UNIVERSITE DE NICE - SOPHIA ANTIPOLIS
UFR Sciences

École Doctorale Sciences et Technologies de l'Information et de la Communication

Thèse

pour obtenir le titre de
Docteur en Sciences
de l'Université de Nice - Sophia Antipolis

Mention: INFORMATIQUE

présentée et soutenue par

Vincent MARTIN

Cognitive Vision: Supervised Learning for Image and Video Segmentation

Thèse dirigée par Monique THONNAT
équipe d'accueil : ORION – INRIA Sophia-Antipolis

Soutenue publiquement le 19 Décembre 2007
devant le jury composé de :

M.	Michel	BARLAUD	Pr, UNSA, France	Président
Mme	Rita	CUCCHIARA	Pr, University of Modena, Italy	Rapporteur
M.	Markus	VINCZE	Pr, University of Vienna, Austria	Rapporteur
M.	Régis	CLOUARD	MCF, Université de Caen, France	Examinateur
M.	Benoit	GEORIS	PhD, Keeneo SAS, France	Examinateur
Mme	Monique	THONNAT	DR, INRIA Sophia Antipolis, France	Directrice

Acknowledgments

I would like to thank Pr. Rita Cucchiara and Pr. Markus Vincze for accepting to review my PhD thesis. I want to thank them for their very pertinent advices and remarks.

Merci à Régis Clouard pour sa participation au jury. J'adresse également tous mes remerciements à Benoit Georis pour sa participation à ce jury de thèse. Merci à Michel Barlaud d'avoir accepté de présider ce jury.

Merci à Monique Thonnat pour m'avoir accepté dans son projet et avoir su si bien diriger ma thèse. J'espère pouvoir mettre à profit tout ce qu'elle a pu me transmettre.

Un grand merci à Catherine Martin pour sa disponibilité et son aide précieuse pour toutes les questions administratives.

Merci à Sabine Moisan pour sa bonne humeur, son aide et ses conseils pertinents.

Merci à François Brémond pour sa gentillesse et son intérêt toujours présent pour mes travaux de recherche.

Merci à Jean-Paul Rigault pour son attention, sa disponibilité et ses récits et autres anectodes qui ont animé nos discussions.

Merci à Paul Boissard pour son enthousiasme et pour avoir toujours cru en nos travaux communs.

Je tiens à remercier tout particulièrement Bernard Boulay, mon co-bureau durant toutes ces années passées à l'INRIA. Il tient une place importante dans la réussite de cette thèse. Ce fut un plaisir de travailler à ses côtés et de profiter de ses talents notamment en informatique.

J'adresse mes chaleureux remerciements à Nadia et Valery Valentin pour leur gentillesse et leur générosité sans limite.

Je remercie de manière plus générale tous les membres de l'équipe ORION : Annie, Anh-Tuan, Etienne, Lan, Luis, Marcos, Mohammed, Ruiha, Suresh ainsi que tous les anciens que j'ai pu croisé : Alberto, Benoit, Céline, Christophe, Julien, Florence, Florent, Gabriele, Magalie et Nicolas. Ils ont su faire régner une ambiance studieuse et détendue qui a largement contribué à mon épanouissement dans l'équipe.

Merci aux personnels de l'INRIA et notamment du SEMIR pour leur compétence et leur services.

Merci à Marie-Hélène pour m'avoir supporté pendant les moments difficiles et

pour avoir partagé avec moi les moments de joie. J'espère pouvoir continuer ainsi avec elle.

Enfin, je remercie profondément ma famille pour m'avoir toujours soutenu dans mes projets depuis tout jeune.

Abstract

In this thesis we address the problem of image and video segmentation with a cognitive vision approach. More precisely, we study two major issues of the segmentation task in vision systems: the selection of an algorithm and the tuning of its free parameters according to the image contents and the application needs. We propose a learning-based methodology to easily set up and continuously adapt the segmentation task.

Our first contribution is a generic optimization procedure to automatically extract optimal algorithm parameters. The evaluation of the segmentation quality is done with regards to reference segmentations. In this way, the user task is reduced to provide reference data of training images, as manual segmentations.

A second contribution is a twofold strategy for the algorithm selection issue. This strategy relies on a training image set representative of the problem. The first part uses the results of the optimization stage to perform a global ranking of algorithm performance values. The second part consists in identifying different situations from the training image set and then to associate a tuned segmentation algorithm with each situation.

A third contribution is a semantic approach to image segmentation. In this approach, we combine the result from the previously (bottom-up) optimized segmentations to a region labelling process. Regions labels are given by region classifiers which are trained from annotated samples.

A fourth contribution is the implementation of the approach and the development of a graphical tool currently able to carry out the learning of segmentation knowledge (automatic parameter optimization, region annotations, region classifier training, and algorithm selection) and to use this knowledge to perform adaptive segmentation.

We have tested our approach on two real-world applications: a biological application (detection and counting of pests on rose leaves) for the static segmentation part, and video surveillance applications for the video figure-ground segmentation part. Results, quantitative evaluations, and comparisons with non-adaptive segmentations are presented to show the potential of our approach.

For the segmentation task in the biological application, the proposed adaptive segmentation approach over performs a non-adaptive segmentation in terms of segmentation quality and thus allows the vision system to count the pests with a better precision.

For the figure-ground video segmentation task, the main contribution of my approach takes place at the context modelling level. By achieving dynamic background model selection based on context analysis, my approach allows to enlarge the scope of surveillance applications to high variable environments.

The main limitation of my approach is its lack of adaptation to unforeseen situations. An improvement could be to use continuous learning technique to adapt the segmentation to new situations.

keywords: Image segmentation, video segmentation, cognitive vision, machine learning, segmentation performance evaluation, optimization techniques.

Résumé

Dans cette thèse, nous abordons le problème de la segmentation d'image dans le cadre de la vision cognitive. Plus précisément, nous étudions deux problèmes majeurs dans les systèmes de vision : la sélection d'un algorithme de segmentation et le réglage de ses paramètres selon le contenu de l'image et les besoins de l'application. Nous proposons une méthodologie reposant sur des techniques d'apprentissage pour faciliter la configuration des algorithmes et adapter en continu la tâche de segmentation.

Notre première contribution est une procédure d'optimisation générique pour l'extraction automatique des paramètres optimaux des algorithmes. L'évaluation de la qualité de la segmentation est faite suivant une segmentation de référence. De cette manière, la tâche de l'utilisateur est réduite à fournir des données de référence pour des images d'apprentissage, comme des segmentations manuelles.

Une seconde contribution est une stratégie pour le problème de sélection d'algorithme. Cette stratégie repose sur un jeu d'images d'apprentissage représentatif du problème. La première partie utilise le résultat de l'étape d'optimisation pour classer les algorithmes selon leurs valeurs de performance pour chaque image. La seconde partie consiste à identifier différentes situations à partir du jeu d'images d'apprentissage (modélisation du contexte) et à associer un algorithme paramétré avec chaque situation identifiée.

Une troisième contribution est une approche sémantique pour la segmentation d'image. Dans cette approche, nous combinons le résultat des segmentations optimisées avec un processus d'étiquetage des régions. Les étiquettes des régions sont données par des classifieurs de régions, eux-mêmes entrainés à partir d'exemples annotés par l'utilisateur. Une quatrième contribution est l'implémentation de l'approche et le développement d'un outil graphique dédié à l'extraction, l'apprentissage, et l'utilisation de la connaissance pour la segmentation (modélisation et apprentissage du contexte pour la sélection dynamique d'algorithme de segmentation, optimisation automatique des paramètres, annotations des régions et apprentissage des classifieurs).

Nous avons testé notre approche sur deux applications réelles : une application biologique (comptage d'insectes sur des feuilles de rosier) et une application de vidéo surveillance. Pour la première application, la segmentation des insectes obtenue par notre approche est de meilleure qualité qu'une segmentation non-adaptative et permet donc au système de vision de compter les insectes avec une

meilleure précision. Pour l'application de vidéo surveillance, la principal contribution de l'approche proposée se situe au niveau de la modélisation du contexte, permettant d'adapter le choix d'un modèle de fond suivant les caractéristiques spatio-temporelles de l'image. Notre approche permet ainsi aux applications de vidéo surveillance d'élargir leur champ d'application aux environnement fortement variables comme les très longues séquences (plusieurs heures) en extérieur.

Afin de montrer le potentiel et les limites de notre approche, nous présentons les résultats, une évaluation quantitative et une comparaison avec des segmentations non-adaptatvie.

mot-clés : Segmentation d'image, segmentation de vidéos, vision cognitive, techniques d'apprentissage, évaluation de la segmentation, techniques d'optimisation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations

This thesis deals with image segmentation in vision systems. Image segmentation consists in grouping pixels sharing some common characteristics. In vision systems, the segmentation layer typically precedes the semantic analysis of an image. Thus, to be useful for higher-level tasks, segmentation must be adapted to the goal, i.e. able to effectively segment objects of interest. The very first problem is that a unique general method still does not exist: depending on the application, algorithm performances vary. This is illustrated in Figure C.1 where two different algorithms are applied on the same image. The first one seems to be visually more efficient to separate the ladybird from the leaf. The second one produces too many regions not very meaningful.

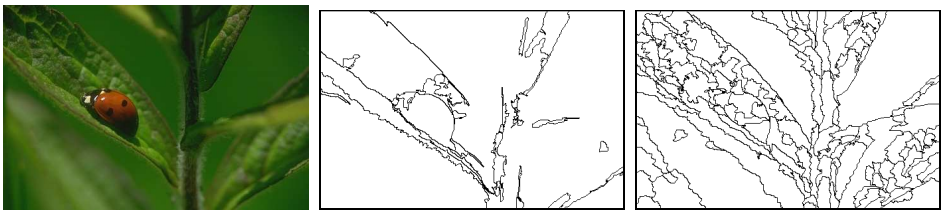


Figure 1.1: An example of the segmentation of an image with two different algorithms. The first algorithm forms regions according to a multi-scale color criteria while the second uses a local color homogeneity criteria.

Basically, two popular approaches exist to set up the image segmentation task in a vision system. A first approach is to develop a new segmentation algorithm dedicated to the application task. A second approach is to empirically choose an existing algorithm, for instance by a trial-and-error procedure. The first approach leads to develop an *ad hoc* algorithm, from scratch, and for each new application. The second approach does not guarantee adapted results and robustness. So, a need exists for developping a new approach to the **algorithm selection** issue. When facing different algorithms, this approach should be able to automatically

choose the one best suited with a segmentation goal.

When designing a segmentation algorithm, internal parameters (e.g., thresholds or minimal sizes of regions) are set with default values by the algorithm authors. In practice, it is often up to an image processing expert to supervise the tuning of these free parameters to get meaningful results. As seen in Figure C.2, it is not clear how to choose the best parameter set regarding the segmented images: the first one is quite good but several parts of the insect are missing; the second one is also good, since the insect is well outlined, but too many meaningless regions are also present. However, complex interactions between free parameters make the behavior of the algorithm fairly impossible to predict. Moreover, this awkward task is tedious and time-consumming. Thus, the **algorithm parameter tuning** is a real challenge. To solve this issue, optimization methods should be investigated in order to automatically extract optimal parameters.

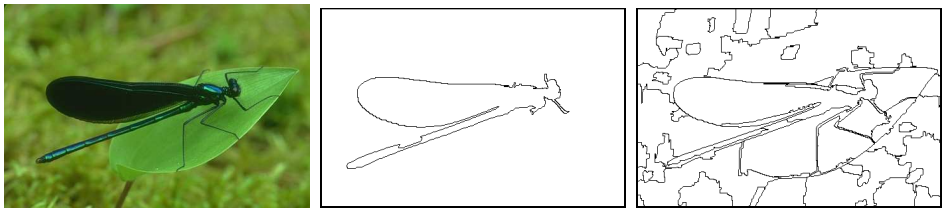


Figure 1.2: Illustration of the problem of algorithm parameter tuning. An image is segmented with the same algorithm (based on color homogeneity) tuned with two different parameter sets.

In real world applications, when the context changes, so does the appearance of the images. This is particularly true for video applications where lightning conditions are continuously varying. It can be due to local changes (e.g., shadows, reflections) and/or global illumination changes (due to meteorological conditions), as illustrated in Figure C.3 where images are extracted from the same scene at different hours of the day. The consequences on segmentation results can be dramatic. This **context adaptation** issue emphasizes the need of automatic adaptation capabilities.

Figure 1.3: illustration of the problem of context variations on a video surveillance application.

1.2 Objectives

My objective is to propose a cognitive vision approach to the image segmentation problem. More precisely, we aim at introducing learning and adaptability capacities into the segmentation task. Traditionally, explicit knowledge is used to set up this task in vision systems. This knowledge is mainly composed of image processing programs (e.g., specialized segmentation algorithms and post-processings) and of program usage knowledge to control segmentation (e.g., algorithm selection and algorithm parameter settings). To this end, three main issues of image segmentation task in vision systems should be solved:

- The first issue is to extract optimal parameters of segmentation algorithms in order to obtain a segmentation adapted to the segmentation task, i.e. a goal-oriented segmentation. The tuning of segmentation algorithm parameters is known to be a tricky task and often requires image processing skills. So, our objective is threefold: first, we want to automate this task in order to alleviate users' effort and prevent subjective results. Second, the fitness function used to assess segmentation quality should be generic (i.e. not application dependent). Third, no *a priori* knowledge of segmentation algorithm behaviors is required, only ground truth data should be provided by users.

- Once all the algorithms have been optimized, a second issue is to select the best one. The selection strategy should be based on a quantitative evaluation of each algorithm performance. However, when images of the application domain are highly variable, it remains quite impossible to achieve a good segmentation with only one tuned algorithm. In this case, a selection strategy depending on the image content analysis should be preferred.

- In many computer vision systems at the detection layer, the goal is to separate the object(s) of interest from the image background. When objects of interest and/or image background are complex (e.g. composed of several sub-parts), a low-level algorithm cannot achieve a semantic segmentation, even if optimized. For this reason, a third issue is to refine the (optimized) segmentation to provide a semantically meaningful segmentation to higher vision modules.

