Chapter 6

A Color Image Segmentation Algorithm

6.1 Introduction

Image segmentation is a essential but critical component in low–level vision, image analysis, pattern recognition, and now in robotic systems. Besides, it is one of the most difficult and challenging tasks in image processing, and determines the quality of the final results of the image analysis. Intuitively, image segmentation is the process of dividing an image into different regions such that each region is homogeneous while not the union of any two adjacent regions. An additional requirement would be that these regions had a correspondence to real homogeneous regions belonging to objects in the scene.

The classical broadly–accepted formal definition of image segmentation is as follows [PP93]. If $\mathrm{P}(\circ)$ is a *homogeneity predicate* defined on groups of connected pixels, then the segmentation is a partition of the set $\mathcal{I}$ into connected components or regions $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ such that

$$\bigcup_{i=1}^{n} \mathcal{C}_i \text{ with } \mathcal{C}_i \cap \mathcal{C}_j = \emptyset\,,\ \forall i \neq j \tag{6.1}$$

The uniformity predicate $\mathrm{P}(\mathcal{C}_i)$ is *true* for all regions $\mathcal{C}_i$ and $\mathrm{P}(\mathcal{C}_i \cup \mathcal{C}_j)$ is *false* when $i \neq j$ and sets $\mathcal{C}_i$ and $\mathcal{C}_j$ are neighbors.

Additionally, it is important to remember here that *the image segmentation problem is basically one of psychophysical perception, and therefore not susceptible to a purely analytical solution*, according to [FM81]. Maybe that is why, literally, there are hundreds of segmentation techniques in literature. Nevertheless, to our knowledge, yet no single method can be considered good for all sort of images and conditions, being most of them created pretty *ad hoc* for a particular task. Despite the importance of the subject, there are only several surveys specific on the image segmentation issue, principally versed on monochrome segmentation [FM81, HS85], giving little space to color segmentation [PP93, LM99]. For more details, Chapter 3 is completely devoted to review the state of the art on the segmentation of color images.

Not until recently has color image segmentation attracted more and more attention mainly due to reasons such as the ones below

- Color images provide far more information than gray–scale images and segmentations are more reliable.
- Computational power of available computers has rapidly increased in recent years, being able even for PCs to process color images.
- Handling of huge image databases, which are mainly formed by color images, as the Internet.
- Outbreak of digital cameras, 3G mobile phones, and video sets in everyday life.
- Improvement in the sensing capabilities of intelligent systems and machines.

Most of the segmentation techniques for monochrome images – histogram thresholding, feature clustering, edge detection, region–based methods, fuzzy techniques, and neural networks – have been extended to segment color images by using RGB color coordinates or some of their transformations, either linear or nonlinear. However, comprehensive surveys on color image segmentation are still scarce in number [SK94, CJSW01].

Work in [SK94] discussed the properties of several color representations and a pretty extensive list of segmentation methods were summarized and analyzed, splitting them into several categories analogous to those already mentioned for gray–scale images. The list of conclusions in that review are worth to be taken into account, specially those saying that

- General purpose algorithms are not robust nor always algorithmically efficient.
- No general advantage in using one specific color space with regard to others has been found.
- Color constancy is needed to improve effectiveness when combining region segmentation with color object recognition.

More recently, the review in [CJSW01] provides an up–to–date summary of color image segmentation techniques available at present, and describes the properties of different kinds of color representation methods and some of the problems encountered when applying those models to segment color images. Some novel approaches such as fuzzy and physics–based methods are discussed as well in that work. There is an interesting taxonomy of methods and color spaces with their description, advantages and disadvantages. For more information about the issue, the reader should refer to Chapter 3.

In order to propose a useful segmentation algorithm that fits our needs, we must say that our choice was among the family of graph–theoretical approaches because of its good mathematical basements and the fact that the segmentation problem is straightforwardly translated into a graph–partitioning problem existing lots of different methods to solve it. Nonetheless, the worst disadvantage of this type of framework is, as can be seen in [WL93, VC93, XU97], that these algorithms are heavy time–consuming, which should prevent us from their application in (nearly) real–time applications. For this last reason, we chose

among the sort of greedy graph–partitioning algorithms, faster than any other one method in that family, as observed in [FH98a].

In this Chapter we present our color image segmentation algorithm that is capable of working on diverse color spaces and metrics. This approach has a nearly linear computational complexity and is based on that in [FH98a] along with a set of improvements, both theoretical and practical, which amend the lacks detected in former results. This algorithm has been successfully applied to segmenting both static images and sequences, where some further enhancements were introduced to achieve more coherent and stabler segmentations of sequences.

Finally, in this Chapter some results are provided whose aim is to test the performance of our segmentation in comparison not only with the results attained by the original algorithm in [FH98a], which has been improved, but also with those obtained by the unsupervised clustering *Expectation–Maximization* (EM) algorithm by Figueiredo [FJ02]. EM is one of the most successful clustering methods in recent years[1], and Figueiredo's version is completely unsupervised, which avoids the problem of selecting the number of components and does not require a careful initialization. Besides, it overcomes the possibility of convergence toward a singular estimate, a serious problem in other EM–like algorithms. We show that our segmentations are fully comparable to those of the Figueiredo's EM algorithm, but at the same time and more importantly, our algorithm is far faster.

6.2 Outline of the Chapter

Next, we summarize the main aspects discussed in each Section of the Chapter. In Section 6.3 we condense the most related former works dealing with the image segmentation problem using a graph–theoretical approach. Section 6.4 is devoted to extensively analyzing our color segmentation algorithm. Our approach has been enlarged to cope with sequences in Section 6.5. Thereafter, in Section 6.6, we reinforce our previous statements with numerous example of image segmentations of static images and sequences, comparing them with those obtained employing other image segmentation algorithms. Finally, Section 6.7 encompasses our conclusions about the work carried out in this Chapter.

6.3 Related Previous Work

An important set of techniques to segment images are those based on graph theory. The main idea consists in building an image representation employing a graph and then applying some graph–theoretical techniques to obtain homogeneous connected components which represent regions in the segmented image. An additional advantage of using graphs is that region–based and edge–based segmentation are dual problems, being able to achieve close contours from the segmentation of regions without any further treatment on the image.

[1] Another extremely interesting clustering algorithm usually applied to the image segmentation problem is the one based on the *mean–shift* transformation [CM97, CM99, CM02]. While this one is nonparametric, EM is a parametric method that provides, as a result, a finite mixture of Gaussian distributions.

Two different groups of methods can be considered depending on the technique employed. On the one hand, there exist all those methods that partition a graph describing the whole image into a set of subgraphs, where there is one component for each image region. Algorithms differ in the particular way of removing superfluous edges. Next, some graph–partitioning approaches are briefly described.

The most efficient graph–based algorithms use fixed thresholds and purely local measures to find regions. For instance, the approach in [Zah71] is based on breaking larger edges in a minimum spanning tree of the graph. The inadequacy of removing larger edges is apparent because edge weights within high variability region tend to be larger than in any other region. This work also developed several heuristics to address such issues by using models of the distributions of weights.

A more recent method is that in [WL93] based on the computation of the *minimum cut* in the graph representing an image. Originally, this kind of algorithms were used to solve problems of maximum flow between two points – the source and the drain – connected by paths with a constrained flow capacity, e.g., water or electric networks. In the case of images, capacities account for the similarity between components and node connectivity represents pixel neighborhoods. Therefore, the *cut* criterion is designed to minimize the similarity between regions that are being split.

This kind of segmentation captures *nonlocal* properties of the image but requires more than nearly linear time, in contrast with more efficient methods described bellow that just employ local information. Other refinements based on spectral partitioning techniques can be found in [SM97, SBLM98], where a normalized version of the minimum cut is computed. For a wider review on these sort of approaches, we refer the reader to [Els97, Fja98].

Another algorithm proposed in [Urq97] uses a measure of local variability to decide which edge to remove from the graph. This measure is based only on the nearest neighbors for each point. When this criterion is applied to segmentation problems, it is claimed that the nearest neighbors alone are not enough to get a reasonable measure of the whole image variability since they only capture local properties of the image. This issue is tackled in [FH98a], as will be seen later.

The interesting graph–theoretical work in [Wan98] presents a method to segment images into partitions with connected components by using computationally inexpensive algorithms for probability simulation and simulated annealing, such as that of Hastings's and the generalized Metropolis algorithm. In order to reduce the computational burden, a hierarchical approximation is proposed, minimizing at each step a cost function on the space of all possible partitions into connected components of a graph.

Finally, there are a number of methods that employ more sophisticated models, such as those based on Markov Random Fields (e.g., [GG84]). However, these methods tend to be quite inefficient in terms of time. In our opinion, the two main goals for an image segmentation algorithm are to capture nonlocal properties of the image and to be efficient to compute, and those algorithms are far too time–consuming for our purposes.

On the other hand, there is another set of graph–based algorithms that takes advantage of *region–growing* methods, being the growing process driven by the attributes of nodes and edges. Thus, edges are aggregated forming a list of connected nodes, which likewise form an image component. Edges are selected in

such a manner they provide homogeneous components. The particular strategy applied to select edges is what differentiate algorithm one another.

It is important to state that in both kinds of methods numerous works are found taking advantage of the *Minimum Spanning Tree* (MST) as a mean to reduce the inherent algorithmic complexity of the graph–partitioning problem as well as the one that may appear in region–growing if all node connections are taken into account. MST captures the minim structure of an image and helps by its partition or growing to obtain efficient segmentation algorithms in terms of time and memory.

In [VC93] vertexes which are connected by the smallest edge weight are afterwards melted by an iterative process. At the end of that process, the list encompassing the smallest edges at each step forms a spanning tree which is further split by way of removing the edges with the greatest weights, while generating a hierarchy of image partitions.

In [XU97] a MST is build up using the Kruskal's algorithm to find a partition that minimizes a cost function afterwards. This task is accomplished by a dynamic approach and diverse heuristics to further reduce the algorithm complexity. The approach in [FH98a] is even more drastic in the use of MSTs since it combines both region–growing and Kruskal's routine. Edge aggregation is driven by a local measure of image variation over arbitrarily large regions in the image.

