# TOWARDS REAL-TIME MPEG-4 SEGMENTATION: A FAST IMPLEMENTATION OF REGION-MERGING

Dirk Farin and Peter H.N. de With
farin@ti.uni-mannheim.de / dewith@ti.uni-mannheim.de

Dept. Circuitry and Simulation (Fac. Comp. Engineering), University Mannheim
B 6,26 D-68131 Mannheim, Germany

*We describe region merging as a flexible spatial segmentation algorithm and introduce a new heuristic to considerably speed up the computation. For further acceleration of the computation, a quadtree decomposition is carried out as a preprocessing step.*

## 1 INTRODUCTION

The emerging MPEG-4 standard is based on a new paradigm for image coding. Instead of coding the picture as a rectangular array of samples, the image is treated as a collection of several independent objects, having arbitrary shape (contours) and motion. These objects can be compressed and manipulated independently, leading to so-called object-oriented coding. This approach not only promises higher compression ratios than MPEG-2, but it also allows content-based access and interactive composition of video sequences at the system level.

When adopting this view, a principal problem at the encoder side is the automatic segmentation of natural video sequences into semantically meaningful video objects. This is a recent field of research where a number of techniques have been investigated. Current approaches can be classified into spatial and temporal image segmentation techniques.

In this paper we describe region merging as a spatial segmentation technique and present an efficient preprocessing step for improving the computation time. Furthermore, we give an additional heuristic that considerably speeds up the merging process.

## 2 REGION MERGING

With region merging, the objective is to group input image pixels to *regions* which are similar with respect to a predetermined criterion. The algorithm proceeds by sequentially merging the two most similar neighbouring regions. The merging process stops when no more regions are found with sufficient similarity, or the minimum number of regions is reached. This stop condition is needed to avoid that the algorithm ends with the whole image as one region.

Region merging can be viewed as a graph-node merging process where the *nodes* represent regions of pixels and the *edges* indicate neighbourship. We assign *edge weights* to the edges to represent the similarity between adjoining regions.

## 2.1 FORMAL DEFINITION OF THE ALGORITHM

Let $P = \{p_i\}$ be the set of pixels in the input image with corresponding luminance $f(p_i)$. Furthermore, let a neighbourship-relation $n(p_i, p_j) = \text{true}$ iff $p_i$ and $p_j$ are neighbours; however $n(p_i, p_i) = \text{false}$.

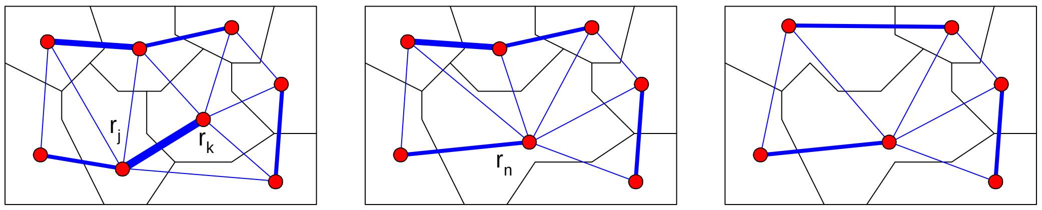
The input of the region-merging algorithm is a set of regions $R = \{r_i\}$ with $r_i \subset P$, $\bigcup r_i = P$ and $r_i \cap r_j = \emptyset$ for $i \neq j$. The way these regions are obtained may vary as follows. They can be the result of a preceding segmentation step such as watershed segmentation, they can be chosen arbitrarily (e.g. blocks of fixed size), or in the extreme case, each input pixel can be considered as a separate region.

The algorithm first builds a neighbourhood graph $G = (R, E)$ with edges $E = \{(r_i, r_j) | \exists_{p_k \in r_i, p_l \in r_j} \; n(p_k, p_l) = \text{true}\}$. Additionally, we define an edge weight $w$ on the edges $w\colon E \to \mathbb{R}$ which describes a measure of similarity of the regions adjacent to the edge (in our paper, smaller values of $w$ correspond to more similar regions). The definition of the edge weights (i.e. the merging criterion) is the crucial part of the algorithm, directly affecting the quality of the segmentation result.