Our final objective is to show the potential of our approach through two different segmentation tasks in real-world applications.

- The first segmentation task we focus on is image segmentation in a biological application related to early pest detection and counting. This implies to robustly segment the objects of interest (mature white flies) from the complex background (rose leaves). Our goal is to demonstrate that the cognitive vision platform coupled with our adaptive segmentation approach achieves a better detection rate of white flies than tuned with an *ad hoc* segmentation.

- The second segmentation task we focus on is figure-ground segmentation in a video surveillance application. The goal is to detect moving objects (e.g., walking people) in the field of view of a fixed video camera. Detection is usually carried out by using background subtraction methods. However, illumination changes make the background modeling problem difficult. Our objective is to show that a dynamic selection of background model allows to enlarge the scope of surveillance applications to high variable environments.

1.3 Context of the Study

This work takes place in the Orion project-team at INRIA Sophia Antipolis Méditerranée, France. Orion is a leading team in scene understanding at the frontier of computer vision, knowledge-based systems, and software engineering. Orion has a cognitive vision approach. It aims to achieve robust, resilient, adaptable computer vision functionalities by endowing them with a cognitive faculty. This means the ability to learn, to adapt, and to weight alternative solutions, and develop new strategies for detection, recognition, and interpretation. Recently, Hudelot [Hudelot, 2005] proposed a cognitive vision platform for semantic image interpretation. This platform is based on the cooperation of three knowledge-based systems of which one is dedicated to the intelligent management of image processing programs. Maillot [Maillot, 2005] has endowed this platform with learning facilities and ontology-based semantic knowledge representation and management for object recognition. Currently, the detection layer of the platform rely on *ad hoc* segmentation. This means that all the segmentation operators have been tuned deep in code once and for all. In this context, my work aims to enrich this cognitive vision platform at the image segmentation level to enable adaptive segmentation.

1.4 Contributions

My main contribution is to propose a cognitive vision approach to image segmentation by solving the issues listed above:

- I propose a generic optimization procedure to automatically extract optimal algorithm parameters. This procedure is based on three independent components: a segmentation algorithm with one or several free parameters to tune, a performance evaluation metric, and an optimization algorithm. The evaluation of the segmentation quality is done with regards to a reference segmentation (e.g. manual segmentation). The performance evaluation metric is generic, has a low-computational cost, and can be used for a broad range of segmentation purposes. In this way, the user task is reduced to provide reference data: manual segmentations of training images.

- I propose two strategies for the algorithm selection issue. These strategies use the results of the optimization stage applied on a training image set

representative of the problem. The first one is based on a global ranking of algorithm performance values. The second strategy is to identify different situations, called contexts, from the training image set and to associate a tuned segmentation algorithm with each context.

- I also propose an approach to semantic image segmentation. In this approach, we consider the segmentation refinement problem as a region labelling problem. It is hence designed for region-based segmentation algorithms only. The goal is to assess the membership of each region to a pre-defined set of regions sharing the same label. The assessment relies on a preliminary supervised learning stage where region-classifiers are trained with training samples. The role of the user is to label the regions of the ground truth segmentations. The originality of this approach is twofold. First, we use the optimized segmentations as input of the region-classifiers. Second, the sub-tasks of the learning process, namely region feature extraction, region feature selection, and classifier training, are automatically optimized in a wrapper scheme to get the best classification performances.

In the scope of the two previously described segmentation tasks, my contributions are the following:

- For the segmentation task in the biological application, the proposed adaptive segmentation approach overperforms the *ad hoc* segmentation in terms of segmentation quality and thus allows the system to count the pests with a better precision.

- For the figure-ground segmentation task, my main contribution takes place at the context modeling level. By achieving dynamic background model selection based on context analysis, my approach allows to enlarge the scope of surveillance applications to highly variable environments.

Each step of the proposed approach is tested and evaluated on several image data sets. This helps us to show the strengths and the limitations of the approach in terms of performance, computational cost, and sensitivity to key parameters.

1.5 Outline

This manuscript is structured as follows. Chapter 2 introduces the reader to image segmentation in the context of computer vision systems. We propose an overview on four topics closely related to our problem: image segmentation in computer vision systems, segmentation approaches, performance evaluation, and segmentation optimization. Chapter 3 introduces the proposed approach, and gives our objectives and assumptions for the different segmentation issues. Chapter 4 details each step of our approach: algorithm parameter optimization, algorithm selection, and semantic region labelling. Chapter 5 is dedicated to the validation of the approach for a real world application. In particular, we are interested in

the segmentation step of a cognitive vision system dedicated to the recognition of biological organisms. In chapter 6, we present how our approach can be used for the adaptive figure-ground segmentation in video surveillance applications. Concluding remarks and suggestions for future work are discussed in chapter 7.

Chapter 2

State of the Art

2.1 The place of the Image Segmentation Task in Vision Systems

In the beginning of the eighties, Marr [Marr, 1982] proposed a theory of the human perceptual vision. This theory is the first complete methodology for the design of information systems. He suggested three levels of abstraction for the analysis of such complex systems:

The computational level: it describes what is the goal of the system. It has a more abstract nature than the next two levels and specifies all informational constraints necessary to map the input data into the desired output.

The algorithmic level: it states how the computational theory can be carried out in terms of methods. It is related to the specification of algorithms with their input and output representations.

The implementational level: it describes how an algorithm is embodied as a "physical" process. It has the lowest description level, e.g. the hardware implementation and the software code.

An important characteristic of this reconstructive approach of vision is the increasing number of solutions while decreasing the abstraction level. For example, there are several algorithms to solve the computational task "edge detection", and there are many possible ways to implement each of them.

Inspired from the Marr's theoretical framework, most existing artificial visual recognition systems, called vision systems, follow the paradigm depicted in Figure 2.1. An image is first pre-processed in order to highlight information which is important for the next stages. Classically, it often refers to the segmentation task. Then, the descriptor mapping module encodes the remaining low-level data into a symbolic form more appropriate for the recognition and analysis stage, which finally identifies the image content.

This architecture has yet one drawback: errors in the first stages, e.g., in the segmentation, will be further propagated into later stages, degrading the quality

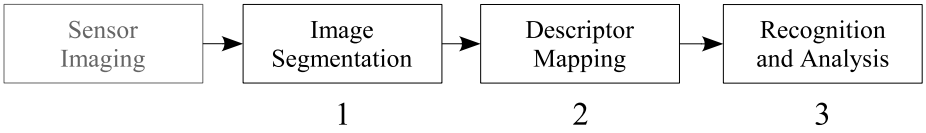


Figure 2.1: The three stages of visual processings usually found in vision systems.

of the whole system. Thus, great attention has been directed to the problem of segmentation. Hundreds of publications in this field appear every year, each trying to find an optimal solution for one specific application or for general purposes. However, a unified, generally accepted definition of image segmentation does not yet exist. Most authors agree on the following facts about segmentation:

- its task is to partition the image into several segments or regions (this point will be developed in section 2.2.1);
- it is an early processing stage in computer vision systems. Within the computational model for computer vision (Figure 2.1), it belongs to the preprocessing module;
- it is one of the most critical tasks in automatic image analysis.

2.1.1 Knowledge-Based Approaches

Early approaches in vision systems use explicit knowledge to define the segmentation task. In [Nazif and Levine, 1984], an expert system for low-level image segmentation is proposed. The system is based on hundreds of production rules that manipulate combinations of regions and lines obtained from two basic segmentation algorithms. Another example can be found in the SIGMA system [Matsuyama and Hwang, 1990] which uses a low-level vision expert module dedicated to handle segmentation and feature extraction tasks for aerial image understanding. One weakness of these systems is their application dependency. The knowledge acquisition necessary to build the rules is also a big problem.

Then, researchers have tried to conceive more versatile systems by incorporating verification and knowledge acquisition components. In [Ossola, 1996], an approach based on the cooperation of two knowledge-based systems (KBS) is presented. Program supervision techniques [Moisan and Thonnat, 1995] are used to process images in an intelligent way, e.g. to dynamically set up the segmentation task with respect to variable conditions. A general program supervision architecture contains three main parts: a library of programs, a knowledge base, and a reasoning engine. The reasoning engine is in charge of selecting and scheduling the programs of the library which are best satisfying a user query. The engine iterates the following loop composed of four steps, until a satisfactory solution is reached: planning (e.g., selection of programs), execution (e.g., initial parameter setting), evaluation (e.g., assessment of output results), and repair (e.g., adjustment of

some parameters). The knowledge base contains a declarative representation (i.e. frame and production rules) of the programs called operators. These operators are hierarchically organized in several levels of abstraction and can be primitives or composites (i.e. combination of several primitives) ones. We can cite the OCAPI environment in [Clément and Thonnat, 1993] as a general tool for the development of KBS dedicated to the supervision of programs. The strength of the program supervision architecture is the ability to reuse programs for various applications as demonstrated in [Crubézy, 1999] for the supervision of medical imagery programs or in [Thonnat, 2002] for the recognition of complex objects.

A related approach for the automatic generation of image processing applications called BORG can be found in [Clouard et al., 1999]. By opposition to the program supervision approach, the system uses hierarchical and opportunistic behavior in order to construct a solution plan. A plan is represented by an action graph of five fixed levels: requests, tasks, functionalities, procedure, and operators. Each level corresponds to a more or less coarse version of the solution. The system dynamically constructs a parametrized plan from an initial user's query. A drawback of this approach is that the action graph is constrained to a fixed number of levels supposed to cover all the solution space and thus limits the flexibility for modeling a problem.

One advantage of knowledge-based approaches is the semantic richness which enables user-friendly interaction with the end-users. Nevertheless, one drawback is that they are application dependent and thus requires a strong expertise in the domain to build the knowledge bases: they are thus limited to a close world.

2.1.2 Learning Approaches

This section deals with the use of decision theory as a basis for intelligent image processing. The main idea is to reduce as much as possible the role of the human expertise in the building of vision systems by machine learning techniques. This principle was introduced by Draper [Draper et al., 1996] who argues that KBS are too *ad hoc* and too dependent on human expertise during their design. Indeed, the use of explicit knowledge is not really suited for modeling the variability, the changes, and the complexity of the world.

Case-Based Reasoning (CBR) is a problem solving approach which solves new problems by adapting previously successful solutions to similar problems. In particular, the case based approach has been used for algorithm parameter learning. Some interesting works can be found in [Ficet-Cauchard et al., 1999] and [Frucci et al., 2007]. A case contains an image, contextual information (as image acquisition information) and algorithm parameters. Finding the best segmentation for the current image is done by retrieving similar cases in the case base. Similarity is computed using non-image and image information. The evaluation is done by a measure of dissimilarity between the original image and the segmented image. If the evaluation is bad, the learning module is activated to build a new case. The main advantage of case based reasoning systems lies in the easiness of their reasoning strategies. Nevertheless, the choice of an adequate

representation of cases is an application dependent problem.

In [Peng and Bahnu, 1998], an adaptive integrated image segmentation and object recognition system is proposed and applied to recognize cars in outdoor imagery. The authors stress the importance of the adaptability to real world changes of the segmentation problem, in order to improve the interpretation process. They propose to use the model matching confidence degree as feedback to influence the segmentation process. A team of stochastic learning automata is used to represent both global and local image segmentation. Reinforcement learning is applied to close the loop between model matching and image segmentation. The main advantage of reinforcement learning is that it only requires knowledge of the goodness of the system performance rather than details on the algorithm. As a consequence, their method is independent of any segmentation algorithm but dependent of the recognition algorithm.

2.1.3 Towards Cognitive Vision

From the previous described approaches, two open problems still remain: first, knowledge acquisition bottleneck when a large amount of knowledge is needed and, second, lack of robustness when faced with varying conditions. Thus, classical vision systems are often brittle. To overcome this brittleness, a new discipline called cognitive vision has recently emerged; a research road-map can be found in [ECVISION, 2005]. A cognitive vision system is defined by its ability to reason from *a priori* knowledge, to learn from perceptual information, and to adapt its strategy to different problems. This new discipline thus involves several existing related ones (computer vision, pattern recognition, artificial intelligence, cognitive science, etc.). Some systems have started to implement cognitive vision ideas, mainly for human behavior recognition relying on different technologies. For example, in [Vincze et al., 2006] a cognitive system combining low-level image components and high-level activity reasoning processes has been developed to recognize human activities. This system integrates various techniques such as connectionism, Bayesian networks, component framework, and robotics. A cognitive vision platform has been proposed in [Hudelot and Thonnat, 2003] for the recognition of complex natural objects in images with reusable components. The authors propose an original distributed architecture based on three KBS for the interpretation, the anchoring, and the image processing levels. Concerning the image processing KBS, they propose an image processing ontology which is application independent but dependent on the data structures of a library of programs. Program supervision techniques are used to manage the knowledge of programs. Finally, in their conclusion, they stress the need of integrating machine learning techniques for image segmentation to reduce the necessary program supervision knowledge and to improve the robustness of the semantic image interpretation.

2.1.4 Discussion

We have presented the segmentation task through computer vision approaches. We have seen that segmentation is a crucial task and demands strong efforts to vision system designers in building complex and exhaustive knowledge bases. However, KBS are not approved unanimously by the computer vision research community. As Draper said [Draper et al., 1996], we must avoid to build *ad hoc* systems, based on close world assumptions. Even if program supervision techniques gain to be used for enabling control and reuse of vision algorithms, they still fail to adapt themselves to unknown situations. The cognitive vision approach has been recently introduced to achieve more robust, reusable, and adaptable computer vision systems. This approach aims at endowing vision systems mostly with learning and adaptability facilities. In this context, the segmentation task has several challenges to be tackled: starting from a generic solution (e.g., from a default parametrization), algorithms can be dynamically tuned by means of learning techniques to reach the specific goal defined by the user.

To fully understand the segmentation problem, a first and essential task is to draw a state-of-the-art on existing approaches. This is the role of the next section.