Moreover, this approach addresses a major shortcoming of previous graph–based methods, i.e., the dichotomy between either using efficient (nearly linear time) algorithms, but avoiding global properties of the image, or capturing global image properties, but being less efficient. Despite in [SM97] it is argued that in order to capture nonlocal properties of an image any segmentation algorithm should start with *larger* regions in the image and then splitting them progressively, rather than starting with *smaller* regions and merging them, work in [FH98a] suggests arguments to the contrary, i.e., a region merging algorithm based on *nonlocal* image properties is as well capable of producing segmentations.

They do so by introducing global definitions of what it means for an image to be subsegmented or oversegmented based on the aggregation of local intensity differences. An image is defined to be *oversegmented* when there is some pair of regions for which the variation between regions is small relative to the variation within each region. Besides, an image is *subsegmented* when there is a way to split some regions into subregions such that the resulting segmentation is not an oversegmentation. These definitions could be used along with other measures of similarity between regions.

The algorithm in [FH98a] satisfies at the same time the two global properties of neither subsegmenting nor oversegmenting an image accordingly to their previous definitions. The algorithm runs in nearly linear time of the number of pixels, and it is really fast in practice. This efficiency is achieved by a bottom–up process that successively merges smaller components into larger ones.

6.4 Segmentation of Color Images

Due to the speed of the algorithm in [FH98a], it is a good starting point to develop a fast algorithm for color segmentation that fits the time constraints

of mobile robotics. Hence, many novelties have been introduced in our new approach in order to improve the final results attained by the original algorithm.

The first change we have introduced is the use color differences instead of independently running an intensity version of the algorithm as many times as the number of color channels and trying to mix the obtained regions afterwards. Secondly, we have developed an energy–based approach to control the component merging process so as to relax the oversegmentation condition to obtain, as a consequence, resultant segmentations with fewer regions.

In addition, we have introduced an index to identify all the spurious regions that appear in segmentation as a result of highly variant regions not corresponding to any actual area in the image. These regions are removed from segmentation and joined to their closest neighboring component. The overall *coherence* at the ending segmentation is improved because the remaining regions correlate better with their counterparts in the real scene.

Finally, the algorithm has also been extended to cope with images coming from video sequences in order to maintain their segmentations as *stable* as possible through time. Part of the results described in this Chapter have already been reported in the papers [VLCS00] and [SAA+02].

6.4.1 Some Definitions

First of all, we give some basic definitions will help us along this Section. In our graph–based approach to image segmentation, *Undirected Weighted Graphs* (UWG) are used to represent color images. Being $\mathcal{V}$ a set of vertexes and $\mathcal{E}$ a set of edges connecting them, an UWG is a graph $\mathrm{G} = (\mathcal{V}, \mathcal{E})$ defined from the set of image pixels $\mathcal{P} = \{p_i\}$ and the set of their colors $\mathcal{I} = \{c_p \colon \forall p \in \mathcal{P}\}$ as follows.

Each pixel $p \in \mathcal{P}$ corresponds to a vertex $v \in \mathcal{V}$ to which a *neighborhood* $N_\rho(p) = \{q \in \mathcal{P} \mid 0 < D_\mathcal{P}(p,q) \leq \rho\}$ can be assigned, being $D_\mathcal{P} \colon \mathcal{P} \times \mathcal{P} \to \mathbb{R}_0^+$ a distance between pixels, usually Euclidean, in image coordinates. Therefore, the set of edges is defined as $\mathcal{E} = \{e_{pq} = (p,q) : \forall q \in N_\rho(p)\}$. ρ is the radius of the neighborhood, in number of pixels. Commonly, $\rho = 1$.

Therefore, the *weight function* ω among edges gives a measure of similarity between two vertexes (pixels) as follows

$$\begin{array}{rcl} \omega \colon E & \longrightarrow & \mathbb{R}_0^+ \\ e_{pq} & \longmapsto & \omega\left(e_{pq}\right) = D_\mathcal{I}\left(c_p, c_q\right) = \omega_{pq} \end{array} \tag{6.2}$$

where $D_\mathcal{I}$ is some distance in a color space. We refer to [WS82, SK94, Fai97] for a wider review on color coordinates and distances, which will be partially reviewed later in this Section. Finally, $\Omega = \{\omega(e) \colon \forall e \in \mathcal{E}\}$ is the set of all weights of the edge set in G. The following algorithm works on a fixed ordering $\tilde{\mathcal{E}} = (e^1, \ldots, e^n)$ such that $\omega(e^i) \leq \omega(e^j)\,,\ \forall i \leq j$, where $n = |\mathcal{E}|$.

A *segmentation* of G is defined as a subgraph $\mathrm{S} = (\mathcal{C}, \mathcal{F}_\mathcal{C})$ where $\mathcal{C} = \{C_i\}$ is the set of components forming a *partition*[2] of $\mathcal{V}$ and $\mathcal{F}_\mathcal{C} = \{F_{C_i}\}$ is a *canonical forest*. A *component* C_i is a set of vertexes that are connected one another by a path of edges of $\mathcal{E}$ minimizing the sum of their edge weights. C_p is the component to where the vertex p belongs.

[2] A partition of $\mathcal{X}$ is a group of subsets $\{\mathcal{X}_i \subset \mathcal{X}\} \mid \mathcal{X} = \cup_i \mathcal{X}_i$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset\,,\ \forall i \neq j$.

A canonical forest $\mathcal{F}_{\mathcal{C}}$ is a set of trees where each $F_{C_i} \in \mathcal{F}_{\mathcal{C}}$ is a *Minimum Spanning Tree* (MST) of $C_i \in \mathcal{C}$. The ordering $\tilde{\mathcal{E}}$ provides a way of selecting a unique MST from the possible minimum weight spanning trees of C_i. We can now define the set Σ of all the segmentations S of a graph G and an equivalence relation, $\leq$, of pairs of elements that is *reflexive*, *anti–symmetrical*, and *transitive*

$$T \leq S \Longleftrightarrow T \in R\,(S) \tag{6.3}$$

where $R\,(\mathrm{S}) = \{\mathrm{Q} \in \Sigma \colon \forall C \in \mathrm{Q}\,,\ \exists C' \in \mathrm{S} \mid C \subseteq C'\}$ is a *refinement* of a segmentation $\mathrm{S} \in \Sigma$. Put in words, a refinement of a segmentation S is the set of all other segmentations which have smaller components in a way that once these components get merged, they generate the same components as in S. Moreover, the strict inequality can be defined as $\mathrm{T} < \mathrm{S}$ if and only if $\mathrm{T} \leq \mathrm{S}$ and $\mathrm{T} \neq \mathrm{S}$.

The set $(S, \leq)$ is a *partially ordered set* because the fact that $\mathrm{T}, \mathrm{T}' \leq \mathrm{S}$ does not imply that $\mathrm{T} \leq \mathrm{T}'$ nor $\mathrm{T}' \leq \mathrm{T}$. Nevertheless, for any two segmentations $\mathrm{T} = (\mathcal{C}, \mathcal{F}_{\mathcal{C}})$ and $\mathrm{T}' = (\mathcal{C}', \mathcal{F}_{\mathcal{C}'})$ it is true that $\mathrm{T} \cap \mathrm{T}' \leq \mathrm{T}, \mathrm{T}'$ and $\mathrm{T}, \mathrm{T}' \leq \mathrm{T} \cup \mathrm{T}'$. Schematically

$$\begin{array}{ccccc} & & \mathrm{T} \cup \mathrm{T}' & & \\ & \nearrow & & \nwarrow & \\ \mathrm{T} & & & & \mathrm{T}' \\ & \nwarrow & & \nearrow & \\ & & \mathrm{T} \cap \mathrm{T}' & & \end{array} \tag{6.4}$$

where $\mathrm{T} \cap \mathrm{T}' = (\mathcal{C} \cap \mathcal{C}', \mathcal{F}_{\mathcal{C} \cap \mathcal{C}'})$ and $\mathrm{T} \cup \mathrm{T}' = (\mathcal{C} \cup \mathcal{C}', \mathcal{F}_{\mathcal{C} \cup \mathcal{C}'})$.

The *maximum* element of $(\mathrm{S}, \leq)$ is $\mathrm{G} = (\mathcal{V}, \mathcal{E})$ and the *minimum* is $\mathrm{G}_{min} = (\mathcal{V}, \emptyset)$, where all components have only one vertex and trees have no edge. If we follow an algorithm that put together two components C and C' at each step in respect to an edge in $\tilde{\mathcal{E}}$, the resultant set of graphs at each step will be in ascendant order in $(S, \leq)$, i.e., from minimum to maximum, forming a chain

$$\Pi \colon \mathrm{G}_{min} = \mathrm{S}_1 \leq \ldots \leq \mathrm{S}_n = \mathrm{G} \tag{6.5}$$

This is the case of greedy algorithms such as the Kruskal's minimum spanning tree algorithm and also that in [FH98a].

6.4.2 Algorithm Analysis

Now in this Section we translate the segmentation of an image I into the problem of finding a proper segmentation S from a graph G among the set of all possible segmentations in Σ. As a starting point, we follow the approach in [FH98a], where a segmentation is sought that fulfills a global property by only carrying out a local search. As mentioned, this approach takes advantage of a greedy algorithm that obeys the previous definitions of what is considered to be an *oversegmented* and a *subsegmented* image. The process keeps merging regions until segmentations which are neither oversegmented nor subsegmented are attained. Ideally, this should occur in an intermediate case corresponding to the notion of having neither too many nor too few components in a segmentation.

