the image. The results for this measure are given in table I for the video sequences "Akiyo", "Table Tennis", "Paris" and "Mobile". When enforcing consistency through the CDM, time consistency is higher, and visually, segmentation is more stable from frame to frame and still fit very well regions boundaries as shown in Fig.9 and Fig.10. We can also see that the time consistency of the spatial segmentation algorithm SRM ([4]) is roughly equivalent to the time consistency of our spatial algorithm without computation of the CDM.

TABLE I

EXPERIMENTAL MEASURES OF TIME CONSISTENCY

Sequence	Akiyo 	Table 	Paris 	Mobile 
Time Consistency (SRM)	0.95	0.80	0.86	0.79
Time Consistency (Our approach without CDM)	0.88	0.73	0.89	0.84
Time Consistency (Our approach with CDM)	0.98	0.92	0.97	0.92

D. Evaluation of the computational cost

In this section, we propose to give the number of Mcycles the algorithm takes on TriMedia for different resolutions and different versions of our algorithm. We propose to compare the spatial computational cost with the one obtained using the Nock algorithm [4].

The computational cost has been evaluated as a function of the image size in Fig. 12. In this figure, the computational cost (in $Mcycles/frame$) has been computed for one image of the video "Akiyo" at different resolutions (QCIF, CIF, SD and two other resolutions). This computation has been performed with and without the optimisations described in section IV-B). First, the results given in Fig.12 show that the complexity is approximatively linear regarding the image size. Indeed, the spatial computational cost is principally induced by the UNION-FIND algorithm and the edges sorting. As explained in section II-C, the sorting step is performed in a linear time $0(|I|)$. As far as the UNION-FIND algorithm is concerned, the complexity is given by $0(\alpha(n_u, n_f)n_f)$ where n_u is the number of UNION operations and n_f is the number of FIND

operations ($n_u < n_f$). The function α is a very slowly growing function [25]. Since the number of FIND operations can be upper-bounded by $c|I|$ where c is a constant, the complexity at worst can be approximated by $0(\alpha(n_u, n_f)|I|)$ which gives an almost linear complexity. This assessment is confirmed by the experimental results given in Fig. 12.

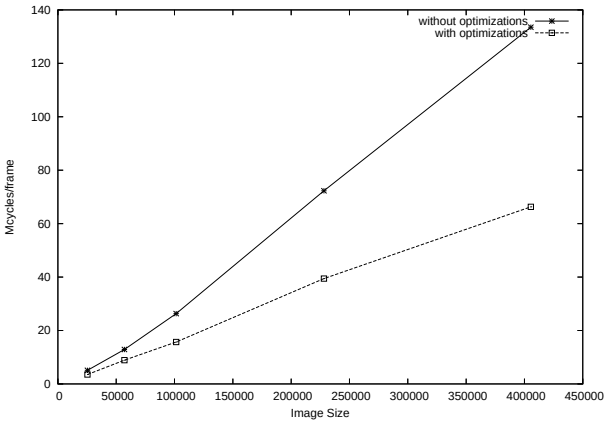


Fig. 12. Evaluation of the computational cost regarding with the image size (with one image of the video Akiyo, $\delta = 0.74$)

We then propose to compare the computational cost of our algorithm to the SRM algorithm [4]. The main difference between the two spatial algorithms lies in the computation of the predicate. The predicate of SRM leads to higher computational cost as demonstrated in the Table II. Our algorithm gives a lower computational cost even without optimisations. When including these improvements, the computational cost decreases. In table II, we also give the number of Mcycles the algorithm takes on TriMedia when enforcing the temporal consistency. The exploitation of the CDM reduces the computational cost. This reduction depends on the correlation between two successive frames.

With a $450\ MHz$ TriMedia, we are able to process more than 25 frames per second. We can then conclude that our algorithm is avalaible in real-time for QCIF or CIF sequences.

