SIAM J. Imaging Sciences
Vol. 6, No. 3, pp. 1274–1317

A Contrario Selection of Optimal Partitions for Image Segmentation[*]

Juan Cardelino[†], Vicent Caselles[‡], Marcelo Bertalmío[‡], and Gregory Randall[§]

Abstract. We present a novel segmentation algorithm based on a hierarchical representation of images. The main contribution of this work is to explore the capabilities of the a contrario reasoning when applied to the segmentation problem and to overcome the limitations of current algorithms within that framework. This exploratory approach has three main goals. Our first goal is to extend the search space of greedy merging algorithms to the set of all partitions spanned by a certain hierarchy and to cast the segmentation as a selection problem within this space. In this way we increase the number of tested partitions, and thus we potentially improve the segmentation results. In addition, this space is considerably smaller than the space of all possible partitions, and thus we still keep the complexity controlled. Our second goal aims to improve the locality of region merging algorithms, which usually merge pairs of neighboring regions. In this work, we overcome this limitation by introducing a validation procedure for complete partitions rather than for pairs of regions. The third goal is to perform an exhaustive experimental evaluation methodology in order to provide reproducible results. Finally, we embed the selection process on a statistical a contrario framework which allows us to have only one free parameter related to the desired scale.

Key words. a contrario, quantitative evaluation, image segmentation, hierarchical segmentation, region merging

AMS subject classification. 68U10

DOI. 10.1137/11086029X

1. Introduction.

Image segmentation is one of the oldest and most challenging problems in image processing. In this work we will focus on the problem of low-level segmentation, which means that our goal will be to find *interesting* partitions of the image, but without a high-level interpretation of the objects present in the scene. This interpretation is application dependent and should be performed by a different kind of algorithm, possibly using this low-level segmentation as input.

[*]Received by the editors December 27, 2011; accepted for publication (in revised form) May 3, 2013; published electronically July 11, 2013.
http://www.siam.org/journals/siims/6-3/86029.html

[†]Instituto de Ingeniería Eléctrica, Universidad de la República, Montevideo 11300, Uruguay, and Department de Tecnologies de la Informació, Universitat Pompeu Fabra, Barcelona 08003, Spain (juanc@fing.edu.uy). This author's research was partially supported by the MICINN project, reference MTM2009-08171, and by GRC reference 2009 SGR 773, funded by the Generalitat de Catalunya. It was also partially supported by the ALFA-CVFA project and the Tecnocom postgraduate scholarship.

[‡]Department de Tecnologies de la Informació, Universitat Pompeu Fabra, Barcelona 08003, Spain (vicent.caselles@upf.edu, marcelo.bertalmio@upf.edu). The second author's research was partially supported by the MICINN project, reference MTM2009-08171, and by GRC reference 2009 SGR 773, funded by the Generalitat de Catalunya. It was also partially supported by the "ICREA Acadèmia" prize for excellence in research, funded by the Generalitat de Catalunya, and by the ERC Advanced grant INPAINTING (grant agreement 319899). The third author's research was supported by the European Research Council, Starting grant reference 306337, and Spanish grants AACC, reference TIN2011-15954-E, and Plan Nacional, reference TIN2012-38112.

[§]Instituto de Ingeniería Eléctrica, Universidad de la República, Montevideo 11300, Uruguay (randall@fing.edu.uy).

Although it is a simpler problem, even for a human observer, it is hard determining a unique meaningful partition of an image, and it is even harder to find consensus between different observers. In general, the biggest problem is determining the scale of the segmentation. The choice of this scale is application dependent, and thus the low-level segmentation task should avoid it. In [20] Guigues, Coquerez, and Le Men remarked the idea that a low-level segmentation tool should remain scale uncommitted and the output of such an algorithm should be a multiscale description of the image. Higher-level information, as one single partition of the image, could be obtained afterwards by using additional criteria or manually inspecting that multiscale representation.

A usual approach is to find a multiscale representation of the image rather than an unique partition. This representation usually takes the form of a hierarchy of nested partitions which are usually constructed by means of region merging/region growing or split and merge algorithms. In the case of region merging algorithms, the hierarchies are constructed in a a bottom-up fashion, starting with small regions and iteratively joining similar ones until a stopping criterion is met.

The literature on image segmentation is vast, and a complete review of all the families of available algorithms is beyond the scope of this work. For this reason, we will focus on the approaches based on merging algorithms, multiscale representation of images, and a contrario models.

Region merging algorithms. According to Morel and Solimini [29], the first merging algorithms presented in the literature aimed to consecutively merge adjacent regions until a stopping criterion was met, thus yielding a single partition. All these algorithms share three basic ingredients: a region model, which tells us how to describe regions; a merging criterion, which tells us whether two regions must be merged or not; and a merging order, which tells us at each step which couple of regions should be merged first. Many of the existing approaches have at least one of the following problems: the use of region information alone (no boundaries are taken into account), a very simple region model, a very simple merging order, or a considerable number of manually tuned parameters. Most of the recent work in this area has been directed to improving the merging criterion and the region model, but little effort has been carried out towards the merging order and the reduction of the number of parameters.

Regarding the merging order, Calderero and Marques [7] presented an improvement over classic merging algorithms which uses a nonparametric region model and a merging order depending on the scale. However, this work still has some free parameters and only one feature (color) is taken into account in the merging criterion.

Regarding the parameter selection, Burrus, Bernard, and Jolion [5] introduced an a contrario model to remove the free parameters and a criterion combining different region descriptors. However, they require a training process which is offline but very slow, and which has to be done for each image size and for each possible parameter combination of the initialization algorithm. In addition, segmentation results are heavily dependent on those initialization parameters.

Multiscale image segmentation and representation. Instead of working at a single scale, a more robust approach is to use the intermediate results of the merging process to build up a hierarchy of partitions, which is often represented as a tree structure. Once the hierarchy is built, the tree is pruned according to some scale parameter [23]. However, most algorithms ignore the fact that this hierarchy spans a subset of all the possible partitions of the image, and that the segmentation problem could be cast as the selection of an optimal partition on this subset. For example, in the approach of Koepfler, Lopez, and Morel [23] the hierarchy is thresholded at a fixed scale, an operation we will call a *horizontal cut*. These kinds of cuts explore only a smaller subset of the possible partitions spanned by the hierarchy.

On the other hand, when filtering the hierarchy to find the optimal partition, the selected regions could exist at different scales on different parts of the image. This optimal filtering has been extensively explored in the field of mathematical morphology under many forms. In one of the first works using these ideas, Najman and Schmitt [32] extended the notion of watershed to produce a saliency map, in which each contour of the watershed was labeled with the *importance* or *saliency* of the contour. This structure encoded all possible output partitions, and a particular segmentation could be obtained by thresholding this structure. In [36] an approach based on constrained connective segmentation was used to produce a hierarchy of partitions of the image by scanning all possible values of the range parameters. In Guigues, Cocquerez, and Le Men [20], this selection was called a *nonhorizontal cut* of the hierarchy and computed by minimizing a Mumford–Shah energy over the hierarchy. In a recent work [28] the authors presented a detailed review of the theory and applications of hierarchical segmentation in the framework of mathematical morphology. In this context the optimal filtering is carried out by using the flooding operator, and the approaches of Najman and Guigues are explained as applications of this framework. In [31] a unified theory is proposed, in which every hierarchical segmentation can be represented as an ultrametric watershed and vice versa. In particular, the saliency maps introduced in [32] are shown to be a particular case of an ultrametric watershed. Finally, in a previous work [11], we used a similar approach to overcome some of the usual problems of merging algorithms. In addition, our algorithm used both boundary and region information within a statistical framework which allowed us to select the optimal partition with only one free parameter, which acted as a scale parameter.

The work of Guigues, Cocquerez, and Le Men [20] is very similar in spirit to the present work. For this reason we will review some of its main features here and we will compare each feature of our algorithm with that work both from the theoretical and experimental points of view. The first contribution of Guigues is defining and characterizing the so-called scale-sets representation. This representation is the region based analogue of the classical scale space used in the context of boundary detection, which was first introduced by Marr [24] and later formalized by Witkin [38] and Koenderink [22]. This representation holds two important properties: causality and geometrical accuracy. These two conditions are equivalent to the *strong causality principle* introduced by Morel and Solimini [29]. The second contribution is characterizing how to build such a representation by means of an energy minimization procedure. Finally, they prove that under certain conditions, the global minimum of these energies can be efficiently computed by means of a dynamic programming algorithm. In particular, it is important to remark that Guigues improves the locality of region merging algorithm by providing a scale climbing algorithm that allows them to compute the global optimum of a certain type of energy.

Objective evaluation. Although segmentation is one of the most studied areas in image processing, there are not too many works that rely on a precise evaluation methodology or that show quantitative results. Another point to be mentioned is the reproducibility of published algorithms, since there are very few binaries or source codes available, making it difficult to determine their real capabilities and limitations.

In this sense, the work of Arbeláez et al. [4] is in the same spirit as the present work. They propose a hierarchical approach constructed on top of a contour detector, which uses spectral clustering techniques to integrate different features for the segmentation task. In addition, they present a comprehensive quantitative evaluation of their experimental results.

Contributions. The main contribution of this work is exploring the capabilities of the a contrario approach for image segmentation. In this framework, only ideas taken from classical region merging algorithms have been proposed, which inherit the main problems of such approaches. In this sense, we further improve the model from [11] by addressing two of its main problems. First, we remove the locality introduced by validating only local merging decisions, computing instead the meaningfulness of an entire partition of the image. Second, we improve the behavior of the free parameter by introducing a theoretical estimation of the number of tests. Some of these ideas were already presented in the morphology field and in the work of Guigues. One of the original contributions of this work is to apply these ideas in the a contrario framework. Finally, we present an exhaustive quantitative evaluation of our results by testing our algorithm with publicly available databases and comparing the results against human segmented images using several performance metrics. We have also created a web page with detailed evaluation results and an online demo to validate our algorithm. Let us mention that the source code can be downloaded, allowing the community to verify our claims.[1]

Outline of the paper. The rest of the paper is organized as follows. In section 2 we review some background concepts related with our approach. In section 3 we review our previous definitions of meaningful regions. An improved a contrario model, called meaningful partitions, is introduced in section 4. In section 5 we build the proposed algorithm based on that model. Section 6 shows experimental results of the proposed approach. In section 7 we discuss the implementation of the algorithm. And finally, in section 8 we summarize our conclusions and discuss future work.

2. Background.

2.1. Hierarchies of partitions.

A hierarchy of partitions $\mathcal{H}$ is a multiscale representation of many possible segmentations of an image. For our purpose, it is a tree structure where each node represents a region, and the edges represent inclusion between regions (see Figure 1). This tree can be constructed in two ways: top-down or bottom-up. In the first case, the algorithm starts with the coarsest possible partition, which is iteratively split until a convergence criterion is met. In the second case, in which this work is contained, region merging algorithms are used to construct the hierarchy. They usually start with an initial set of regions, often called seeds, which conform the finest possible partition. Then two or more adjacent regions

[1] http://iie.fing.edu.uy/rs/wiki/ImageSegmentationAlgorithms

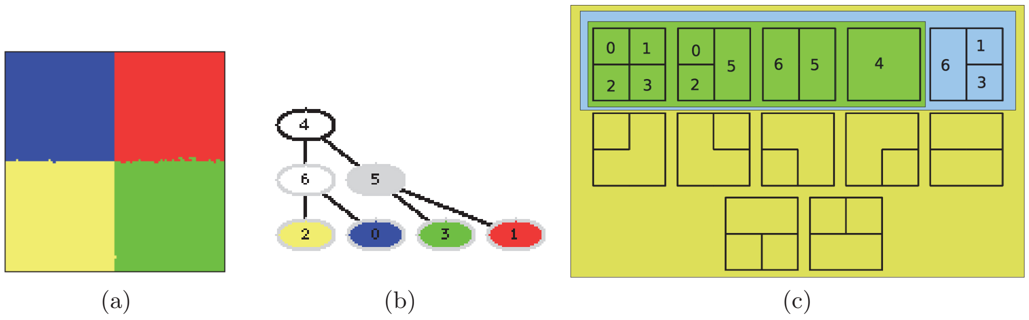

Figure 1. (a) *Sample image composed of four regions of random noise with different means.* (b) *Corresponding Mumford–Shah hierarchy, constructed from an initial partition composed of the four original regions (labeled* 0, 1, 2, 3*).* (c) *Different choices for the partition search space. The yellow set represents the set of all possible partitions that could be generated with the initial four regions. The blue set represents all possible partitions spanned by the tree. The green set represents the partitions considered by the region merging algorithm used to construct the hierarchy. Each region is labeled with the corresponding node number in the hierarchy. Note that there are possible partitions that are not spanned by the tree.*

are iteratively merged, and the resulting regions on the graph are represented as a new node which is the parent of the merged regions. One of the most popular ways to merge regions is to minimize the well-known Mumford–Shah (MS) functional [30]. The binary tree resulting from this merging procedure is often called a binary partition tree [34] or dendrogram in the clustering literature.

In this work we will use an initial hierarchy $\mathcal{H}$ constructed by means of a greedy iterative region merging algorithm, which minimizes the piecewise constant MS functional. However, our results are easily extensible to other kinds of hierarchies, like trees of shapes (also called component trees) [13].

2.2. A contrario framework. A contrario models were introduced by Desolneux, Moisan, and Morel in [16], within the framework of the computational Gestalt theory (CGT). Given a set of events the aim is to find the *meaningful* ones, according to the so-called Helmholtz principle, which states that "we naturally perceive whatever could not happen by chance." To apply this principle, CGT algorithms rely on a *background* or noise model and then define the interesting events as those extremely improbable under this model. To measure how exceptional an event is, they define the *number of false alarms (NFA)*, which is an upper bound on the expected number of occurrences of a certain event under the background model. If this expectation is low, it means that the considered event is meaningful and could not arise by chance. Given an event $\mathcal{O}$, its NFA is computed as $NFA(\mathcal{O}) = N_{tests}.P(\mathcal{O})$, where N_{tests} is the number of possible events in the set and $P(\mathcal{O})$ is the probability of occurrence of event $\mathcal{O}$. Defined in this way, it can be proven that if $NFA(\mathcal{O}) < \epsilon$, then the expected number of detections of event $\mathcal{O}$ in an image of noise generated by the background model is lower than ϵ. This reasoning is analogous to classical hypothesis testing, where one assumes a null hypothesis (images are generated by the background model) and given an observation one computes the probability of occurrence of the event under these assumptions. The null hypothesis is then rejected if the computed probability is low enough (given a threshold). In this context, the NFA plays the role of the type I errors, i.e., rejecting the null hypothesis when the event was actually generated by it. In the a contrario jargon, this means to detect meaningful objects in a noise image, which is a false alarm (or a false positive). For a more detailed explanation of the a contrario approach please refer to [16].

3. Meaningful regions and mergings.

The algorithm presented in this work is the natural evolution of the algorithm presented in [11], which we will denote as GREEDY. For this reason, we will start by reviewing the basic concepts introduced in that work, and we will discuss its main problems. The understanding of these concepts is crucial to understanding the present work, and thus it is important to review them here. Please note that the material shown in this section contains an excerpt of the aforementioned work, but with a more in-depth review of some concepts and newly added sections.

The main idea of the GREEDY algorithm is to combine region and boundary information to select the optimal partition from the set of all partitions spanned by the given hierarchy. This selection is embedded into an a contrario model which states that a region is meaningful if its gray level is homogeneous *enough* and its boundary is contrasted *enough*.

3.1. Region term.

The region term of the model will define a *meaningful region* as a region which is well adjusted to a certain model. In this work we use the simplified MS and we will say that a region is meaningful when its *error* is small, in a way similar to [21]. To explain this, let us review the data term in the MS model for a single region:

$$E_R = \sum_{x \in R} (I(x) - \mu_R)^2, \tag{1}$$

where μ_R is the mean gray value of region R. This can be seen as the L_2 error when we approximate each pixel of the region by μ_R. We can also define the pixelwise error as $e_R(x) = (I(x) - \mu_R)^2$, so with this notation the error becomes $E_R = \sum_{x \in R} e_R(x)$.

If we consider that E_R is a random variable generated by the background model and we define $\hat{E}_R$ as the observed region error, we can define the number of false alarms of a region as

$$NFA_r(R) = N_r.P(E_R < \hat{E}_R), \tag{2}$$

where N_r is the number of possible regions to consider. In practice, the number N_r can be estimated only for simple problems, so a common practice is to replace the number of possible regions by the number of *tested* regions. Defined in this way, this NFA measures the goodness of fit of the region pixels to the model given by the MS mean μ_R. If the probability P is very low, it means that the error is extraordinarily small and could not arise by chance. Thus, the NFA is small and the region will be marked as *meaningful*.

In order to complete the definition of (2) we need to compute the probability $P(.)$ of the error in each region. We do this in two steps: First, we estimate the probability density function $p(e)$ of the pixelwise error $e(x)$. After that, we compute the probability that region R has error less than or equal to $\hat{E}_R$.