2.2 Segmentation Approaches

Many segmentation methods are based on two basic properties of the pixels in relation to their local neighborhood: discontinuity and similarity. Methods based on some discontinuity property of the pixels are called boundary-based methods, whereas methods based on some similarity property are called region-based methods. Before it can be properly stated, some fundamental concepts have to be specified.

2.2.1 Definition of Image Segmentation

Image segmentation can be formalized through its region-based definition as follows:

Definition 1 (Image region) *An image region $\mathcal{R}$ is a non-empty subset of the image I, such that $\mathcal{R} \subseteq I, \mathcal{R} \neq \emptyset$*

A region does not need to be topologically connected. The existence of an unbroken path from one region element (i.e. a pixel) to another one inside the region is sufficient.

Definition 2 (Image partition) *A partition of I is a set of n regions $\mathcal{R}_i, i = 1, \ldots, n$ such that $\bigcup_{i=1}^{n} \mathcal{R}_i = I$ and $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, $\forall i \neq j$*

This definition states that the partition has to cover the whole image and two regions cannot overlap.

Definition 3 (Image segmentation) *For a certain defined homogeneity predicate $\mathcal{H}$, a segmentation $\mathcal{S}$ of I is a partition of I which satisfies: $\mathcal{H}(\mathcal{R}_i) = 1, \forall i$ and $\mathcal{H}(\mathcal{R}_i \cap \mathcal{R}_|) = \emptyset$ for $\mathcal{R}_i$ and $\mathcal{R}_j$ adjacent, $i \neq j$.*

The first condition states that each region has to be homogeneous with respect to the predicate $\mathcal{H}$. The second condition states that two adjacent regions cannot be merged into a single region that satisfies the predicate $\mathcal{H}$.

The nature of the predicate $\mathcal{H}$ is the key-element of the definition of segmentation. It can be based only on pixel values, or it can judge the high-level relevance of the partition. Since the solution is not unique, this makes the segmentation an ill-posed problem in the sense of Hadamard. Then, to solve the problem, a solution consists in defining the segmentation, i.e. defining a predicate $\mathcal{H}$, for each level of abstraction. Figure 2.2 depicts possible segmentation results at each level of Marr's computational model. At the image-based level, pixels are grouped according to their feature values (e.g., their gray value). The surface-based level detects surfaces, but not objects; for example the background keeps its patches. The object-based level detects a region per object.

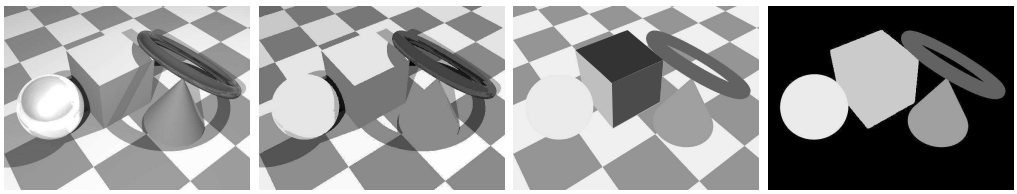


Figure 2.2: Ideal segmentation results at different levels of Marr's vision computational model. From left to right: original image, image-based level, surface-based level, and object-based level.

2.2.2 Static Image Segmentation

Several surveys of segmentation techniques have been published. Three of them [Pal and Pal, 1993, Skarbek and Koschan, 1994, Lucchese and Mitra, 2001] review about 300 publications giving a fair overview of the state-of-the-art in segmentation at the image-based processing level. Pal and Pal [Pal and Pal, 1993] mainly evaluate algorithms for gray-valued images and introduce three of the first attempts to exploit color information.

Skarbek and Koschan [Skarbek and Koschan, 1994] concentrate their survey on color image segmentation. They classify the algorithms according to the underlying concepts of the homogeneity predicate $\mathcal{H}$ and identify four categories: pixel-based, area-based, edge-based and physics-based approaches. Pixel-based approaches consider a region as homogeneous, if the features of its elements belong to the same cluster in the feature-space. Area-based techniques define a region as a set of connected pixels obtained for instance by growing from seeds, by joining smaller pixel blocks or by splitting non-uniform regions. The third,

edge-based group, defines regions as those sets of pixels delimited by inhomogeneities or discontinuities. This is the complementary concept to area-based segmentation. Physics-based methods include knowledge about physical properties of the image formation process to improve the detection of regions corresponding to object surfaces. Physics-based methods are categorized in the current work as surface-based techniques. They do not belong to the image-based stage, since all additional knowledge about physical properties of object surfaces cannot be regarded as part of a low-level homogeneity predicate, but rather as external higher level information about the analyzed scene.

Lucchese and Mitra [Lucchese and Mitra, 2001] also review exclusively color segmentation approaches and use a similar categorization: feature space based, image domain based and physics based techniques. The combination of area and edge-based methods into one image domain class makes more sense nowadays, since many modern approaches try to satisfy both concepts simultaneously.

2.2.2.1 Feature-Space Based Approaches

Feature-space approaches generally neglect spatial relationships between image pixels and analyze exclusively the configuration of their feature values. Algorithms in this category delimit sections in the feature space and assign the same region label to all image pixels falling into the same section. Two principles are common. The first one finds sections detecting peaks in unidimensional or multidimensional feature histograms. The second one uses traditional clustering algorithms.

Histogram thresholding

Historically, histogram thresholding is one of the first used technique for segmenting images. Gray-level images histograms can be commonly decomposed into peaks and valleys which characterize objects and backgrounds. A good survey on these techniques can be found in [Sahoo et al., 1988]. Early methods for color segmentation work with several one-dimensional histograms, which implies that the correlation between different dimensions is ignored. More recent algorithms work in two or three dimensional color spaces and are characterized by different techniques to robustly detect peaks and their corresponding boundaries in the feature space. The choice of the color representation often plays a major part. An additional problem of this approach is the usually required smoothing of the feature space in order to keep the size of data structures tractable. Many algorithms search for peaks by approximating the histograms with a mixture of Gaussian, and fail if this assumption does not hold (a fact that, in real images, is almost always the case).

Clustering techniques

Clustering approaches can be interpreted as unsupervised classification methods. Several concepts are based on the k-means and fuzzy c-means clustering algorithms applied on different color and texture spaces. One of the major drawbacks

of the original clustering methods is that the number of clusters (k) must be known *a priori*. Several heuristics have been suggested to compute k automatically based on some image statistics. A well-known clustering-based segmentation algorithm is the meanshift [Comaniciu and Meer, 2002] approach which introduces a method to automatically detect different bandwidths from the data for each section of the feature space. The major drawback of this concept is its computational cost compared to simple k-means approaches. The generalization of the k-means algorithm for color images including spatial constraints is introduced in [Chang et al., 1994]. This algorithm considers the segmentation as a maximum *a posteriori* probability estimation problem. The algorithm starts with global estimates and progressively adapts the cluster centers to the local characteristics of each region.

2.2.2.2 Image-Domain Based Approaches

Another way to cope with the image-based segmentation problem is to compare the feature values of each pixel in the image-domain, i.e. pixels are compared within predefined spatial neighborhoods. Two major groups of algorithms can be identified: the first one defines regions through the feature similarity between their elements (area-based approaches). The second one identifies feature discontinuities as boundaries between homogeneous regions (edge-based approaches). Many modern segmentation strategies try to satisfy both concepts simultaneously [Munoz et al., 2003].

Region Growing techniques

Traditional area-based techniques utilize one of two principles: region growing or split-and-merge. Region growing methods assume the existence of some seed-points, to which adjacent pixels will be added if they fulfill a homogeneity criterion. An extensive review is detailed in [Fan et al., 2005]. The main advantage of these methods is the creation of spatially connected and compact regions, which contrast with the usually noisy image partition obtained with pure feature-based segmentation approaches. They are frequently applied to separate one single homogeneous object (e.g., background) from the rest of the image, but using several seeds positioned at different objects it is also possible to perform more sophisticated segmentations. The required seed selection is a subtask of this approach, which can be solved by taking advantage of clustering methods or morphological operations, among others.

Split-and-Merge techniques

Split-and-merge algorithms proceed to successively divide an image into smaller non-overlapping regions while some similarity criterion is not met. A common data structure used to implement this procedure is the quadtree representation which is a multi-resolution scheme. Delauney triangulation [Gevers and Smeulders, 1997] or Voronoi diagrams [Itoh and Matsuda, 1996]

are also employed as an alternative technique to the rigid rectilinear nature of the quadtree structure. The end result of the splitting is an over-segmented image. A merging procedure is then applied to join neighboring regions under the same homogeneity predicate that was used for splitting. The comparison between adjacent regions can use simple statistics or can be based on more elaborated mathematical models, like Markov Random Fields (MRF), which also permit merging regions of similar texture [Panjwani and Healey, 1995].

Edge based techniques

Edges are discontinuities in the feature characteristics (e.g., intensity) of adjacent pixels. The magnitude of the gradient of a gray-valued image has been typically employed, since it is a relatively robust edgeness representation form. Its approximation for discrete digital images has been analyzed in detail in the past. Most methods involve the use of well-known convolution kernels, like the Roberts, Robinson, Prewitt, Kirsch, or Sobel operators. The detection of edge pixels is just the first stage of any edge-based segmentation approach. Further processing is necessary in order to provide a valid segmentation as stated by Definition 3. Since standard detectors like Canny's [Canny, 1986] or SUSAN [Smith and Brady, 1997] usually leave some gaps between object boundaries, some mechanisms are required to fill them appropriately. Recently, a new generation of edge detectors based on the Earth Mover's Distance have been proposed [Ruzon and Tomasi, 2001]. They show a better performance due to their capability to detect junctions and corners. However, their computational cost is very high compared to traditional techniques. A classified and comparative study of edge detection algorithms can be found in [Sharifi et al., 2002].

Morphological watershed segmentations [Vincent and Soille, 1991] can also be categorized as an edge-based approach. They work on a topographical edgeness map, where the probability of a pixel to be an edge is modeled by its altitude. A "flooding" step begins which fills the valleys with water. The watershed lines are detected when the water of two different valleys encounters. The principal advantage of the watershed segmentation scheme over other edge based techniques is that it generates closed boundaries. The regions defined by the closed boundaries represent an over-segmentation of the image, since the algorithm is sensitive to noise. If the gradients are computed at successively higher scales, the number of local minima (i.e. flood basins) in the gradient magnitude image will decrease. The available techniques work on gray-valued images obtained usually as the gradient of the intensity.

Active contour models, also known as "snakes", is another family of edge-based algorithms [Kass et al., 1988, Ronfard, 1994]. An interesting and powerful property of an active contour model is its ability to find subjective contours and interpolate across gaps in edge chains. An active contour model represents an object boundary or some other salient one dimensional image feature as a parametric curve that is allowed to deform from some arbitrary initial positions towards the desired final contour. The problem of finding this initial contour is cast as an

energy minimization problem with the intention that it yields a local minimum of the associated energy functional. The original model incorporates two internal energy terms related to contour smoothness and regularity. Active contour models are well-adapted for segmenting objects in noisy images but they require *a priori* knowledge of the object shapes. Good illustrations of such algorithms are frequently found in medical applications such as in [Jehan-Besson et al., 2004] and in optical flow segmentation as in [Herbulot et al., 2006].

Hybrid Approaches

All previous methods have intrinsic drawbacks that can be partially compensated by combining different techniques. For instance, clustering methods detect homogeneous regions in the feature space. However, since spatial relationships are ignored, the region boundaries in the image-domain are highly irregular. In recent years, numerous techniques for integrating region and boundary information have been proposed. A detailed review of techniques to combine area-based and edge-based approaches can be found in [Munoz et al., 2003]. One of the main features of the hybrid approaches is the timing of the integration: embedded in the region detection, or after both processes are completed. The most common way to perform integration in the embedded strategy consists of incorporating edge. Region growing and split-and-merge are the typical region-based segmentation algorithms [Zugaj and Lattuati, 1998]. Post-processing integration is based on fusing results from single segmentation methods, attempting to combine the map of regions (generally with thick and inaccurate boundaries) and the map of edge outputs (generally with fine and sharp lines, but dislocated) with the aim of providing an accurate and meaningful segmentation. Another example of hybrid approach can be found in [Chen and Wang, 2004] which combines color and texture-based segmentations using border refinement.

2.2.2.3 Object Based Approaches

While the image-based approach has been dealt with a relative success, the challenge of aggregating pixels into segments representing meaningful parts of objects is much difficult. In fact, segmentation is also closely related to the problem of extracting object from images. One of the oldest approaches to object-based segmentation is template matching. The idea of template matching is to create a model of an object of interest (the template, or kernel) and then to search over the image of interest for objects that match the template. The simplest methods, based on correlation or comparable matching operators, can only determine the position of the template. The main difficulty in this technique stems from the large variability in the shape and appearance of objects within a given class. Consequently, the segmentation may not accurately delineate the object's boundary.

A recent development in this area is presented in [Borenstein and Ullman, 2004] then updated in [Borenstein and Malik, 2006].

The proposed approach relies on learnt patches from training image samples and a *bottom-up* process used to derive a segmentation graph. Partial templates are used to detect object parts of a given class (horses in the experiment) by matching to the segmentation graph, even though the global appearance of the objects in the test images slightly differs from the learnt material. The methods become more complex and time consuming if further parameters like orientation or scale need to be estimated. Since the number of objects and their orientation in an image are unknown in the current application, the search space for matching approaches becomes intractable.

In [Schnitman et al., 2006], an approach inducing semantic segmentation from examples is described. They argue that determining whether an entity belongs to a particular semantic part is easier done at the fragment level than on a pixel-by-pixel basis. Starting from an example, patch sets representing a collection of homogeneous fragments are built. Then, a test image is first over-segmented and the labelling of each fragment is induced from the minimization of a global labelling cost. They apply the graph-cuts multi-label optimization technique for finding the globally optimal labelling. Since this example-based approach allows to use a non-parametric model of the object's parts, they assume that the learnt fragment-label pairs are representative of the possible image variations, i.e. illumination, resolution, and scale characteristics. Finally, they concede that their approach is only appropriate for images depicting closely similar scenes. A similar approach is described in [He et al., 2006] where a probabilistic model assigns labels to each region of an over-segmented image based on local, global, and pairwise features. As depicted by the author, their model accuracy is limited by the reliance and the amount on training data.