Intuitively, an image is oversegmented when there are still too many components that could be further merged into bigger regions. Consequently, the algorithm should grow components until the image failed to be oversegmented,

that is, whenever merging more components were a likely error. Hence, an image is no more oversegmented if the differences between any two adjacent components are greater than their differences within

$\diamond$ S $\in \Sigma$ **is NOT oversegmented if**

$$\left.\begin{array}{r}\forall C_i, C_j \in \mathrm{S} \\ \text{adjacent} \\ C_i \neq C_j\end{array}\right\} \Longrightarrow Dif\,(C_i, C_j) > Hom\,(C_i, C_j) \tag{6.6}$$

where $Dif(\circ,\circ)$ is a function measuring the difference between two adjacent components and $Hom(\circ,\circ)$ accounts for the internal homogeneity of both components. Be $\Sigma^c_{OS} \subset \Sigma$ the set of all graphs observing Eq. (6.6)[3]. If $\mathrm{T}_0 \in \Sigma$ is the greatest segmentation in the chain Π being oversegmented, then we can rewrite Σ^c_{OS} in an intervalwise manner as $\Sigma^c_{OS} = (\mathrm{T}_0, \mathrm{G}) = \{T \in \Sigma\colon \mathrm{T}_0 < T < \mathrm{G}\}$.

In a similar way, an image is subsegmented whenever region–growing has gone too far and there are too few components left. This implies that too different components have been erroneously joined. Therefore, an image will not be subsegmented if there exists a proper refinement which is neither oversegmented, meaning that a smaller segmentation can be still found fulfilling Eq. (6.6). Hence, we can take as an interval the set $(\mathrm{G}_{min}, \mathrm{S}) = \{T \in \Sigma\colon \mathrm{G}_{min} < T < S\}$ of all proper segmentations smaller than S. So, we get that

$\diamond$ S $\in \Sigma$ **is NOT subsegmented if**

$$(\mathrm{G}_{min}, \mathrm{S}) \cap \Sigma^c_{OS} \neq \emptyset \tag{6.7}$$

The algorithm proposed by Felzenszwalb&Huttenlocher in [FH98a] – F&H's algorithm, from now on –, which is a modification of the Kruskal's algorithm to compute minimum spanning trees, used the two criteria above to control the segmentation process. Moreover, it was proved that the resulting segmentations were unique, that is, for a particular image the process always ends at the same segmentation and follows the same chain of segmentations Π.

Nevertheless, what is important in this algorithm is the fact that the segmentation process takes decisions based on *local* properties of the image, such as pixel differences, and, yet, the resulting segmentation reflects *global* properties of the image since both oversegmentation and subsegmentation are global image features.

However, we are convinced that these constraints are still too restrictive, which causes aggregation to stop prematurely, giving as a result a class of segmentations with too many components for our purposes. Our approach on the forthcoming Sections touches upon these defects both in a theoretical and a practical manner, as explained straight away.

6.4.3 Theoretical Approach

Stating the fact that the F&H's algorithm causes a resultant segmentation S as soon as both previous constraints are fulfilled and that any two successive

[3] Σ_{OS} is the set of all oversegmented graphs and Σ^c_{OS} is its complement.

segmentations S_i and S_{i+1} accomplish that $\mathrm{S}_i \leq \mathrm{S}_{i+1}$, we deduce that the algorithm will stop whenever

$$(\mathrm{G}_{min}, \mathrm{S}) \cap (\mathrm{T}_0, \mathrm{G}) \neq \emptyset \Longleftrightarrow \mathrm{T}_0 < \mathrm{S} \tag{6.8}$$

This means that the F&H's algorithm stops at the first segmentation S that is not oversegmented, which is in some way quite arbitrary and restrictive since the segmentation S usually has too many components in practice, i.e., it is still oversegmented for our proposes.

If the nonoversegmented criterion were relaxed, it would be possible to attain segmentations S' with fewer components, i.e., $\mathrm{S} \leq \mathrm{S}'$. In case S' were still oversegmented, again the algorithm would follow the aggregation until another nonoversegmented S'' appeared, i.e., $\mathrm{S}' \leq \mathrm{S}''$. Otherwise, we could deter the constraint again or just stop at that segmentation, which would be effectively greater than S and nonoversegmented, as expected.

Nevertheless, oversegmentation can not be pushed too far since as regions grow, so do their internal dissimilarities, which are more than likely to surpass their mutual differences. This would cause the nonoversegmented condition not to be satisfied once a point of no return were crossed, in view of the fact that aggregation would keep on until only one region remained. So, in practice, the interval Σ_{OS}^{c} would be $(\mathrm{T}_0, \mathrm{T}_1)$ and a resulting segmentation should be obtained before T_1 were dangerously too close to G.

In order to manage this leap over the constraints while avoiding the problem of going too far, we first reformulated the nonoversegmented criterion as a problem controlled by an energy function U in the following way

$\diamond$ $\mathrm{S} \in \Sigma$ **is NOT oversegmented if**

$$\forall\, C_i, C_j \in \mathrm{S}, \text{ adjacent, and } C_i \neq C_j \implies \Delta U_{\mathrm{S}\to\mathrm{S}'} > 0 \tag{6.9}$$

where $\mathrm{S} \leq \mathrm{S}'$. $\Delta U_{\mathrm{S}\to\mathrm{S}'}$ stands for the energy of the system involved in the transition between two consecutive segmentations S and S'. If the transition is done by joining components C_i and C_j together, we note this as $\Delta U_{\mathrm{S}\to\mathrm{S}'} = \Delta U\,(C_i \cup C_j)$. In the case of F&H's, we get that

$$\Delta U\,(C_i \cup C_j) = Dif\,(C_i, C_j) - Hom\,(C_i, C_j) \tag{6.10}$$

where $Dif(\circ,\circ)$ increases as regions grow while $Hom(\circ,\circ)$ tends to fall along the segmentation because components differentiate each other more and more as they propagate. Those functions are based on local information provided by edges in $\tilde{\mathcal{E}}$, which is not modified once computed at the starting point because of the greediness of that approach.

The merging step of the algorithm employs the following aggregation condition. At any step k, two components merge if the edge $e^k = e_{ij} \in \tilde{\mathcal{E}}$ connecting them fulfills that

$$C_i^{k-1} \neq C_j^{k-1} \quad \text{and} \quad \Delta U\left(C_i^{k-1} \cup C_j^{k-1}\right) \leq 0 \tag{6.11}$$

then, at step k, segmentation S_k has a new component formed by

$$C_i^{k-1} \cup C_j^{k-1} \quad \text{and} \quad \mathcal{F}_{C_i^{k-1}} \cup \mathcal{F}_{C_j^{k-1}} \cup \left\{e^k\right\} \tag{6.12}$$

Now a condition is needed to be fulfilled for any energy difference ΔU that will make possible to attain global properties of images by means of a greedy algorithm, which is only capable of tracing local features. If for any *discarded*[4] edge $e^k = e_{ij} \notin \mathrm{S}$ such that $C_i \neq C_j$ occurring at position k in the ordering, with $C_i^{k-1} \subseteq C_i$ and $C_j^{k-1} \subseteq C_j$, it is true that

$$\Delta U\left(C_i^{r-1} \cup C_j^{r-1}\right) > 0\,,\ \forall e^r = e_{ij} \notin \mathrm{S} \text{ with } r \geq k \tag{6.13}$$

then, the segmentation produced by conditions in Eq. (6.11) is also nonoversegmented in terms of Eq. (6.9) because $\Delta U\left(C_i \cup C_j\right) > 0$ for any pair of adjacent components.

That is to say, if at a point k two adjacent components do not merge because of their mutual differences, these components will any longer be as similar as to be put together in the final segmentation. Otherwise, it would mean that somewhere in the segmentation process the two regions started to resemble. If using Eq. (6.10), where ΔU rises accordingly to edge values, it is proven in [FH98a] that $C_i^{k-1} = C_i$ and $C_j^{k-1} = C_j$, which satisfies the above condition. Hence, any other energy function should act similarly in order to provide nonoversegmentations.

The energy–based approach makes possible to introduce the probability of an event $S \rightarrow S'$, namely, the union of two adjacent components $C_i \cup C_j$, in a similar way as it is computed in a simulated annealing process using the Metropolis dynamics [Wan98]

$$Pr\left(C_i \cup C_j\right) = exp\left(-\frac{\max\{\Delta U\left(C_i \cup C_j\right), 0\}}{t}\right) \tag{6.14}$$

If $\Delta U\left(C_i \cup C_j\right) \leq 0$ then $Pr\left(C_i \cup C_j\right) = 1$. Otherwise, $Pr\left(C_i \cup C_j\right)$ is compared to a random number to decide whether or not to joint.

The probability thus computed is employed as a condition in Eq. (6.11) to decide whether to merge two components. As a result, it is possible to find other nonoversegmentations S' such that $\mathrm{S} \leq \mathrm{S}'$. Since it is a probabilistic scheme, Eq. (6.13) may not be guaranteed to be always fulfilled. Nevertheless, in each step, the energy needed to break through the constraint is greater so the leap is less likely, being practically impossible from certain point on which satisfies Eq. (6.13). Besides, width of the interval $(\mathrm{T}_0, \mathrm{T}_1)$ can be selected by tuning the *temperature* t. Consequently, at the end we always get segmentations which are neither oversegmented nor subsegmented, as desired.

6.4.4 Practical Approach

It is time to further specify functions $Dif(C_i, C_j)$ and $Hom(C_i, C_j)$ in terms of edge weights. As said, $Dif(C_i, C_j)$ accounts for the difference between two adjacent components and is defined as the lowest weight edge connecting them

$$Dif(C, C') = \min_{\substack{v_i \in C \\ v_j \in C'}} \{\omega\left(e_{ij}\right) : e_{ij} = (v_i, v_j) \in \mathcal{E}\} \tag{6.15}$$

On the other hand, function $Hom(\circ, \circ)$ measures the internal homogeneity of the two components as the lowest value for the variation within, that is,

$$Hom(C, C') = \min\{Int(C), Int(C')\} \tag{6.16}$$

[4]Not fulfilling conditions in Eq. (6.11).