TABLE II
COMPUTATIONAL COST

Sequence	Akiyo	Table	Paris	Mobile
				
Mcycles/frame (SRM)	34.57	66.53	38.03	25.41
Mcycles/frame (without CDM, without optimisations)	26.33	32.21	28.18	24.73
Mcycles/frame (without CDM, with optimisations)	15.68	15.84	16.59	16.20
Mcycles/frame (with CDM)	9.87	11.37	11.02	10.06

VI. DISCUSSION

Designing usable algorithms for video processing requires low computational methods. Directed by this constraint, we propose here an efficient time consistent algorithm for video segmentation. Let us discuss the strengths and limitations of our algorithm regarding the three main points of this work :

- Spatial segmentation: We propose here an alternative statistical modelisation to the work of Nock et al [4]. This leads to a simpler predicate for merging that is more adapted to a real-time implementation and gives good results for the spatial segmentation. However, as in [4], such a statistical model is dedicated to the segmentation of flat regions and may produce an over-segmentation on textured area of an image.
- Temporal consistency: The proposed algorithm allows to obtain both stable segmentation results and a reduction of the computational cost. This method is based on the use of a CDM and of region information deduced from the first frame. Regions are not linked from one frame to another leading to a video segmentation algorithm that is robust to scene cut and occlusion. However, if never this algorithm has to be exploited for video object tracking, region matching will be useful. It can be obtained by comparing regions of two consecutive frames using statistical inequalities.
- Hardware implementation : Our algorithm runs in real-time for CIF sequences. For SD (Standard Definition) or HD (High Definition) sequences some further efforts are needed. In order to obtain a real-time implementation, we have directed our attention to the parallelisation by blocks of the spatial segmentation. However, we still investigate this part and notably the merging of the different spatial segmentation obtained for the different blocks. This last step remains delicate.

We finally want to outline that, such a real-time video segmentation algorithm would help many video algorithms by leading to a better comprehension of the image content. Among applications, we can think to time conversion, peaking (also named unsharp masking), video compression or deinterlacing. The region segmentation algorithm can be exploited directly using regions boundaries and region color properties or as a source of information on the image content (level of noise, complexity of the scene, main colors) which can be exploited to better design existing algorithms [33]. Our on going research is also directed to the design of such region-based algorithms for electronic devices (e.g. : Set Top Box).

ACKNOWLEDGEMENT

The authors would like to thank Patrick Meuwissen, O.P. Gangwal and Zbigniew Chamski for their constructive suggestions. We also would like to thank the reviewers for their very useful comments and suggestions.

REFERENCES

[1] G. Iannizzotto and L. Vita, “Fast and accurate edge-based segmentation with no contour smoothing in 2-d real images,” *IEEE Transactions on Image Processing*, vol. 9, Issue 7, pp. 1232 – 1237, 2000.

[2] Y. Haxhimusa, A. Ion, and W. G. Kropatsch, "Evaluating minimum spanning tree based segmentation algorithms," in *Proceeding of the 11th International Conference on Computer Analysis of Images and Patterns*, A. Gagalowicz and W. Philips, Eds., no. 3691. France: LNCS, September 2005, pp. 579–586.

[3] L. Brun, M. Mokhtari, and F. Meyer, "Hierarchical watersheds within the combinatorial pyramid framework," in *Proc. of DGCI 2005*, IAPR-TC18. LNCS, 2005, pp. 34–44.

[4] R. Nock and F. Nielsen, "Statistical region merging," *IEEE PAMI*, vol. 26, no. 11, pp. 1452–1458, November 2004.

[5] S. Lallich, F. Mulhenbach, and J.-M. Jolion, "A test to control a region growing process within a hierarchical graph," *Pattern Recognition*, vol. 36, no. 10, pp. 2201–2211, 2003.

[6] S. Pateux, "Spatial segmentation of color images according to MDL formalism," in *International conference on Color in Graphics and Image Processing*, Saint Étienne, France, october 2000, pp. 89–93.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, Issue 8, pp. 888 – 905, 2000.