2.2.2.4 Summary

In this section, we have presented a didactic survey on image segmentation techniques. The goal of this review was to familiarize the reader with classical techniques rather than to give an extended review of all existing algorithms. To give an overall view, a summary is drawn up in Table 2.1, inspired by the one presented in [Alvarado Moya, 2004a].

Finally, we can conclude this study by making some important remarks, closely akin to the conclusions of [Skarbek and Koschan, 1994] in their survey:

1. General purpose algorithms are not robust and usually not algorithmically efficient.

2. All techniques are dependent on parameters, constants and thresholds which are usually fixed on the basis of few experiments. Tuning and adapting parameters is rarely performed.

3. As a rule, authors ignore comparing their novel ideas with existing ones.

4. As a rule, authors do not estimate the algorithmic complexity of their methods.

Feature Space	+ Detection of homogeneity in a global context. − Spatial relationship between pixels is ignored.		
	Histogram	+ Multiple 1D histogram methods are computationally inexpensive − Noise sensitive. − 1D approaches ignore correlation between different feature space dimensions. − Models used to fit histograms (e.g., multi-gaussians) usually do not correctly match the real distributions. − Limited to binary segmentation problems.	
	Clustering	+ Simultaneous consideration of all dimensions of the feature space. + Suitable for color and texture segmentation. + Relatively efficient algorithms exist. − Size or number of clusters must be known *a priori*.	
Image Domain	+ Produce smoother and more accurate region boundaries than feature space-based approaches. − Edge detectors falls into the edge linking problem.		
	Area-based	+ Creation of connected compact regions. + Fast algorithms available. − Key-parameters tuning can be a tricky task.	
		Region growing	+ Suitable for segmentation of complex objects having homogeneous background. − Prior information on optimal number and position of seeds may be needed. − Result depends on order in which pixels are examined
		Split & Merge	+ Fast and flexible implementation. − Traditional tessellation mechanisms produce too coarse spatial quantization artifacts.
	Edge-based	Edge detectors	+ Accurate local discontinuity detection − Sensitive to noise and parameter changes.
		Watershed	+ Detection of closed contours. − Image is often over-segmented.
		Snakes	+ Robust to noise. − Difficult automatic initialization of the contour.
Hybrid	+ Combination of several methods can be appropriately adapted to the needs of each application. − High computational cost.		
Object-based	+ Combine *top-down* and *bottom-up* approaches to achieve semantically meaningful segmentation. − Robustness relies on the learning capacities from examples or patches. − Applicability is restricted by appearance constraints on objects such shape and scale.		

Table 2.1: Comparison between different image segmentation techniques.

5. It seems that separating processes for region segmentation and for object recognition is the reason of failure of general purpose segmentation algorithms.

6. Several different color spaces are employed for image segmentation. Nevertheless, no general advantage of one of the color spaces with regard to the other color spaces has been found yet.

2.2.3 Image Sequence Segmentation

Identifying moving objects from a video sequence is a fundamental and critical task in video applications such as video surveillance, traffic monitoring and analysis, human detection, tracking, and gesture recognition. We have seen in previous sections that segmenting semantically meaningful components of a static image is fairly impossible with conventional approaches based on primitive grouping. In image sequences, it is more practical to segment moving objects from dynamic scene with the aid of motion information contained in it. The goal of this section is to give an overview of existing techniques devoted to the segmentation of moving objects in image sequences. We limit our review on techniques devoted to segment video frames captured from a single static camera. A more detailed review of segmentation of moving objects in image sequence can be found in [Zhang and Lu, 2001] and [Cheung and Kamath, 2004].

Optical Flow

The displacement or optical flow of a pixel is a vector representing the motion between a pixel in one frame and its correspondent pixel in the following frame [Barron et al., 1994]. Traditionally, the motion calculation is pixel-based, exploiting the gradient cues [Horn and Schunck, 1992, Nagel and Gehrke, 1998, Stiller and Konrad, 1999]. An advantage of this technique is that it can be used even in presence of camera motion. A drawback of this technique is that the computation of derivatives for each pixel is often required, thus making the method computationally expensive. First-order motion sensors also suffer from the aperture problem, which means that they can detect motion only perpendicular to the orientation of the contour that is moving. While segmentation based on finding flow discontinuities is straightforward, it is unlikely to achieve expected results without combination with a spatial segmentation technique. However, alternatives to this pixel-based approach exist. For instance, in [Coimbra et al., 2003] the authors propose to use MPEG-2 motion vectors as a basis for obtaining the motion field. Then, they apply specific rules and filters to obtain a smooth motion field. In consequence, this alternative is able to work in real-time conditions and to achieve region-based segmentation.

Background Modeling

A common approach for identifying the moving objects is background subtraction, where each video frame is compared against a reference or a background model. Pixels in the current frame that deviate significantly from the background are considered to be moving objects. These "foreground" pixels are further processed for object localization and tracking. In order to distinguish between relevant changes due to motion of objects or brightness changes, and irrelevant temporal changes due to noise, the frame difference has to be compared to a threshold T. The reliable decision, that a spatial position (x, y) belongs to a changed region, is only possible if the frame difference exceeds this threshold. Basically, the background model at each pixel location is based on the pixel's recent history. In many work, such history is just the previous n frames, or a weighted average where recent frames have higher weight. The background model is computed as a chronological average from the pixels history.

Basic methods consider background as the average or the median [Prati et al., 2003] of the previous n frames. If this method is rather fast, the memory requirement is $n \times size(frame)$. Without no more memory requirements, background is often modeled as the running average:

$$B_{t+1}(x, y) = \alpha * F_t(x, y) + (1 - \alpha) * B_t(x, y) \tag{2.1}$$

where α is the learning rate typically equals to 0.005, $B_t(x, y)$ is the background model value at the position (x, y) and at the time t, and $F_t(x, y)$ is the current pixel value at the same position. In order to prevent the background model to be polluted by foreground pixels, the running average can be achieved with a selectivity criteria:

$$B_{t+1}(x, y) = \begin{cases} \alpha * F_t(x, y) + (1 - \alpha)B_t(x, y) & \text{if} F_t(x, y) \text{background} \\ B_t(x, y) & \text{otherwise} \end{cases} \tag{2.2}$$

Based on a single value, the previous techniques fail to model multiple modal background distributions. To cope with this issue, probability density functions (pdf) of the background model can be estimated by fitting one Gaussian distribution $G(\mu, \sigma)$ over the histogram [Wren et al., 1997]. In that way, the background pdf is updated by the running average on μ and σ. This technique has been extended to deal with multimodal background histograms by using generalized Mixture of Gaussians (MoG) [Stauffer and Grimson, 1999] where the number of modes is arbitrarily pre-defined (usually from three to five). An incremental *expectation-minimization* (EM) algorithm is used to learn and update the parameters $(\mu_{i,t}, \sigma_{i,t}, \omega_{i,t})$ of the model, where $\omega_{i,t}$ is the portion of the data accounted for by the i-th component. If MoG are widely used, the tuning of initial parameters remains a difficult task. Moreover, depending on the learning rate and the number of gaussians, MoG faces problems to find the best trade-off between adaptability to fast variations versus robustness.

To overcome the weaknesses of MoG-based approaches, a non-parametric approach based on kernel density estimators (KDE) is proposed

in [Elgammal et al., 2000] and updated in [Mittal and Paragios, 2004]. The background pdf is given by the histogram of the n most recent pixel values, each smoothed with a Gaussian kernel. This technique is able to adapt very quickly to changes in the background model and to detect targets with high sensitivity. However, this non-parametric approach cannot be used when long-time periods are needed to sufficiently sample the background due mostly to memory constraints.

Among the widely-used techniques, we can also cite those based on Kalman filtering [Wren et al., 1997] and Meanshift estimation [Han et al., 2004]. Another interesting background subtraction approach is proposed in [Kim et al., 2005]. In this paper, inspired by Kohonen [Kohonen, 1989] Self-Organizing Map (SOM), sample background values at each pixel are quantized into codebooks which represent a compressed form of background model for a long image sequence. The quantification criteria is based on a color distortion metric together with brightness bounds. The clusters represented by quantized values (called codewords) do not necessarily correspond to a single Gaussian or other parametric distributions. The codebook representation has the advantage to be efficient in memory and speed compared with the previously described approaches. The major problem of these techniques is that no explicit method is provided to choose the foreground threshold or to tune the initial parameters. Moreover, no spatial correlation is used between different (neighboring) pixel locations.

Spatio-temporal Approaches

To address this issue, an integrated region and pixel-based information approach to background modeling is discussed in [Cristani et al., 2002]. The goal is to combine a region-based static segmentation to a pixel-based motion segmentation in an adaptive manner in order to better manage sudden changes in background. In [Seki et al., 2003], the authors exploit the strong correlation of image variations at neighboring image blocks to narrowly construct their background model. In [Toyama et al., 1999] a three level background maintenance system is proposed. The pixel level component performs Wiener filtering to make probabilistic predictions of the expected background; the region-level component fills in homogeneous regions of foreground objects; and the frame-level component detects sudden, global changes in the image and swaps in better approximations of the background. If spatio-temporal approaches overperform motion-based approaches in difficult situations, they are much slower due to the computation time needed by their spatial segmentation components and are sensitive to key-parameters tuning.

2.2.4 Discussion

We have reviewed the different families of image sequence segmentation algorithms. A summary with performance comparison is given in Table 2.2. It can be seen from this table that all robust segmentation algorithms are achieved at

the cost of high computation complexities. The codebook approach seems to be well-adapted to cope with both mixture of gaussians and kernel density estimators issues but require a tedious parameter tuning stage to obtain optimal results. Combining spatial and temporal features is the key-element for improving image sequence segmentation. However, more research is needed to improve robustness against environment noise with an acceptable computational cost.

Approaches		Strength	Weakness	Computation Complexity
Optical Flow		motion detection accuracy	aperture problem, assumption on local smoothness	high
Background Modeling	Running Average	simple to implement, fast	aperture problem	low
	MoG	able to model complex backgrounds	parameters tuning, fail to model fast variations	medium
	KDE	high sensitivity, quick adaption, non-parametric	limited by memory constraints	high
	Codebooks	non-parametric, fast, robust	parameters tuning	low
Spatio-temporal		meaningful segmented regions	slow	high

Table 2.2: Comparison between different image sequence segmentation techniques.

2.3 Segmentation Performance Evaluation

Considering the increasing number of segmentation algorithms, the problem of performance segmentation evaluation becomes a primordial task. Two reasons motivate this statement: researchers must be able to compare their algorithm to another ones, and end-users must be able to choose an algorithm depending on the problem to solve. Usually, segmentation results are visually assessed by the algorithm's designer, which only allows subjective and qualitative conclusions on the algorithm performance. A generic method for the segmentation evaluation task does not exist, but many approaches have been proposed and can be classified into two principal classes: unsupervised methods and supervised methods (see Figure 2.3). The first class gathers the methods which do not require any *a priori* knowledge of segmentation results to evaluate. Their principle consists in estimating empirical criteria based on image statistics. The second class groups together evaluation methods based on *a priori* knowledge as a reference segmented image, usually named a ground truth (GT). A good survey of all these methods can be found in [Zhang, 1996] and in [Rosenberger et al., 2005].

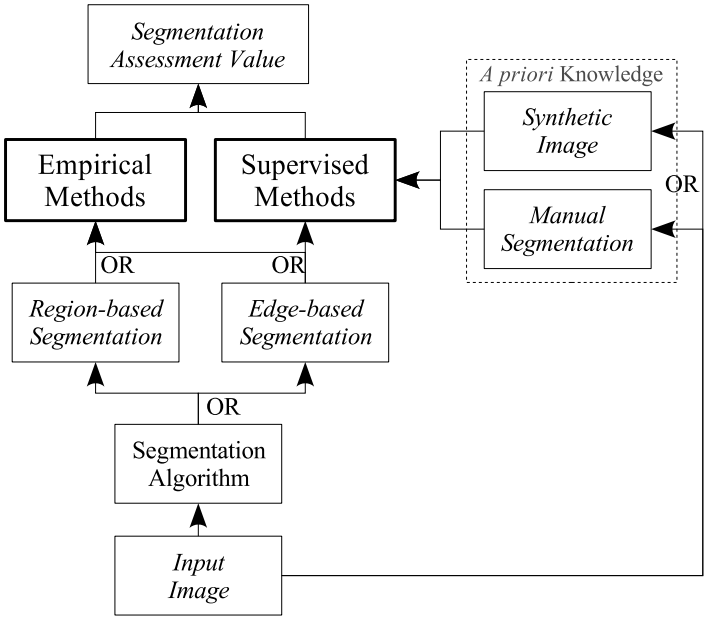


Figure 2.3: Segmentation evaluation diagram starting from an input image and returning a segmentation assessment value.

The first goal of this section is to give a non-exhaustive overview of classical and popular methods in order to draw their advantages and drawbacks. The ultimate goal is to show that among the vast number of proposed approaches, some of them could be applied to reach both a good level of genericity for algorithm performance comparison and a good level of flexibility to fit the user's requirements.

2.3.1 Unsupervised Methods

2.3.1.1 Empirical Methods

Empirical methods rely on quantitative criteria from segmented images. Most of these criteria are established in accordance with the definition of (a good) region-based segmentation which is based on the inter-region variability and the intra-region homogeneity. Among the most used and cited methods in the literature, we present hereafter some of them.

