A fast and robust image segmentation scheme

Thomas Kunlin Lionel Lacassagne Alain Mérigot
Institut d'électronique fondamentale
Université Paris sud
91405 Orsay Cedex, FRANCE
kunlin@ief.u-psud.fr lacas@ief.u-psud.fr am@ief.u-psud.fr

Abstract

This paper presents a modified version of the classical split and merge algorithm. Instead of performing a regular decomposition of the image, it relies on a split at an optimal position that does a good interregion separation. The implementation of the algorithm uses an initial image preprocessing to speed-up computation. Experimental results show that number of regions generated by the split phase is largely reduced and that the distorsion of segmented image is smaller, while the execution time is slightly increased.

1. Introduction

The *Split and Merge* segmentation method [2] is a well known method to extract homogeneous region from an image. It consists to recursively split the image in four regular quadrants until every segment is homogeous enough, then to merge the obtained regions as long as the merging respects some homogeneity criterion. Despite several advantages (in terms of simplicity and computational efficiency), this method exhibits several drawbacks, mostly based on the rigid space subdivision that it uses.

This paper presents a new segmentation scheme based on this method. The main difference is that, instead of performing a regular decomposition of the (sub)image, at every step, the split will be performed at a variable position, in order to optimize a given criterion. It happens that some implementation tricks, relying on an initial prefix sum computation preprocessing step, allow this method to be almost as efficient in terms of computational efficiency as the classical split and merge, while giving better segmentation results and being much more flexible.

Section 2 presents the basic issues on the method. Section 3 describes how this optimal splitting scheme can be efficiently implemented. Section 4 presents the results obtained with this method.

2. Split and merge segmentation

2.1. The Horowitz-Pavlidis split and merge algorithm

The classical split and merge algorithm (HP-SM), presented in 1976 by Horowitz and Pavlidis [2] relies on the following algorithm:

Algorithm 1 Horowitz Pavlidis Split and Merge

```
1  Split(Region $R$)
2      if $R$ is not homogeneous enough then
3          split $R$ in four equally sized regions $R_1$,$R_2$,$R_3$, $R_4$
4          for $r$ in $R_1$,$R_2$,$R_3$, $R_4$ do
5              Split(r)
6          end do
7      end if
8  Merge(RegionSet $R_s$)
9      repeat
10         get two regions $r$ and $r'$ in $R_s$ that are neighbor and
           similar
11         merge $r$ and $r'$
12     until no regions can merge
```

This algorithm presents several advantages :

1. The split phase is very fast. The homogeneity predicate (line 2), that generally relies on a comparison of region R variance with a predefined threshold can be very rapidly computed in an upward pyramid-like pass.
2. The data structure obtained after the split phase, a quadtree[4], leads to efficient and flexible implementations[3].

HP-SM also has a set of drawbacks :

1. It can only determine regions with horizontal or vertical frontiers. Hence, to deal with the general situation, a large number of small regions are generated in the split phase.
2. Regions boundaries are at fixed positions. Accordingly, there is no invariance of the resulting data struc-

ture with simple geometrical transformations as translation or even scaling.

3. The methods relies on a square image, whose sides must be a power of two.

2.2. Optimal splitting

To partially cope with the limitations of HP split and merge, we propose a couple of modifications of the split phase of the algorithm. First, instead of performing a regular decomposition of an heterogenous region in equally sized regions (line 3 of algorithm 1), we will perform the decomposition at a position that optimizes a certain split criterion. Second, as this kind of decision is easier to take in one dimension, and can be more precisely determined, the split phase will decompose a region in *two* parts, instead of four. Here is the optimal split algorithm:

Algorithm 2 Optimal split

```
1  Optimal-Split(Region $R$)
2      if $R$ is not homogeneous enough then
3          split $R$ either vertically or horizontally in two regions
           $R_1$ and $R_2$ at a position $P$ that optimizes a given split
           criterion $SC$
4          for $r$ in $R_1$ and $R_2$ do
5              Optimal-Split(r)
6          end do
7      end if
```

The split criterion SC (line 3) must perform a good interregion separation. For instance, one could try to find a point such as the difference of the average grey level between regions R_1 and R_2 is maximized. Several methods can also be used to determine if the split must be vertical or horizontal.