Region merging is a greedy algorithm that follows the intuitive process to continuously merge the two most similar regions into a single region. Merging stops when the lower bound of regions $\#r_{min}$ is reached or the minimum edge weight exceeds a threshold $w_{max}$. The algorithm is outlined in Algorithm 1 and illustrated in Figure 1.

**Algorithm 1** Basic merging algorithm

```
1 while $|R| > \#r_{min}$ do
2       $e_{min} = (r_j, r_k) \leftarrow \operatorname{argmin}_{e \in E} w(e)$
3       if $w(e_{min}) > w_{max}$ then $STOP$ fi
4       Create new region $r_n \leftarrow r_j \cup r_k$
5       Insert new edges $E \leftarrow E \cup \big(E_{new} = \{(r_n, r_i) \mid (r_j, r_i) \in E \vee (r_k, r_i) \in E\}\big)$
6       Remove old edges $E \leftarrow E \cap \{R \setminus \{r_j\,;\, r_k\}\} \times \{R \setminus \{r_j\,;\, r_k\}\}$
7       Remove regions $r_j$ and $r_k$   $R \leftarrow R \setminus \{r_j\,;\, r_k\}$
8       foreach $e \in E_{new}$ do Update edge weight $w(e)$ end
9 end
```



(a) initial regions (b) merging step 1 (c) merging step 2

Figure 1: First two steps of a region-merging process. Thicker edges represent more similar regions.

## 2.2 MERGING CRITERIA

A simple similarity measure that nevertheless obtains good results is to calculate the difference of the mean luminance of two adjacent regions:

$$w\big(e=(r_j,r_k)\big)=\Big|\big(\frac{1}{|r_j|}\sum_{p\in r_j}f(p)\big)-\big(\frac{1}{|r_k|}\sum_{p\in r_k}f(p)\big)\Big|.$$

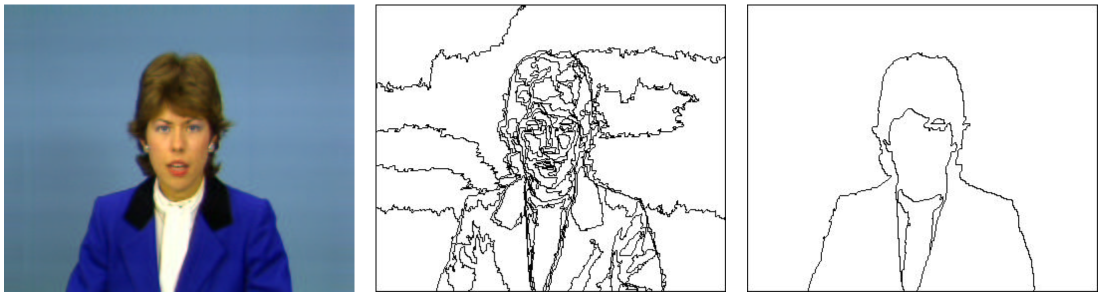
An advantage of this measure in applications aiming at real-time execution is its simplicity, because updating the edge weights after a merging step can be easily accomplished. If we store for each region $r_i$ the sum of the luminance of all pixels $l(r_i)=\sum_{p\in r_i}f(p)$ and the region size $s(r_i)=|r_i|$, the edge weights calculate as:

$$w\big(e=(r_j,r_k)\big)=\Big|\frac{l_j}{s_j}-\frac{l_k}{s_k}\Big|.$$

The nice property of this representation is, that after merging the two regions $r_j$ and $r_k$ into the combined region $r_n$, we only have to set $l(r_n)=l(r_j)+l(r_k)$ and $s(r_n)=s(r_j)+s(r_k)$ and update the edge weights accordingly. This measure can be extended to all three color components (e.g. Y,Cb,Cr) and combined into a weighted sum. We used this measure with weighting (3/5 ; 1/5 ; 1/5) in the generation of the results in Figure 2.