The inner variation of a component is taken as the highest edge weight in any minimum spanning tree of that component

$$Int(C) = \max_{\forall e \in F_C} \{\omega(e)\} \tag{6.17}$$

The use of such a function $Int(C)$ has, indeed, some problems [FH98a]. Due to the fact that a component C will not grow for any edge e such that $\omega(e) \geq Int(C)$, and since $Int(C) \geq \omega(e')$, $\forall e' \in \mathcal{F}_C$, it is only possible that all the edges in $\mathcal{F}_C$ have the same weight $\omega(e) = Int(C)$. Given that the first edge value is 0, regions can not grow beyond this value because $\omega(e) > Int(C) = 0$ for any edge left in $\mathcal{F}_C$.

To solve this defect in such a way that function $Int(C)$ be greater in small components whereas decreases as components grow, a better version for the function $Int(C)$ is

$$Int(C) = \max_{\forall e \in \mathcal{F}_C} \{\omega(e)\} + \frac{\tau}{|C|} \tag{6.18}$$

This function overestimates the internal variation of components when they are small. Despite helping homogeneous regions to grow, it artificially increases the internal variation of regions with an already great variation, such as borders and textured regions. Hence, some spurious regions may appear having no correspondence to actual regions of homogeneous color, rather than to high variable and textured regions.

To cope with a pernicious effect that might helplessly increase the number of segments, we identify all those pixels belonging to these regions by means of an index I_C computed for every region C. Only spurious border regions are taken under consideration since most of texture is eliminated using a proper smoothing filter. Index I_C accounts for the shape of the region, the amount of variability, and its size. Therefore, it is directly proportional to the compactness of the region K_C and to the maximum internal variation $max\{\omega\}$, and inversely proportional to its area $|C|$, i.e.,

$$I_C = \frac{K_C \cdot \max_{\forall e \in \mathcal{F}_C} \{\omega\,(e)\}}{|C|} \tag{6.19}$$

Once all those regions get identified, their pixels are randomly distributed into the adjacent components with most neighboring pixels. Hence, if the set of all neighbor components to pixel p is defined as $\mathcal{N}_p = \{C_q \in \mathcal{C} \colon (p, q) \in \mathcal{E}\}$, pixel p will be added to component C' if and only if

$$C' = \underset{\forall C \in \mathcal{N}_p}{\operatorname{argmax}} \{\, |N(p) \cap C| \,\} \tag{6.20}$$

If the number of spurious pixels is too big this step can cause some distortions to region borders. Hence, in order to have as few spurious pixels as possible it might be sensible to temporarily deter the oversegmentation constraint, granting that, at least for $\omega(e^k) \leq thr$ the aggregation be freely done. The combination of these two heuristics make possible to grow homogeneous regions, while reducing the population of spurious regions.

6.4.5 Algorithm Sketch

Finally, if all those considerations are put together in a proper way, we accomplish an algorithm capable of segmenting color images based on a greedy algorithm which computes the minimum spanning tree of an undirected weighted graph encompassing the differences between the colors of any pair of neighboring pixels as edge weights. The segmentation thus obtained is a subgraph $\mathrm{S}_n \subset \mathrm{G}$. The sketch for the whole algorithm is considered hereafter.

---

1. Sort edges in $\mathcal{E}$ into an ordering $\tilde{\mathcal{E}} = \left(e^1, \ldots, e^n\right)$, where $n = |\mathcal{E}|$, by nondecreasing edge weights $\omega(e^k)$.
2. Start segmentation with $\mathrm{S}_0 = \mathrm{G}_{min}$ and $k = 0$.
3. Blind aggregation while $\omega(e^k) \leq thr_1$. Nonoversegmentation condition is deterred and components grow freely.
4. Repeat step 5 and 6 for $thr_1 < \omega(e^k) \leq \omega(e^n)$.
5. Select a random number $\nu \in [0, 1]$
6. Construct S_k from previous segmentation S_{k-1}. If edge $e^k = e_{ij}$ connects two components such that
$$C_i^{k-1} \neq C_j^{k-1} \quad \text{and} \quad Pr\left(C_i^{k-1} \cup C_j^{k-1}\right) > \nu \tag{6.21}$$
then S_k is computed using Eq. (6.12). Otherwise, e^k is rejected to compute the contour image afterwards. Probability is computed with Eq. (6.14).
7. Compute index I_C for each component applying Eq. (6.19). Regions with $I_C > thr_2$ are labeled as spurious components.
8. Distribute pixels belonging to spurious components to neighboring regions applying the heuristic in Eq. (6.20).

Both thr_1 and thr_2 are thresholds provided by the user controlling blind aggregation and spurious regions identification, respectively. Two more parameters are needed in order to put the routine to work, namely, growing threshold τ and temperature t. Generally, both thr_1 and t are maintained constant, while the result is controlled by tuning parameters thr_2 and τ.

The implementation maintains the segmentation using a disjoint–set forest with union by rank and path compression as the original Kruskal's algorithm in [CCLR01]. The running time for the algorithm can be split into three parts. First, in Step 1 it is necessary to sort the weights into a nondecreasing ordering. Since the weights are continuous values we used the *bucket sort* algorithm, which requires a $O(n)$ time, being $n = |\mathcal{E}|$ the number of edges. Steps 2 to 6 of the algorithm take a time complexity of $O(n\,\alpha(n))$, where α is the very slow–growing Ackerman's function [CCLR01]. This is because the number of edges is $O(n)$ since the neighborhood size δ is constant. Finally, Step 7 and 8 are $O(m)$, where $m \leq n$ is the number of pixels in spurious components. To determine those pixels, the set of discarded edges is employed, which is easily available from Step 6. At the end, pixel redistribution is done in a raster way simulating a random assignment to speed up the process.

6.4.6 Color Spaces and Distances

The world of color spaces and metrics is far wider than one could imagine at first glance. There are literally dozens of them, usually in a straight relation to their specific use. Thus, there are color spaces for the fabric industry, paper industry, press, psychology, television, computers, physics, and even for foods. Despite the numerous efforts to find a definitive one, there is no single all-terrain color space nor even a simple way to compare colors valid enough to everyone.

Here, we are not going to rehash them all over again, not even some of them. *Ars longa, vita brevis.* We just summarize those found essential for our interests and means, basically digitalized color images given in *RGB* coordinates. For a more extensive study on color, we suggest Wyszecki and Stiles' book [WS82]. In case this is too much and only a slight coat of paint is needed, work in [SK94] would suffice. For the latest knowledge on color models, have a look into [Fai97].

RGB

These are the color coordinates provided by most capture and imaging sets nowadays. They consist basically in the sensor response to a set of filters, as explained in Chapter 4. Those filters are an artificial *counterpart* of the human mechanism of color perception and reproduction of most colors can be achieved by modulating three channels roughly corresponding to colors red, green, and blue.

The natural way to compare two colors would be the use of the Euclidean distance. Thus

$$\Delta C = \sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2} \tag{6.22}$$

Nevertheless, some problems rise when trying to emulate the human judgement of color differences. First, we are more sensitive to some colors than others, which means that for them our sense of difference is finer. This is not the case when using the above distance. Moreover, some color changes affects differently on some areas of the color space. Nonetheless, since the Euclidean distance is homogeneous and isotropic for the RGB color space, the aforementioned kind of nuances in the differences between colors can not be reproduced.

Next, we consider three possible alternatives coping with those difficulties, namely, HSI, Lab, and Luv color spaces. All of them try to translate the human perception of color into figures. Besides, both Lab and Luv aspire to define a space where the Euclidean metric can be used straight away to estimate subtler color differences.

In addition to these approaches, there also exists a number of other works on color representation being the most important among them those of Smeulders and Gevers [GS99, GBSG01]. The authors try to generate there a set of color invariants by all sort of derivatives of a fundamental color invariant extracted from certain reflectance model. We are not considering those endeavors in our work because their involvement limits a practical application as well as results only show their performance on a pretty small set of images of too unrealistic and homogeneous objects.

Our greatest objection to these class of invariants, however, has to do with the way a given color is transformed independently of what happens in the rest of the color space and of the illuminant conditions that produced such measure. As a consequence, the invariant will always produce the same result

for the same input no matter this color comes from two different surfaces under different light conditions which happen to coincide in this color at least. This problem is usually referred to as *metamerism* and is greatly reduced if the whole set of colors is considered instead.

HSI

There are many color models based on human color perception or, at least, trying to do so. Such models want to divide color into a set of coordinates decorrelating human impressions such as hue, saturation, and intensity. Next expressions compute those values from raw sensor RGB quantities [SK94]

$$\begin{aligned} I &= \tfrac{1}{3}\left(R+G+B\right) \\ S &= 1-\tfrac{\min\{R,G,B\}}{I} \\ H &= \arctan\left(\tfrac{\sqrt{3}(G-B)}{2R-G-B}\right) \end{aligned} \tag{6.23}$$

I models the intensity of a color, i.e., its position in the *gray diagonal*[5]. Saturation S accounts for the distance to a pure white with the same intensity, that is, to the closest point in the gray diagonal. H is an angle representing just a single color without any nuance, i.e., naked from its intensity or vividness. Some approaches erroneously to our taste use the Euclidean directly to compute color differences in HSI coordinates forgetting that hue is an angle and not strictly a spatial measure. Hence, as suggested in [SK94], probably a better distance would be the following expression

$$\Delta C = \sqrt{\left(I_2-I_1\right)^2 + S_2^2 + S_1^2 - 2S_2S_1\cos\left(H_2-H_1\right)} \tag{6.24}$$

At small intensities or saturations, hue is very imprecisely determined with those expressions and it is a better idea to compare colors by means of their intensity in that case.

CIELAB

The CIE[6] 1976 (L^*, a^*, b^*) is a *uniform color space* developed as a space to be used for the specification of color differences. It is defined from the tristimulus values normalized to the *white* by next equations

$$\begin{aligned} L^* &= 116\left(\tfrac{Y}{Y_w}\right)^{\frac{1}{3}} - 16 \\ a^* &= 500\left[\left(\tfrac{X}{X_w}\right)^{\frac{1}{3}} - \left(\tfrac{Y}{Y_w}\right)^{\frac{1}{3}}\right] \\ b^* &= 200\left[\left(\tfrac{Y}{Y_w}\right)^{\frac{1}{3}} - \left(\tfrac{Z}{Z_w}\right)^{\frac{1}{3}}\right] \end{aligned} \tag{6.25}$$

[5] The line from $(0,0,0)$ to $(R_{max}, G_{max}, B_{max})$, where the maximum coordinate value is 255 or 1, if normalized coordinates are used.