This algorithm withdraws most of the disadvantages of HP-SM, except the horizontal or vertical region shape limitation. The number of regions after the split phase is much smaller than in the HP-SM algorithm. As the optimal point depends on image data, and not on geometrical characteristics, the split image exhibits a good stability for simple image transformation (translation, scaling). Last, any image size can be used. This technique can also be used on specific rectangular parts of the image, if we want to specifically perform a segmentation on some subimages.

We show in the next section, that despite the necessity to optimize a a split criterion, it is possible to realize a fast implementation of this segmentation scheme.

3. Implementation issues

Let us consider image $I(x, y) : 0 \leq x < N_x, 0 \leq y < N_y$, and a rectangular region R described by the coordinates of its top-left and bottom-right points $(x_0, y_0)(x_1, y_1)$. We want to split horizontally R at a position y_m, in two regions $R_A = (x_0, y_0)(x_1, y_m)$ and $R_B = (x_0, y_m + 1)(x_1, y_1)$. To perform that two problems must be solved :

- We need to determine region R homogeneity, to decide if it requires splitting (line 2). The most frequently used criterion is the variance of I over region R.
- We need to determine the optimal split position (line 3). Point y_m must be selected in order to provide a good separation between different segments of the image. For instance, it can be determined to maximize the difference of the average value of I over regions R_0 and R_1.

In either case, we can see that the problem is mostly to compute efficiently global sums over rectangular regions. We describe in next section how it can be very efficiently realized thanks to so-called *prefix-sum*, also known as *scans*[1].

3.1. Scan operations and global sums

Scan operations are defined as follows: given an associative operator $\diamond$ and a vector $v(x) : 0 \leq x < x_M$, the $\diamond$-scan of v returns a vector $w = \diamond\!-\mathrm{scan}(v)$, such as $w(x) = v(0) \diamond v(1) \diamond ... \diamond v(x-1) \diamond v(x)$.

As we are concerned with computing sums, this can expressed as $+\!-\mathrm{scan}(v) = w$ with $w(x) = \sum_{i \leq x} v(i)$.

On a two dimentional array, such as image I, these operations can be performed either vertically on the columns of I or horizontally along the lines. If we perform sucessively this operation in both directions on image I, we get a new array σ_I, such as $\sigma_I(x, y) = \sum_{0 \leq i \leq x} \sum_{0 \leq j \leq y} I(i, j)$. We can deduce the sum over any rectangular region $R = (x_0, y_0)(x_1, y_1)$ from σ_I.

$$\sum_{(x,y) \in R} I(x, y) = \sigma_I(x_1, y_1) + \sigma_I(x_0 - 1, y_0 - 1) - \sigma_I(x_1, y_0 - 1) - \sigma_I(x_0 - 1, y_1) \quad (1)$$

We assume that $\sigma_I(-1, y) = \sigma_I(x, -1) = 0$. Hence, a global sum over any region of image I can be easily and rapidly computed provided we have previously determined the prefix sums σ_I. For variance computation, we will also need to have the sum of the square of pixel in I. The same preprocessing leads to the sum image σ_{I^2}.

3.2. Computing the homogeneity predicate

As a direct application, let us see how to compute the homogeneity predicate for a region $R = (x_0, y_0)(x_1, y_1)$, in order to determine if it requires splitting. We assume that the considered predicate is based on the variance V of the image on the region.

$$V(R) = \frac{1}{N} \sum_{(x,y) \in R} I^2(x, y) - \frac{1}{N^2}\Big(\sum_{(x,y) \in R} I(x, y)\Big)^2 \quad (2)$$

where $N = Card(R)$. This can be simply rewritten with the sum arrays σ_I and σ_{I^2} of image I and its square as :

$$\begin{aligned} V(R) \;=\; & \frac{1}{N}(\sigma_{I^2}(x_1,y_1) + \sigma_{I^2}(x_0-1,y_0-1) \\ & -\sigma_{I^2}(x_1,y_0-1) - \sigma_{I^2}(x_0-1,y_1)) \\ & -\frac{1}{N^2}(\sigma_I(x_1,y_1) + \sigma_I(x_0-1,y_0-1) \\ & -\sigma_I(x_1,y_0-1) - \sigma_I(x_0-1,y_1))^2 \quad (3) \end{aligned}$$

Hence, the variance computation on *any* rectangular region can be done with a constant small number of operations on the sum arrays σ_I and σ_{I^2}.