Note that the error $\hat{e}_R(x)$ depends on μ_R, which in turn depends on the region the pixel belongs to. Before the segmentation starts, we do not know which region the pixel will be assigned to, so we cannot compute this quantity. To overcome this, we can compute the error with respect to all possible regions the pixel x could possibly be assigned to. In this way, we are not computing the error with respect to a single partition, but to all the possible partitions spanned by $\mathcal{H}$. In Figure 2 we show the pseudocode of the pixelwise error computation, in order to illustrate the estimation of the probability density function of the error in detail.

```
1. Let $\Omega \in R^2$ be the image domain. Let $R(H)$ be the set of all possible regions
   belonging to the hierarchy $H$.
2. For each pixel $x \in \Omega$ find the set of regions $X \subset R(H)$ such that $R \in$
   $X$ iff $x \in R$.
3. The maximum possible error $e(x)$ will occur in the case that $I(x) = 255$
   and $\mu_R = 0$ or vice versa. The minimum possible error will occur when
   $I(x) = \mu_R$. Thus $x(x) \in [0, e_{max}] = [0, 255^2]$.
4. Create an empty histogram $h_e$ of the error $e(x)$ in the range $[0, e_{max}]$ with
   $n$ bins.
5. For each $R \in X$
     • Compute the error of pixel $x$ with respect to $X$. That is, $e_R(x) =$
       $(I(x) - \mu_R)^2$.
     • Add $e_R(x)$ to the corresponding bin of $h_e$.
6. Normalize $h_e$ such that $\Sigma h_e(t)dt = 1$.
```

Figure 2. *Estimation of the probability density function of the pixelwise error over a hierarchy.*

Computed in this way, h_e could be regarded as an estimation of probability density function of an error of magnitude ϵ given the hierarchy H, $h_e(\epsilon) = p(e_R(x) = \epsilon|H)$. We could argue that this model is a contrario in the following sense. As we do not choose any segmentation, the histogram of the error is computed from every possible region each pixel could belong to. Note that, computed in this way, the probability density function of the error is independent of any particular choice of a partition but depends on the initial hierarchy $\mathcal{H}$ chosen.

Finally, to compute $P(E_R < \hat{E}_R)$ we make the assumption of independence between image intensities at the different locations of the n pixels of the region. Thus, looking at (2), the random variable E_R is a sum of n independent and identically distributed (i.i.d.) random variables e_R. Then, the probability can be approximated (for large values of n) by a normally distributed random variable, using the central limit theorem (CLT). In practice, with $n > 20$ the Gaussian approximation of the CLT is very accurate.

3.2. Meaningful regions vs. meaningful mergings. From our definition of meaningful region, it can be seen that the NFA associated with each region depends on the given initial partition. Given a certain tree, it is a common practice to remove the leaves with small area, in order to reduce the computational burden of the algorithm. Usually, leaves have small errors, so removing one of them will decrease the probability of observing a small error. If we remove a big number of small leaves, we will make the small errors less probable, thus making all nodes more meaningful. This is not a desirable behavior, because we want the result of our algorithm to be independent of the preprocessing performed.

This problem arises because our definition of meaningful region is absolute, which makes it strongly dependent on the histogram of the error. One way to overcome this problem is to compare pairs of regions instead of validating each of them independently. For this reason, we introduce a similar definition of meaningfulness but applied to mergings. We will say that a certain merging is meaningful if it *improves* the previous representation, that is, if the meaningfulness of the merged region is greater than the meaningfulness of considering the regions as separate entities.

To compute the meaningfulness of a merging we must compute two quantities: the NFA of the union of two regions, and the NFA of their separate existence. To compute the first quantity, we can apply the definition of the previous section. However, we need to adjust the number of tests, because now we are not testing all possible regions, but only those created as a result of a union. Let R_1 and R_2 be the regions to be merged, and let $R_u = R_1 \cup R_2$. Thus, the NFA of the union is

$$NFA_r(R_1 \cup R_2) = N_u.P(E_{R_u} < \hat{E}_{R_u}), \tag{3}$$

where N_u is the number of possible unions in the tree, which is exactly $\frac{N}{2}$, where N is the number of nodes of the tree. Here, as we modeled the union as a single region, the error $\hat{E}_{R_u}$ is computed using the mean μ_u of the union.

Now we need to consider the existence of two separate and independent regions R_1 and R_2. In this case, we have a different model for each region, given by the means μ_1 and μ_2, and the corresponding errors $\hat{E}_{R_1}$ and $\hat{E}_{R_2}$. As the MS error is additive, we can consider that the total error of approximating both regions by their means is $\hat{E}_{R_1;R_2} = \hat{E}_{R_1} + \hat{E}_{R_2}$. Thus we can define the NFA of two separate regions as

$$NFA_r(R_1; R_2) = N_c.P(E_{R_1;R_2} < \hat{E}_{R_1;R_2}), \tag{4}$$

where N_c is the number of possible couples (R_1, R_2) which can also be approximated by $\frac{N}{2}$. As the involved quantities are probabilities which could take very small values, it is usual to take the logarithm of the NFA, which we will call the LNFA. Thus, we will say that a merging is meaningful if

$$\mathcal{S}_r = LNFA_r(R_1 \cup R_2) - LNFA_r(R_1; R_2) < 0. \tag{5}$$

This condition is very similar to the Patrick–Fisher distance used for region merging in classic approaches. For instance, in section 2.2 of [41], a parametric form of this test for the Gaussian case is used to test whether two regions should be merged or not. This approach relies on deciding whether both samples come from the same underlying distribution, and if true (up to a confidence parameter), they should be merged. The main difference resides here in the relative nature of our term, which does not compare both regions but decides whether the union is a better explanation than the separate regions.

3.3. Boundary term. Regarding region boundaries, we propose merging two regions when the boundary between them is not contrasted enough. In addition, we want to obtain the regularizing effect of the boundary term in the MS functional, which favors short curves [30]. For this reason, we use a definition similar to that introduced by Desolneux, Moisan, and

Morel [15] and Cao, Musé, and Sur [9], in the sense that we say a curve is meaningful if it has an *extraordinarily* high contrast. However, we also penalize long curves by using the accumulated contrast along the curve instead of the minimum as in Cao's model. Thus, we define *meaningful regions* as those having a short and contrasted boundary. To measure the length of the curves we use the geodesic curve length

$$\mathcal{L}(\Gamma) = \int_\Gamma l(x(s))ds, \tag{6}$$

where $l(x) = g\left(|\nabla I(x)|\right)$ is a pixelwise contrast detection function. Here $g(x)$ yields small (near 0) values in well-contrasted pixels and values near 1 in the low contrasted ones. In this case we learn the background model from the image to be segmented, so we use the histogram of the gradient as proposed in [9]. From the image, we can obtain the empirical distribution of the gradient norm, or the distribution of $l(x)$ which is a function of $|\nabla I|$. However, we need to compute the distribution of the sum over all the pixels of the curve, so our new random variable will be $L = \sum_{x\in\Gamma} l(x)$. As we did in section 3.1 we can compute the probability by means of the CLT from the distribution of l, and define the NFA of a curve Γ as

$$NFA_b(\Gamma) = N_{curves}.P(L < \hat{L}), \tag{7}$$

where N_{curves} is the number of possible curves in the image, which is again approximated by the number of curves tested. In our case, it is the number of curve segments separating two neighboring regions in $\mathcal{H}$.

3.4. Combination of terms.

Using the two previous definitions of meaningful regions, we developed an unified approach which is able to take into account both definitions at the same time. For that purpose, we propose computing the following quantity:

$$NFA_j(R) = N_r.P\left(E_R < \hat{E}_R; L_{\partial R} < \hat{L}_{\partial R}\right), \tag{8}$$

where N_r is the number of tested regions and P is the joint probability of the region having a small error and a contrasted boundary at the same time. A way to make this model computable is to make the (strong) assumption of independence between boundary and region terms. This allows us to factorize P and write the new NFA as

$$NFA_j(R) = N_r.P(E_R < \hat{E}_R).P(L_{\partial R} < \hat{L}_{\partial R}). \tag{9}$$

Taking this into account, we say that a merging is meaningful using region and boundary information if

$$\mathcal{S}_j = LNFA_j(R_1 \cup R_2) - LNFA_j(R_1; R_2) < 0. \tag{10}$$

Definition (10) allows us to construct a parameterless algorithm; however, in practice we verified that our algorithm tends to oversegment images. The explanation of this phenomenon relies on our estimation of the number of tests. This estimation is very hard to compute analytically, so we used a very rough estimate ($N_u \approx N_c$). To overcome this problem, we consider an alternate formulation based on the following observation. Remembering that

$$P(R_1 \cup R_2) = P(E_{R_u} < \hat{E}_{R_u}).P(L_{\partial R_u} < \hat{L}_{\partial R_u}), \\ P(R_1; R_2) = P(E_{R_1;R_2} < \hat{E}_{R_1;R_2}).P(L_{\partial R_1;R_2} < \hat{L}_{\partial R_1;R_2}) \tag{11}$$

we can expand (10) to explicitly show the number of tests, obtaining

$$\mathcal{S}_j = \log N_u + \log P(R_1 \cup R_2) - \log N_c - \log P(R_1; R_2). \tag{12}$$

Thus, if we do not want to estimate both numbers of tests, we can merge them into a single variable called α. So, our definition of meaningful merging becomes

$$\mathcal{S}_j = \log P(R_1 \cup R_2) - \log P(R_1; R_2) < \alpha(R_1; R_2), \tag{13}$$

where $\alpha(R_1; R_2) = \log(\frac{N_c}{N_u})$. For the sake of simplicity, we assume that α is a constant value for every couple of regions (R_1, R_2). In [11] this parameter was set manually, but it could be estimated as in [5].

3.5. The algorithm. To explain the GREEDY algorithm, we need to introduce some notation. We define the height of a node n as the length of the longest branch starting on a leaf and ending in node n. We also define $N(h)$ as the set of nodes of height h. Starting from an initial partition, composed either of all pixels or of a set of given regions, we iteratively merge pairs of regions, as shown in Figure 3.

```
1. construct the initial partition $\mathcal{P}$
2. build the hierarchy $\mathcal{H}$
3. for each couple $C = (R_1, R_2)$ compute its LNFA
4. for each possible height $h$
   (a) while $N(h) \neq \emptyset$
   (b) select the most meaningful couple $C = (R_1, R_2)$
   (c) merge the couple $C$
   (d) remove $C$ from $N(h)$
```

Figure 3. *Outline of the GREEDY algorithm.*

3.6. Initial hierarchy. The presented algorithm does not need a particular choice of the hierarchy $\mathcal{H}$ to be applied, as long as it can be encoded as a tree structure. However, the results are dependent on the initial $\mathcal{H}$ chosen. As we mentioned in the introduction, in this work we will use a greedy region merging algorithm which obtains a local minimum of the piecewise constant MS functional. The implementation used is inspired by [23]. The first step of the algorithm is to compute a region adjacency graph (RAG) of the pixels of the image; let $\mathcal{G}$ be this graph. At the same time, we initialize the tree structure $\mathcal{T}$, where each node of the RAG is a leaf. To speed up the computation, we create a heap with the initial edges of the RAG, where the cost associated with each edge is the minimum value of λ in which the node will be removed. Then we will iteratively merge adjacent regions, picking in each step the one that will decrease the MS energy the most, taken from the top of the heap. Each time a pair of regions is merged, the corresponding edge is removed from the graph, and a new node is created in the tree. This node will be linked as the parent of the two merged ones and have as an attribute the λ value at which it has been created. This value is called *scale of appearance* in Guigues, Cocquerez, and Le Men [20]. The algorithm stops when only one region is left and the RAG is reduced to a trivial form; this node will be the root of the tree.

3.7. Complexity.

The computational cost of the algorithm can roughly be divided in two parts: computing the tree and selecting the meaningful regions on it. A usual way to construct the tree is to use all pixels as the initial partition. So, for an n-pixel image, the tree will have $2n-1$ nodes. The computational cost of the second part is proportional to the number of nodes of the tree, and thus we can reduce the computational cost by pruning the tree.

In practice we rarely segment regions of small area, and thus we remove the lower nodes of the tree corresponding to the smaller regions. To explain this, let us recall the simplified MS functional: given a partition $\mathcal{P}$ of the image, the energy $F(\mathcal{P}) = E_R + \lambda\mathcal{L}_{\partial R}$ balances the data adjustment term E_R with the regularity term $\mathcal{L}_{\partial R}$. The parameter λ controls the relative importance of each term. The pruning of the tree is performed with a fixed (and small enough) value of the scale parameter λ, which avoids important regions being lost.

We start by estimating the complexity of the construction of the initial tree. The first step is to construct the RAG, which has a complexity of $\Theta(n)$. Then we build a priority queue, using a binary heap as the underlying data structure, which will also cost $\Theta(n)$. If we merge only one region at a time, we will have $n-1$ steps, and at each step we have to remove the top of the heap and reorder it, which will take $\Theta(\log n)$ operations. In addition, we have to update the λ values of all the edges incident to the merged regions. Assuming an average degree of $k \ll n$ neighbors per node, and taking into account that changing the value of an edge implies reordering, the complexity of this part will be $\Theta(k \log n) = \Theta(\log n)$. So the total complexity of the construction of $\mathcal{H}$ is $\Theta(n \log n)$. Assume then a pruning that keeps only N nodes of the tree. Usually, $N \ll n$ and the reduction is more than linear, a rough but reasonable estimate being $N \approx n^\gamma$, with $\gamma \leq 0.5$. Regarding the complexity of the region selection part, we need to visit all pixels to compute the MS terms on the initial partition, which costs $\Theta(N)$. Then we need to visit the nodes once to compute the error and the probabilities. Then, at each step of the algorithm, we need to find the minimum LNFA node, which costs $\Theta(\log N)$. Thus the complexity of this part is $\Theta(N)$. Given our assumption that $N \ll n$, the overall complexity of the algorithm is dominated by $\Theta(n \log n)$.

3.8. Implementation.

For example, in the *Church* image (Figure 4) the total number of pixels is 38400, but using $\lambda = 50$ we have to process only 2297 regions, which results in a 94% reduction of the computational cost. In spite of this pruning, the resulting image still retains enough level of detail and no important objects are lost. Note that this preprocessing step is optional and that the final result of the algorithm is independent of it; it is performed just to speed up the algorithm.

4. Meaningful partitions.

One of the main problems of the previous approach is its locality: as we are validating only one merging at a time in a bottom-up fashion, all further

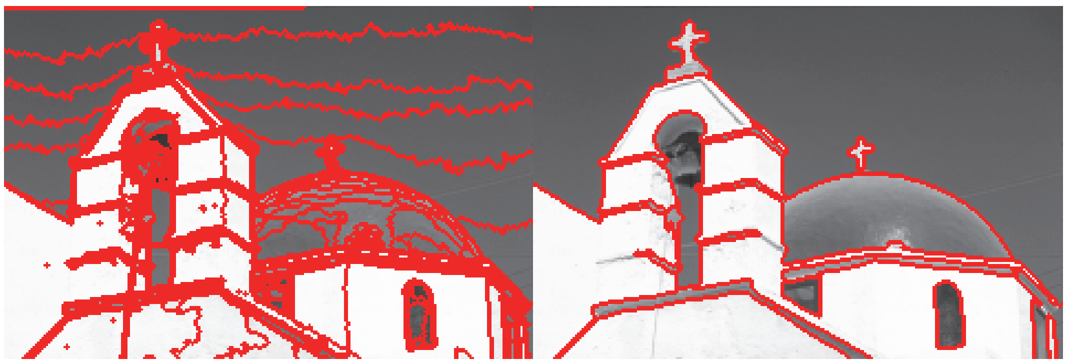

Figure 4. *Results of the GREEDY algorithm on the* (240×160) Church *image. The original image has* 38400 *regions. Left: Initial regions after pruning with* $\lambda = 50$, 2297 *regions. Right: Segmentation results with* $\alpha = 60$, 24 *regions.*

decisions are conditioned by the previous merging performed. For this reason, in this section we seek a more global definition of meaningfulness which would measure the goodness of fit of a whole partition instead of the goodness of fit of a single merging operation.

4.1. Multiregions and meaningful partitions. We start by defining the basic concepts of *multiregions* and *multipartitions*. Given a set of k disjoint meaningful regions R_i, one can ask whether the union R of them is a better explanation of the image than the separate regions alone. To deal with this we define a multiregion k–$\mathcal{R} = \{R_i\}_{i=1}^k$ as the entity resulting from considering the k regions as separate entities. A multipartition, on the other hand, is a special case of a multiregion where the union of the composed regions gives the whole image domain Ω, i.e., $\cup_i R_i = \Omega$.

We will define a k-multipartition $\mathcal{P}_k$ as a partition composed of k regions, and the number k will be called the order of the partition.

As explained in section 3.2 we can model the separate existence of k regions, using k means μ_i and k corresponding errors $\hat{E}_i$ obtained by approximating those regions with their means. Thus the total error committed when approximating the image with k regions will be $\hat{E}_{\mathcal{P}_k} = \sum_{i=1}^k \hat{E}_i$.

Then we can define the NFA of a multipartition as

$$NFA(\mathcal{P}_k) = N(n,k)P(E_{\mathcal{P}_k} < \hat{E}_{\mathcal{P}_k}), \tag{14}$$

where $N(n,k)$ is the number of possible k-partitions that could be formed from an image with n pixels. Again, the two key points here are the computation of $N(.)$ and $P(.)$. In the following, we explain these computations in detail.