[6] Comité International d'Éclairage.

In these equations (X, Y, Z) are the tristimulus values of the pixel and (X_w, Y_w, Z_w) are those of the reference white. We approximate these values from (R, G, B) by the linear transformation in [SK94]

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{6.26}$$

Our reference white is $(R_w, G_w, B_w) = (255, 255, 255)$. L^* represents lightness, a^* approximates redness–greenness, and b^*, yellowness–blueness. These coordinates are used to construct a Cartesian color space where the Euclidean distance is used, i.e.,

$$\Delta E^*_{ab} = \sqrt{\Delta {L^*}^2 + \Delta {a^*}^2 + \Delta {b^*}^2} \tag{6.27}$$

CIELUV

The CIE 1976 (L^*, u^*, v^*) is also a uniform color space defined by equations

$$\begin{aligned} L^* &= 116 \left(\tfrac{Y}{Y_w}\right)^{\frac{1}{3}} - 16 \\ u^* &= 13 L^* \left(u' - u'_w\right) \\ v^* &= 13 L^* \left(v' - v'_w\right) \end{aligned} \tag{6.28}$$

In these equations u' and v' are the *chromaticity coordinates* of the stimulus and u'_w and v'_w are those of the reference white. These values actually are the CIE 1976 Uniform Chromaticity Scales (UCS) defined by equations

$$\begin{aligned} u' &= \tfrac{4X}{X+15Y+3Z} \\ v' &= \tfrac{9Y}{X+15Y+3Z} \end{aligned} \tag{6.29}$$

As before, (X, Y, Z) are the tristimulus values of a pixel computed from RGB values with Eq. (6.26). Analogously to (L^*, a^*, b^*) coordinates, those coordinates also construct a Cartesian color space where to use the Euclidean distance

$$\Delta E^*_{uv} = \sqrt{\Delta {L^*}^2 + \Delta {u^*}^2 + \Delta {v^*}^2} \tag{6.30}$$

We must state that in [Fai97] is argued that (L^*, a^*, b^*) are better coordinates than (L^*, u^*, v^*) since the adaptation mechanism of the latter – a subtractive shift in chromaticity coordinates, $(u' - u'_w, v' - v'_w)$, rather than a multiplicative normalization of tristimulus values, $(X/X_w, Y/Y_w, Z/Z_w)$ – can result in colors right out of the gamut of feasible colors. Besides, (L^*, u^*, v^*) adaptation transform is extremely inaccurate with respect to predicting visual data. However, what is worst for our purposes is its poor performance at predicting color differences. We consequently prefer to use Lab coordinates, whenever an alternative to the RGB space is needed.

6.5 Segmentation of Sequences

We must now face the problem of segmenting a sequence of images keeping in mind that those segmentations should satisfy at least two general properties, namely, components should correspond to actual regions in the image of homogenous color (coherence) and remain as stable as possible through the sequence. In other words, we do not want either segmentations with too many small regions or components which fluctuate too much through time.

However, the process of reducing the number of components by aggregating similar adjacent regions may cause unstable segmentations because some of them may be joined differently in contiguous frames. From some preliminary results it seems that a more coherent segmentation would be necessary to prevent this shortcoming.

We suggest an approach which takes advantage of the segmentation of the immediately previous frame in order to obtain that of the next one. The idea is pretty simple and, for each new frame, consists in grouping similar regions into bigger ones in the same way as it was done in the preceding frame. Thus, we kill two birds with one stone, i.e., we get greater coherence and stability. Obviously, an intermediate step dedicated to matching regions which seem equal in two consecutive images is needed.

In general, using a correspondence stage in a segmentation process would be seen as a drawback because of being a time consuming and a usually prone–to–error process. Nevertheless, we propose to use the ideas laying behind the IRM distance between regions [WLW01], which provides both robustness to poor segmentations and effortlessly integrates features from many regions.

Next, we consider the two steps that are needed in our segmentation of sequences, namely, the computations of correspondences among components and the propagation of previous segmentations into the new ones for each frame in the sequence.

6.5.1 Computation of Component Correspondences

The correspondence between two components, $C_i^{k-1} \sim C_{i'}^k$, in two correlative frames I_{k-1} and I_k is defined as

$$C_i^{k-1} \sim C_{i'}^k \Longleftrightarrow C_i^{k-1} = \underset{\forall C_l^{k-1} \in \mathcal{I}_{k-1}}{\operatorname{argmin}} \left\{ D\left(C_l^{k-1}, C_{i'}^k\right) \right\}, \quad \forall\, C_{i'}^k \in \mathcal{I}_k \qquad (6.31)$$

where $D(\circ, \circ)$ is a measure of distance between components in $\mathcal{I}_{k-1}$ and $\mathcal{I}_k$.

As said, we follow the ideas of the IRM similarity measure[7] in [WLW01] to compute a content–based distance between two components from different images. Our approach combines, at the same time, features of *appearance* and *position*. We use the mean color as the appearance feature, while the component center of mass is the position feature.

Then, the difference $D(C_l^{k-1}, C_{i'}^k)$ between two components in two successive frames is computed using the simple Euclidean distance over the features above. In order to compare side by side two features that apparently are rather heterogeneous, such as color and position, we normalize the coordinates to fit the interval $[0, 1]$ dividing each component by the maximum range of each feature.

[7] In Chapter 7 there is a wider explanation about this measure.

This way, things which are *a priori* different and have dissimilar units can be compared as if they were basically the same. Theoretically, computations should be done for all $C_i^{k-1} \in \mathcal{I}_{k-1}$ and $C_{i'}^{k} \in \mathcal{I}_k$ so that we finally got all the correspondences between components in two correlative frames. Nonetheless, to speed up computations it is interesting to focus comparisons only to a certain area surrounding the likeliest position where to find those component.

6.5.2 Propagation of Component Correspondences

For each new frame, once the image has been individually segmented into components, we would like to use the previous regrouping of components to reduce the number of existing regions in the present image while preserving the regions which have already come up, maintaining the degree of coherence along the sequence as a consequence of it.

Formally, let us suppose that two consecutive frames I_{k-1} and I_k provide us with two segmentations $\mathcal{I}_{k-1} = \{C_i^{k-1}\}_{i=1,\dots,n_{k-1}}$ and $\mathcal{I}_k = \{C_{i'}^{k}\}_{i'=1,\dots,n_k}$, respectively. Let us also assume we know that the segmentation $\mathcal{I}_{k-1}$ has been reduced to a new segmentation with bigger components $\tilde{\mathcal{I}}_{k-1} = \{\tilde{C}_j^{k-1}\}_{j=1,\dots,m_{k-1}}$, where $m_{k-1} \leq n_{k-1}$ and for each component $C_i^{k-1} \in \mathcal{I}_{k-1}$ there exists a bigger component $\tilde{C}_j^{k-1} \in \tilde{\mathcal{I}}_{k-1}$ so that $C_i^{k-1} \subseteq \tilde{C}_j^{k-1}$. We define the set of indexes Ind_j of all components in $\mathcal{I}_{k-1}$ that have been put together forming one single region $\tilde{C}_j^{k-1} \in \tilde{\mathcal{I}}_{k-1}$. There hence exist as many index sets as components in $\tilde{\mathcal{I}}_{k-1}$.

The problem then is to propagate the segmentation in $\tilde{\mathcal{I}}_{k-1}$ into the one in $\mathcal{I}_k$ forming, as a consequence, a new segmentation $\tilde{\mathcal{I}}_k = \{\tilde{C}_{j'}^{k}\}_{j'=1,\dots,m_k}$. This is carried out by grouping the regions in $\mathcal{I}_k$ in such a way that if any component $C_{i'}^{k} \in \mathcal{I}_k$ *corresponds* to a component $C_i^{k-1} \in \mathcal{I}_{k-1}$ in the previous frame that was joined forming a bigger region $\tilde{C}_j^{k-1} \in \tilde{\mathcal{I}}_{k-1}$, then the component $C_{i'}^{k}$ will be grouped with the others satisfying the same property and creating the bigger component $\tilde{C}_{j'}^{k}$, which is the propagation of the component $\tilde{C}_j^{k-1}$ in the $(k-1)^{th}$ frame into the k^{th} frame, that is, $\tilde{C}_j^{k-1} \sim \tilde{C}_{j'}^{k}$. Formally, the component $\tilde{C}_{j'}^{k}$ is build as follows

$$\tilde{C}_{j'}^{k} = \bigcup_{i' \in Ind_{j'}} C_{i'}^{k}, \qquad \forall C_{i'}^{k} \in \mathcal{I}_k \mid C_i^{k-1} \sim C_{i'}^{k} \wedge C_i^{k-1} \subseteq \tilde{C}_j^{k-1} \tag{6.32}$$

In other words, components in a given frame will be joined together as their corresponding components were joined in the anterior frame. Finally, a new segmentation $\tilde{\mathcal{I}}_k$ is achieved at k^{th} frame, which is in general less oversegmented than the original one, $\mathcal{I}_k$, while maintaining the stability of regions in respect to the previous frame.

This scheme does not need to treat in any particular manner the components that appear or disappear in every new frame. Since component correspondence is done backwards, disappearing regions simply have no matching in the new frame. On the other hand, new regions will look for the closest region in the previous frame in terms of color and position. If the resulting distance is too great, then it is not adjoined to any component in $\tilde{\mathcal{I}}_{k-1}$ and is considered as a new region in the segmentation $\tilde{\mathcal{I}}_k$.

6.6 Experiments and Results

The main concern in this Chapter resides in the segmentation of color images considered both as static images as well as belonging to a sequence that may have been obtained, e.g., from an autonomous robot. In order to achieve this goal, we display in this Section the set of experiments that have been carried out and the results obtained.