In [Weszka and Rosenfeld, 1978], the authors proposed an intra-region uniformity criterion that quantify the effect of noise to evaluate thresholded images. Based on the same idea, Levine and Nazif also defined in [Levine and Nazif, 1985] a criterion that calculates the uniformity of a region characteristic (e.g., gray-level, color, etc.) based on the variance of this characteristic. Another criterion to measure the intra-region uniformity was developed by Pal and Pal [Pal and Pal, 1989]. It is based on a thresholding that maximizes the local second-order entropy of regions in the segmentation result. In the case of slightly textured images, these

criteria of intra-region uniformity prove to be effective and very simple to use. However, the presence of textures in an image often generates improper results due to the overinfluence of small regions. Complementary to the intra-region uniformity, Levine and Nazif [Levine and Nazif, 1985] defined a disparity measurement between two regions to evaluate the dissimilarity of regions in a segmentation result. Borsotti et al. [Borsotti et al., 1998] identified this limitation of Liu and Yang's evaluation criterion [Liu and Yang, 1994] and modified it, so as to more strictly penalize the segmentation results presenting many small regions as well as heterogeneous ones. These modifications permit to make the criterion more sensitive to small variations of the segmentation result. Zeboudj [Zeboudj, 1988] proposed a measure based on the combined principles of maximum inter-regions disparity and minimal intra-region disparity measured on a pixel neighborhood. This criterion has the drawback of not correctly taking into account strongly textured regions. Considering the types of regions (textured or uniform) in the segmentation result, Rosenberger presented in [Rosenberger, 1999] a criterion that enables to estimate the intra-region homogeneity and the inter-regions disparity. Recently, [Zhang et al., 2004] proposed an objective evaluation metric based on information theory. The method uses the entropy as the basis for measuring the uniformity of pixel characteristics (the luminance in the paper) within a segmentation region. However, since entropy is a global measure, it does not consider local information or incorporate any measure about the shapes of the regions themselves.

2.3.1.2 Summary

The major advantage of unsupervised methods is that they do not require the intervention of an expert, just the definition of a metric of quality measure by the user is needed. Thus, these methods are totally automatic. However, defining a metric that could match all the segmentation objectives defined by the user is not a tricky task. Hence, quality measures are at best heuristic, since no specific knowledge of object(s) to segment is available. In general, the authors note that such methods are not well-adapted for textured images because the texture properties and the application are closely linked. Authors also point out that a bias exists when using these methods for the assessment of algorithms based on the same technique: for instance, the evaluation of a segmentation algorithm relying on a criterion of intra-region uniformity with an evaluation metric based on the same criterion will inevitably produces biased measures. This tends to consider unsupervised performance evaluation method not very pertinent.

2.3.2 Supervised Methods

Reference segmentations are achieved generally by hand or by generating synthetic images. In the last case, the ground truth data are objective and precise, in the contrary of subjective and imprecise hand-made expert drawing. These methods try to determine how far the actually segmented image is from the reference image

in a quantitative manner.

2.3.2.1 Region-based Methods

Yasnoff et al. proposed in [Yasnoff et al., 1977] an intuitive set of classification error measures computed from the pixel-wise class confusion matrix and based on comparison of both pixel class proportions and spatial distributions. The Vinet's distance [Vinet, 1991] is also a well-known measure. This distance computes a dissimilarity measure between two segmentations based on the maximal covering of regions. However, it does not take into account all the information (among others the spatial dispersion of pixels). It assumes that two regions can be matched if they have a maximal number of common pixels. This hypothesis is restrictive and favors big regions.

2.3.2.2 Edge-based Methods

For edge-based methods, there are three discrepancy measures (under-detection, over-detection, and localization error) between edge pixels of the segmented image and edge pixels of the ground truth. One of the most used method is the empirical measure proposed by Pratt [Pratt et al., 1978] based on the distance between an edge pixel in the segmented image and the closest one in the ground truth. This measure is not sensible to under-detection errors and to erroneous shape but is sensible to over-detection and localization errors. Odet [Odet and Benoit-Cattin, 2002] proposes an interesting scalable divergence measure for a multi-scale error evaluation of binary segmentation by using a spatial notion.

2.3.2.3 Multi-objective Methods

[Correia and Pereira, 2000] proposed a method which relies on the evaluation of each region by verifying several similarity conditions relating to shape, geometric, edge pixel, and region content statistics. All theses measures are weighted in order to match with human evaluation results. In the same way, [Mezaris et al., 2003] proposed an objective evaluation metric which takes into account not only the accuracy of the boundary localization but also the under and over-segmentation effects. In [Martin et al., 2004], the authors define an interesting metric based on global and local consistency between segmentation of different granularity (i.e. refinement). The correspondence procedure is tolerant to small localization errors and achieve good results in the collection of the Berkeley image database. In the same way, Cardoso [Cardoso and Corte-Real, 2005] introduces, in a strong theoretical study, two metrics based on symmetric and asymmetric distances. Everingham [Everingham et al., 2002], more than defining a new measure, attempts to aggregate fitness functions using the Pareto front. A solution is said to be Pareto optimal if it is not dominated by any other solution in the search space. In complex search spaces wherein an exhaustive search is infeasible, it is very difficult to guarantee Pareto's optimality. Therefore, instead of the true set of optimal

solutions (Pareto Set), one usually aims to derive a set of non-dominated solutions as possible (Pareto Front) of the Pareto Set. More recently, [Zhang et al., 2006] proposed a meta-evaluation framework in which any set of base evaluation methods are combined by a machine learning algorithm that coalesces their evaluations based on learnt weighted decision trees. The learning component tailors its performance to a particular set of images through the training data, which is composed of a set of examples, each of which includes a raw image, two segmentations, and a label, provided by a human, indicating which of the segmentations is the best one. After the labelling of each training image, a decision tree is computed for each base evaluator. Each internal node in a decision tree is set by considering different partitions of global image features (e.g., based on the LUV color space, wavelet coefficient) and segmentation attributes (e.g., number of regions, texture features, etc.). This method improves the evaluation accuracy compared to the stand-alone evaluators but it also requires heuristic knowledge of global image features to extract (which color space, which texture features, etc.?) and thresholds for the tree partitioning. Finally, objective evaluation of video segmentation quality has also been studied. The main difference with static image segmentation evaluation is the use of temporal accuracy and stability measures. A good review can be found in [Correia and Pereira, 2003].

2.3.2.4 Summary

The use of a ground truth is double-edged: it makes this class of methods potentially the most general and the less biased but this also supposes that ground truth are easily available. From this study, it also clearly appears that multi-objective methods yield better results than stand-alone methods (edge-based or region-based). However, the manner to combine measures remains an issue.

2.3.3 Discussion

If we take a look at the number of publications around the segmentation evaluation problem, we can see that at present, this number is about one thousand concerning the segmentation algorithms, one hundred concerning the evaluation methods, and does not raise ten concerning the comparison of evaluation methods. If more efforts have been recently put on segmentation evaluation, it is still difficult to define wide-ranging performance metrics and statistics. Several explanations justify this limitation: (1) no common mathematical model or general strategy for evaluation is available especially for analytic methods, (2) no single evaluation can cover all aspects of segmentation algorithms, (3) appropriate ground truths are hard to determine objectively. Then, to overcome such limitations, potential research directions may explore methods combining multiple metrics in an effective manner (e.g., using learning) and methods considering the final goal of the segmentation.

Research is currently underway in terms of using these metrics as a mean to optimize parameters within a segmentation algorithm or to select the best

adapted algorithm. This involves to use an optimization procedure which is also a challenge in the context of image segmentation. The next section discusses this issue.

2.4 Segmentation Optimization

In this section, we address the problem of segmentation optimization by means of algorithm parameter tuning and algorithm selection. We first draw up a background on optimization techniques, then relate some optimization approaches used to optimize the segmentation by parameter tuning and algorithm selection.

2.4.1 Background on Optimization Techniques

The basic goal of an optimization process is to systematically find the values of real or integer variables that minimize or maximize an objective function (see an example on Figure 2.4). This result is called an optimal solution. There are many optimization algorithms available (more than four thousands). However, some methods are only appropriate for certain types of problems. It is important to be able to recognize the characteristics of a problem and identify an appropriate solution technique. Within each class of problems, there are different minimization methods, which vary in computational requirements, convergence properties, and so on. Optimization problems are classified according to the mathematical characteristics of the objective function, the constraints, and the decision variables. Interesting surveys on optimization techniques can be found in [Fletcher, 1987] and more recently in [C. A. and P. M., 1996] and [Bliek et al., 2001]. In this subsection, we do not intend to draw up a complete review on optimization techniques but rather to summarize methods with special emphasis on the ones compatible with our segmentation optimization purpose.

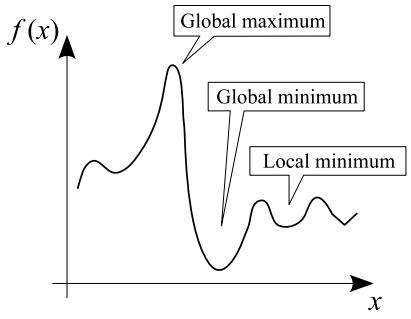


Figure 2.4: Simple unconstrained optimization

The main elements of any optimization problem are:

Variable(s): The variables usually represent tunable free parameters of the problem. They are not known when the problem starts.

Objective function: This is the mathematical expression that combines the variables to express the goal. The objective function will be either maximized or minimized in order to find the best solution of the problem (see Figure 2.4).

Constraint(s): In the case of a constrained problem, the constraints are mathematical expressions that combine the variables to express limits on the possible solutions. For example, they may express that the value of the variable x_1 should always be smaller than the variable x_2.

Formally, an optimization problem can be described by:

$$\min / \max_{x \in X} f(x), \quad f : \mathbb{R}^n \to \mathbb{R}, \quad X \subseteq \mathbb{R}^n$$

where $X = x_1, \ldots, x_n$ is a n-dimensional variable and f is the objective smooth function to minimize/maximize.

The search of function extrema is equivalent to solve a system of n equations with n variables, linear or not:

$$\frac{\partial f}{\partial x_i}(x_1; \ldots; x_n) = 0 \tag{2.3}$$

A linear function subject to linear constraints defines a *linear programming* [1] (LP) optimization problem stated in the form:

$$\min_{x \in X} \{c^T x : Ax = b,\ x \geq 0\} \tag{2.4}$$

where $c \in \mathbb{R}^n$ is the cost vector and $A \in \mathbb{R}^{n \times m}$ is the constraint matrix. The feasible region described by the constraints is a polytope, or Simplex, and at least one member of the solution set lies at a vertex of this polytope. If the objective function is not linear, it is a *nonlinear programming* (NLP) optimization problem. From the large literature on this subject, we can cite two recent surveys on LP [Ignizio and Cavalier, 1994] and NLP [Vasaru and Hong, 1996].

Beyond these mathematical considerations, optimization methods are also classified within some computing restrictions. When users are faced with problems for which function evaluations are very expensive (i.e. results of complex computer simulations), and/or it is not appropriate to determine derivatives directly (e.g., results from physical measurements), and/or data are noisy (e.g., the calculated value of f depends on discretization or sampling on a grid) the following strategies can be considered.

Direct Search Methods

The first, and maybe simplest, is to apply *direct search* optimization methods. This term appears in the paper [Hooke and Jeeves, 1961] but since then, has

[1] The word "Programming" is used here in the sense of "planning": the necessary relationship to computer programming was incidental to the choice of name.

become a catch-all phrase that is often applied to any optimization method that does not require an explicit representation of the gradient. Direct search methods are characterized by the absence of the construction of a model of the objective. In one hand, when the function to be optimized is smooth and its calculated values have full precision, a standard option is to use finite-differences (with a small interval) to obtain derivative estimates that are accurate enough to be treated as exact gradients in a quasi-Newton method [Gill et al., 1981]. In the other hand, this brings us to the problems of noise and nonsmoothness. The term *non-smooth* optimization is typically used in connection with functions that are discontinuous, for example, in simulating a system that undergoes a discrete change of state. The basic logic of the method is depicted in Figure 2.5.

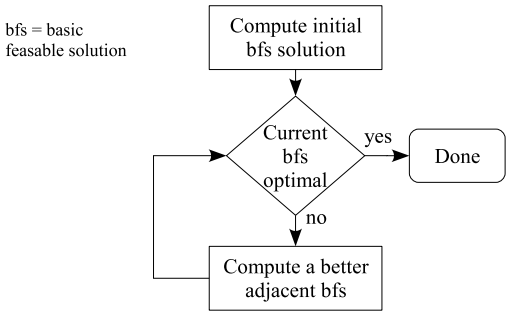


Figure 2.5: The basic Direct-search logic

The most known and widely used derivative free optimization method is the Simplex reflection algorithm of Nedler and Mead [Nelder and Mead, 1965] or its modern variants [Lewis et al., 2000]. The algorithm starts with an initial *basic feasible solution* (bfs) and tests its optimality. If some optimality condition is verified, then the algorithm terminates. Otherwise, the algorithm identifies an adjacent bfs, with a better objective value. The optimality of this new solution is tested again, and the entire scheme is repeated, until an optimal bfs is found. Since every time a new bfs is identified the objective value is improved and the set of bfs's is finite, it follows that the algorithm will terminate in a finite number of steps (iterations).