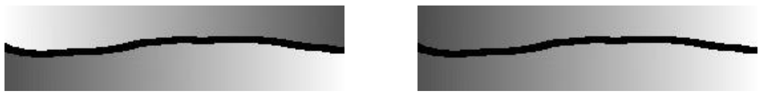
Unfortunately, the mean-luminance difference measure also has several drawbacks; consider for example the image in Figure 3a. The regions on both sides of the boundary have exactly the same mean luminance, but clearly do not match as good as the two regions in Figure 3b. Nevertheless, the similarity measure evaluates to exactly the same value.

A better criterion is to examine the common boundary between two regions and define a similarity measure favouring an equal luminance distribution along the common boundary. To formalize this measure, let the sequence



(a) input image (claire) (b) 120 regions remaining (c) 5 regions remaining

Figure 2: Region merging applied to test image 'claire'.



(a) gradients with opposite directions (b) gradients with same direction

Figure 3: Gradients along a region boundary.

of $N$ pairs of opposing pixels along the boundary between region $r_j$ and $r_k$ be $b_i(e) = (p_a, p_b)$, $1 \leq i \leq N$ (with $e = (r_j, r_k)$ and $p_a \in r_j$, $p_b \in r_k$). Now we can define a similarity measure e.g. as follows:

$$w(e) = \sum_{1 \leq i \leq N} |f(p_a) - f(p_b)|, \text{ with } p_a, p_b \text{ given by } b_i(e).$$

Other similarity measures are possible as well: e.g. region luminance variance, similar texture, a well-shaped boundary of the hypothetically merged regions, or similar motion of the regions. To exploit the advantages of different measures, it may be attractive to switch to another merging criterion at some point in the merging process. For example, one could start with a pure spatial merging criterion, like mean-luminance difference, and switch to a similar-motion criterion when the regions have grown to a reasonable size.

The motivation for switching the merging criterion is that it can improve the segmentation results. The mean-luminance difference measure works well in small local areas, but it fails on larger regions, because the region description is too simplistic. On the other hand, motion as a merging criterion works well for larger regions, where motion can be determined with high accuracy. However, a motion criterion fails on small regions, because the features contained in the region may not be sufficient for an accurate motion estimation.

## 2.3 REDUCING THE COMPUTATIONAL COMPLEXITY

Let $n = |R|$ be the number of input regions. Usually segmentation stops when only a small number of regions is remaining. Thus, the total number of merging steps performed is $\mathcal{O}(n)$. In a simple implementation where edges are stored in an unsorted array, step 2 in Algorithm 1 takes $\mathcal{O}(n)$ time. In the worst case, shown in Figure 4a, steps 5,6 and 8 each require $\mathcal{O}(n)$ time when the comb-shaped region is merged. The overall complexity is therefore $\mathcal{O}(n^2)$.

As the number of merging steps is fixed, the only way to decrease computation time is to reduce the number of edges that have to be updated in each step. If the regions would grow in a more balanced way (as shown in Figure 4b), the number of outgoing edges of each region becomes more or less constant. Consequently, steps 5,6 and 8 would run in $\mathcal{O}(1)$ time.

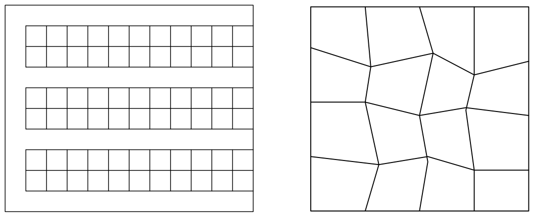

(a) worst-case pattern (b) heuristically favoured pattern

Figure 4: Worst case pattern for region merging and pattern that we want to favour with the heuristic.