4.1.1. Number of multipartitions. In most existing a contrario models (see [16]) the number of tests is computed analytically, because the events to be considered are simple enough to enable the computation. However, in many cases, these computations are too difficult to carry out analytically. This leaves us with two options: either estimate bounds on these quantities or approximate them empirically. In our particular case, we need to count, for each of the image dimensions p, q and for each order of partition k, the number $N(p, q, k)$ of possible partitions of k regions that could be constructed in this image. This is a very hard combinatorial problem which can be found under various forms in many fields of mathematics, and it is a problem still open with no suitable solutions available for our case. One of the closest problems studied comes from the clustering literature, where the number of separable clusters that can be constructed with n unorganized points in $\mathcal{R}^d$ are counted. In [1], the authors propose bounds on the number $N_c(n, d, k)$ of ways in which k clusters can be constructed by splitting those points with hyperplanes. The regions that can be built in this way are simple, in the sense that they cannot contain holes. Thus, the number of regions estimated in this way will be a rough lower bound on the total number of regions. The number of points to be organized into regions in our case is $n = pq$, and our space has dimension $d = 2$; thus the number of possible partitions could be expressed as $N(n, k)$, and the lower bound under these hypotheses is $\hat{N}(n, k) = O(n^{6(k-2)})$.

4.1.2. A true a contrario model. The background model explained in section 3.1 computed the empirical probability density function of the errors, considering all the regions a pixel could belong to. However, those were not all possible regions but only those spanned by the hierarchy $\mathcal{H}$. This has two main problems: First, the statistics are dependent on the hierarchy used, and thus it is not a true a contrario model; second, the computed statistics are dependent on the preprocessing performed. The second problem was mostly alleviated by the validation of mergings introduced in section 3.2; however, the first problem is still present. To address this, we propose a new a contrario model, which states that under the hypothesis of noise, we should not detect any region. A detection in our case is to find a meaningful partition of the image, so if the image is generated by an i.i.d. noise, we should not find any partitions and we should say that the image is composed of a single region. This is equivalent, using our multipartition model, to saying that our random image is a $\mathcal{P}_1$. Thus, to compute the probability, we will consider the error of approximation of the whole image by only one global mean μ.

4.1.3. Probability computation. For the computation of the probability in (14) we rely again on the fact that errors are additive and use the CLT to estimate the probability, as we did in previous sections. Since a $\mathcal{P}_k$ is composed of k independent regions R_i, under the assumption of statistical independence we can compute its probability as

$$P(E_{\mathcal{P}_k} < \hat{E}_{\mathcal{P}_k}) = \prod_i P(E_{R_i} < \hat{E}_{R_i}), \tag{15}$$

and to compute the probability of each region, $P(E_{R_i} < \hat{E}_{R_i})$, we can use the CLT, as we did in section 3.1.

5. The MULTIPARTITION (MP) algorithm. We have seen in the previous section a way to compute the NFA for a given partition. After that, the straightforward procedure is to evaluate this NFA for all possible partitions. However, as we have shown, the number of partitions is enormous, and thus we cannot afford to explore all of them. For this reason, we introduce a fast algorithm to compute probabilities over the hierarchy $\mathcal{H}$. In the first place we compute the mean value and the error for each node on the tree. Then the NFA for each

```
1. Construct the initial partition $\mathcal{P}$.
2. Build the hierarchy $\mathcal{H}$.
3. For each internal node $r_i$: compute the number $n_{r_i}$ of possible partitions
   spawned by its subtree, and its probability $P(.)$ according to section 4.1.3.
4. Start from the root node.
5. Recursive step:
   (a) For the current node $r$ (let $a, b$ be its siblings):
   (b) Create a table of probabilities $T_r$ of length $n_r$; in each position $k$ of the
       table, we will store the probability of a $k$-partition.
   (c) For each possible partition of order $k > 1$ between 1 and $n_r$: compute
       the probability $p$ calling the recursive function $p = prob(r, k)$.
         i. Find all possible couples of subpartitions of orders $i, j$ (of nodes $a$
            and $b$) such that $i + j = k$.
        ii. Compute the probability as $p(i, j) = prob(a, i) * prob(b, j)$.
       iii. Return $p$ as $\min_{i,j} p(i, j)$.
   (d) If $k = 1$, $p(r, 1)$ is the probability computed in 3,
   (e) Store $p$ in $T_r$ at position $k$.
6. For each order $k$ in the root node table ($T_{root}$), compute the NFA from
   the probabilities and the number of tests. Call these computed quantities
   $NFA(root, k)$.
7. Select the final partition order as $o = \text{argmin}_k \{NFA(root, k)\}$.
```

Figure 5. *Outline of the MP algorithm.*

multipartition is computed recursively over the tree starting from the root. The key point to understanding our algorithm is to bear in mind that we are not only obtaining the best overall partition, but we also keep the best k-multipartition for every possible k.

To explain this algorithm (outlined in Figure 5), let us start from the results it yields. Given a hierarchy $\mathcal{H}$ with n leaves, the order of all possible partitions is bounded between 1 and n. For each order k there is a subset of $N(n, k)$ possible multipartitions to choose from, and our algorithm will pick the *best* (lowest NFA) partition from that subset. The output of this algorithm can be summarized as a table with the best NFA for each partition order, as shown in Table 1. So at the end, to choose the best partition we will need only choose the order k with the lowest NFA, among all multipartitions of each order, from $\mathcal{P}_1$ to $\mathcal{P}_n$. Note in Table 1 the opposite effects of the probability (which in turn relates to data adjustment) and the number of tests (which represents a notion of regularity or complexity of the partition). Thus, the optimal partition comes from the balance between the number of tests and probability or, as the previous interpretation suggests, a balance between complexity and data fidelity. An interesting side effect of the way this algorithm works can be seen in Table 1. As we keep the best partition for each order k, we could do a variant of the algorithm where we specify the desired number of regions. This can be particularly useful in some applications where we know beforehand how many objects we are looking for. Some examples of this approach over real biological images are shown in section 6.11.

Table 1

Sample result of the MP algorithm for the image in Figure 1. For each order k we compute the lower bound of the number of partitions $\hat{N}(n,k)$, we select the best partition *$\mathcal{P}^*$ (the one with lowest probability), and we compute its corresponding NFA. Note that the lower the LNFA value, the more meaningful the partition.*

Order (k)	$\log \hat{N}(n,k)$	Error	$\log P(\mathcal{P}^*)$	$LNFA(\mathcal{P}^*)$
1	0	134.5	-0.68	-0.7
2	18.5	18.4	-2909.8	-2891.4
3	55.3	24.5	-7687.1	-7631.9
4	110.5	1.8	-11173.9	-11063.3

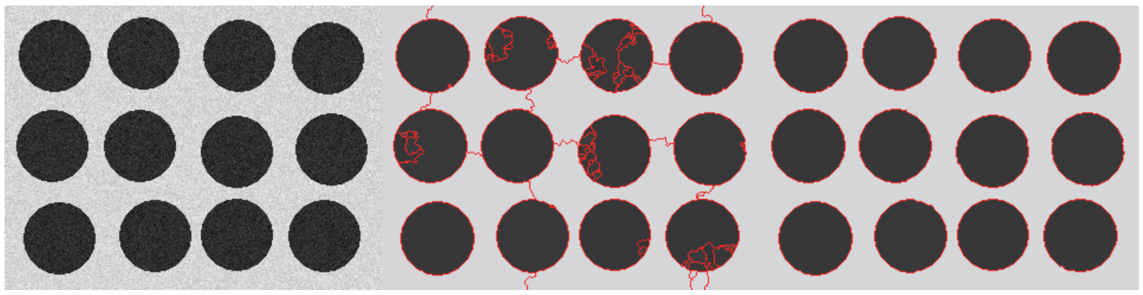
(a) (b) (c)

Figure 6. Blobs*: Example of segmentation over a synthetic image.* (a) *Original image: $N = 76800$ pixels.* (b) *Initial regions after pruning with $\lambda = 7$, $N_i = 117$.* (c) *Segmentation results using only the data term $N_d = 13$. N, N_i, and N_d are the number of regions of each stage, respectively.*

5.1. Synthetic images. To explain the basic behavior of our algorithm, we show in Figure 6 a simple synthetic example where the image is a piecewise constant image with Gaussian noise added. In this case, the image fully complies with the model, so it is the best case for the algorithm. The objects to detect are dark blobs over the light background. As explained in section 5, the output of our algorithm is the NFA of each partition. In Figure 7 (left column) the NFA is plotted against the order of the partition. As can be seen, the minimum NFA corresponds to order 13, which is the number of objects in the image. Looking at the probability graph in Figure 7 we can see that, starting from a 1-partition, the probability shows a great decrease every time we add a new region corresponding to one of the blobs. Once we add all the blobs, we start adding subdivisions of the blobs (or the background) which improve the model only slightly, so the rate of decrease is dramatically slowed. This phenomenon occurs in all images, but it is more salient when working with this kind of synthetic image. The right side of Figure 7 illustrates the same behavior in a more realistic case, which is the *Church* image of Figure 4. In this case, we can see that the decrease of the probability shows softer transitions around the desired number of objects, and it is not that clear how to choose the optimal partition. In addition, as the probability is always decreasing, the algorithm would select the finest partition available. But as the NFA graph shows, the number of tests acts as a regularizer, penalizing finer partitions and thus helping to choose the correct segmentation (around 36 regions in this case).

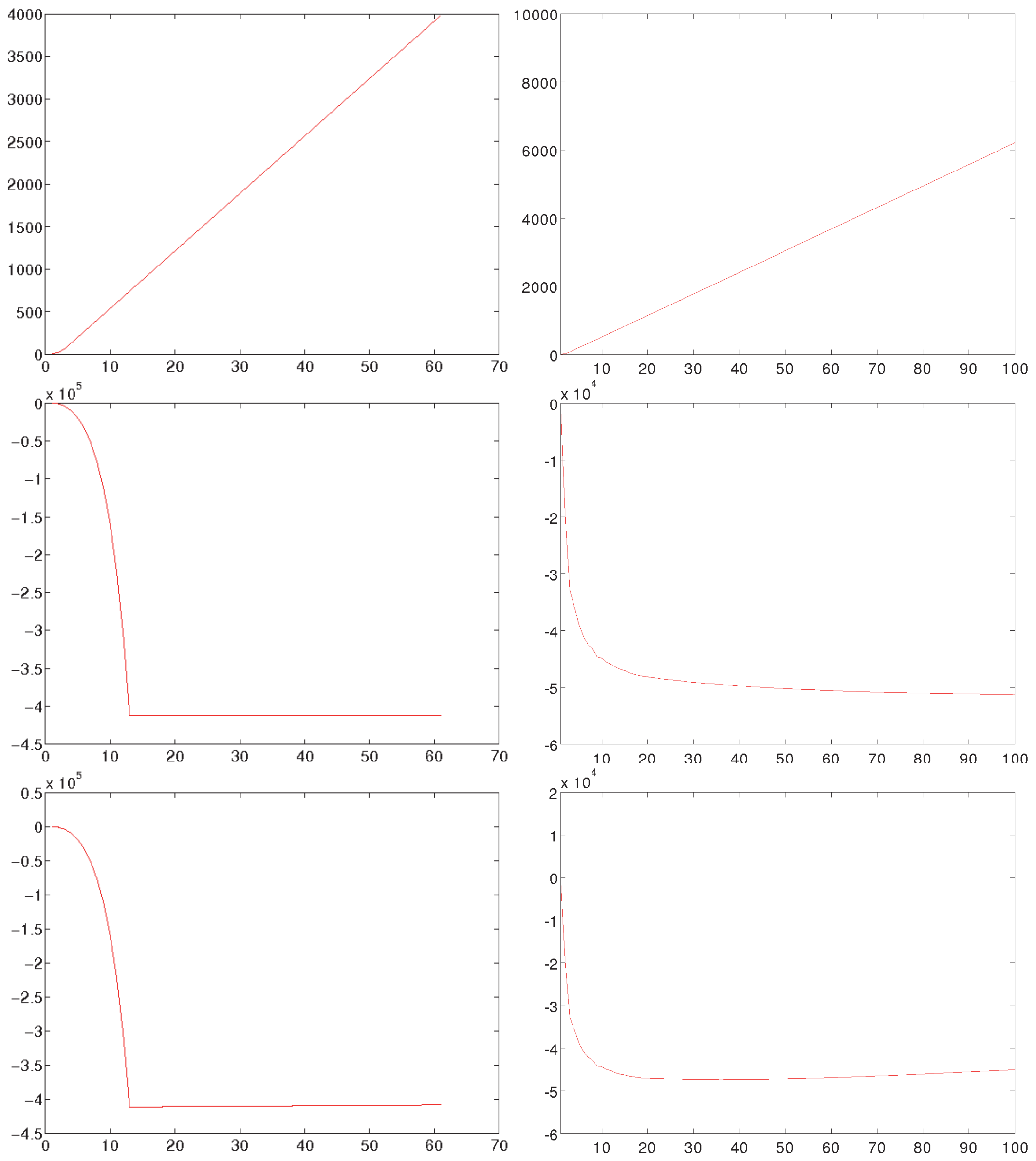

Figure 7. *Analysis of the results on the* Blobs *image of Figure* 6 *(left) and the* Church *image of Figure* 8 *(right). In each row we show the values of the different terms in* (14). *First row: logarithm of the number of tests. Second row: logarithm of the probability. Third row: LNFA. Note that the lower the LNFA value, the more meaningful the partition.*

5.2. Real images.

In the case of the *Church* image, the algorithm has selected the partition with 36 regions as the optimal one. However, as the difference in probabilities is low we are not certain which one is the best partition. In this case we can also output a ranking of segmentations and provide the user with the top M segmentations and let him/her choose the one most suitable for his/her application. In Figure 8 we show the results of such an approach.

The numerical results corresponding to this experiment are shown in Table 2, where we can confirm the fact that most of the top selected partitions have similar meaningfulness.

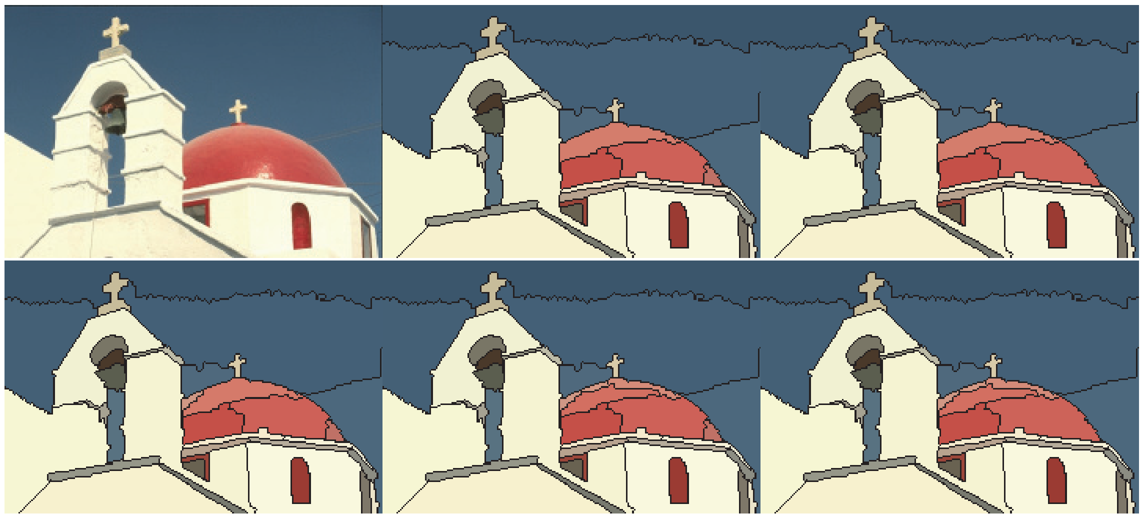

Figure 8. *Ranking of partitions ordered by the NFA for the* Church *image. From left to right: Original image and the five best partitions selected (using the data term only).*

Table 2
Best partitions selected by the MP algorithm for the Church *image.*

Order (k)	LNFA (best $\mathcal{P}$)
36	-47374.1
35	-47373.5
37	-47371.7
39	-47369.9
38	-47369.1

5.3. Recursive computation. The results in Table 1 can be computed in a recursive way as follows. First we compute the NFA for each region in $\mathcal{H}$, using (14). Consider the subtree $\{6, 2, 0\}$ in the hierarchy of Figure 1(b): it can span only two partitions, a $\mathcal{P}_1$ composed by node 6 and a $\mathcal{P}_2$ composed by nodes $\{2, 0\}$, whose probabilities are shown in Table 3. The same reasoning can be applied to the $\{5, 3, 1\}$ subtree. Thus, when analyzing the whole tree, we can see that the two possible $\mathcal{P}_3$ can be obtained by combining $\mathcal{P}_1$ and $\mathcal{P}_2$ from both subtrees. Then, as we are interested in the most meaningful $\mathcal{P}_3$, we will pick the one with lowest probability (which is $\{6, 3, 1\}$). In that way, we can compute the probability of any $\mathcal{P}_k$ from the probabilities of i-$\mathcal{P}$ and j-$\mathcal{P}$, with $i + j = k$, obtained from the subtrees associated with its siblings. The expensive part of this algorithm is the computation of the MS data adjustment term (error) from (1), which needs to be evaluated for every pixel on a given region. In addition, each pixel can be assigned to many different regions in $\mathcal{H}$, and thus the computation is repeated many times for each pixel. One way to avoid this overhead is to take advantage of the fact that the error (but not the probability) is additive. In this way, we can

Table 3

Intermediate results of the MP algorithm for the image in Figure 1. For each internal node (5 and 6) we compute all possible partitions spanned by the corresponding subtree.