In the N-dimensional space, a Simplex is a polyhedron with $N + 1$ vertices. Starting with an initial Simplex, the method iteratively updates the worst point by four operations: reflection, expansion, contraction, and shrinkage. Figure 2.6 illustrates these operations in a three-dimensional variable space. Reflection involves moving the worst point (vertice) of the Simplex (where the value of the objective function is the highest) to a point reflected through the remaining N points. If this point is better than the best point, then the method attempts to expand the Simplex along this line. This operation is called expansion. On the contrary, if the new point is not much better than the previous point, then the Simplex is contracted along one dimension from the highest point. This proce-

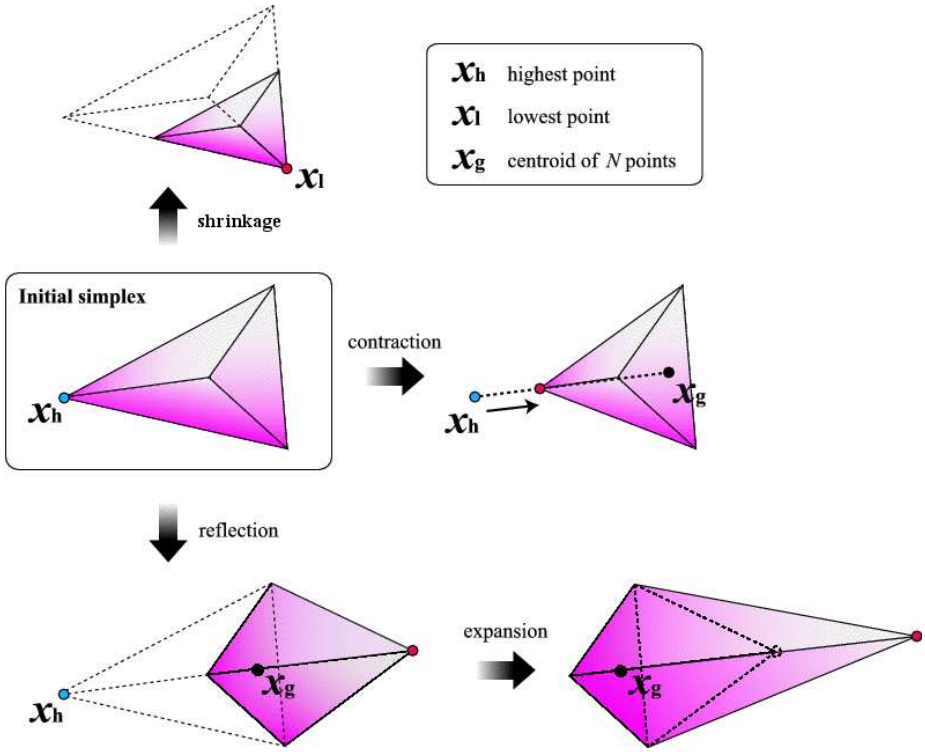


Figure 2.6: Four basic operations in the Simplex method

dure is called contraction. Moreover, if the new point is worse than the previous points, the Simplex is contracted along all dimensions toward the best point and steps down the valley. This procedure is called shrinkage. By repeating this series of operations, the method finds the optimal solution.

We give the notation used to describe formally the algorithm: Let q_o be a starting point in segmentation algorithm parameter space, and let $\lambda_i, i = l, \ldots, n$ be a set of scales. Let $e_i, i = 1, \ldots, n$ be n orthogonal unit vectors in n-dimensional variable space, let $p_O, \ldots, p_n$ be $(n+1)$ ordered points in n-dimensional variable space such that their corresponding function values satisfy $f_o \leqslant f_1 \leqslant \ldots \leqslant f_n$, let $\bar{p} = \sum_{i=0}^{n-1} \frac{p_i}{n}$ be the centroid of the n best (smallest) points, let $[p_i p_j]$ be the n-dimensional Euclidean distance from p_i to p_j, let $\alpha = \frac{[p_r \bar{p}]}{[p_n \bar{p}]}, \beta = \frac{[p_c \bar{p}]}{[p_n \bar{p}]} < 1, \gamma = \frac{[p_e \bar{p}]}{[p_n \bar{p}]} > 1, \sigma \in [0, 1]$ be the reflection, contraction, expansion and shrinkage coefficient, respectively, and let T be the threshold for the stopping criterion. For a problem with n control variables, the Nelder-Mead algorithm works as indicated in Figure 2.7.

Direct search methods and particularly the Simplex algorithm remain popular because of their simplicity, flexibility, and reliability. That is why they have been widely applied in contemporary techno-socio-economic applications. The main weakness of the Simplex algorithm is the requirement of initial parameter values

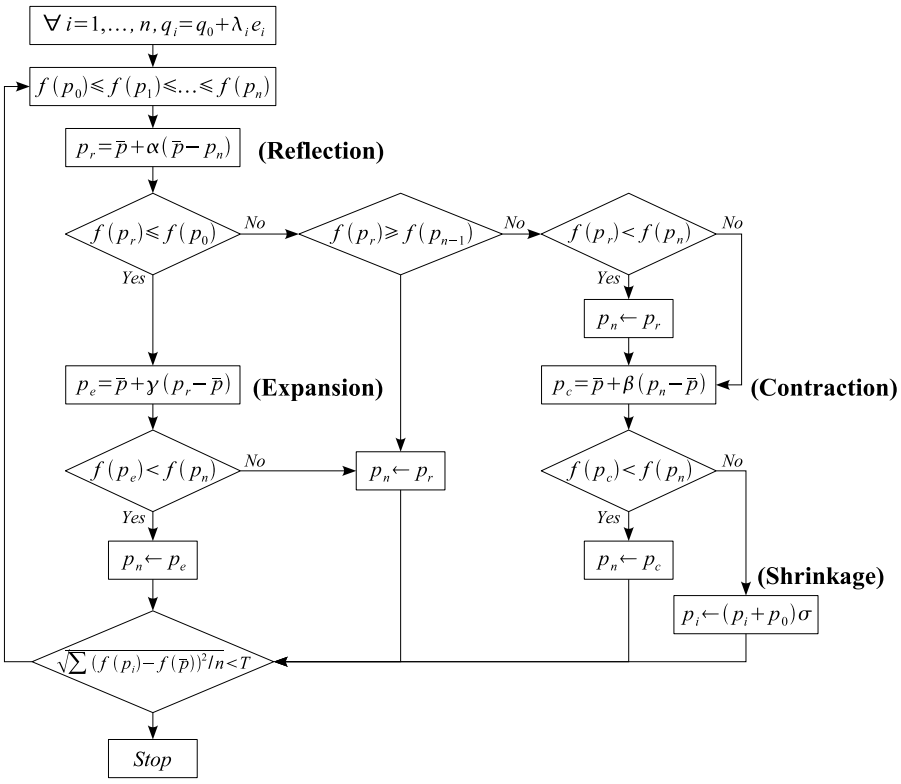


Figure 2.7: The Simplex algorithm, with its four operations of reflection, contraction, expansion, and shrinkage.

for search exploration.

Evolutionary Algorithms

Evolutionary algorithms (EAs) are adaptive heuristic search methods that take their inspiration from natural selection and survival of the fittest in the biological world. A good introduction to this area is given in [Aschlock, 2006]. EAs differ from more traditional optimization techniques in that they involve a search from a *population* (called chromosomes or the genotype or the genome) of solutions (called individuals or phenotypes), not from a single point. Each iteration of an EA involves a competitive selection that weeds out poor solutions. The solutions with high *fitness* are stochastically *recombined* with other solutions by swapping parts of a solution with another. Solutions are also *mutated* by making a small change to a single element of the solution. Recombination and mutation are used to generate new solutions that are biased towards regions of the space for which good solutions have already been seen. Pseudo-code for a genetic algorithm (GA), which is the most popular type of EA is presented in algorithm 1.

An effective GA representation and meaningful fitness evaluation are the keys of the success in GA applications. A standard representation of the solution is an

Algorithm 1: Genetic Algorithm pseudo-code

```
1: Initialize the population
2: Evaluate the fitness of the initial population members
3: repeat
4:    Select pairs from the population to be parents, with a fitness bias
5:    Copy the parents to make children
6:    Perform cross-over on the children (optional)
7:    Mutate the resulting children (probabilistic)
8:    Place the children in the population
9:    Apply genetic operators to generate new solutions
10:   Evaluate the fitness of the children
11: until some convergence criteria are satisfied
```

array of bits as string of 0s and 1s. The main property that makes these genetic representations convenient is that they facilitates simple crossover operation. Traditionally, the initial population is generated randomly, covering the entire range of possible solutions (the search space). The next operations as selection and reproduction can be more specific, depending on the nature of the application. For instance, certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming. Popular and well-studied selection methods include roulette wheel selection and tournament selection. The processes of selection and reproduction ultimately result in the next generation population of chromosomes that is different from the initial one. Generally the average fitness will have increased by this procedure, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions to keep the diversity of the population large, preventing premature convergence on poor solutions. This generational process is repeated until a termination condition has been reached. It can be based on a fixed number of generations, the highest ranking solution's fitness value or a combination of the above.

In many problems, GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. This means that it does not "know how" to sacrifice short-term fitness to gain longer-term fitness. Obviously, it depends on the shape of the fitness landscape: certain problems may provide an easy ascent towards a global optimum, others may make it easier for the function to find the local optima. To alleviate this problem, a solution is to use different fitness functions, increasing the rate of mutation, or to apply selection techniques that maintain a diverse population of solutions. To this end, it can be also quite effective to combine GA with other optimization methods. For instance, simple hill climbing techniques are quite efficient at finding absolute optimum in a limited region. Thus, alternating GA and hill climbing can improve the efficiency of GA while overcoming the lack of robustness of hill climbing.

Simulated Annealing

Simulated annealing (SA) is a generalization of a Monte Carlo approach for minimizing multivariate functions. Monte Carlo approaches are based on random walks in reference to the Monte Carlo casinos. Simulated annealing is a stochastic search method modeled according to the physical annealing process which is found in the field of thermodynamics. Annealing refers to the process of a thermal system initially melting at high temperature and then cooling slowly by lowering the temperature until it reaches a stable state (ground state), in which the system has its lowest energy. The sequence of temperatures and the number of iterations applied to thermalize the system at each temperature comprise an annealing schedule. [Kirkpatrick et al., 1983] initially proposed an effective connection between simulated annealing and combinatorial optimization, based on original work by [Metropolis et al., 1953].

To apply simulated annealing, the system is initialized with a particular configuration. A new configuration is constructed by imposing a random displacement. If the energy of this new state is lower than that of the previous one, the change is accepted unconditionally and the system is updated. If the energy is greater, the new configuration is accepted probabilistically. This is the Metropolis step, the fundamental procedure of simulated annealing. This procedure allows the system to move consistently towards lower energy states, yet still 'jump' out of local minima due to the probabilistic acceptance of some upward moves. If the temperature is decreased logarithmically, simulated annealing guarantees an optimal solution. A pseudo-code is given in Algorithm 2.

SA's major advantage over other methods is an ability to avoid becoming trapped at local minima. The algorithm uses a random search which not only accepts changes that decrease objective function f, but also some changes that increase it. The downside to SA is the need to set multiple parameters before execution: initial temperature, cooling schedule, and halting criteria. Choosing good parameters is a search task in itself. The initial temperature must be large enough to allow some freedom to make backward moves, but not too large as to become totally random exploration of the search space. An exponential cooling schedule is standard, but by no means necessary. The halting criteria is just as arbitrary as the initial temperature. Most of these settings require domain-specific knowledge about the problem to choose appropriate values.

Reinforcement Learning

Reinforcement learning (RL) is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment [Kaelbling et al., 1996]. There are two main strategies for solving reinforcement-learning problems. The first is to search in the space of behaviors in order to find one that performs well in the environment. The second is to use statistical techniques and *dynamic programming* (DP) methods to estimate the utility of taking actions in states of the world. In the standard reinforce-

Algorithm 2: Simulated Annealing pseudo-code

```
1: Select an initial state i ∈ S {S is the search space}
2: Select an initial temperature T > 0
3: Set the best state i* ← i
4: Set temperature change counter t ← 0
5: repeat
6:    Set Repetition counter n ← 0
7:    repeat
8:       Generate state j = random(S)
9:       Calculate δf ← f(j) − f(i) {f is the energy function}
10:      if δf < 0 then
11:         i ← j
12:         if (f(i) < f(i*) then
13:            i* ← i
14:         end if
15:      else if random(0,1) < exp(−δf/T) then
16:         i ← j
17:      end if
18:      n ← n + 1
19:   until n = S(t) {S is the cooling schedule function}
20:   t ← t + 1
21:   T ← T(t) {T is the temperature decrease function}
22: until stopping criteria true
23: return best found solution i*
```

ment learning model (i.e. when RL can be formulated as class of Markov decision problems), an agent is connected to its environment via perception and action. This model is schematized in Figure 2.8, on each step of interaction the agent receives as input some indication of the current state, s, of the environment T; the agent then chooses an action, a, to generate as output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar reinforcement signal, r. The agent's behavior, B, should choose actions that tend to maximize the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms. The most classical model-free algorithms are *temporal-difference* learning (TD) and *Q-learning*. TD learning is a combination of Monte Carlo ideas and dynamic programming ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like DP, TD methods update estimates based in part on other learnt estimates, without waiting for a final outcome (they bootstrap). The relationship between TD, DP, and Monte Carlo methods is a recurring theme in the theory of reinforcement learning. A good study of these approaches can be found in [Sutton and Barto, 1998] and [Kaelbling et al., 1996].

Reinforcement learning differs from the more widely studied problem of supervised learning in several ways. The most important difference is that there is no presentation of input/output pairs. Instead, after choosing an action the

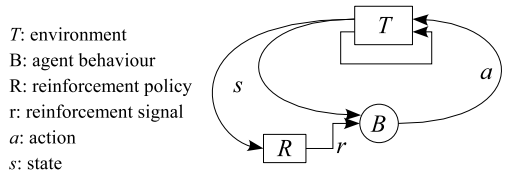


Figure 2.8: The standard reinforcement learning model.

agent is told the immediate reward and the subsequent state, but is not told which action would has been in its best long-term interests. It is necessary for the agent to gather useful experience about the possible system states, actions, transitions and rewards actively to act optimally. Another difference from supervised learning is that on-line performance is important: the evaluation of the system is often concurrent with learning.

This section has reviewed several famous optimization methods. Although this study is non-exhaustive, it will serve to appreciate the following sub-sections where a state-of-the-art on segmentation optimization is drawn up.

2.4.2 Algorithm Parameter Optimization

In this sub-section, we relate some work dealing with segmentation algorithm parameter optimization. All the following approaches rely on three independent components: a segmentation algorithm with its free-parameters to tune, a segmentation quality assessment function and a global optimization algorithm as seen in Figure 2.9.