We have achieved a more balanced growing of regions by adding a secondary criterion to the edge weights. For this purpose, we use a new weight $w'$ which favours the merging of small regions in undecided cases $(w(e_i) = w(e_j))$. In total, we define

$$w'\big(e_i = (r_a, r_b)\big) < w'\big(e_j = (r_c, r_d)\big) \iff \\ w(e_i) < w(e_j) \vee \big(w(e_i) = w(e_j) \wedge |r_a| + |r_b| < |r_c| + |r_d|\big).$$

Since $w'$ differs from $w$ only by a secondary weight, this approach does not degrade the segmentation quality in any way.

However, this proposal alone is not sufficient for a significant speedup, as finding the minimum edge weight (step 2) still requires $\mathcal{O}(n)$ time. By storing the edges in a heap data-structure, sorted by edge weights, we can reduce the time spent for finding the edge to $\mathcal{O}(1)$, but on the other hand enlarge the time spent for updating the edge weights (step 8) again to $\mathcal{O}(\log n)$, since the heap has to be kept sorted. However, this results in an overall computation time of only $\mathcal{O}(n \log n)$[1]. As the edge weights mostly will not change much in a merging step,

[1]Note that by using a Fibonacci-heap implementation instead, we could reduce overall computation time to $\mathcal{O}(n)$. But this implementation is so complex that it is unlikely to be faster in practice.

the number of levels that an edge has to be moved in the heap up- or downwards will be small. Therefore, the actual time spent in step 8 can even be considered as constant, leading to a linear-time algorithm for ordinary picture material[2].

## 2.4 EXPERIMENTAL RESULTS

Table 1 shows the speed improvement obtained when using the new heuristic criterion-extension on region sizes. Merging started at the original pixel level and the merging criterion was a weighted sum of the difference of the mean values of the luminance and chrominance components in the regions.

| image sequence | dimension | standard-heap | heuristic-heap |
|---|---:|---:|---:|
| kettle | 352x288 | 76.19s | 2.71s |
| coastguard | 352x288 | 6.99s | 2.74s |
| foreman | 352x288 | 5.20s | 2.80s |
| claire | 352x288 | 4.70s | 2.62s |

Table 1: Computation time of region merging (starting with single pixel regions). Number of input regions: 101376, input edges: 202112. Calculation has been performed on an Intel Pentium-III 550 MHz.

Note that images that generally required a long computation time, give the fastest results with the new criterion. This phenomenon occurs because many edges with equal weights appear during the computation of these images. This results in large regions and thus slow computation, using the standard implementation. In contrast to this, our heuristic applies more often in these cases and the region sizes are growing more balanced, which results in reduced computation time.

## 2.5 QUADTREE-DECOMPOSITION PREPROCESSING AND RESULTS

Another way to enhance performance is to start with larger regions rather than taking the original input pixels. If this preprocessing step is considerably faster than the region-merging process, an overall speed improvement can be achieved. One way to get larger input regions is to perform a *quadtree decomposition* of the image, as shown in Algorithm 2. In practice, this algorithm can be implemented most easily as a recursive procedure. Some degree of freedom lies in the definition of the function *uniform*: $R \rightarrow \text{bool}$ in step 2. We have used a metric that coincides with the mean-luminance difference measure from Section 2.2. The definition equals: $r_i$ is uniform $\iff \big(\max_{p \in r_i} f(p) - \min_{p \in r_i} f(p)\big) < T$ (with $T$

[2]Note that steps 5 and 6 can be combined by mostly reusing the data structures of the edges to be removed for the new edges. Therefore, these steps do not require $\mathcal{O}(\log n)$ time.

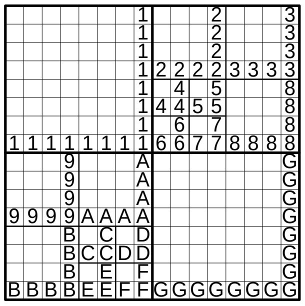
being a fixed threshold). It can be seen that this definition represents a simple calculation.