Principally, these experiments consist in the segmentation of such images by the algorithm we suggest in this Chapter and the comparison of the resulting outcomes with those attained by means of the two other approaches already mentioned in previous paragraphs, namely, the original F&H's algorithm in [FH98a] and the Figueiredo's EM clustering method in [FJ02].

The goal of doing so is, first of all, to illustrate the improvements that have been attained in relation to the results by the original F&H's algorithm, while maintaining its speed at a similar level. Likewise, our algorithm has been put side by side to that of Figueiredo, which is known to perform fairly well, to comparatively study the quality of our segmentations. Since our segmentations are definitively far faster than those of Figueiredo's unsupervised EM, it is important for us to show that the same range of quality is kept.

6.6.1 Segmentation of Static Images

The images shown in Fig. 6.1 correspond to different stages in the segmentation of the picture exhibited in Fig. 6.1 (a). First, we display the results obtained using the original F&H's algorithm in Fig. 6.1 (b). It can be appreciated how this segmentation is not completely satisfactory since big homogeneous regions are split into several components, specially in the background. This is partially solved in Fig. 6.1 (c), where now homogeneous regions are completely merged in a coherent way into bigger components.

Nevertheless, the total number of regions is still high in respect of the relatively small number of potential real regions in the image. This is because of the spurious regions generated in highly variable areas such as borders. These regions are detected using the index defined in Eq. (6.19) and can be observed in Fig. 6.1 (d). Finally, the resulting segmentation can be appreciated in Fig. 6.1 (e) after removing spurious regions, closely fitting actual homogeneous areas in the scene.

An analogous situation is the one shown in Fig. 6.2, where the well–known picture of peppers is segmented. Again, the original image is portrayed in Fig. 6.2 (a). Fig. 6.2 (b) is the segmentation before removing the spurious regions that are pictured in Fig. 6.2 (c). The final result is exhibited in Fig. 6.2 (d). It must be noted that spurious pixels are eliminated by layers, starting at outer layers and ending with inner pixels. In this manner, regions tend to phagocytize any small spurious region within and to grow outwards until another region is found. This is not a genuine dilatation since pixels prefer regions with the highest number of neighbors in common.

In order this segmentation to be useful in an object recognition system, it is important that images of a given object, which have been taken from different angles, be segmented in a similar way. We verify that behaviour in Fig. 6.3 and Fig. 6.4 where two series of images are shown. Fig. 6.3 displays a toy bear under six views. The upper row shows the original pictures while the lower row offers

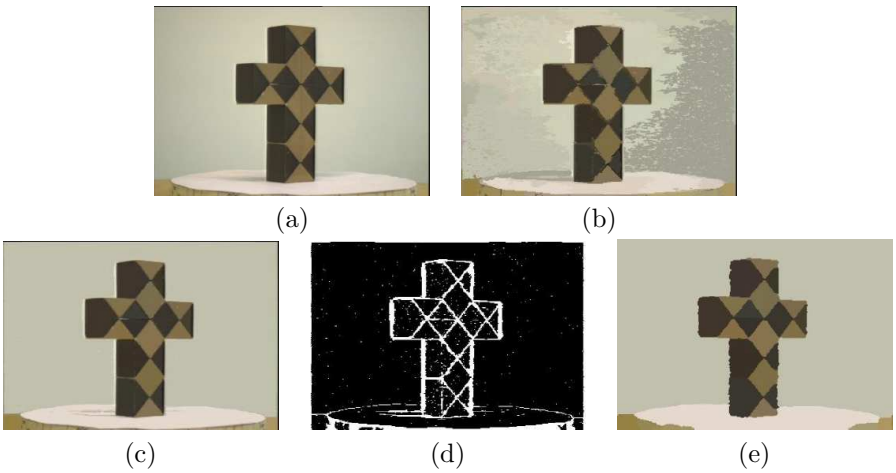

Figure 6.1: Comparing our algorithm to that of F&H: (a) Original Image. (b) F&H's segmentation. (c) Our segmentation before spurious regions elimination. (d) Spurious regions. (e) Final result after spurious regions elimination.

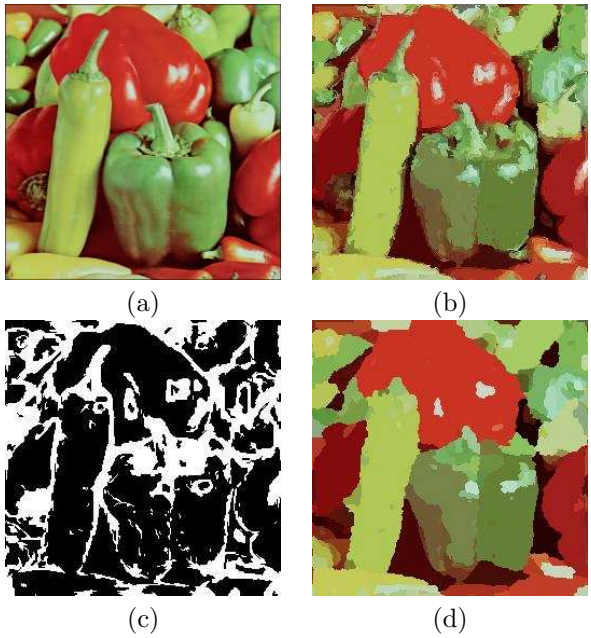

Figure 6.2: Example of our segmentation: (a) Original Image. (b) Our segmentation before spurious regions elimination. (c) Spurious regions. (e) Final result after spurious regions elimination.

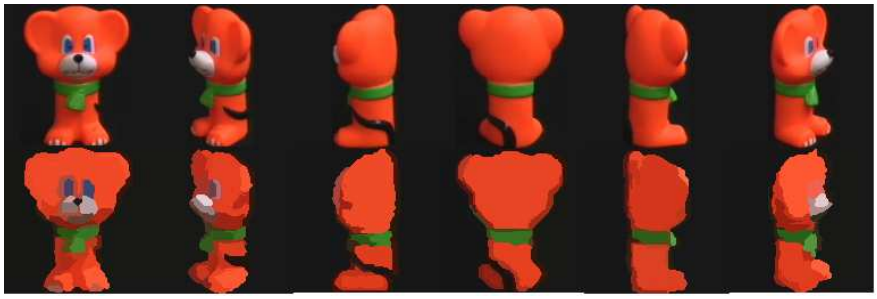

Figure 6.3: Example of our segmentation. Upper row: Original image. Lower row: Segmented image.

Figure 6.4: Example of our segmentation. Upper row: Original image. Lower row: Segmented image.

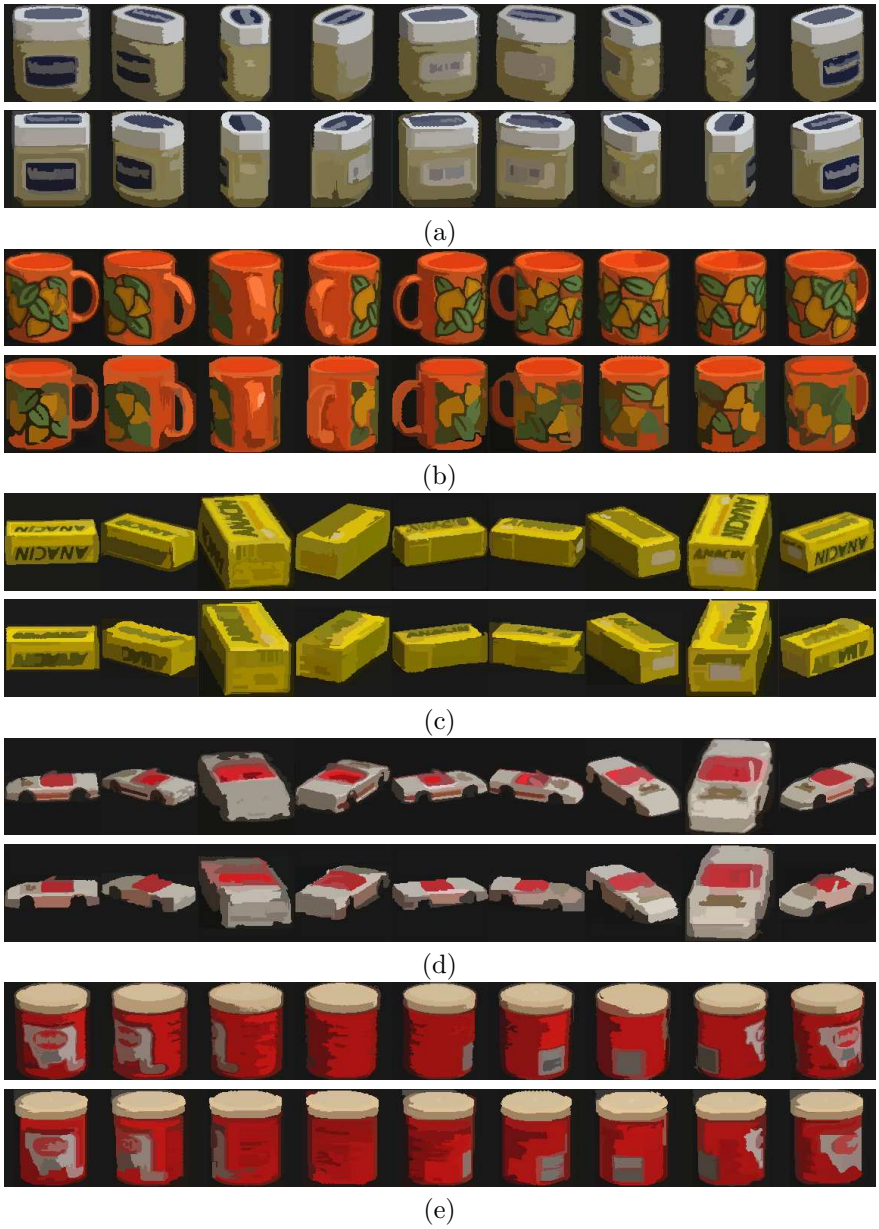

Figure 6.5: Comparing our algorithm to that of Figueiredo. Upper row: Figueiredo's segmentation. Lower row: Our segmentation.

the segmented images. A similar series is exhibited in Fig. 6.4, where a set of ten different views is supplied. In both series, regions formed in neighboring views are similarly segmented. Shades and highlights are collected into separate regions, which is quite natural since we, as humans, can also perceive them as separate areas. In our opinion, it is not a segmentation concern the issue of identifying such regions and discerning to which component they belong, a question that should be implemented in a different level task.