[8] E. Sharon, A. Brandt, and R. Basri, "Fast multiscale image segmentation," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 70 – 77, 2000.

[9] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, Issue 6, pp. 583–598, 1991.

[10] M. Couprie, L. Najman, and G. Bertrand, "Quasi-linear algorithm for topological watershed," *Journal of Mathematical Imaging and Vision*, vol. 22, no. 2-3, pp. 231–249, 2005.

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

[12] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, Issue 2, pp. 167–181, 2004.

[13] Y. Deng and B. Manjunath, "Unsupervised segmentation of colour-texture regions in images and video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, Issue 8, pp. 800 – 810, 2001.

[14] C. Fiorio and R. Nock, "Sorted region merging to maximize test reliability," in *IEEE International Conference on Image Processing*, Vancouver, Canada, 2000.

[15] H.-Y. Wang and K.-K. Ma, "Automatic video object segmentation via 3D structure tensor," *International Conference on Image Processing, ICIP*, vol. 1, 14-17, pp. 153–156, 2003.

[16] D. DeMenthon, "Spatio-temporal segmentation of video by hierarchical mean shift analysis," in *Statistical Methods in Video Processing Workshop*, Copenhagen, Denmark, 2002.

[17] L. Duan, M. Xu, Q. Tian, and C. Xu, "Mean shift based video segment representation and applications to replay detection," in *ICASSP*, vol. 5, May 2004.

[18] F. Moscheni, S. Bhattacharjee, and M. Kunt, "Spatio-temporal segmentation based on region merging," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, Issue 9,, pp. 897 – 915, 1998.

[19] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, ISSUE 5, pp. 539 – 546, 1998.

[20] E. Patras, E. Hendriks, and R. Lagendijk, "Video segmentation by map labeling of watershed segments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 326–332, march 2001.

[21] C. McDiarmid, "Concentration," *Probabilistic Methods for Algorithmic Discrete Mathematics, Springer*, pp. 1–54, 1998.

[22] A. M. Tekalp, *Video Segmentation*. Handbook of Image and Video Processing, Elsiever, 2005.

[23] A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services - the European COST 211 framework," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 802–813, novembre 1998.

[24] A. Caplier, L. Bonnaud, and J. Chassery, "Robust fast extraction of video objects combining frame differences and adaptative reference image," in *International Conference on Image Processing*, Thessaloniki, Greece, octobre 2001, pp. 785–788.

[25] C. Fiorio and J. Gustedt, "Two linear time union-find strategies for image processing," *Theoretical Computer Science*, vol. 154, pp. 165–181, 1996.

[26] G. E. Healey, S. A. Shafer, and L. B. Wolff, Eds., *Color*. Jones and Bartlett, 1992, ch. A Physical Approach to Color Image Understanding, pp. 134–165.

[27] L. Wolf, X. Huang, I. Martin, and D. Metaxas, "Patch-based texture edges and segmentation," in *ECCV06*, 2006, pp. II: 481–493.

[28] L. Brun and M. Mokhtari, "Two high speed color quantization algorithms," in *Proceedings of CGIP'2000*, Cépaduès, Ed., Saint Etienne, October 2000, pp. 116–121.

[29] A. Desolneux, L. Moisan, and J. Morel, "Meaningful alignments," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 7–23, 2000.

[30] A. Mitiche and P. Bouthemy, "Computation and analysis of image motion: a synopsis of current problems and methods," *International Journal of Computer Vision*, vol. 19, pp. 29–55, 1996.

[31] K. Wu, E. Otoo, and A. Shoshani, "Optimizing connected component labeling algorithms," *Proceedings of SPIE*, vol. 5747, pp. 1965–1976, 2005.

[32] "pnx1500 databook," *available at http://www.tcshelp.com/public_files.html*.

[33] M. E. Hassani, M. Duranton, and S. Jehan-Besson, "Dynamic peaking," submitted patent NXP, 2007.