Node 5 (subtree {5,3,1})			
Order (k)	$\log N(n,k)$	MS error	$\log P$ (best $\mathcal{P}$)
1	0	23.86	−2394.6
2	18.47	1.20	−5539.0
Node 6 (subtree {6,2,0})			
Order (k)	$\log N(n,k)$	MS error	$\log P$ (best $\mathcal{P}$)
1	0	42.99	−751.8
2	18.47	0.60	−5640.3

compute the error of the $\mathcal{P}_3$ composed of regions $\{5,3,1\}$ from the errors of partitions $\{5\}$ and $\{3,1\}$. In this way, each error is computed only once per region.

5.4. Optimality and complexity. Our procedure for computing the probabilities is based on a dynamic programming scheme and thus computes the globally optimal partition by taking local decisions at each level. It is important to determine whether we can achieve the global optimum with this procedure, which is discussed in the following theorem.

Theorem 5.1 (optimality). *Given the number of regions N of the partition of the image, the recursive computation of probabilities gives the optimal partition.*

Note that the background model ensures that the probabilities of false alarms are always strictly greater than zero. Indeed, the only way to obtain a zero probability is to compute $P(E_R < -\infty)$, but that could not happen because the error E_R is always finite.

Proof. Suppose A is a node of the hierarchy and B, C are its sons.

(Binary case). Let P_A be the partition of A which is optimal for n sets (the partition is formed by n sets and is optimal between all of the partitions of n sets).

Let $P_A \cap B$ be the partition of B made of the sets (say n_1) of P_A that are in B, and let $P_A \cap C$ be the partition of C made of the sets (say n_2) of P_A that are in C. Note that $n = n_1 + n_2$.

Let us prove that $P_A \cap B$ is an optimal partition of B with n_1 sets. Otherwise, there exists a partition of B with n_1 sets P_B such that $NFA(P_B) < NFA(P_A \cap B)$. By replacing $P_A \cap B$ by P_B, we have that $NFA(P_B)NFA(P_A \cap C) < NFA(P_A \cap B)NFA(P_A \cap C) = NFA(P_A)$, and this is a contradiction.

Similarly, $P_A \cap C$ is an optimal partition of C with n_2 sets. Thus, given the number of regions N of the partition of the image, the algorithm gives the optimal partition in the hierarchy with N sets. ■

As we are sure that we could reach the optimal partition, we can use this very efficient dynamic programming scheme to compute our probabilities.

From the practical point of view, we can also ensure that the probabilities are strictly positive. The problem could arise when they very small, because we can reach the precision of the data type and have representation errors leading to an exact zero value. However, as we internally work directly computing the logarithms of the probabilities, we circumvent this representation problem and never get an exact zero value.[2]

Regarding the complexity of the algorithm, again it has two parts: constructing the hierarchy and selecting the optimal partition. Assume an n-pixel image and a pruned tree of N leaves. From the analysis of the GREEDY algorithm in section 3.7, we have a cost of $\Theta(n \log n)$ to construct the tree. However, in this case, the selection part is slower, because for every node we need to compute the probabilities of every possible partition spanned by it which has a great number of combinations to explore, because we keep the best partition for every possible level. On the other hand, in the GREEDY case we need only find a minimum NFA couple at each step. The analysis of the complexity of the NFA computation is not trivial, and it is shown in Appendix A, as that computation shows that the complexity of this part is $\Theta(N^3)$. If we make no initial pruning, the complexity will be $\Theta(n^3)$, which is unusable in practice for medium sized images. If we make a reasonable pruning of $N \approx n^\gamma$ with $\gamma = 0.5$, as mentioned in section 3.7, the complexity will be $\Theta(N^{1.5})$. Finally, if we want our algorithm to run fast, we need to prune the tree to $N \approx n^{1/3}$. This could be a reasonable amount of pruning in practice (around 100 regions for a 1024×1024 image), and it is better as n grows. If that condition holds, the worst case complexity of this selection could be reduced to $\Theta(n)$. From this we can conclude that the complexity of the algorithm can be reduced to $\Theta(n \log n)$ if proper pruning is applied.

5.5. Boundary term. In the MP case, our data term is based on a piecewise constant model and has no regularization term. For this reason, when faced with images which do not comply with the model, it presents some problems. Mainly, it tends to oversegment almost flat regions with a slow varying intensity gradient. This phenomenon is shown in Figure 9(c). To overcome this problem, we could add a boundary term in just the same way we did with our previous algorithm in section 3.3. That is, instead of computing the probability $P(R1) = P(E_{R1} < \hat{E}_{R1})$, we could add another term measuring the meaningfulness of the boundary so that $P(R1) = P(E_{R1} < \hat{E}_{R1}).P(L_{R1} < \hat{L}_{R1})$, where $\hat{L}_{R1}$ is the length of the boundary of region $R1$ computed with (6). However, this introduces the need to consider the relative importance of each term, which in turn leads us to introducing some kind of weighting parameter. In addition, due to the way probabilities are constructed, we were not able to find a definition of this term holding the decreasing property needed for a proper scale-climbing algorithm. For this approach to work, the length term should always be decreasing with scale, and our term is decreasing in almost every case, so we cannot guarantee this property.

For these reasons, we propose the use of the boundary term as a postprocessing, in order to refine the partition selected by the data term. Thus after the best partition is selected, as shown in section 5, we test every edge separating each pair of neighboring regions, and if the boundary between them is not meaningful, we merge them into a single region.

So far, we have shown examples of the MP algorithm using the data term only, and, as we have mentioned before, in images like *Church* we see that our algorithm tends to oversegment large uniform regions, which are clearly far from the piecewise constant model. In Figure 9 we show how the boundary term solves this problem and gives a more accurate segmentation.

[2]We can also limit the probabilities using a small threshold $\epsilon > 0$, $P = \max(P, \epsilon)$, or we can add ϵ to the probabilities and then normalize them to sum one.

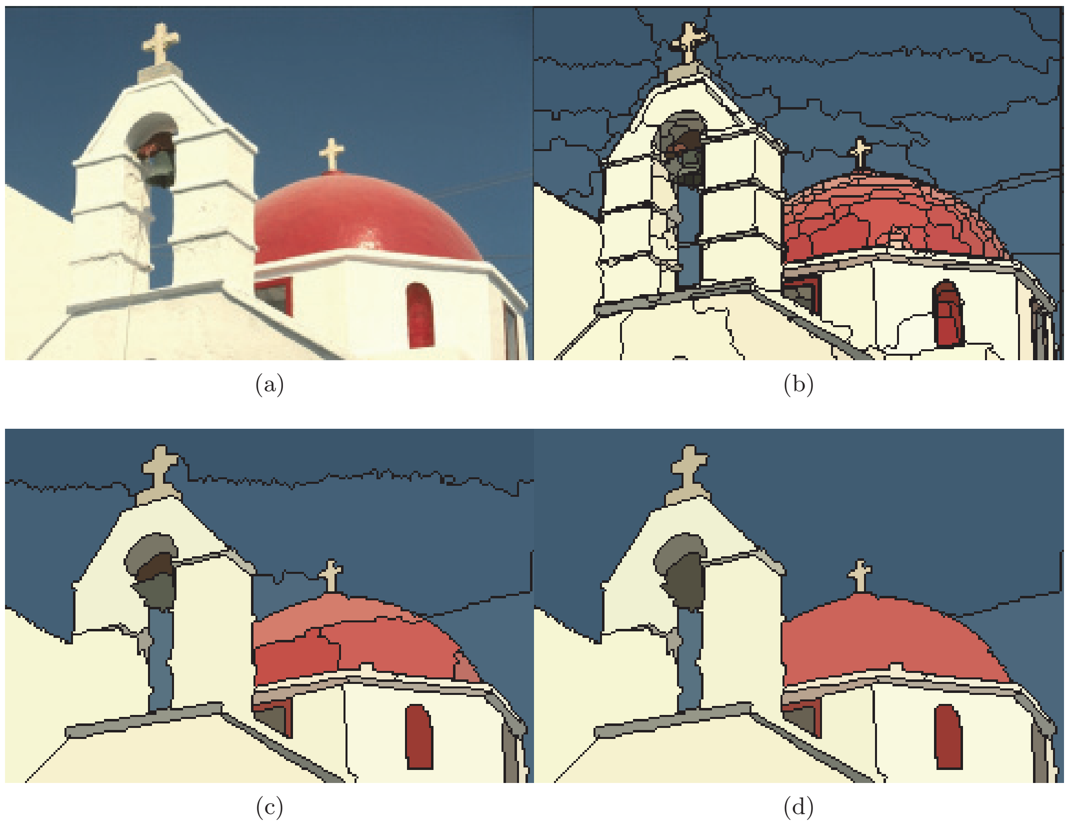

Figure 9. *Results of the MP algorithm on the* Church *image.* (a) *Original image:* $N = 38801$. (b) *Initial regions after pruning with* $\lambda = 10$, $M = 628$. (c) *Segmentation results using only the data term* $N_d = 36$. (d) *Segmentation results after adding the boundary term* $N_c = 29$.

5.6. Extension to multivalued images. So far, we have presented our algorithms for gray-level images. In this section we will extend them to color images and to the more general case of vector-valued images. In this case, an image will be a vector-valued function $I : \Omega \to R^d$. We start by rewriting the MS data term in (1) as

$$E_R = \sum_{x \in R} d(I(x), \mu_R), \tag{16}$$

where $\mu_R \in R^d$ and $d(x)$ is any monotone function of a norm in R^d. In addition, this formulation leaves the door open to see this term as the goodness of fit of the image value $I(x)$ to the region R, which leads us to the more general formulation

$$E_R = \sum_{x \in R} p(I(x)|R), \tag{17}$$

where $p(.)$ is the probability that pixel x belongs to region R. This formulation gives us a general and flexible way to improve our results by choosing different region models and metrics. In the particular case of color-valued images ($d = 3$), to complete our model we need to choose a color space and a suitable metric in that space. Two important requirements for this space are to have a clear separation between color and intensity channels and to be coherent with human perception. For this reason, we choose the CIELab color space, in which equal color differences approximate equal differences perceived by a human observer, thus making it possible to compare colors using a Euclidean norm. For this reason, to compute the error term we choose $d(x)$ as the squared Euclidean norm in R^3.

5.7. Color-valued gradient. The final step on the extension to the multivalued case is to extend the boundary detection algorithm of Cao, Muśe, and Sur [9] to work with vector images. The notion of gradient for multichannel images is not well defined; however, there are some works in the literature that try to define similar notions. In this work we follow the ideas of Di Zenzo [17], which proposes computing the structure tensor and obtaining its eigenvalues. In this setting, the largest eigenvector represents the direction of maximum change and its corresponding eigenvalue the magnitude of that change. As we are interested in finding contrasted boundaries, we will compute the probability of the minimum value of the magnitude of the largest eigenvalue along the curve as we did with the gradient in the scalar case.

5.8. Parameter choice. The algorithm, as presented in previous sections, does not have any free parameters. For that reason, it does not allow us to select the desired scale. Without any parameters, it works at a fixed scale which usually gives oversegmented images. This is explained by the collaboration of two phenomena. First, we have a lower bound on the number of tests that is a very loose estimate. Second, as we are using the approximation of independence between pixels to use the CLT, in many cases we obtain lower bounds of the joint probability by factorizing it into independent terms. These two approximations have the effect of making finer partitions more meaningful than coarser ones.

Summarizing, the main problem of the algorithm is to find the correct scale of an image. From a practical point of view, it would be interesting to have a way to select the optimal scale given a set of examples. We recall from section 4.1.1 that our lower bound for the number of partitions was $N(n, k) \geq \Theta(n^{6(k-2)})$. We have simulated this bound for the $k = 2$ case and varying n, and we have found that the difference between the actual quantity and the bound increases with n. As the number of tests is related to the complexity of the partition, underestimating this quantity will lead the algorithm to oversegment partitions.

With this in mind, we propose adding a simple correction of the form $N(n, k) \geq O(n^{\alpha(k-2)})$, where parameter α is introduced as an exponent to n to reduce the increasing gap between both quantities. This gives us the flexibility to adjust the α parameter to match the correct scale; this parameter is tuned according to the dataset used and the application. The value $\alpha = 6$ is the theoretical estimate of the lower bound of the number of tests, and it will be called the *default* value of the parameter.

Tuning parameters is a difficult and time consuming task. For the sake of simplicity for the end user, we provide here an automatic way to determine α for a given dataset. We achieve this by optimizing a simple cost function given by the difference of the number of regions detected by the algorithm and by human subjects. Thus, given a parameter value α, which gives the segmentation Q_i for image i and a ground truth segmentation P_i, the error function is

$$E(\alpha) = \sum_{i}^{N} (\#P_i - \#Q_i)^2.$$

Then we perform a simple empirical error descent on this cost function and find the *optimal* value for α. In section 6.5 we will discuss how to find suitable values for this parameter for the database used in the experiments, how the algorithm behaves with different values of α, and how it compares with the automatically tuned value. In the following, the variant of the algorithm using the default value will be called MP-D, and the variant using the modified α will be called MP-M. When α is selected automatically, we denote the algorithm as MP-S (S stands for "supervised").

6. Results.

6.1. Real images. In the first place, we present here qualitative results of the MP algorithm. In Figure 10 we show the results of the algorithm in a few selected images of the Berkeley Segmentation Database (BSDS300 subset) [27], with different values of the parameter. As this figure shows, the results are quite good even with the great variability presented by the images. However, we also observe that using the default parameter value, our algorithm tends to give oversegmented results in most of the cases. In addition, these illusory boundaries are not completely removed by the boundary term. However, when adjusting the scale parameter the results are greatly improved and the results are visually pleasant. Finally, as in this example we are not using texture descriptors; in natural images presenting a *camouflage* effect (last row of Figure 10), the algorithm performs poorly regardless of the choice of the parameter.

In Figure 11 we show results over real images taken from the test set of the BSDS500 database. For each image, we show the corresponding saliency map which serves to assess the quality of the hierarchical segmentation obtained. The saliency map is constructed by labeling each contour with its scale of disappearance α^-, which is the maximum scale at which a region is present in the output partition. These results show two important properties. First, the obtained stack of partitions is causal; i.e., as the scale parameter increases the segmentation gets coarser. This can be verified by observing the values of α^- assigned to each contour. If a contour has a high α^- value, it means that it will survive for a long time in the merging process. On the other hand, small values mean that it will be removed at early stages. As the figure shows, the highest values of α^- are assigned to the most coarse segmentation. And second, the stack of partition is geometrically accurate; i.e., the borders present in each scale are a strict subset of the previous (finer) scale. If they were not included, they would be seen as new contours in the saliency map. These results empirically show that the obtained hierarchy holds the strong causality property.

In Figure 12 we show a comparison of the hierarchy of partitions generated by the MP algorithm against the initial hierarchy over the same set of real images. The results obtained by the MP algorithm are similar to those from the initial partition, with the difference that less salient boundaries are detected in the slowly varying regions. This is mainly due to the introduction of the boundary term weighted by the gradient. An important property of this model is that even with the smallest possible parameter value the finest possible partition in

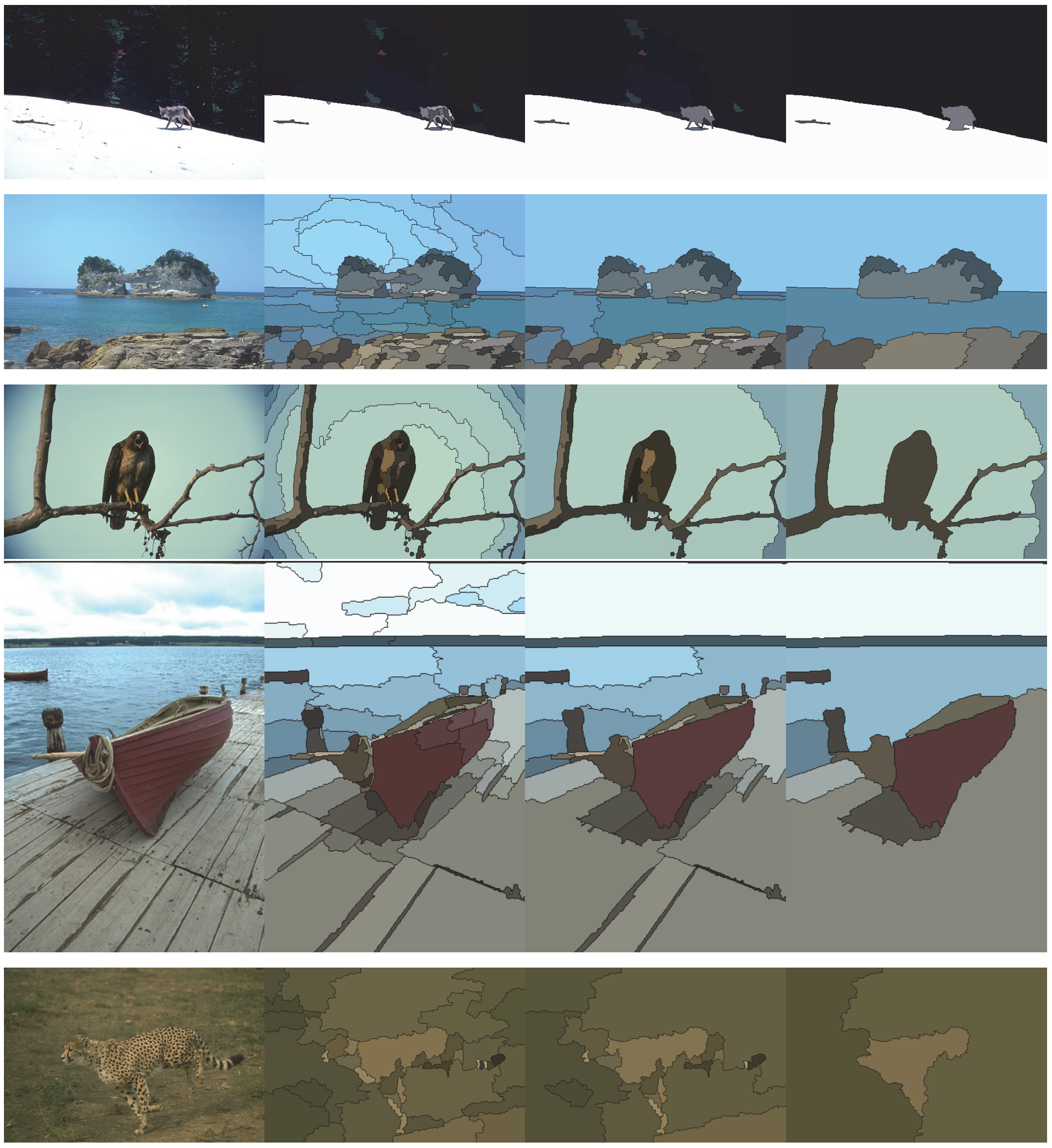

Figure 10. *Results of the algorithm over real images extracted from the BSDS*300 *(images numbered* 167062, 249061, 42049, 138078, 134008, *respectively). From left to right: original image, initial partition, segmentation using the default parameter* ($\alpha = 6$)*, segmentation using a manually tuned parameter* ($\alpha = 50$)*. In the last three images of each row, segmented regions are filled with their mean color and the boundaries are superimposed in black. The corresponding number of detected regions for each image is shown in Table* 4.

the initial hierarchy cannot be obtained. This is due to the regularizing effect of the CLT approximation of the probabilities, which favors partitions with bigger regions.