Finally, in Fig. 6.5 we check up on whether our segmentation algorithm is capable of attaining results comparable to those obtained by the Figueiredo's unsupervised clustering algorithm [FJ02]. This is an excellent version of the EM technique, very useful to segment images of unknown content since there is no need to know the exact number of clusters to run the routine. Moreover, this algorithm provides us with a family of Gaussian distributions as a result. Nevertheless, it takes quite a lot of time to complete an image. For example, a 360×288 image takes about 25 sec. to get segmented in a 800 MHz PC. Our algorithm only takes about $0.10 \div 0.20$ sec. in the same computer, which is almost less than two orders of magnitude.

In these series, the upper rows of each object are formed by the results corresponding to the Figueiredo's segmentation, whereas the lower rows belong to the ones obtained by our algorithm. The aim in placing these images this way is to illustrate mainly two important questions, i.e., how different views of the same object are comparatively segmented and whether these segmentations differ too much depending on the kind of algorithm used. At first sight, it seems that both algorithms supply very similar segmentations, despite the elimination of spurious regions in our approach can produce slightly differing results wherever textured areas appear in images, as it is the case of fruity drawings and letters in Fig. 6.5(b) and Fig. 6.5(c), respectively.

6.6.2 Segmentation of Sequences

We now move on to the description of some of the results that have been obtained after segmenting a video sequence captured from a mobile robot in an indoor environment. Yet, our aim is to illustrate the performance of our algorithm in such a task if compared to Figueiredo's approach. At this point, we must state the difficulty we found to put these results in paper. Although the sense of all that is at once grasped once the videos are viewed[8], we try to provide the same information in the following pages by only showing a set of images from a short interval out of the whole sequence.

This piece of sequence is in Fig 6.6 and consists of a reduced set of 16 images from a longer sequence ($\approx$ 1 min.) of 1001 images at a rate of 15 images/sec. This small set span for about 10 sec. and represent only one every 10 images. Images are filtered using the *median* filter with a neighborhood of 3×3 pixels to remove noise and to get smooth images without enlarging region contours. Color information is stabilized using the color constancy algorithm in Chapter 5 together with the *Mean* heuristic. The first image in the sequence is employed as the canonic one.

The first step is to examine how the Figueiredo's algorithm performs in segmenting sequences in order to later compare them with those achieved by our

[8]These videos will be provided in a CD–ROM for a better appreciation along with the rest of the graphics and images used in the conformation of this document.

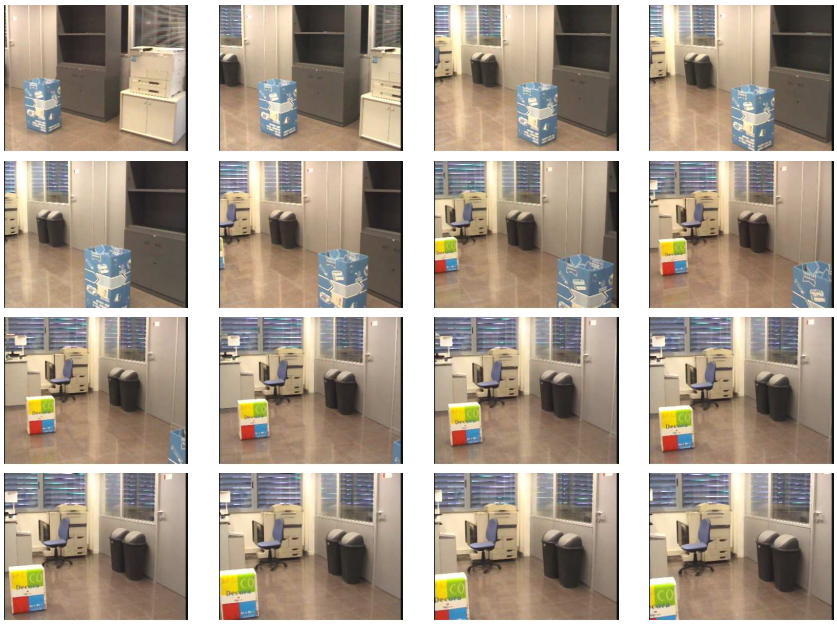

Figure 6.6: Set of images from the video sequence of a mobile robot moving about in an indoor environment.

approach. Two set of images are presented in Fig. 6.7 in groups of two rows. The upper row are the same images in Fig 6.6 that have been independently segmented, meaning that each image is segmented using a randomly initialized Gaussian mixture. As can be seen, this method presents a number of problems since clustered colors are not exactly the same in consecutive frames. To minimize this lack of *stability*, the initialization routine is changed so that it could take advantage of previous segmentations.

This is very easily attained using at each new frame the finite mixture of Gaussian distributions from the previous EM execution. When a certain color disappears, its corresponding Gaussian simply gets a zero weight and dies out. Letting spare Gaussian distributions initialize at random allows the algorithm to incorporate new clusters into the next segmentation step. Results obtained in that manner are displayed at the lower row in Fig. 6.7. The improvement is obvious in both segmentation and computation time, since convergence of the EM routine is faster due to the minor number of distributions and their closeness to the quiescent point.

Afterwards, in order to complete the series of segmentations we carry out the same experiment as before, but using this time our algorithm in the next two cases, namely, without and with the enforcement of stability based on the computation and propagation of correspondences between components explained in Section 6.5. To perform these experiments, we use two color spaces, i.e., RGB along with the Euclidean distance, and Lab with the ΔE_{ab} metric, both oh them reviewed in Section 6.4.6. Results obtained this way are exhibited in Fig. 6.8 for case of Lab space, and in Fig. 6.9 for RGB coordinates.

As explained for Fig. 6.7, segmentations produced as if images were inde-

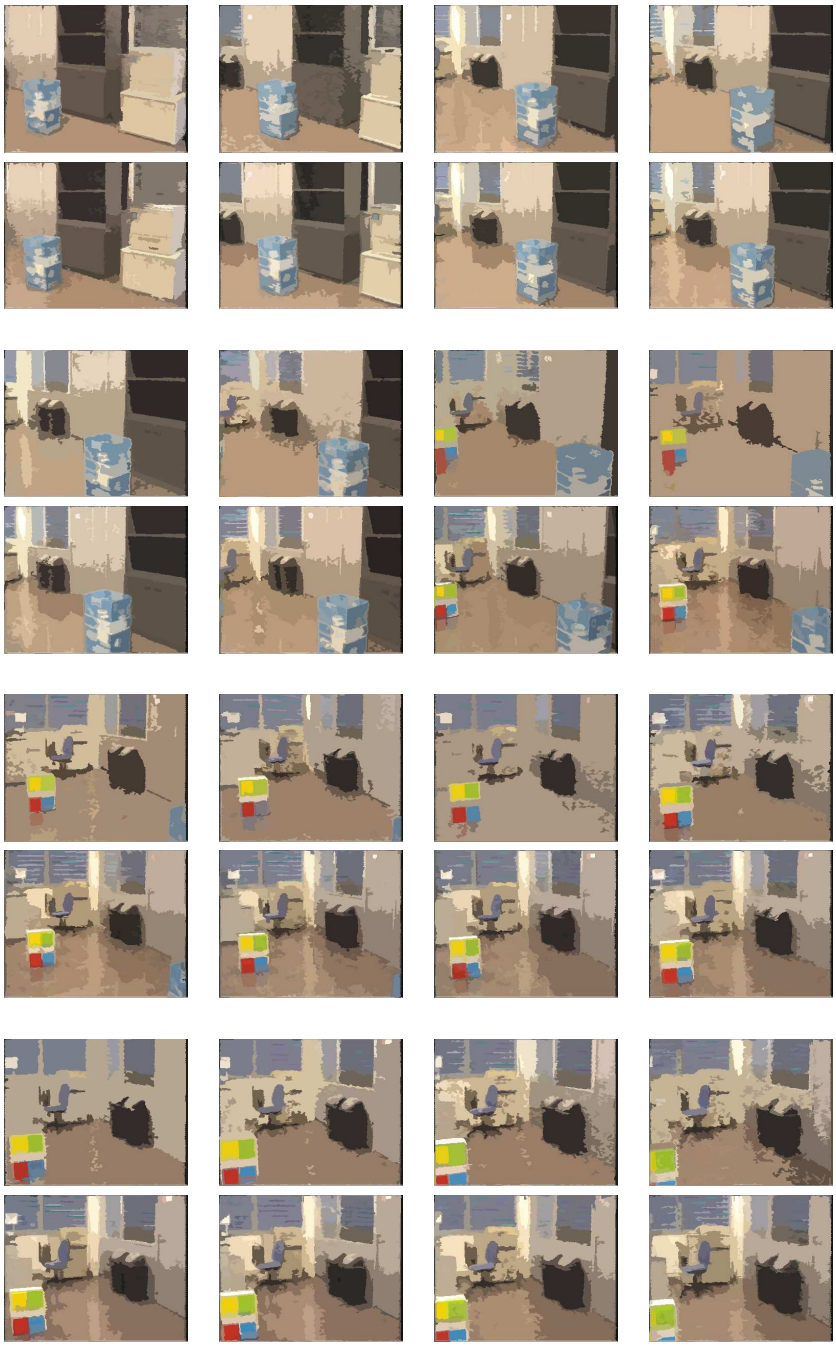

Figure 6.7: Images from the video sequence segmented using Figueiredo's algorithm. Upper row: independent images. Lower row: using previous segmentation.