3.3. Determining the optimal split position

To determine the optimal split position of a region, we use the following criterion (we assume that the region will be split horizontally). Given a region $R = (x_0, y_0)(x_1, y_1)$, find the position y_m that maximizes the absolute value of the difference of the average of I over adjacent k-pixel high stripes. More formally, find y_m that maximizes

$$D(y) = |\sum_{(i,j)\in S_B(x)} I(i,j) - \sum_{(i,j)\in S_T(x)} I(i,j)| \qquad (4)$$

where $S_T(x)$ and $S_B(x)$ are the stripes respectively ending and starting at position x, eg $S_T(x) = (x_0, y-k+1)(x_1, y)$ and $S_B(x) = (x_0, y+1)(x_1, y+k)$. The difference can thus easily be expressed with σ_I.

$$\begin{aligned} D(y) = \sigma_I(x_1, y+k) + \sigma_I(x_1, y-k) - \sigma_I(x_0-1, y+k) \\ - \sigma_I(x_0-1, y-k) + 2\times(\sigma_I(x_0-1, y) - \sigma_I(x_1, y)) \end{aligned} \qquad (5)$$

Typycal values of k are 20% of the height of R, bounded to 1–7. The calculation of $D(y)$ along the range $[y_0 - k + 1 : y_1 - k]$ can be vectorized for further gain on the computation time.

4. Segmentation results

We present here some segmentation results and compare the original Horowitz-Pavlidis split algorithm with several versions of optimal split. In every case, we consider as the main parameter the threshold v_t, the square of which is compared to the variance to decide if a region needs additional splitting. Values in the range 10–30 give good results. Above, the resulting image is too degraded, below, the number of regions is very large. To limit the number of regions, regions below a certain size (5 pixels) are not split.

4.1. Considered algorithms

4.1.1 Original Horowitz-Pavlidis algorithms HP4 and HP2

The first considered algorithm is the original Horowitz-Pavlidis one. We test homogeneity of a region and, if it is required, we split it in four equal subregions. This algorithm will be denoted as HP4.

A small variation on this method consists to realize a binary split. If the region is square, we split it in two equal horizontal rectangles, if it is rectangular, the split will be vertical. Such a variation will be denoted HP2.

4.1.2 Optimal split OS1D, OS2D and OS2Dx

These algorithms will be compared to several flavors of optimal split. These versions differ in the way the split point and direction is determined. For all these versions, we will use the difference of the grey value of adjacent stripes as the split criterion.

OS1D is a version close to HP2, in the sense that the shape of the region determines the split direction. If the region is square, it is split horizontally, otherwise, it is split across the largest dimension. The split point is selected at an optimal position.

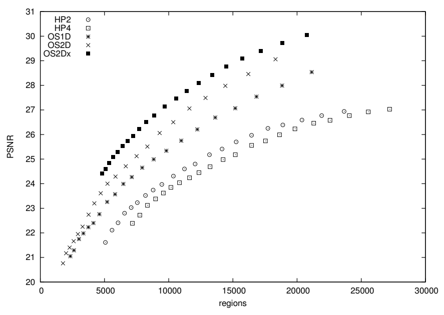
OS2D is a slight amelioration of the previous method that consists to perform a double search for the optimal split point in either vertical and horizontal directions. The direction with the largest interstripes average difference is selected.

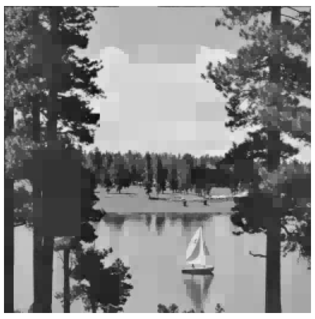
One drawback of the described methods is that they fail to segment small regions drowned in larger ones. OS2Dx is a modification of OS2D to improve this behavior and consists in performing a *detailled examination* on a region whenever it is considered as homogeneous. We conceptually tile the region with small $k \times k$ squares and check for the homogeneity of *every* square. Whenever a single square is not homogenous, we decide to split the region. Typical values of k in the range 5–11 give good results. Of course, the variance computation on every square can be rapidly performed thanks to the scanned sum image.

4.2 Test image segmentation results

To compare the results, we use two features. The number of generated regions n, that should be as low as possible, and the PSNR that measures the likeness of I to $I^\star$, respectively the original image and its approximation by replacing each region by the average value of its pixels.