6.2. Noise levels and threshold analysis. In Figure 13 (top) we show the behavior of the algorithm under different noise powers, on a 1000 $\times$ 1000 gray-level version of Figure 1. The regions have the following means: 50, 100, 150, 200. Eight test images were created, with a

Table 4

Number of detected regions corresponding to images shown in Figure 10.

Image	Partition sizes		
	Initial	MP-D	MP-M
Wolf (167062)	39	19	4
Sea (249061)	135	62	11
Bird (42049)	102	28	10
Boat (138078)	91	43	14
Cheetah (134008)	41	22	3

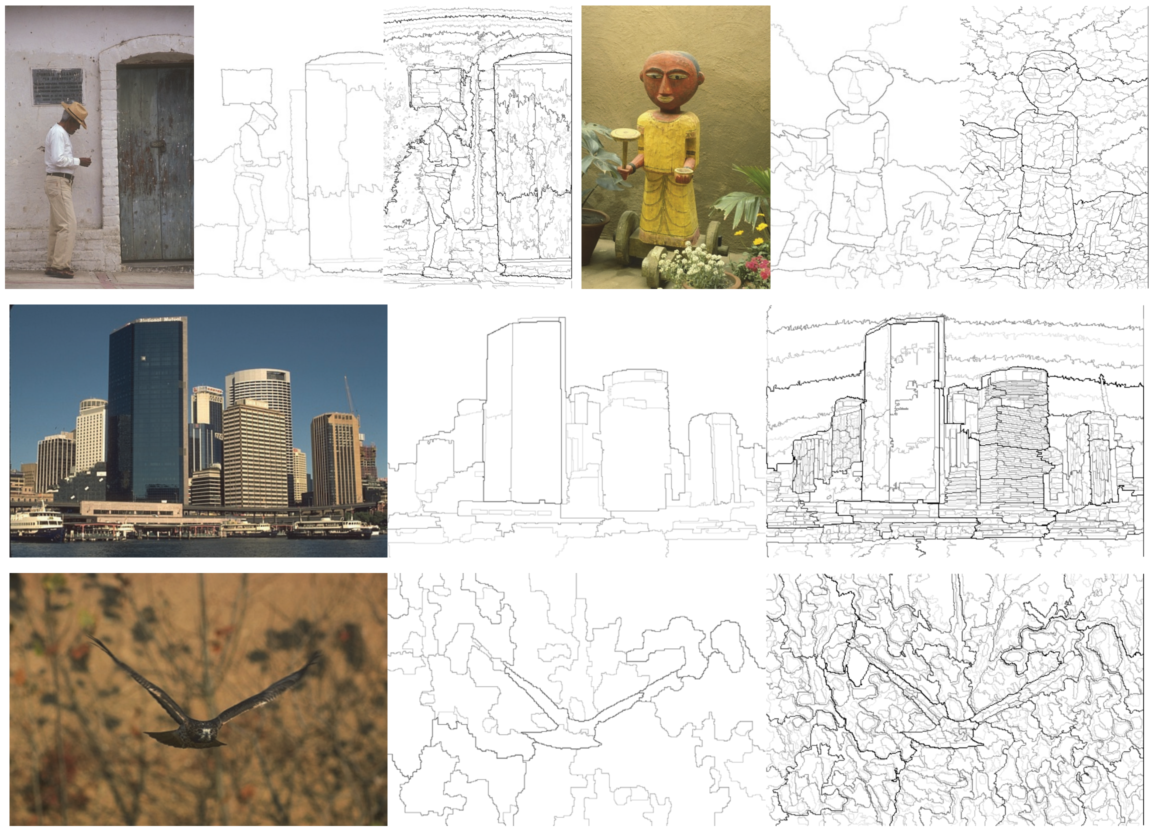

Figure 11. *Results of the algorithm over selected images of the BSDS500 (test set). For each example we show the original image on the left and a saliency map of the MP algorithm in the middle. The value of each contour corresponds to the scale of appearance* α^- *of each region. In the right images we show a pseudosaliency map for the Guigues algorithm* [20]. *In this case, each contour is labeled with the number of times the contour appears in the hierarchy.*

zero mean Gaussian noise of power σ added. The lower (logarithmic) bound for the number of tests has the same value of 165.8 for all cases. As we are modeling regions by their means, when we increase noise power the adjustment to that model becomes worse, and thus the meaningfulness of the selected partition is lower. However, the algorithm is able to select the correct partition (four regions) in each case, even under heavy noise conditions.

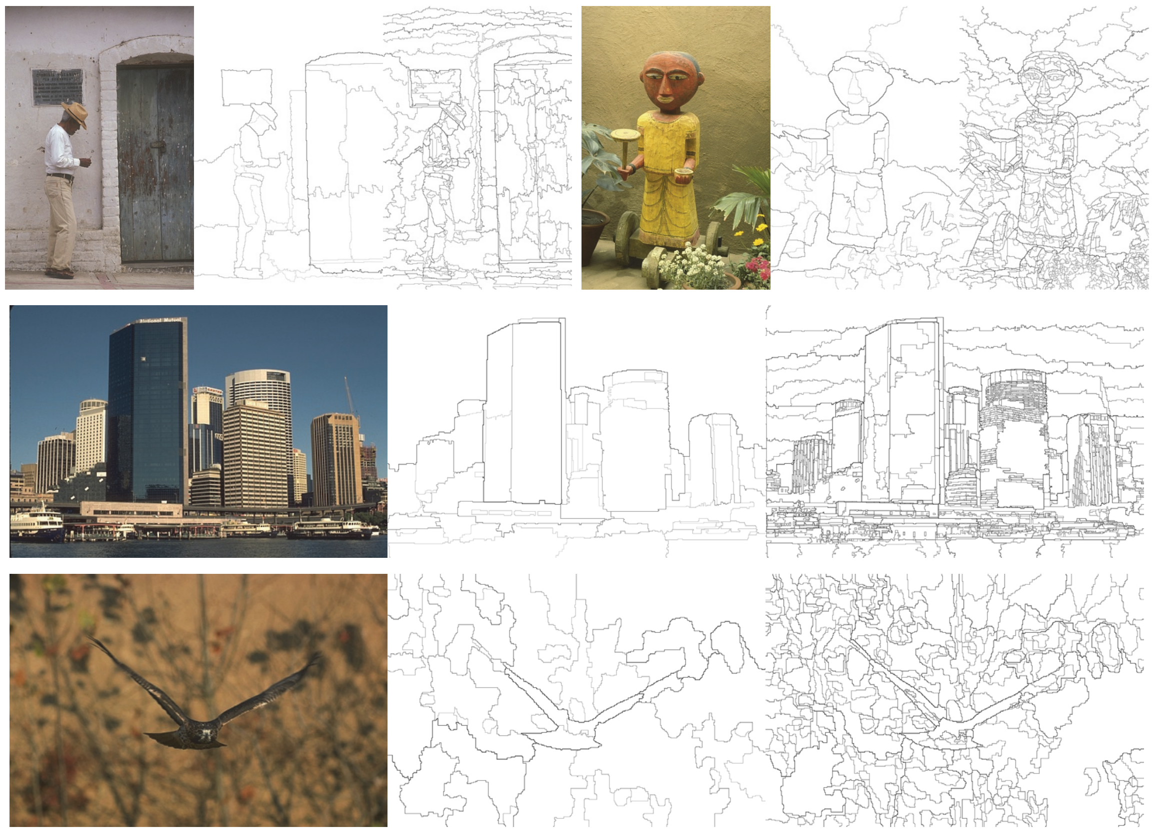

Figure 12. *Results of the algorithm over selected images of the BSDS500 (test set). For each example we show the following images. Left: original image. Middle: the saliency map of the MP algorithm. Right: saliency map of the initial hierarchy. For the MP algorithm the value of each contour corresponds to the scale of appearance α^- of each region. For the initial hierarchy the value of each contour corresponds to the scale of appearance λ^- of each region (where λ is the scale parameter in the MS model).*

In most a contrario approaches there is a notion of detection: if the meaningfulness is lower than a threshold, they detect objects, and if it is greater, they give no detections. On the contrary, our algorithm always selects a partition, and it does not have a notion of no-detection. This happens because we are not thresholding the NFA of each partition, but selecting the one with the lowest NFA. As Table 1 shows, all the partitions are meaningful which means that our threshold on the NFA is not well adjusted. This is a common problem in a contrario approaches and was studied in detail in [19]. Another problem with this approach has to deal with the size of the images: if we have two images with the same content and different sizes, the NFA values will change due to the use of the CLT and with bigger images all structures will be more meaningful. In Figure 13 (middle and right) we show an experiment with an image composed of one region with a mean of 128. In addition, Gaussian noise with different powers was added, and two different image sizes were considered. As the figure shows, the values for a bigger image are more extreme: meaningful partitions become more meaningful and nonmeaningful ones become even less meaningful. However, our approach

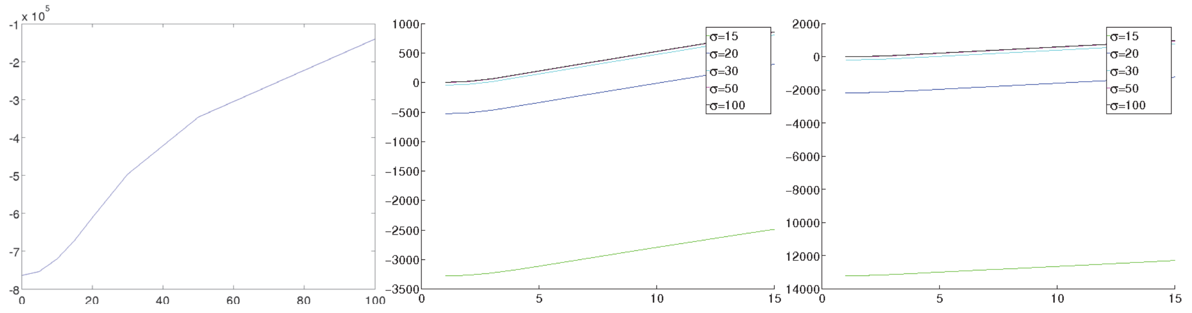

Figure 13. *Left: LNFA of the best partition (four regions) selected against noise power. Middle and right: Effects of the size of the images and noise power on the NFA. In each image the LNFA is plotted against the number of regions of the partition for two image sizes (*250×250 *and* 500×500 *in middle and right images, respectively).* σ *is the power of the added noise. The color codes green, blue, cyan, magenta, and black correspond to the increasing* σ *values of* 15, 20, 30, 50, 100, *respectively.*

does not attempt to say whether an individual partition is meaningful or not, but tries to get the partition that better explains the data in a way relative to the other partitions, so in the end, we are immune to these problems.

6.3. Performance metrics. In order to compare our results against human segmentations, we need suitable metrics to compare two different partitions of the same image. There are many metrics proposed in the literature, but most of them are heuristic and lack theoretical background and validation. One of the most interesting works in this area is by Martin [25], where the BSDS was built and some metrics consistent with human perception were proposed. In a posterior work Cardoso and Corte-Real [12] introduced a new set of metrics which aim to improve Martin's proposal, while maintaining the same basic idea. One way to evaluate the performance of a segmentation algorithm is to model the (qualitative and quantitative) properties of human made partitions and then verify whether a given segmentation has these properties. On the contrary, rather than model human segmentation, these works compare two given segmentations of the same image in a way which is consistent with our perception.

In the following we will briefly review Martin's work [25]. One key concept introduced is the notion of mutual refinement. Given two partitions P and Q, P is a refinement of Q if for every region $R_i \in P$, there exists $S_i \in Q$ such that $R_i \subset S_i$. On the other hand, P and Q are mutual refinements if $R_i \cap S_i$ is either equal to R_i or S_i or it is the empty set. This means that locally, either P refines Q or vice versa.

One of the conjectures in Martin's work is that human segmentations of the same scene are mutual refinements of each other and that all possible human segmentations of a scene could be organized in a perceptual tree. Thus, each human made partition is just a selection of regions around this tree. It is important to note that this tree induces a notion of *scale*, not in the scale-space sense, but regarding the granularity of the partition. Put in this way, each human segmentation differs from each other in the scale selected. However, this scale notion is somehow loose, because it is not global but local. The conclusion from these observations is that differences due to mutual refinements are perceptually less important than those produced by misplaced regions.

Cardoso and Corte-Real [12] propose three different metrics to quantitatively evaluate results. We give here only an intuitive idea about how they work; for a detailed definition please refer to [12]. Given two partitions P and Q to be compared, the following quantities are defined:

- *SPD(P,Q)*: this quantity roughly reflects the percentage of incorrectly classified pixels. It does not tolerate mutual refinements.
- *APD(P,Q)*: same as SPD, but it does not count errors on pixels where P is a refinement of Q.
- *MPD(P,Q)*: same as SPD, but it does not count errors on pixels where P and Q are mutual refinements.

These metrics are bounded between 0 and 1, and they count errors; thus small values mean good performance and vice versa. It is also important to note that SPD and MPD are symmetric with respect to their arguments, while APD is not. In the following we will denote the human partition as P and the algorithm result as Q; and we will call this set of metrics partition distances (PD).

Although these metrics are not additive, SPD should roughly reflect the total amount of error, which in turn is related to the sum of $APD(P,Q)$ and $APD(Q,P)$. This means that two partitions with the same SPD have the same performance, and the differences between the APD show us the balance between oversegmentation and undersegmentation. However, this interpretation alone could be misleading, because SPD includes errors of different nature. For instance, if a region in the human partition is divided into two regions in the computer segmentation, for the SPD this error counts the same as if the region were completely misplaced. Even between human observers (see [25]) partitions are not consistent in terms of an exact matching but rather in terms of mutual refinements. For this reason, global performance could be measured with the SPD, but the value of MPD should be verified too. While it is tempting to summarize both measures into only one, we believe that this would be too simplistic and we would lose the richness of the information of these metrics. For this reason our measure of overall performance will be a qualitative interpretation of both SPD and MPD.

6.4. Quantitative evaluation. Most of the images used in this paper for the validation process come from the BSDS, which is a subset of 1200 images from the Corel Database, with human made segmentations available. Each input image was manually segmented by at least five different subjects. One of the publicly available datasets, the BSDS300, is composed of 300 images of 481 $\times$ 321 pixels, divided into train and test subsets of 200 and 100 images, respectively. Ground truth segmentation is provided for all images. The instructions given to the subjects were not to identify objects but uniform regions on the images, which makes this database well suited for low-level image segmentation. In order to test the independence of our results from the BSDS300, we also conducted experiments with the Lotus Hill Institute (LHI) database [40], which is a publicly available database with a ground truth labeling of the objects present in the image, mainly designed for object recognition tasks. Recently, a new version of the BSDS was released under the name of BSDS500, which adds 200 new test images, along with a new set of benchmarks. In the final part of this section, we also present results using this database.