**Algorithm 2** Quadtree decomposition

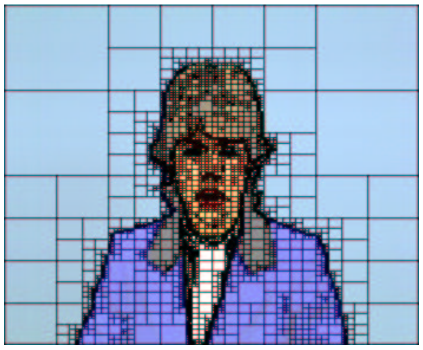
```
1 $R = \{r_1\} = \{P\}$
2 while $\exists r_i$ with $uniform(r_i) = false$ do
3         Subdivide $r_i$ into 4 subregions $r_a, r_b, r_c, r_d$
4         $R \leftarrow (R \setminus \{r_i\}) \cup \{r_a\,;\, r_b\,;\, r_c\,;\, r_d\}$
5 end
```

To initialize the region-merging algorithm, we need to build the neighbourhood graph of our quadtree decomposition. To quickly find all necessary edges, we label all pixels to the bottom and the right of a region with a region ID whenever a uniform region is found (see Figure 5a). To determine the required edges, we scan for each region the pixels one to the left and one to the top of its boundary. Whenever the region ID found changes, we generate an edge between the two regions. Take region 'G' for example: scanning the top row, we first find region ID '6' which is a new ID. Thus, an edge $(6\,;\,G)$ is generated. The next pixel is labeled '6' too, so that we do not create a new edge. We advance to '7', which is different from the last ID, so a new edge $(7\,;\,G)$ is generated. Finally, an edge $(8\,;\,G)$ is constructed and the same algorithm applies to the left side, resulting in the edges $(A\,;\,G)$, $(D\,;\,G)$ and $(F\,;\,G)$.

Table 2 shows the computation times when the quadtree decomposition is added as a preprocessing step. As compared to Table 1, a significant reduction



(a) Labeling of pixels at the edge of regions for efficient neighbourhood-graph construction.



(b) Result of decomposition step applied to claire sequence.

Figure 5: Quadtree decomposition preprocessing step.

of the computation time is obtained with comparable results. The accuracy and computation time are both depending on the threshold $T$ in the function *uniform*. We have to balance between a small value of $T$, giving exact segmentation results, and a larger $T$, lowering the overall computation time.

| image sequence | avg. $\|R\|$ | avg. $\|E\|$ | q.-tree+standard | q.-tree+heuristic |
|---|---|---|---|---|
| kettle | 13057 | 28850 | 1.04s | 0.39s |
| coastguard | 38552 | 83529 | 1.31s | 1.16s |
| foreman | 24853 | 54695 | 1.32s | 0.73s |
| claire | 5688 | 12832 | 0.26s | 0.19s |

Table 2: Computation time of region merging preceded by the decomposition stage. Also listed are the average number of regions and edges after the quadtree decomposition step. Calculation has been performed on an Intel Pentium-III 550 MHz.

## 3 CONCLUSIONS

We described region merging as a spatial segmentation algorithm and illustrated its flexibility by using different merging criteria. An improvement has been achieved by including region sizes into the merging criteria. This heuristic measure favours balanced region growing, leading to a significant reduction of the computation time when it is combined with a heap data-structure implementation.

A further reduction of computation time has been obtained by adding quadtree decomposition as a preprocessing step. We have presented an efficient algorithm to transform the quadtree data structure into a neighbourhood graph, thereby initializing the input data structure for the subsequent region-merging step. The resulting computation time is mostly below one second per CIF-resolution frame and makes region merging more attractive for real-time applications.

## REFERENCES

[1] F. Moscheni, S. Bhattacharjee, "Robust Region Merging for Spatio-Temporal Segmentation", Proceedings of International Conference on Image Processing ICIP'96, Vol. 1, 1996.

[2] F. Moscheni, F. Dufaux, "Region Merging Based on Robust Statistical Testing", SPIE Proc. Visual Communications and Image Processing '96, Orlando, Florida, USA, March 1996.

[3] T.Kurita, "An Efficient Clustering Algorithm for Region Merging," IEICE Trans. of Information and Systems, Vol. E78-D, No.12, 1995.