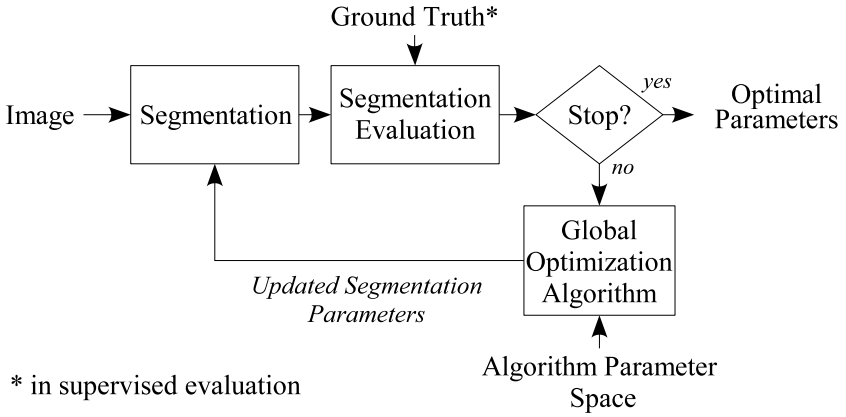


Figure 2.9: The segmentation parameter optimization framework.

In [Bahnu et al., 1995], an adaptive image segmentation system using genetic

and hybrid search (GA plus hill climbing) methods for optimal parameter extraction and learning is presented. The system incorporates a feedback loop consisting of a machine learning subsystem, a segmentation algorithm with two free-parameters, and an evaluation component which is a weighted combination of different global and local criteria. Experimental results on outdoor TV imagery are presented. The main advantage of the approach is that image features and external image variables are represented and manipulated using both numerical and symbolic forms within the generic knowledge structure. For example, this allows to construct a multiobjective evaluation function based on image color features (i.e. numerical values) and environmental factors such as the presence of rain or fog (symbolic values). Although this interesting approach is described as being very fundamental in nature, it deserves deeper experiences, i.e. to be tested on other algorithms and image databases, to fully demonstrate its potential.

Another attempt to control parameter of segmentation algorithms was conducted in [Peng and Bahnu, 1998]. The presented system applies delayed reinforcement learning to induce a mapping from input images to corresponding segmentation parameters. This is accomplished by using the confidence level of model matching as a reinforcement signal for a set of learning agents to search the optimal parameters during training. The model is a polygonal representation of the object of interest and the evaluation process is not achieved at the segmentation level but at the object recognition level.

In [Mao and Kanungo, 2000], the authors poses the automatic training of a page segmentation algorithm as an optimization problem. A textline-based performance metric is defined to construct a multivariate non-smooth function to be minimized with the Simplex algorithm. Starting from randomized initial parameters, the method founds optimal values in accordance with the objective function. The optimization is performed for the four parameters simultaneously with a number of function evaluations of 100 in mean. This makes the method suitable for other applications. Actually, this framework has been successfully applied for the parameter optimization of a video segmentation algorithm in [Gelasca et al., 2003] with a different evaluation metric based on objective spatial accuracy of regions.

In [Pignalberi et al., 2003], a genetic algorithm is used to optimize the parametrization of two range image segmentation. The objective fitness function is supervised and takes into account two levels of errors (pixel level and surface level). The tested algorithms have up to ten parameters to tune. Results obtained with the method over-performs the segmentation quality for one algorithm against default parameters, and reaches similar quality for the other one but without having any knowledge about the nature of the parameters. They have also proposed an interesting extension of the approach in [Cinque et al., 2002] where the search strategy combines two techniques in cascade: a genetic algorithm to obtain a rough seed point set and a simulated annealing to have a more precise refinement of suitable solutions. They achieve quite similar results but above all shorten the required number of iterations. However, this implies to finely tune the SA's parameters which is a tricky task.

In [Abdul-Karim et al., 2005], an automated method is presented for selecting optimal parameter settings for vessel/neurite segmentation algorithms using the minimum description length principle and a recursive random search (RRS) algorithm. It trades off a probabilistic measure of image-content coverage against its conciseness. The method is applied to 223 images of human retinas and cultured neurons, using a single algorithm with eight free parameters. Most of the improvement in segmentation quality occurs in the first hundred iterations of the RRS. However, the method is not fully automated since the user have to set a parameter which controls the trade-off between coverage and conciseness.

2.4.3 Algorithm Selection

In this section, we focus on the algorithm selection problem. Here, the goal is not to find the best parameter setting but rather to find the most suitable algorithm among several ones for a given segmentation task. Due to the still increasing number of algorithms, this problem has taken a big interest during the last decade. Basically, researchers tackle the problem with two different philosophies: model representation approach versus expert system approach.

In [Xia et al., 2005], Xia et al. make the assumption that the choice of a segmentation algorithm can be predicted from a global feature vector. In other words, this means that a relationship between algorithm behaviors and global image characteristic variations can be easily established. More precisely, they attempt to directly find the best adapted algorithm from image features by means of learning techniques. Given a training image set and a set of algorithms, segmentation results are evaluated by users within four classes (from worst to best) used to rank the algorithms. Then a predictor (a support vector machine, SVM) is trained using as input a feature vector (a gray-level histogram) for each training image and as output the best ranked segmentation algorithm identifier. In use mode, the feature vector is computed on the test image then the SVM returns the assessment value for each tested algorithm. This approach is tested on a synthetic image base (1000 images with various noise levels) and with four classical unsupervised thresholding algorithms. Results demonstrates the accuracy of the proposed algorithm selection model with 85% of correct classifications. The principal advantage is that this approach does not require (in using mode) any segmentation evaluation process as in trial-and-errors methods and thus, is computationally much more efficient. The principal drawback is that the training process is imitated by the user assessment reliability. The task of visual algorithm ranking is time-consumming and then hardly conceivable in the case of large image and algorithm sets. As depicted by the authors, objective performance evaluation criteria (i.e. automatic) should be investigated to free users from the tedious training stage.

In [Zhang and Luo, 2000], the authors propose a framework for automatic algorithm selection based on knowledge driven hypothesis-and-test optimization model. An expert system is designed to use evaluation knowledge, heuristic knowledge, and high-level knowledge to segment an image with the best adapted

segmentation algorithm. The knowledge base is constructed by extracting basic segmentation assessment criteria from comparison between segmentation results and synthetic image models. Another type of knowledge, called high-level knowledge is also incorporated into the base. It refers to *a priori* restrictions about domain dependent object features (e.g., object's size, shape, etc.). Based on such knowledge, three generic frame types are used in a blackboard representation: request frame for input and output data estimation and measurements, target frame for control operator choice and evaluation, and operator frame for the parameter initialization and operator adjustment. Tests are conducted on various simple biological images with classical thresholding algorithms. However, few details are given concerning the evaluation metric and the degree of domain dependent knowledge used for the experimentation. Moreover, the difficulty to model segmentation knowledge into production rules, makes this approach unsuitable for sophisticated algorithms.

Globally, the two approaches rely on strong hypothesis concerning their field of applications: variations between images must be easy to model, algorithm behaviors within the images must be well-established, and high-level knowledge of objects to segment must be provided as a key-element of the performance evaluation. Actually, the lack of theory on segmentation rules out these approaches to be universally applicable. Indeed, application domains with image variations difficult to model disable the model representation approach, and the expensive knowledge acquisition task needed to build expert systems limits their applicability.

2.4.4 Discussion

Researchers have experienced many segmentation optimization approaches during the last decade. Almost all of the free derivative optimization techniques have been tested. In the worst case, results of optimized segmentations are equivalent to the ones obtained with default parameters. In most of the cases, segmentation quality is improved and time spent to tune algorithms is drastically reduced. The authors present their frameworks as generic by nature and then widely applicable. This affirmation is well-founded in an analytical point of view since the three main components are considered separately. Nonetheless, each described framework has been set up for a particular segmentation task where the fitness function has been specifically elaborated for the application using implicit domain knowledge. Thereby, it has not be proved how the fitness function can affect the performance of the optimization. Moreover, if authors have often assessed their optimization methods against default segmentations, they did not make any quantitative evaluation regarding to other optimization techniques. A comparative study of optimization algorithms has to be done. Concerning the algorithm selection problem, the model representation approach appears to be more realistic in a computing point of view as compared to expert systems. However, the learning framework based on image statistics seems too brittle for the majority of applications where variations between images are difficult to model.

2.5 Conclusion

In this chapter, we have reviewed the segmentation task in the field of computer vision systems. If researchers agree that segmentation is one of the fundamental problem in computer vision, the efforts devoted to cope with this issue since the last four decades have still not led to a unified solution. Most of the vision systems are application dependent and their segmentation step is based on heuristic rules for, as example, the tuning of algorithm parameters. It is, however, well-established that such *a priori* knowledge is determined by domain experts from the context in which the segmentation takes place. Hence, the generalization to other domain of application is strongly limited. Nonetheless, it appears that the recent cognitive vision approach [ECVISION, 2005] has identified some avenues of researches to cope with these limitations, as integration of machine learning techniques into the knowledge acquisition task.

The non-uniqueness solution of the segmentation problem has also been exposed through the review of principal existing segmentation techniques (for both static and sequence of images) drawn in section 2.2. From pure low-level approaches to elaborated multi-cues object-based frameworks, none of them is able to provide a complete solution to general segmentation purposes without making some assumptions. Particularly, we can highlight two recurrent issues: first, the quality of the final results relies on the tuning of key-parameters, almost all algorithms; and second, the lack of direct algorithm comparison possibilities, like the availability of source code or generalized standard testbeds makes the problem even worse.

In section 2.3, a study on the segmentation performance evaluation problem, as a primordial task for algorithm comparison, is presented. The lack of general theory on segmentation has also induced a plethora of techniques for assessing the performance of segmentation algorithms. In one hand, supervised approaches suffer from the subjectivity of manual segmentations. In the other hand, unsupervised approaches are either designed for a specific application, or too generic to consider the final goal of the segmentation. The most promising approaches are propably the co-evaluation frameworks [Zhang et al., 2006] which attempt to coalesce different basic evaluation methods in order to build a goal-adapted fitness function.

The section 2.4 has explored the segmentation tuning paradigm. To optimize the segmentation parametrization, researchers have developed several optimization frameworks mainly based around three independent components: a segmentation algorithm, a performance evaluation function, and an optimization algorithm. All the authors argue that their frameworks are generic and could be applied to wide range of applications and algorithms. To rule on the generic nature of the optimization framework, a deeper analysis of the impact of both the chosen performance evaluation metric and the optimization algorithm on the results is yet needed.

Chapter 3

Approach Overview

3.1 Introduction

In this chapter, we present an overview of our cognitive vision approach to image and video segmentation. We have defined in chapter 1 the expectations of the segmentation task in computer vision systems (algorithm selection and tuning, context adaptation). We have seen in chapter 2 that these challenging issues have been tackled by many different approaches. Our goal is to propose a methodology that takes the best of each approach.

In the context of cognitive vision, we propose a framework with a **reusability** property to ease the set up of the segmentation task in vision systems. More precisely, our framework does not require image segmentation skills: the complexity of this tricky task is hidden by means of automatic algorithm parameter tuning and segmentation assessment. Moreover, the acquisition of the segmentation knowledge is made **convenient** by user-friendly interactivity.

The second property of cognitive vision we are aiming at is the property of **genericity**. In our framework, the different components are not application-dependent. Consequently, this framework can be used with different segmentation algorithms and for different real-world applications. In this chapter, we describe the framework for adaptive image segmentation and adaptive video segmentation.

Another property of our framework is its **adaptation** faculty to image content and to application needs. To this end, we use learning techniques for dynamic algorithm selection and parameter tuning.

The following sections focus on the proposed methodology. The detailed description and analysis of our solutions will be the topics of the three next chapters.

3.2 The Proposed Approach

Our approach is composed of two modules: a supervised learning module where knowledge of the segmentation problem is extracted and f from training data, and a second module where this knowledge is dynamically used to perform an adaptive

segmentation of new images. This approach can be applied to image segmentation tasks (section 3.2.2) and to video segmentation tasks (section 3.2.3).

3.2.1 Hypotheses

Our approach makes some hypotheses concerning the segmentation algorithms and the training data:

- Segmentation algorithms: we suppose that the free parameters of the segmentation algorithms are known as well as their range values.
- Training image set: we suppose that training images are available and representative of the expected situations.
- Ground truth data: the supervised learning stage uses two kinds of ground truth data: manual segmentations and semantic region labels. A manual segmentation represents the expected final result. We suppose that the user is able to provide such manual segmentations for all of the training images. Region labels help to refine a segmented image into a semantically meaningful result. The user's task is thus to annotate some training samples (i.e. regions of the manual segmentations) in accordance with its needs.

3.2.2 A Framework for Adaptive Image Segmentation

Our proposed approach relies on two segmentation frameworks: a parameter optimization framework and a region-based classification one. The first framework aims to optimize bottom-up image segmentation by controlling of the algorithm selection and parametrization. The second framework relies on high-level segmentation knowledge (i.e. semantic region labels) to refine the segmentation in a top-down process. The goal is to train region classifiers w.r.t. the annotated manual segmentations of the training images. The learning module of the framework is sketched in Figure 3.1. It consists in building a segmentation knowledge base.

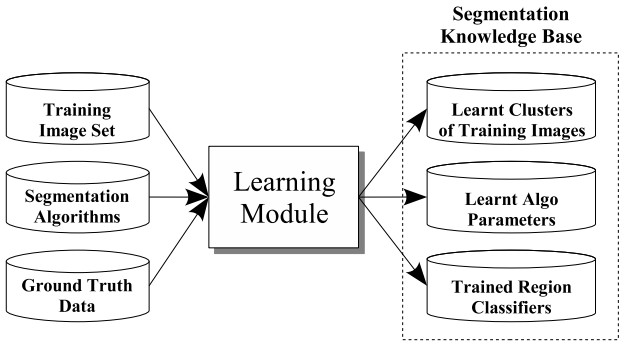


Figure 3.1: The learning module schema of the proposed framework for adaptive image segmentation.