We present here just a brief excerpt of the extensive validation conducted; full results and detailed analysis of each algorithm can be found at http://iie.fing.edu.uy/rs/wiki/Image

Table 5

Quantitative evaluation of the proposed algorithms over the BSDS300 using PD metrics. The number n is the average number of detected regions for each algorithm.

Alg.	SPD	$APD(P,Q)$	$APD(Q,P)$	MPD	N
initial ($\lambda = 50$)					222.5
MP-D ($\alpha = 6$)	0.56	0.54	0.09	0.07	63.8
MP-M ($\alpha = 50$)	0.40	0.32	0.17	0.10	16.0

Table 6

Quantitative evaluation of the proposed algorithms over the BSDS300 using using PD metrics.

Alg.	SPD	$APD(P,Q)$	$APD(Q,P)$	MPD
Results over the complete database				
MP-D ($\alpha = 6$)	0.56	0.54	0.09	0.07
MP-G ($\alpha = 6$)	0.49	0.43	0.16	0.12
GREEDY	0.57	0.54	0.12	0.10
Results over the 100 test images				
MP-D ($\alpha = 6$)	0.56	0.54	0.10	0.08
MP-S ($\alpha = 21$)	0.46	0.41	0.13	0.09
MP-M ($\alpha = 50$)	0.41	0.31	0.19	0.11
MP-M ($\alpha = 150$)	0.38	0.18	0.28	0.09
GREEDY	0.58	0.54	0.13	0.10
MP-G ($\alpha = 6$)	0.50	0.45	0.17	0.13
MP-G-S ($\alpha = 21$)	0.42	0.31	0.23	0.14

SegmentationAlgorithms. We have also made available the source code and an online demo to run the algorithm via a web page. We developed a new benchmark based on PD metrics, and the code to run it is also available online.

In Table 5 we show the average value of the performance metrics over all the images in the BSDS for the MP algorithm with two different parameter values. The small MPD value shows that most of the partitions given by the algorithm are coherent with the human segmentation, meaning that we get only a few overlapping regions that are not refinements, and thus our result is a mutual refinement of the human segmentation. Moreover, as we have seen in the qualitative evaluation, the MP-D algorithm tends to over-segment. This conclusion is reinforced by the small value of $APD(Q,P)$, which indicates that there are very few undersegmented regions. On the other hand, the big value of $APD(P,Q)$ also indicates that the number of oversegmented regions is high. In addition, the similarity between SPD and $APD(P,Q)$ and the small value of MPD tell us that most of the errors come from choosing the incorrect scale (finer) rather than having misplaced regions.

When we increase the parameter value α, we see that the number of false positives ($APD(P,Q)$) is reduced and the number of false negatives is increased ($APD(Q,P)$). In addition, the number of detected regions is lower, which suggests that α is acting as a scale parameter.

In Table 6, we show a comparison of performance among all the presented algorithms. We can see that the MP algorithm outperforms the GREEDY algorithm, improving almost every performance metric by around 2%. However, as the MP algorithm uses color information, in order to make the comparison fair we have made a gray-level variant of it called MP-G.

The first point to be noticed is that the gray-level version of the algorithm presents a more balanced result in terms of the SPD and MPD, but the overall performance is slightly worse than the MP color case, because it presents higher values for both SPD and MPD. However, the difference is not as big as we could have expected. One of the main problems of the MP algorithm is oversegmentation, and since MP-G does not use color information, all regions are more similar and we expect the algorithm to make more mergings and give coarser partitions. In this sense, discarding color information diminishes oversegmentation, and thus the results are better than expected.

There are two remarkable points in the results of the comparison between the MP-G algorithm and the GREEDY one: first, the MP-G algorithm shows better performance overall, showing a considerable improvement in SPD and a lower decrease in MPD; second, and probably more important, these results were computed by tuning the α parameter of the GREEDY algorithm and without tuning any parameter in the MP-G case. These two observations together show the importance of validating complete partitions rather than single merging decisions.

Up to this point, we have shown only average values over the BSDS300 database, but it is also interesting to show how these quantities are distributed over the database. In Figure 14 we show such distributions for each measure. This figure confirms the conclusions drawn from the average values shown before. The SPD is concentrated towards high values, while the MPD lies around small values. The shift to the left of the values of $APD(P, Q)$ shows that the algorithm with this parameter value has very few undersegmented images. On the contrary, the higher values of $APD(Q, P)$ show that we have many oversegmented images.

In addition, in the first row of Figure 15 we show the distribution of the number of regions detected by human subjects and by the MP algorithm, which clearly confirms that our algorithm with the default parameter value is oversegmenting.

6.5. Parameter learning. What is left to complete the algorithm is to choose a suitable value of the scale parameter α. As we stated in the introduction, there is no such thing as the *correct scale* for the general case. This notion is application and dataset dependent. Even given a dataset like the BSDS, not every image has the same scale, and thus we should define an optimal set scale, which is the best scale one can choose for the complete dataset, and an optimal image scale, which is different for every individual image. In this work, α represents the notion of optimal set scale.

In the first place, we proceeded to tune the α parameter manually, using the training subset of the BSDS to find a suitable value for α. The goals established were the following: obtaining more balanced results in terms of the APD, the diminishing SPD, and keeping the increase of MPD controlled. As a result, we found that $\alpha = 50$ is a good compromise for this set of metrics and for this database. The results of this choice of the parameter are shown in Table 6, where we can see that the results are better than those of the previous algorithms.

Regarding the automatic parameter tuning, for the particular case of the BSDS, we ran the parameter estimation algorithm presented in section 5.8 with the 200 images from the training set and obtained a value of $\alpha \approx 21$. We also ran the same procedure with the gray-level version of the algorithm, and we called these variants multipartition-supervised (MP-S) and multipartition-gray-supervised (MP-G-S). Their results over the test set of 100 images are

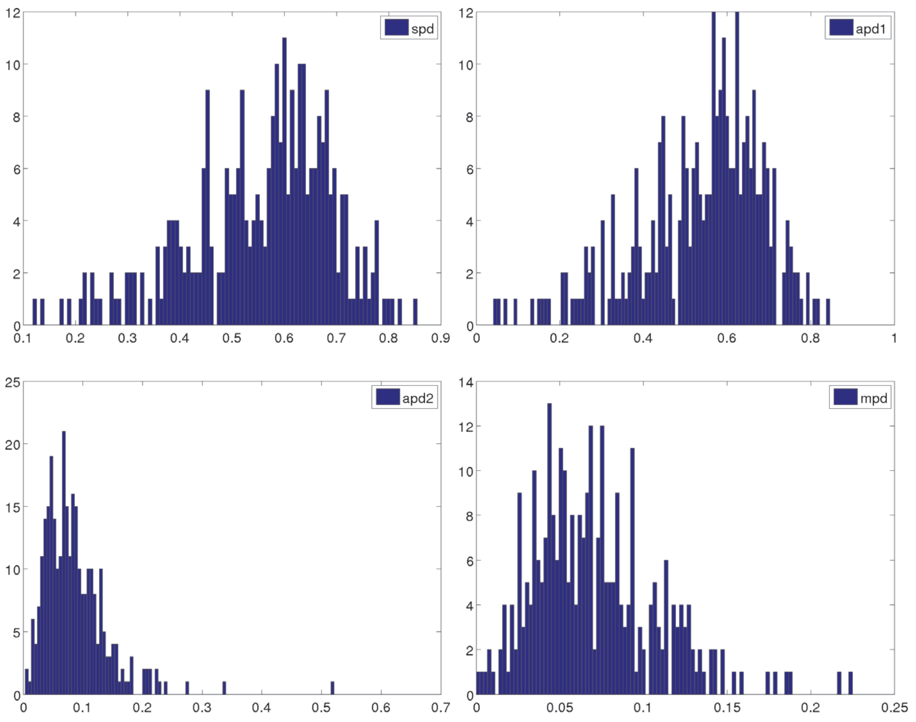

Figure 14. *Distribution of performance measures over the BSDS*300 *database. From left to right, and from top to bottom:* SPD*,* $APD(P,Q)$*,* $APD(Q,P)$*,* MPD*.*

shown in Table 6. As this table shows, the training procedure greatly reduces the number of false positives (FP) and slightly increases false negatives (FN), so the overall performance is clearly improved (10% reduction in SPD) and the FP and FN are more balanced. Moreover, in Figure 15(c) we can see how the distribution of the number of detected regions is shifted to the left, meaning that on average we are detecting fewer regions than before.

The training procedure presented before provides a considerable improvement of the results over the default value of the parameter, but its performance is lower than the manually selected parameter. Looking at both APD we can see that the results are still slightly unbalanced. Please note that the optimization goal was to optimize the number of regions, but not any particular metric. Regarding this supervised procedure, two more points need to be remarked. First, the kind of parameter that needs to be learned is very simple, so we do not really need a big number of training images, and the performance of that training is not affected when using only a few images, as opposed to [26], which requires a very complex training process. Second, our optimization goal is not a particular performance metric as in most reference works, but to choose the correct scale. This means that our optimization procedure does not guarantee a decrease in the APD, SPD, or any other performance metric, just in the number of detected regions. This optimization goal makes the results independent of the metrics used. In addition, in Figure 15 we show the distribution of the number of regions detected by human subjects and by the proposed algorithms. From this figure we can conclude that each version

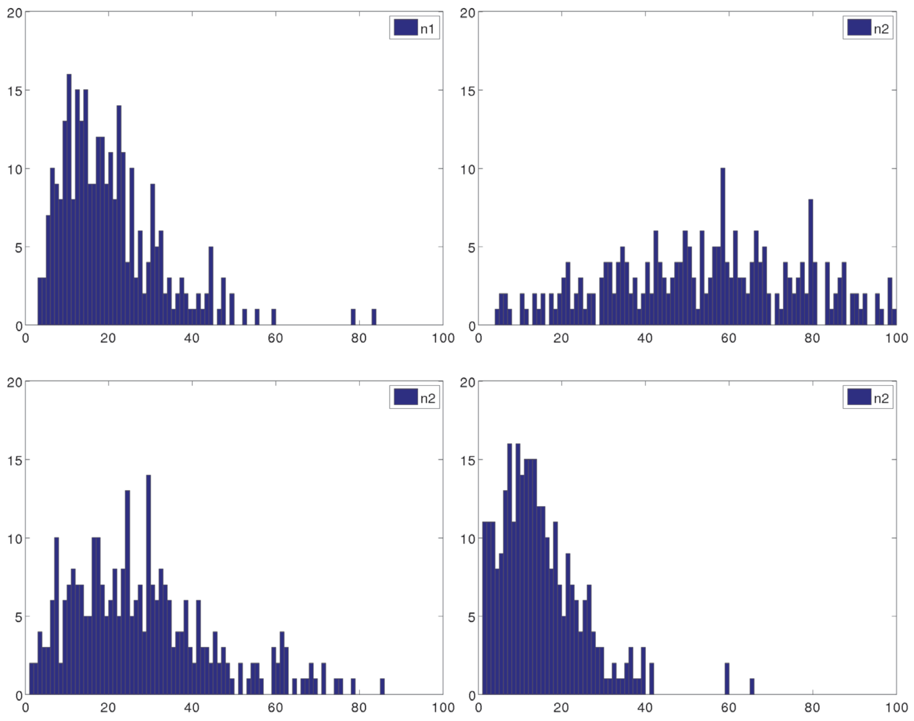

Figure 15. *Distribution of the number of detected regions over the database. From left to right, and from top to bottom: human detected regions, result of the unsupervised (MP-D) algorithm, result of the supervised (MP-S) algorithm, result of the manually tuned (MP-M) algorithm.*

of the algorithm improves the previous ones in terms of detected regions and that the MP-M achieves the distribution most similar to the human subjects.

Although the effort of quantifying the results of the algorithms is praiseworthy and constitutes a good step towards the objective evaluation of scientific work, we should bear in mind that every set of metrics has its drawbacks and they are still far from our subjective perception. Overoptimizing the algorithms with respect to a particular metric is a dangerous path to follow and could lead to results which are far from human perception. In the particular case of this algorithm, the best numeric results are obtained with $\alpha = 150$ (see Table 6); however, from a subjective point of view we think that those results are not as good as those obtained with $\alpha = 50$. In our experience this is because we tend to penalize subsegmentation more than oversegmentation, and thus the results using $\alpha = 50$ are more pleasant to the human eye.

Finally, when comparing with human made segmentation, caution should be exercised, because there are fundamental differences among the processes of segmenting an image by a human observer and a machine. The human segmentation uses a higher level of abstraction than our low-level tools, and thus the latter have a natural limit on their performance, which explains the differences we see with the human subjects.

Table 7

Quantitative evaluation of the proposed algorithms over a selected subset of the LHI database, using performance metrics from Cardoso and Corte-Real [12].

Alg.	SPD	$APD(P,Q)$	$APD(Q,P)$	MPD
MP-D	0.62	0.61	0.08	0.07
MP-S	0.55	0.52	0.10	0.08
MP-M	0.50	0.46	0.18	0.15

6.6. Other datasets. In order to extend our results we conducted the same validation procedure in the LHI database. One important remark about this database is the fact that it was designed for object recognition, and the human segmentations provided are of a higher level of abstraction; i.e., objects are marked as persons, cars, etc. In Table 7 we show the results of the proposed algorithms on 75 images from the following LHI subsets: Badminton, Highway, Ibis, Polo, Snowboarding, Street. The obtained results are quite good in quality, although the scale is generally finer than the ground truth segmentations. In the case of the MP-S algorithm, we did not train the algorithm again but used the parameter obtained on the BSDS300. The first observation about these results is that all the conclusions extracted from the BSDS are still valid here, namely, the oversegmentation shown by the MP algorithm, and the further improvements provided by the MP-S and MP-M algorithms, which present more balanced results. The second observation is that the performance is lower than in the BSDs. This is coherent, since the human segmentations provided by the LHI database are of a higher level of abstraction, which in general corresponds to coarser partitions than those from the BSDS.

6.7. Comparison. As we stated before, there are very few published algorithms presenting quantitative results. The first work on this area which presented an extensive evaluation was [26], followed by [2, 39]. These three works use the BSDS database and the precision-recall framework, which is good because it allows the community to compare results from a quantitative perspective. However, this comparison is performed from a boundary detector oriented perspective, and the results are not well suited to benchmark partitioning algorithms. The objections raised to this approach are discussed in detail in [12]. In [39] they also use the probabilistic Rand index (PRI) and the variation of information (VOI) to measure the quality of the results. In addition, [6] shows quantitative results on a custom database (not publicly available at the moment of writing) using the metrics from Cardoso and Corte-Real [12].

In addition, in [3, 33] the authors also did the great work of running the benchmarks for some state of the art algorithms:

- FH: "'Efficient Graph Based Segmentation," Felzenszwalb and Huttenlocher [18].
- CM: "Mean shift," Comaniciu and Meer [14].
- NC: "Normalized Cuts," Shi and Malik [35].
- MS: "Multiscale algorithm for image segmentation," Koepfler, Lopez, and Morel [23].
- WS: "Geodesic saliency of watershed contours and hierarchical segmentation," Najman and Schmitt [32].
- MFM: "Learning natural image boundaries," Martin, Fowlkes, and Malik [26] and Canny [8].

Those authors have made their code or binaries publicly available, but no quantitative results are given. Note that they had to tune the parameters of the algorithms manually to perform a fair comparison. We also want to comment on the fact that many researchers are comparing with NC, FH, and CM, since they have made their codes and/or results available, although the results are not always the state of the art.

Even counting the aforementioned results, the number of algorithms with quantitative evaluation amounts to a couple of dozen, counting this work, which is clearly insufficient for a broad area like low-level image segmentation.

Finally, partition-based performance measures are rarely used, so the results are hard to compare in this sense. We hope that this work can contribute to spreading these kinds of evaluation metrics and that we see works using them in the future more.

For the aforementioned reasons, in this section we show the comparison using Cardoso's metrics only for two algorithms: texture and boundary encoding-based segmentation (TBES) [33] and ultrametric contour maps (UCM) [2, 3]. We have chosen these algorithms because they have state of the art results and they provide a means to reproduce their results. In order to benchmark these algorithms, we ran the code provided by the authors with the parameters provided by them. The results of such a comparison are shown in Table 8, and they show that our algorithm presents a comparable (and slightly better) performance with the reference works.

The work of Guigues is very similar in spirit to our algorithm, and thus it is important to compare with his approach. We did our best to compare in a quantitative way, but at the moment of writing we did not have access to a suitable executable/source code to run the benchmarks.[3] For these reasons we show in Figure 11 only a qualitative comparison for a few random images taken from the BSDS500; the only criterion used to select them was to choose one image from each different category (city, animals, people, objects). The current implementation of the scale-sets representation uses a piecewise constant MS model, using the three RGB channels as features. As this figure shows, Guigues has the same problem in slowly varying intensity regions, where they appear fragmented due to the piecewise constant model. This can be seen in the sky of the *city* image, where some salient boundaries are detected. In addition, some salient boundaries are detected in the background *bird* image which are more salient than the bird itself, due to the camouflage effect.

These results show that the quality of the hierarchy obtained by the MP algorithm is better than that from Guigues, because the most salient boundaries better match the structures present in the images.