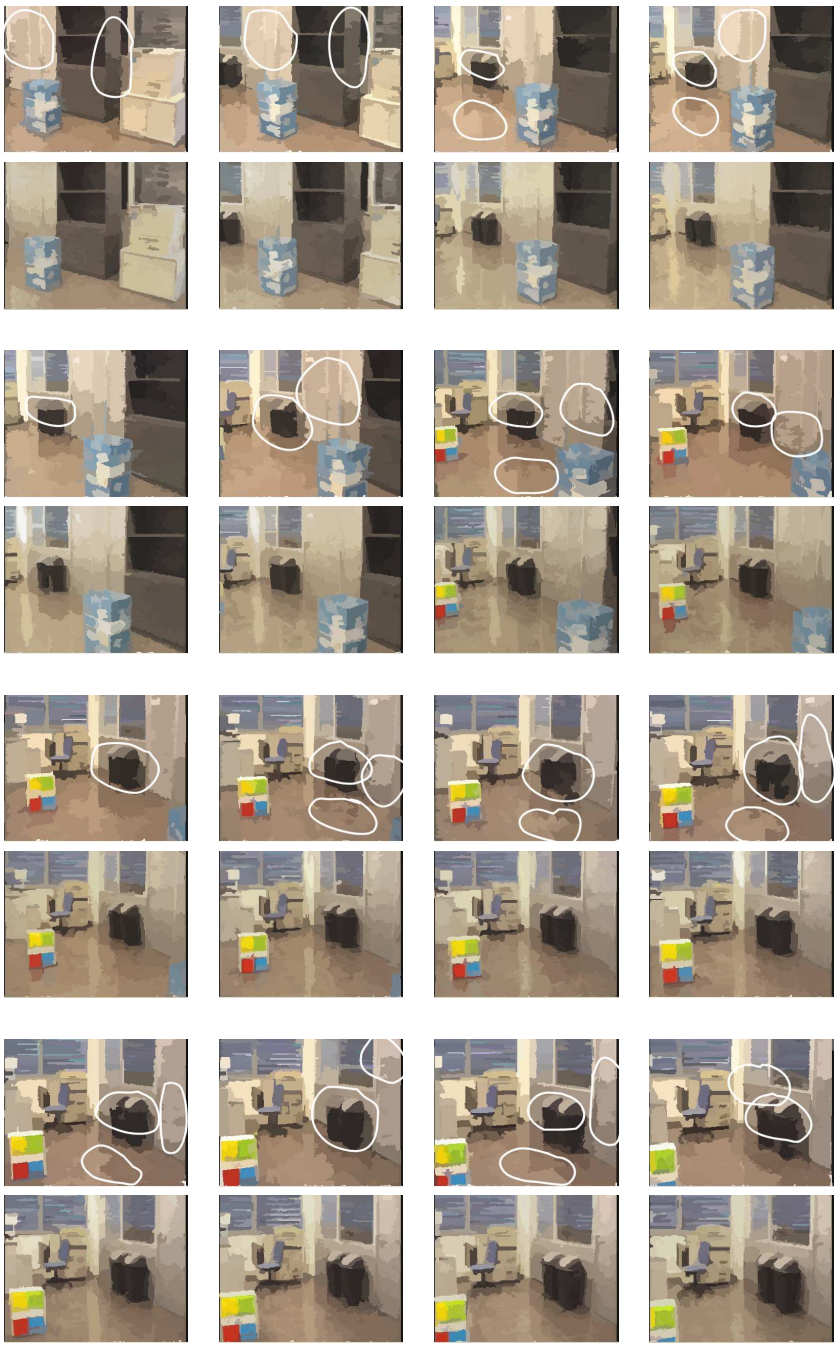

Figure 6.8: Images from the video sequence segmented using our algorithm and Lab color space. Upper row: independent images. Lower row: component correspondence.

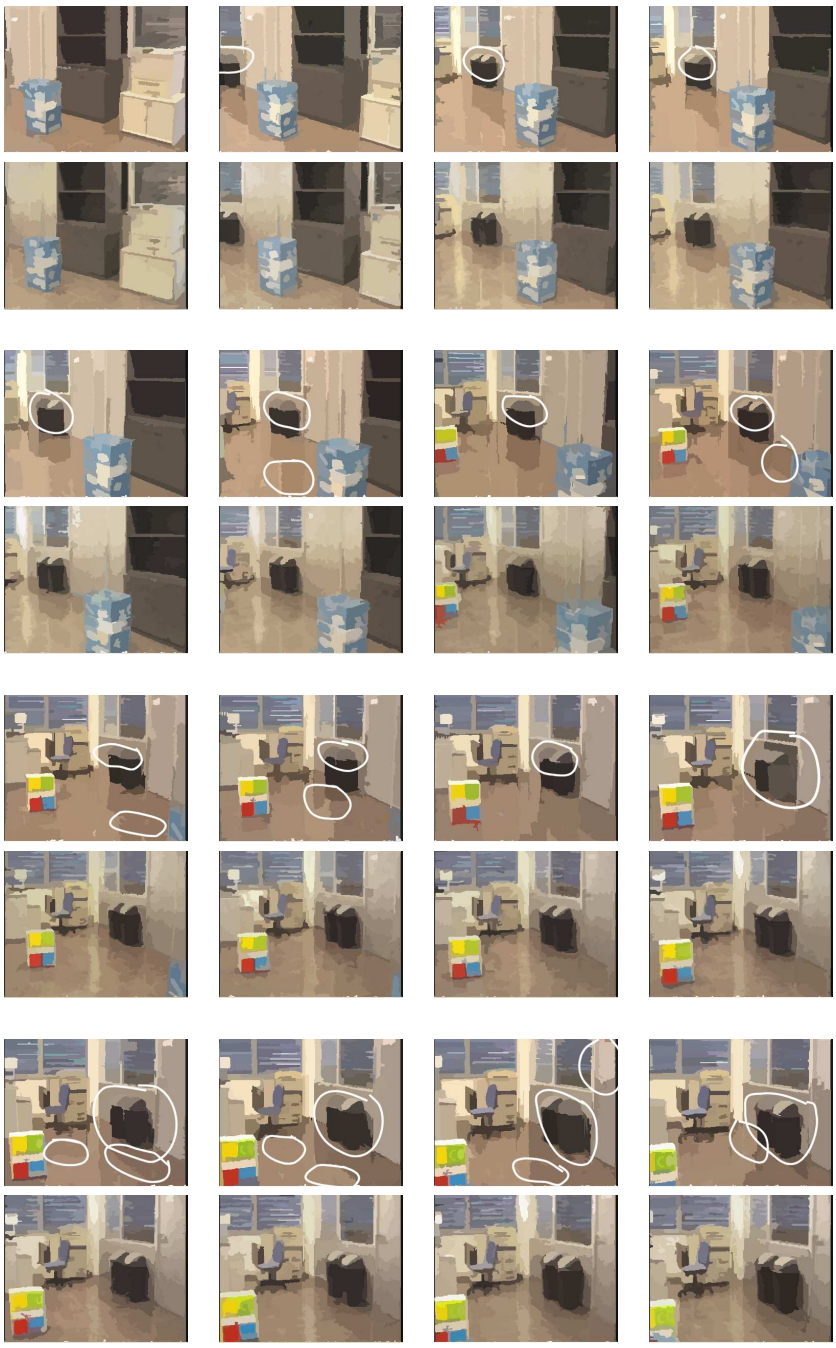

Figure 6.9: Images from the video sequence segmented using our algorithm and RGB color space. Upper row: independent images. Lower row: component correspondence.

pendently considered are placed in the upper rows. The lower rows are reserved to segmentations after applying the component correspondence. White circles have been painted around some areas in the upper row of Fig. 6.8 and Fig. 6.9 to focus on the regions that shift back and forth uncertainly compared to those in the lower row, which remain far stabler.

Despite it is difficult to catch this behaviour at once in paper, what we must understand from these results is that some areas in Fig. 6.8 and Fig. 6.9, such as those corresponding to doors, the floor, and the pair of black wastepaper baskets, present a *swinging* segmentation, since some regions are differently joined in two consecutive frames.

This bad consequence of subsegmenting images mainly occurs in poorly defined regions and is greatly reduced by component correspondence, as it can be appreciated in the lower rows of Fig. 6.8 and Fig. 6.9. These results are even better than those exhibited in the lower row of Fig. 6.7 corresponding to the case of Figueiredo's routine being fed with Gaussian distributions from previous steps. And what is more important, images get segmented in far less time.

6.7 Conclusions

As a conclusion to this Chapter, we claim that the problem of segmenting color images is faced, no matter their origin is static or from a video sequence, in a way that both coherent and stable segmentations are sought. For us, coherence means that components in a segmentation must correspond as close as possible to actual regions of the segmented scene, whereas stability has to do with the existence of components through time in a sequence, meaning that two consecutive frames must generate similar segmentations where corresponding components encompass similar areas in the scene.

To that purpose we suggest a *greedy* algorithm based on the computation of the minimum spanning tree which grows components attending to local properties of pixels. The process is fully controlled by an energy function that estimates the probability whether two components may be put together or not. Spurious regions that are helplessly generated during the growing process are removed accordingly to a quality index identifying such class of regions. Hence, a fast algorithm is achieved providing image segmentations that are good enough for identification purposes, as will be seen later in Chapter 7.

The segmentation algorithm is additionally extended to handle sequences in order to get stabler segmentations through time. For each new frame, this job is done by propagating forward the segmentation in the previous image, i.e, regions which get joined in a frame forming a bigger component are matched to other segments in the posterior frame by way of a distance that weights both *position* and *color appearance*, and then, these segments are grouped into a new component. Thus, it is granted that a pair of corresponding components in two consecutive frames of the sequence look similar.

Results show that segmentations using the Felzenszwalb&Huttenlocher's algorithm [FH98a], from which our method is inspired, have been improved and are similar in coherence and stability to those achieved by Figueiredo's EM in [FJ02], though being far faster. Furthermore, our segmentation algorithm will be used in the next Chapter to obtain the segmentations needed to carry out a set of experiments related with image retrieval and object recognition.