Figure 1-a plots the PSNR vs n of each method for the 512×512 image *lake*. We see the similar behavior of HP2 and HP4 and a significant improvement in the OS methods. Searching in both directions (OS2D) improves the results with respect to OS1D. OS2Dx generates a slightly larger number of regions, but the PSNR gain is quite significant. Images 1-b and 1-c shows a comparison of the OS2Dx and HP4 method on *lake* : for v_t=16, OS2Dx produces 33% less regions and the visual quality of the segmented image is obviously better than HP4. For less contrasted images, the visual quality gain of the OS methods compared to HP4


(a) lake: PSNR/regions


(b) HP4: n=18634,PSNR=25.99

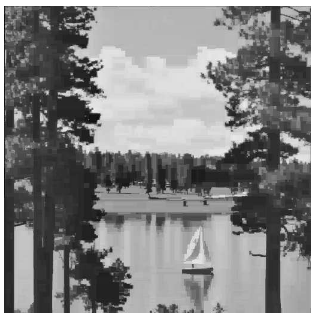

(c) OS2Dx: n=12342,PSNR=28.01

Figure 1. Segmentation results on the 'lake' picture (512×512).

is smaller, but the gap on the number of regions produced remains important.

4.3 Computation time

We use the CPP (number of cycles per pixel) to measure the computation times of the different methods. All the results were obtained on a 2.4 GHz Pentium 4, using gcc 3.3.

4.3.1 Comparison of HP4 and optimized HP

The fast execution time of the HP algorithm is achieved thanks to a pyramid data structure also obtained from a preprocessing stage. The results indicate that the computation times of such an implementation of HP and of algorithm HP4 - that uses σ_I and σ_{I^2} prefix sums - are very near : the preprocessing time is 35 CPP for both methods, and the split phase is 5% slower in HP4 than in the optimized implementation of HP.

4.3.2 Comparison of the considered algorithms

Figure 2 represents the computation time of the algorithms presented in 4.1. It gathers the results of every method for several 512×512 pictures.

We can see that the OS methods lie in a factor 2-3 of HP4. These computation times stay quite compatible with real time constraints : for 512×512 images on a 2.4 GHz CPU, 200 CPP permits a framerate of 45. Right now, the sigma transform has been vectorized both for Intel SSE2 and PowerPC Altivec SIMD extensions. The speed-up is, respectively 1.7 and 2.4. The optimal split position computation represents 50% of the overall CPU time and should be accelerated by a factor 4. According to Amdhal's Law the global speed-up of the OS methods would be 1.6, a fully SIMD optimized version is currently under developpement and should confirm this statement.

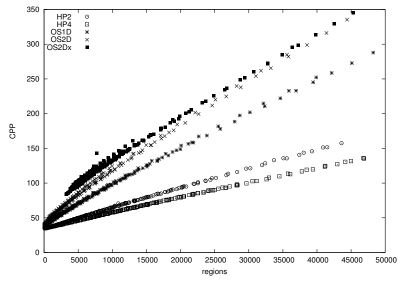


Figure 2. Complexity of the algorithms.

5. Conclusion

We have presented here a new algorithm based on the standard split algorithm of Horowitz-Pavlidis, that relies on the determination of an optimal split position. While preserving the most interesting features of the original method (speed, simplicity of the obtained data structure), it presents significative improvements. The number of regions is largely reduced and the generated image is closer to the original one. More, this method can be applied to any rectangular shape. It can for instance be applied to a subregion of the image where a specific object is searched. Another interesting aspect is the good invariance by translation and scaling that this method presents, and that could allow to use it for structural pattern recognition. There is a small time overhead compared to the original method, but the method is still compatible with real-time constraints. Its speed and the possibility to adapt the split position and precision to specific tasks, make it a good candidate for time-critical image analysis applications.

References

[1] G. E. Blelloch. *Vector Models for Data-Parallel Computing*. The MIT Press, Cambridge,Massachusetts, 1990.

[2] S. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23:368–388, 1976.

[3] L.Balmelli, J.Kovačević, and M. Vetterli. Quadtree for embedded surface visualization: Constraints and efficient data structures. *Proceedings of IEEE Int. Conf. Image Processing (ICIP)*, 2:487–491, October 1999.

[4] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Readin, MA, 1990. ISBN 0-201-50255-0.