6.8. Alternative performance metrics. Although we believe that the metrics presented by Cardoso and Corte-Real [12] are the best choice for the evaluation of low-level segmentation algorithms, we faced the fact that very few works used them to present results. For that reason, and to ensure reproducibility, we present here performance results using the metrics proposed by Martin [25]. In that work, two sets of metrics are introduced: region based

[3]The authors pointed us to the open source Seascape software [37], which has a graphical user interface which runs Guigues's algorithm. However, it was not possible to run it in batch mode in order to compute the results for the whole database. We also tried to develop our own executable based on Seascape code but without luck.

Table 8

Quantitative comparison of algorithms over the BSDS using performance metrics from Cardoso and Corte-Real [12].

Results over the 100 test images				
Alg.	SPD	$APD(P,Q)$	$APD(Q,P)$	MPD
MP	0.56	0.54	0.10	0.08
GREEDY	0.58	0.54	0.13	0.10
MP-S	0.46	0.41	0.13	0.09
MP-M ($\alpha = 50$)	0.41	0.31	0.19	0.11
MP-M ($\alpha = 150$)	0.38	0.18	0.28	0.09
UCM	0.39	0.32	0.14	0.09
TBES	0.41	0.34	0.17	0.11

Table 9

*Quantitative evaluation of the proposed algorithms over the BSDS*300 *using the F-measure.*

Results over the 100 test images of the BSDS300			
Alg.	F-meas	P	R
Human	0.79	X	X
MP	0.64	0.736	0.564
GREEDY	0.544	0.772	0.420
MP-S	0.624	0.606	0.644
MP-M	0.590	0.505	0.710

and boundary based. From these metrics, we want to highlight the F-measure, which is the most widely used at the moment. The definition of the F-measure [25] relies in turn on the definition of two quantities: *precision* and *recall*. Precision (P) is the probability that a machine-generated boundary pixel is a true boundary pixel. Recall (R) is the probability that a true boundary pixel is detected. The F-measure is a way to summarize both quantities in a unique performance metric, and it is computed as the harmonic mean of P and R. The F-measure is bounded between 0 and 1, and values closer to 1 mean better performance. Note that precision and recall are boundary-based metrics, and thus they are not well suited (nor are they the F-measure) to evaluate region-based algorithms.

The results using the aforementioned metrics are shown in Table 9. In spite of some differences, these results confirm the conclusions obtained from our first evaluation. First, they confirm that the MP algorithm clearly outperforms the GREEDY one. And second, they confirm that the MP-S algorithm improves MP performance and gives more balanced results in terms of FP and FN.

Regarding the supervised procedure, as we explained in section 6.5, it is not tuned to any particular metric, which could explain why our MP-S algorithm does not improve the F-measure obtained by the original MP algorithm.

In order to extend our comparison, we compare our algorithms with the state of the art using the F-measure. In Table 10 we present an excerpt of results presented in [3, 33] with the addition of our results. Note that c-UCM and g-UCM refer to the canny-owt-UCM and the gPb-owt-UCM algorithms presented in [3]. We report here only the F-measure for the optimal dataset scale (ODS), because some of the precision-recall values from the reference works are not available.

Table 10

*Quantitative comparison of the proposed algorithms with the state of the art. Results computed over the test BSDS*300 *subset using the F-measure (ODS). Results were taken from Figure* 4 *of* [2] *and Table* 1 *of* [3].

Alg.	F-meas	Alg.	F-meas	Alg.	F-meas
Human	0.79	Results from [2]		Results from [3]	
MP	0.639	UCM	0.67	c-UCM	0.58
GREEDY	0.544	MS	0.63	g-UCM	0.71
MP-S	0.624	WS	0.62	CM	0.63
MP-M	0.590	MFM	0.65	FH	0.58
TBES [33]	0.645	canny	0.58	NC	0.62

As these results show, our algorithm has a performance similar to that of the other state of the art algorithms, measured with the F-measure. Some algorithms have a significantly better performance than the MP algorithm.

6.9. Multiscale evaluation.

So far we have shown single scale results using the PD metrics for a selected number of scales. In this section we show a multiscale evaluation based on these metrics, in the same spirit as the BSDS multiscale evaluation. In Figure 16 we show the results of varying the α parameter in the range $[6, 200]$ for the BSDS300 database. As this figure shows, with the smaller parameter choice we have an oversegmentation (high $APD(P, Q)$, low $APD(Q, P)$), as the parameter changes the balance of FP vs. FN towards an undersegmentation. This supports our claim that α behaves as a scale parameter.

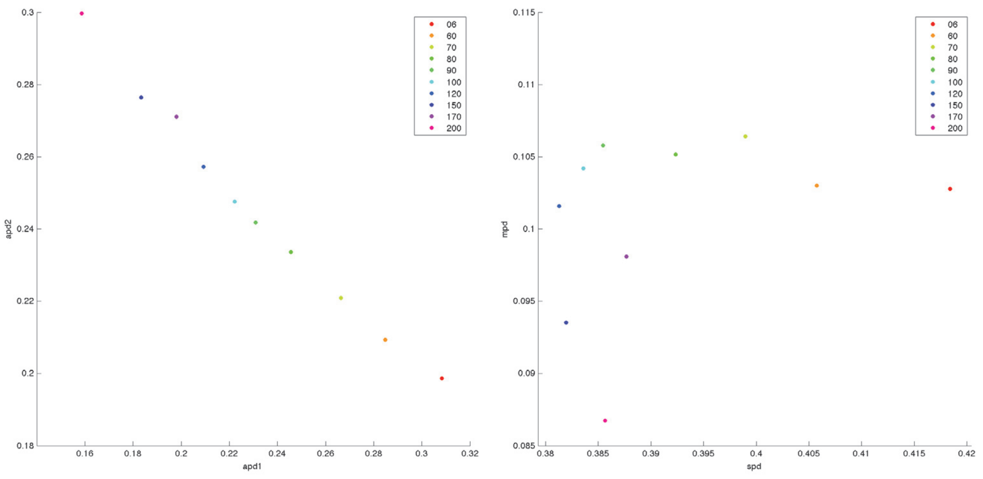

Figure 16. *Results of the region-based multiscale evaluation proposed in this work, using the BSDS*300 *database. Top: APD(P,Q) vs. APD(Q,P). Bottom: SPD vs. MPD.*

6.10. Results over the BSDS500 database.

In this section, we extend the evaluation to a third dataset, the 200 test images from the BSDS500 database. These images are new (not included in the BSDS300), and no training or tuning of the algorithms was carried out in this database. We use the exact configuration tested in the BSDS300. This evaluation was computed with the new version of the Berkeley Benchmark, which now includes the

Table 11

Quantitative evaluation of the proposed algorithms over the BSDS500 using Berkeley-boundary based measures.

Results over the 200 test images of the BSDS500							
Alg.	ODS ($\alpha = 10.8$)			OIS			
Alg.	P	R	F-meas	P	R	F-meas	AP
MP	0.70	0.61	0.65	0.72	0.66	0.69	0.51

computation of some region-based quantities (PRI, VOI, Covering).

The boundary-based results are shown in Table 11. These results confirm what we previously commented about the F-measure; i.e., it is unfair with region-based approaches. It is also interesting to note that the chosen ODS corresponds to $\alpha = 10.8$, which is very different from the chosen scale to optimize the PD metrics $\alpha = 150$. In Figure 17 we show the results of the boundary-based multiscale benchmark of Berkeley. These results confirm that α acts as a scale parameter: when α is small, the algorithm shows high precision and low recall values, which corresponds to a segmentation with a big number of FN and a low value of FP; when α increases the balance leans towards oversegmentation, i.e., precision falls and recall rises. Finally, to complete the boundary-based evaluation, in Table 12 we compare with the reference works using the F-measure on the BSDS500. As the results show, we obtain state of the art results, surpassed only by Arbeláez et al. [4]. Note that the difference in performance between Arbeláez's algorithm and ours is bigger when quantified with the F-measure, which is explained by the fact that Arbeláez's algorithm is trained with the F-measure as the optimization goal. As our algorithm uses an initial pruning, the $R = 0$ value is never attained, and the same happens with $P = 0$; thus the P-R curve is not *complete*. This means that the measured AP value is not accurate.

In Table 13 we show the results of the Berkeley region-based metrics. In the ODS case, we show the value of α that achieved that result. In Table 14 we compare our results with the available results from [4] using the Berkeley region-based metrics. As these results show, our algorithm obtains good results, surpassed only by the work of Arbeláez et al. [4], which is significantly better. For the rest of the compared algorithms we obtain better values for almost every metric.

6.11. Fixed number of regions. As we stated before, our approach gives us the flexibility of specifying the number of desired regions beforehand. This could be particularly useful in a number of applications, such as skin lesion detection, were we wanting to detect one single mark in the skin [10]. In Figure 18 we show the result of such an approach for the detection of melanomas in dermatological images. As this figure shows, the original algorithm gives the correct number of regions in some cases and fails in some others. In these cases the fixed region variant helps solve the problem.

6.12. Summary of experimental results. We evaluated our algorithm in a very exhaustive way, using many different sets of metrics and various databases. From all the results we can conclude that our algorithm presents good results, which are comparable with the state of the art. However, the results of Arbeláez et al. [4] are better than ours. This is explained by various factors, but mainly by the fact that we focused on the model of partition selection developed

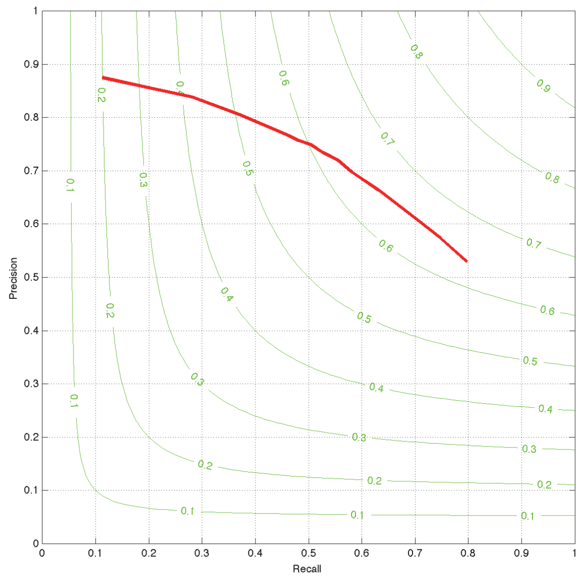

Figure 17. *Results of the boundary-based multiscale evaluation from Berkeley* [4]*. The red curve depicts the behavior of the F-measure of the MP algorithm as the scale parameter α varies in the range $[0.01, 5000]$. The curves in green are isocontours of the F-measure.*

Table 12

Quantitative evaluation of the proposed algorithms over the BSDS500 using boundary-based measures. The results of the related works were extracted from Table 1 of [4]*.*

Results over the 200 test images of the BSDS500			
Alg.	F-meas		AP
	ODS	OIS	
Human	0.79	0.79	X
gPb-owt-ucm	0.73	0.76	0.73
MP	0.65	0.69	0.51
CM	0.64	0.68	0.56
NC	0.64	0.68	0.56

Table 13

Quantitative evaluation of the MP algorithm over the BSDS500 using Berkeley region-based measures.

Results over the 200 test images of the BSDS500						
	ODS			OIS		
	GT	PRI	VOI	GT	PRI	VOI
Metric value	0.52	0.79	1.87	0.58	0.83	1.65
α values	80	20	300			

and we did not tune the ingredients. Choosing a better initial segmentation, a different set of features (we use only color information), a better region model (we use a piecewise constant one), and a better boundary detector, we could improve our results considerably. In Arbeláez's work, all these factors were carefully chosen. In addition, these choices and the values of the

Table 14

Quantitative evaluation of the MP algorithm over the BSDS500 using Berkeley region-based measures. The results of the related works were extracted from Table 2 of [4].

Results over the 200 test images of the BSDS500							
Alg.	Covering			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
Human	0.72	0.72		0.88	0.88	1.17	1.17
MP	0.52	0.58	0.68	0.79	0.83	1.87	1.65
gPb-owt-ucm	**0.59**	**0.65**	**0.74**	**0.83**	**0.86**	**1.69**	**1.48**
CM	0.54	0.58	0.66	0.79	0.81	1.85	1.64
FH	0.52	0.57	0.69	0.80	0.82	2.21	1.87
Canny-owt-ucm	0.49	0.55	0.66	0.79	0.83	2.19	1.89
NCuts	0.45	0.53	0.67	0.78	0.80	2.23	1.89
Quad-Tree	0.32	0.37	0.46	0.73	0.74	2.46	2.32

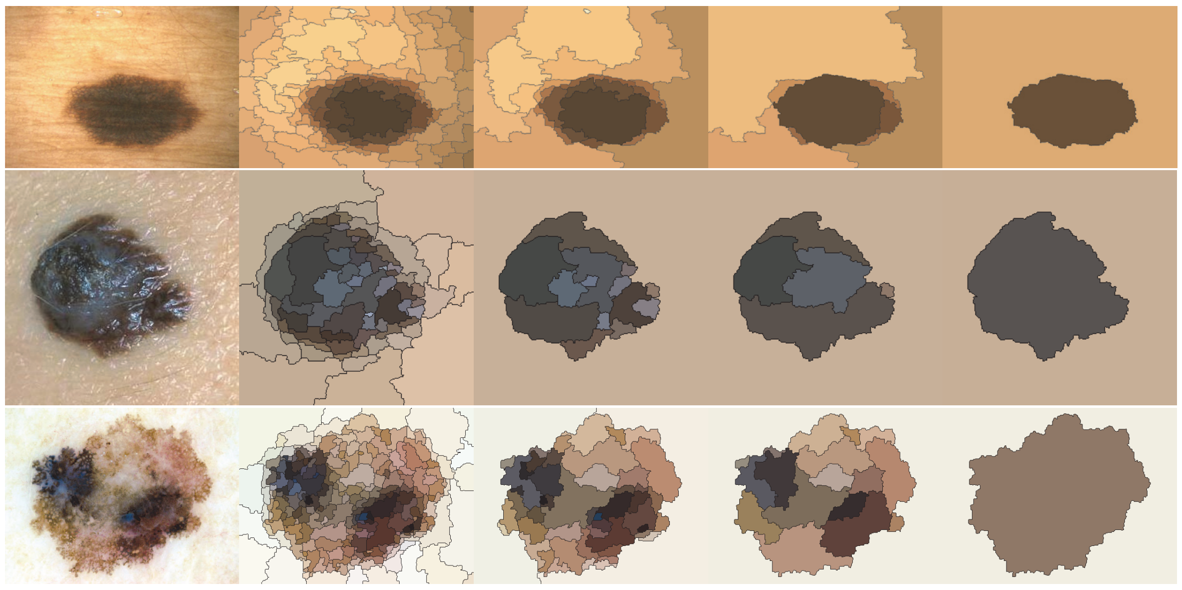
Figure 18. *Results of three of the proposed algorithms over skin lesion images. From left to right: original image, initial partition, result of the MP algorithm, result of the MP-S algorithm, result of the algorithm imposing a 2-region constraint. These images were provided courtesy of Dr. Pablo Musé.*

parameters were made by training over the train subsets of the BSDS, whereas we did not use any training process.

7. Implementation.

Our algorithms are implemented in C and C++, with automatic code optimization turned on. The execution times of the algorithms depend on image content; they have a strong dependency on the size of the initial partition and a weak dependency on the size of the image. In Figure 19 we show the distribution of the execution times over the BSDS300 database. The average times and their standard deviations are shown in Table 15. These times were computed on an Intel(R) Core(TM)2 Duo CPU E7300 @ 2.66GHz, with 3Gb RAM, which was a standard desktop computer in 2008. The first conclusion is that the family of developed algorithms has reasonable execution times, approximately 5 seconds in average. However, with a careful implementation these times could be considerably lowered. Second, the different variants of the MP algorithms (obtained by varying the scale parameter) do not change the execution time, which is coherent because the main computational effort is invested in the construction of the tree and the computation of the NFA. This is the opposite behavior of many merging algorithms where the execution time depends on the amount of merging performed, which in turn depends on the value of the scale parameter. Third, the execution time of all the algorithms depends only on the size (N_i) of the initial partition, as predicted by our formulation in sections 3.7 and 5.4.

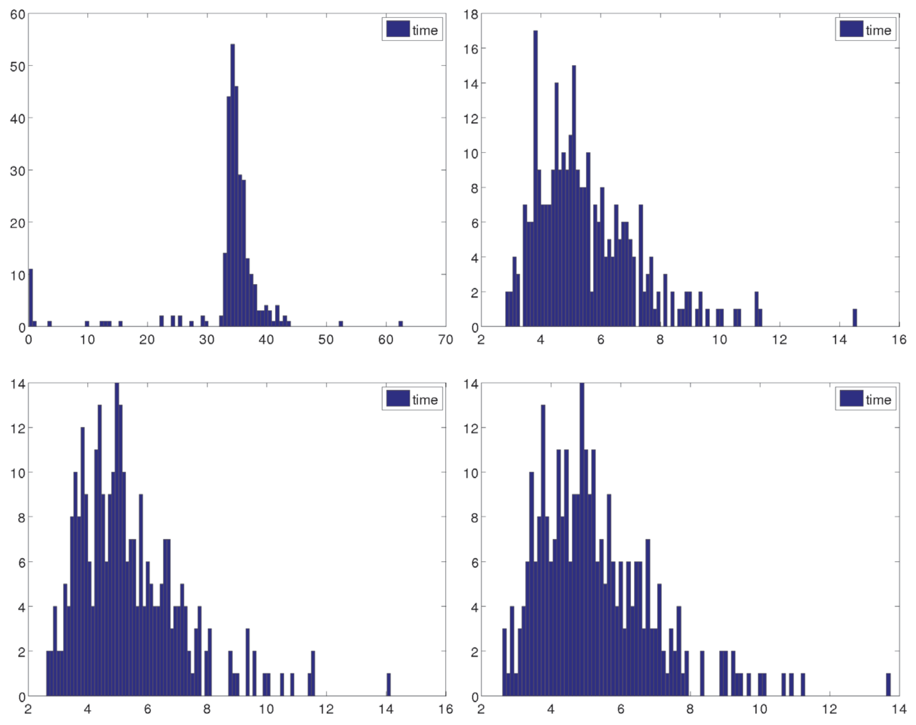

Figure 19. *Distribution of the execution times (in seconds) over the BSDS300 database for the different algorithms presented. From left to right, and from top to bottom: GREEDY, MP, MP-S, MP-M.*

Table 15

Execution times for the different versions of the algorithms on the complete BSDS300 database. In each case N_i is the average number of regions of the initial partition used. μ and σ are the corresponding mean and standard deviation of the execution times (in seconds).

Alg.	N_i (regions)	μ (secs.)	σ (secs.)
GREEDY	605	14.14	3.74
MP-D	222	5.50	1.71
MP-S	222	5.37	1.74
MP-M	222	5.27	1.66

8. Conclusions.

In this work we presented a method for image segmentation based on a hierarchy of partitions of an image. We introduced an a contrario statistical model based on the combination of region- and boundary-based descriptors, which allows us to validate the partitions present in the hierarchy. In this way, we reduce the number of free parameters and alleviate the problem of local minima shown by greedy algorithms. The results obtained by our algorithm are comparable to those presented in the literature. We have conducted an extensive quantitative evaluation using many different performance metrics and two publicly available datasets.

The main drawback of our approach is the underestimation of the NFA, which leads our algorithm to oversegment images. We have addressed this problem using an example-based training process to adjust the optimal scale for a given database. However, part of the problem is still there, and we need to address this in a future work. Another point that needs further work is the way we include boundary information in the process, because we are currently using a simple version of the boundary detector of Desolneux, Moisan, and Morel [15]. Thus, we could improve this by modifying the boundary detector to introduce all the improvements presented in [9], which would help to remove the apparent boundaries found by our algorithm.

The main focus of this work was to present a generic partition validation approach, which has proven to be useful for the segmentation tasks. However, our results are conditioned by the quality of the algorithm used for the initial segmentation (MS in this case) and the features used to describe regions (color information). In addition, our boundary detection term is somehow weak, as mentioned before. In this sense, there is still room for improvement in many ways. In the first place, we could use the gPb contour detector presented in [4] to improve the initial segmentation, and we could also add texture information to the region descriptors.

We also want to test our framework with three-dimensional and general multivalued images (to deal with textured images). While our model is in fact independent of the dimension of the image domain and of the pixel dimension, and we have some preliminary results, we still need extensive experimentation in these cases.

It would also be interesting to add a summarizing quantity to the PD metrics analogous to the F-measure, which could allow us to compute a single score for a segmentation in our benchmark. Finally, the computational cost of the partition selection mechanism needs to be improved, which will make the algorithm useful in more cases.

Appendix A. Complexity analysis.

In this appendix we will analyze the complexity of the dynamic programming scheme used to update the NFA tables. We will analyze what we believe are the most extreme cases with respect to complexity, which are dependent on the structure of our binary tree. Those cases are a completely degenerated or a completely balanced tree. This is not a formal proof but merely a conjecture. The hypothesis behind this reasoning is that the balanced tree is the worst case with respect to computational cost and the degenerated tree is the best case. We will discuss this assumption at the end of this section.

Given a node A, of a tree T, let us define T_A as the subtree which has A as root. Suppose that we want to fill the NFA table of a node A, which will be denoted as $table_A$, which has children B and C (see Figure 20). Suppose that we have already computed both tables

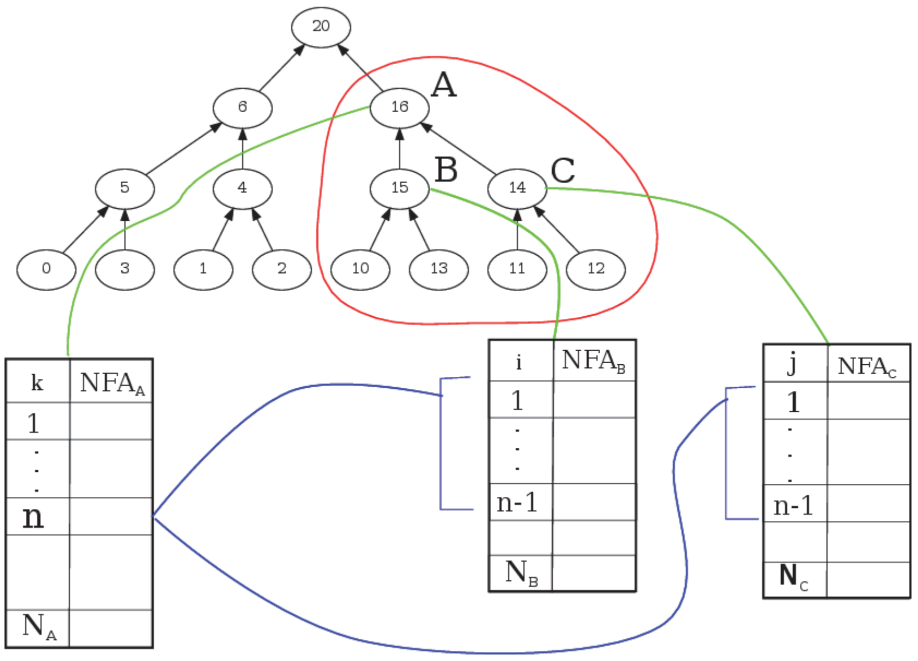

Figure 20. *Example computation of the NFA.*

corresponding to the subtrees T_B and T_C. Given a node A, the number of possible partitions that span is bounded by $[1, N_A]$, where N_A is the number of leaves T_A. We will compute this number later.

A.1. Computation of the NFA table for a node. We need to fill a table of the NFA of N_A entries. To compute the NFA at level n, we will explore all the NFA values at levels i, j such that $i + j = n$, taking i from $table_B$ and j from $table_C$. In the following we will compute the number of possible ways to do that, denoted as $f_A(n)$.

Remember that $N_A = N_B + N_C$. Without losing generality, we can assume that $N_B < N_C$ and we will split the range of N_A into three ranges:

- $n \leq \min(N_A, N_B)$. In this case the number of combinations is $f_A(n) = n - 1$.
- $n > \min(N_A, N_B)$ and $n \leq \max(N_A, N_B)$. In this case, after we use all the values of N_B we do not have more options, so $f_A(n) = N_B$.
- $n > \max(N_A, N_B)$. This case is the same as the previous. $f_A(n) = N_B$.

Thus, the total number of computations needed to fill the table is

$$F_A = \sum_{n=1}^{N_A-1} f_A(n) = \sum_{n=1}^{N_B-1} (n-1) + \sum_{n=N_B}^{N_A-1} N_B \\ = \frac{(N_B-1)N_B}{2} + (N_A - 1 - N_B)N_B \\ = \frac{(N_B-1)N_B}{2} + (N_C - 1)N_B. \tag{18}$$

The numbers N_A, N_B, N_C depend on the structure of the tree, but there are two extreme cases: the degenerate tree and the balanced tree. In the degenerate case, $N_B = 1$ and

$N_C = N_A - 1$; on the other hand, in the balanced case, $N_B = N_C = \frac{N_A}{2}$. Substituting this into (18) we obtain the two boundary values for the number of cases:

$$F_A = \begin{cases} N_C - 1 = N_A & \text{if } T_A \text{ is degenerated,} \\ \frac{3}{8}N_A^2 - \frac{1}{4}N_A & \text{if } T_A \text{ is balanced.} \end{cases} \tag{19}$$

A.2. Computation for every node in the tree. In order to carry on the computation, we need to add up the cost of computing the table for every node; that is,

$$\sum_{A \in T} F_A. \tag{20}$$

This of course depends on the structure of the tree, so it can be counted in this form. However, if we take into account that the complexity depends on N_A and on the structure of the tree, we can rewrite this equation in a more compact way if we arrange the nodes by the type of their subtrees:

$$\sum_t F_A(t).\mathcal{N}(t), \tag{21}$$

where t is a type of subtree and $\mathcal{N}(t)$ is the number of those subtrees. Of course this notion is vague, and cannot be counted, but in certain cases we could do it. For instance, if the tree is balanced, every possible subtree is also balanced; the same is true if the tree is degenerated. The important thing to be noted is that the complexity depends on the number of that type, the size of that type, and the F_A of that type. Thinking in that way, the balanced tree is the worst case for two (out of three) of those quantities. For example, this tree has the minimum number of different sizes of subtrees, the biggest number of nodes per level, and the highest F_A per node.

One possible way to count this number is classifying nodes by their height h. We have a total of $\log(n)$ height values, and for each subtree of height h there are $\mathcal{N}(h) = 2^{-h}$ nodes, and each A node has $N_A = 2^h$ leaves. Summarizing, in the balanced case, the number of operations is

$$\sum_{h=1}^{\log(N/2)} F_{A_h}.\mathcal{N}(A_h), \tag{22}$$

where N is the total number of nodes of the tree. The key observation is that F_A and N_A in these cases depend only on h, because all the subtrees have the same structure:

$$\sum_{h=1}^{\log(N/2)} \left(\frac{3}{8}N_h^2 - \frac{1}{4}N_h\right)\mathcal{N}(h). \tag{23}$$

Taking into account that $N_h = 2^h$ and $\mathcal{N}(h) = N/2^{h+1}$ the total number of operations yields

$$\mathcal{C}(N) = \Theta(N^3). \tag{24}$$

The main reason for the high complexity of this part is due to the fact that we maintain a table of all possible partitions for each node, which is far more complex than keeping only the best partition for each node.

Acknowledgments. We thank J. Cardoso for kindly sharing and explaining the code of his algorithm to us. We thank P. Arbeláez for sharing his benchmarks and results with us. We thank L. Guigues for pointing us to the Seascape software, where his algorithm runs. We also thank Starlab's staff, in particular A. Soria and A. Albajes-Eizaguirre, for helping with the Seascape code.

REFERENCES

[1] N. Alon and S. Onn, *Separable partitions*, Discrete Appl. Math., 91 (1999), pp. 39–51.

[2] P. Arbelaez, *Boundary extraction in natural images using ultrametric contour maps*, in Proceedings of the IEEE Workshop on Perceptual Organization in Computer Vision, 2006.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, *From contours to regions: An empirical evaluation*, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2009, pp. 2294–2301.

[4] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, *Contour detection and hierarchical image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 33 (2011), pp. 898–916.

[5] N. Burrus, T. M. Bernard, and J. M. Jolion, *Image segmentation by a contrario simulation*, Pattern Recognition, 42 (2009), pp. 1520–1532.

[6] F. Calderero and F. Marqués, *General region merging approaches based on information theory statistical measures*, in Proceedings of the IEEE International Conference on Image Processing, 2008, pp. 3016–3019.

[7] F. Calderero and F. Marques, *Region merging techniques using information theory statistical measures*, IEEE Trans. Image Process., 19 (2010), pp. 1567–1586.

[8] J. F. Canny, *A computational approach to edge detection*, IEEE Trans. Pattern Anal. Mach. Intell., 8 (1986), pp. 679–698.

[9] F. Cao, P. Musé, and F. Sur, *Extracting meaningful curves from images*, J. Math. Imaging Vision, 22 (2005), pp. 159–181.

[10] G. Capdehourat, A. Corez, A. Bazzano, R. Alonzo, and P. Musé, *Toward a combined tool to assist dermatologists in melanoma detection from dermoscopic images of pigmented skin lesions*, Pattern Recognition Lett., 32 (2011), pp. 2187–2196.

[11] J. Cardelino, M. Bertalmio, V. Caselles, and G. Randall, *A contrario hierarchical segmentation*, in Proceedings of the IEEE International Conference on Image Processing, 2009, pp. 4041–4044.

[12] J. S. Cardoso and L. Corte-Real, *Toward a generic evaluation of image segmentation*, IEEE Trans. Image Process., 14 (2005), pp. 1773–1782.

[13] V. Caselles, B. Coll, and J. Morel, *Topographic maps and local contrast changes in natural images*, Int. J. Comput. Vis., 33 (1999), pp. 5–27.

[14] D. Comaniciu and P. Meer, *Mean shift: A robust approach toward feature space analysis*, IEEE Trans. Pattern Anal. Mach. Intell., 24 (2002), pp. 603–619.

[15] A. Desolneux, L. Moisan, and J-M. Morel, *Edge detection by Helmholtz principle*, J. Math. Imaging Vision, 14 (2001), pp. 271–284.

[16] A. Desolneux, L. Moisan, and J. M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, Springer, New York, 2008.

[17] S. Di Zenzo, *A note on the gradient of a multi-image*, Comput. Vis. Graph. Image Process., 33 (1986), pp. 116–125.

[18] P. Felzenszwalb and D. Huttenlocher, *Efficient graph-based image segmentation*, Int. J. Comput. Vis., 59 (2004), pp. 167–181.

[19] R. Grompone and J. Jakubowicz, *On computational Gestalt detection thresholds*, J. Physiol., 103 (2009), pp. 4–17.

[20] L. Guigues, J.-P. Cocquerez, and H. Le Men, *Scale-sets image analysis*, Int. J. Comput. Vis., 68 (2006), pp. 289–317.

[21] L. Igual, J. Preciozzi, L. Garrido, A. Almansa, V. Caselles, and B. Rouge, *Automatic low baseline stereo in urban areas*, Inverse Probl. Imaging, 1 (2007), pp. 319–348.

[22] J. J. Koenderink, *The structure of images*, Biol. Cybernet., 50 (1984), pp. 363–370.

[23] G. Koepfler, C. Lopez, and J. M. Morel, *A multiscale algorithm for image segmentation by variational method*, SIAM J. Numer. Anal., 31 (1994), pp. 282–299.

[24] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman, New York, 1982.

[25] D. Martin, *An Empirical Approach to Grouping and Segmentation*, Ph.D. thesis, EECS Department, University of California, Berkeley, Berkeley, CA, 2003.

[26] D. Martin, C. Fowlkes, and J. Malik, *Learning to detect natural image boundaries using local brightness, color, and texture cues*, IEEE Trans. Pattern Anal. Mach. Intell., 26 (2004), pp. 530–549.

[27] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, in Proceedings of the 8th IEEE International Conference on Computer Vision, 2001, pp. 416–423.

[28] F. Meyer and L. Najman, *Segmentation, minimum spanning tree and hierarchies*, in Mathematical Morphology: From Theory to Applications, L. Najman and H. Talbot,, eds., ISTE-Wiley, London, 2010, pp. 229–261.

[29] J. M. Morel and S. Solimini, *Variational Methods in Image Segmentation*, Birkhäuser, Boston, 1995.

[30] D. Mumford and J. Shah, *Optimal approximations by piecewise smooth functions and variational methods*, Comm. Pure Appl. Math., 42 (1989), pp. 577–685.

[31] L. Najman, *On the equivalence between hierarchical segmentations and ultrametric watersheds*, J. Math. Imaging Vision, 40 (2011), pp. 231–247.

[32] L. Najman and M. Schmitt, *Geodesic saliency of watershed contours and hierarchical segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 18 (1996), pp. 1163 –1173.

[33] S. Rao, H. Mobahi, A. Yang, S. Sastry, and Y. Ma, *Natural image segmentation with adaptive texture and boundary encoding*, in Proceedings of the Asian Conference on Computer Vision, 2009.

[34] P. Salembier and L. Garrido, *Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval*, IEEE Trans. Image Process., 9 (2000), pp. 561–576.

[35] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22 (2000), pp. 888–905.

[36] P. Soille, *Constrained connectivity for hierarchical image partitioning and simplification*, IEEE Trans. Pattern Anal. Mach. Intell., 30 (2008), pp. 1132–1145.

[37] N. Teixidó, A. Albajes-Eizagirre, D. Bolbo, E. Le Hir, M. Demestre, J. Garrabou, L. Guigues, J.-M. Gili, J. Piera, T. Prelot, and A. Soria-Frisch, *Hierarchical Segmentation-Based Software for Cover Classification Analyses of Seabed Images (Seascape)*, Tech. Report 431, Marine Ecological Progress Series, Inter-Research Science Center, Oldendorf/Luhe, Germany, 2011.

[38] A. W. Witkin, *Scale space filtering*, in Proceedings of the Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983, pp. 1019–1021.

[39] A. Yang, J. Wright, Y. Ma, and S. Sastry, *Unsupervised segmentation of natural images via lossy data compression*, Comput. Vis. Image Understand., 110 (2008), pp. 212–225.

[40] B. Yao, X. Yang, and S. Zhu, *Introduction to a large scale general purpose ground truth database: Methodology, annotation tool, and benchmarks*, in Proceedings of the 6th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, 2007, pp. 169–183.

[41] S. C. Zhu and A. Yuille, *Region competition: Unifying snakes, region growing and Bayes/MDL for multi-band image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 79 (1996), pp. 12–49.