The originality of our approach is to combine these two frameworks in a complementary manner: in a first step, segmentation is optimized by dynamic algorithm selection and parameter tuning. Then, the bottom-up segmentation is refined thanks to region labelling to achieve the expected semantic segmentation. We describe below the different steps of the learning module: segmentation parameter tuning, algorithm selection, and semantic image segmentation.

3.2.2.1 Learning for Segmentation Parameter Tuning

Our framework is able to optimize several free parameters of a segmentation algorithm w.r.t. the parameter space bounds. The optimization procedure is not embedded into the segmentation algorithm so as to be independent of its internal mechanisms. Segmentation performance is evaluated using a global measure of the segmentation quality. To this end, manual segmentation of training images is used to assess segmentation errors. The definition of the performance evaluation metric is thus a key-element of the procedure. We use a boundary pixel-based metric which rates the missed and false detected region boundary pixels against manual segmentation results. Then, a global optimization algorithm explores the parameter space of the segmentation algorithm driven by the segmentation assessment, as sketched in Figure 3.2. At the end of the process, for each training image and for each segmentation algorithm, an optimal parameter set and the corresponding final assessment value are returned. The main advantage of this procedure is that the search process is independent of both the segmentation algorithm and the application domain. Therefore, it can be systematically applied to automatically extract optimal segmentation algorithm parameters. To enforce

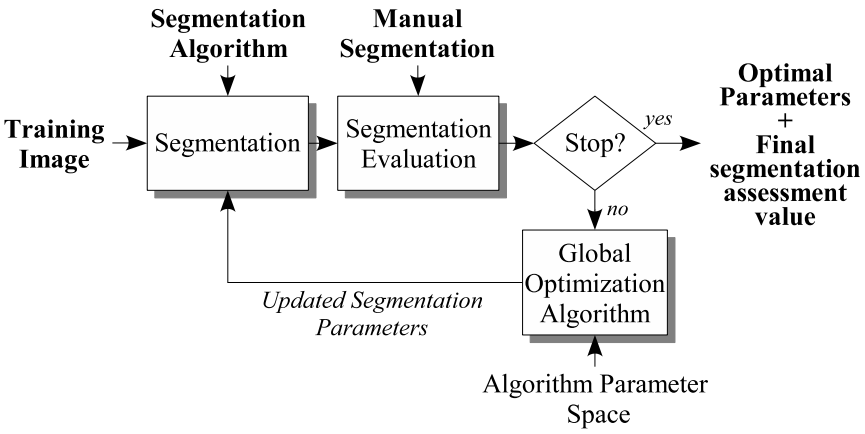


Figure 3.2: Proposed segmentation parameter optimization framework. Input and output are in bold font.

the relevance of our approach, we have tested it on several configurations including different optimization algorithms.

3.2.2.2 Learning to Select a Segmentation Algorithm

In our approach, the selection of a segmentation algorithm is free of *a priori* knowledge of algorithm properties. The different algorithms are compared according to the segmentation quality. Obviously, getting a fair comparison involves that each algorithm has been optimized beforehand. This step thus follows the parameter tuning step. Segmentation is very sensitive to image variations. Hence, the selection (and tuning) of an algorithm must be dynamically fitted to different situations (e.g., different lighting conditions), that we called contexts. This selection emphasizes the need of context modelling. We tackle this problem by performing an unsupervised clustering of the training images based on an analysis of global image characteristics like color variations. At the end of the clustering process, each cluster gathers training images sharing similar global features, i.e. images of the same context. This process is shown in Figure 3.3.

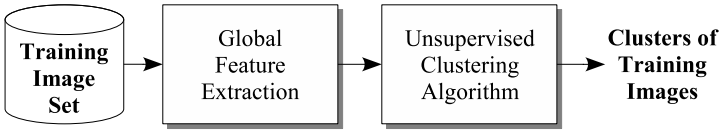


Figure 3.3: Training image set clustering based on image-content analysis. Input and output are in bold font.

Then, by ranking the final performance scores of the different candidate segmentation algorithms, we can select the one which performs the best segmentation for each cluster. Finally, for each cluster we associate a context identifier with a segmentation configuration, i.e. the best ranked algorithm tuned with a mean parameter set computed from optimal values (see Figure 3.4).

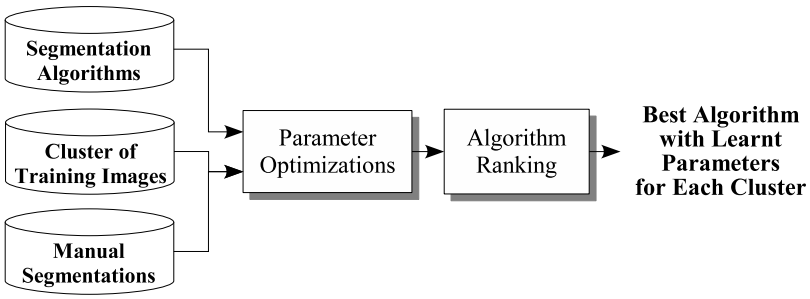


Figure 3.4: Learning schema for algorithm selection. Input and output are in bold font.

In the adaptive segmentation stage, the selection of an algorithm is only based on the image-content analysis. So, the main advantage of this approach is that the algorithm selection does not need to perform any segmentation, it is an *a priori* decision.

3.2.2.3 Learning for Semantic Image Segmentation

The last step of our framework consists in learning the mapping between high level knowledge of the segmentation task (i.e. expected region labels) and region characteristics (i.e. region features) as sketched in Figure 3.5. The main objective of this approach is to reach a semantically meaningful segmentation from an initial optimized pixel-based segmentation. The user first defines a set of classes according to the segmentation goal (e.g. *background, foreground, object of interest #1, object of interest #2*, etc.). This set is used to annotate regions of the manual segmentations. Then, for each training image, the regions of the previously optimized segmentations are automatically annotated according to the annotations of the manual segmentation. The goal is to train region classifiers in order to improve the quality of the segmentation by providing a semantic segmentation. Based on region features (e.g., color distribution, texture features), a region classifier returns a predicate on the region label with probability estimates. The class with the best estimated probability is returned.

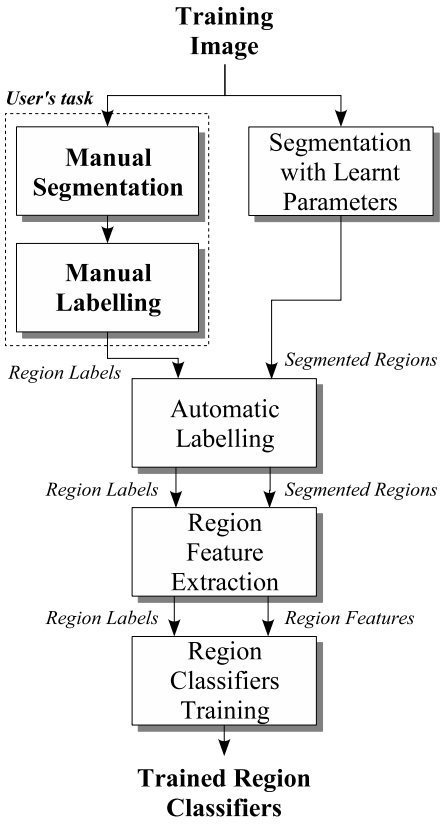


Figure 3.5: Proposed region classifier training schema. Input and output are in bold font.

3.2.2.4 Adaptive Image Segmentation

Segmentating a new image (i.e. not belonging to the training set) is achieved by the adaptive segmentation module in four steps (see Figure 3.6) using the segmentation knowledge base (learnt clusters of training images, learnt parameters, and trained region classifiers):

1. **Context Identification:** a global feature vector is extracted from the image. The feature vector is classified among the previously identified clusters. The classification is based by assessing the distance of the feature vector to the cluster centers.

2. **Algorithm selection:** from the identified context, the corresponding segmentation algorithm with learnt parameters is selected.

3. **Segmentation:** the image is segmented using the selected algorithm. This algorithm is tuned with the learnt parameters specific to the identified context.

4. **Region labelling:** for each region of the segmented image, features are extracted and given as input to the region classifiers. The most probable label is assigned to the region. The final labelled partition representing the semantic segmentation of the image is returned to the user.

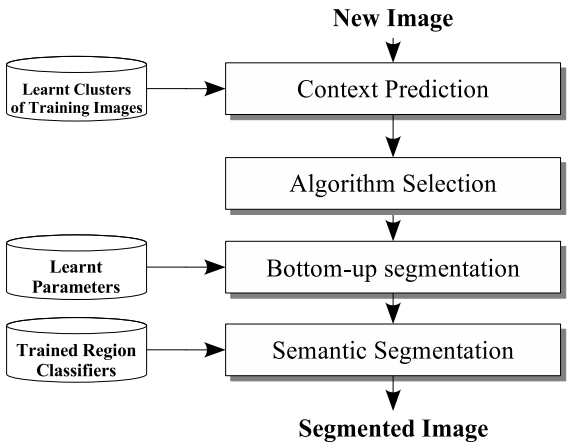


Figure 3.6: Adaptive segmentation of an input image based on algorithm selection, parameter tuning, and region labelling.

3.2.3 Adaptive Video Segmentation

In this task, the goal is to detect moving objects (e.g. a person) in the field of view of a fixed video camera. Detection is usually carried out by background modelling methods. In this situation, annotations of the different classes of foreground

objects are useless. Only background is modeled. Consequently, the semantic image segmentation step of our framework described in section 3.2.2.3 is not used.

Video segmentation algorithms can be decomposed into two classes: algorithms which rely on a training stage for background modelling (e.g., mixture of Gaussians or codebook models) and others (e.g., optical flow, running average). In the first case, the quality of segmentation mostly depends on the quality of the background model training. In the second case, it mostly depends on the parametrization of some key paremeters, such as the detection threshold. The learning step of our framework for parameter tuning could be used to learn the parametrization of such algorithms. However, this implies to manually segment a lot of training samples with foreground objects. We prefer to spare the user this tedious task and we focus more on the learning-based video segmentation algorithms. In this case, strong efforts have been done to cope with quick-illumination changes or long term changes, but coping with both problems altogether remains an open issue. For this task, we propose an approach for dynamic background model selection based on context analysis.

Our approach is based on a preliminary weakly supervised learning module (see Figure 3.7) during which the knowledge about context variations is acquired. The role of the user is limited to establish a training image set composed of background samples that point out context variations. The clustering process of the training image set is the same as the one described in Figure 3.3.

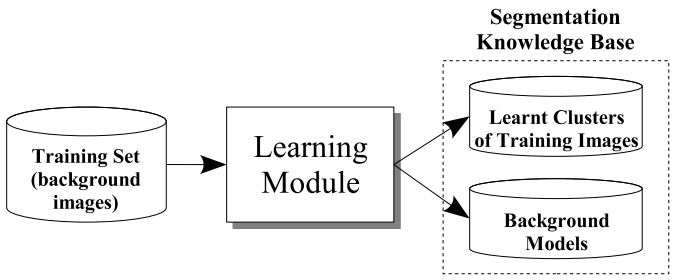


Figure 3.7: The learning module in video segmentation task.

Here, the goal of clustering is to make the background modelling task more reliable by restricting the model parameter space. This approach is particularly interesting for motion segmentation algorithms relying on a training stage of models as mixture of Gaussians [Stauffer and Grimson, 1999] or codebook models [Kim et al., 2005].

For a new input image, global features are first extracted; then, a background model is selected and figure-ground segmentation is performed. A temporal context filtering step is applied before segmentation to prevent from incoming erroneous context identification as sketched in Figure 3.8.

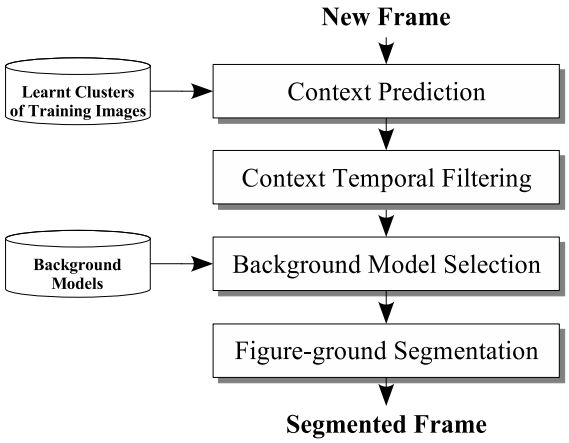


Figure 3.8: Adaptive figure-ground segmentation schema based on context identification and background model selection.

3.3 Conclusion

This chapter has given an overview of our cognitive vision approach to image and video segmentation. We have presented a methodology which aims at endowing the segmentation task of vision systems with **reusability**, **convenience**, **genericity**, and **adaptation** faculties. The proposed framework can be applied for both image and video segmentation tasks. Our approach mainly relies on supervised learning techniques for segmentation algorithm selection and parameter tuning according to users' needs and image contents. Training data, composed of representative image samples with their manual segmentation and region annotations are requested from the users. The main contribution of our approach to adaptive image segmentation is to combine, in a general scheme, a bottom-up approach for parameter tuning and algorithm ranking with a top-down approach for semantic image segmentation. Our framework can also be used for video segmentation. However, we propose an alternative learning module based on weak supervision, more appropriate to video segmentation tasks. Our main contribution is at the level of background model selection. Our approach enables to control the selection of different background models by image content analysis.

Chapter 4

A Framework for Adaptive Image Segmentation

4.1 Introduction

In this chapter, we detail our framework for adaptive image segmentation introduced in Chapter 3. The framework is composed of two modules: a learning module dedicated to extract and learn segmentation knowledge for algorithm selection, parameter tuning, and semantic segmentation; a module for adaptive image segmentation relying on the learnt segmentation knowledge.

The first section of the chapter focuses on the learning for segmentation parameter tuning. We describe our performance evaluation metric for the segmentation quality assessment and our optimization procedure. We also discuss the choice of the optimization algorithm. The second section deals with our strategy for learning to select a segmentation algorithm based on image-content analysis and algorithm ranking. The third section is devoted to the description of our learning approach for semantic image segmentation. The goal is to train region classifiers to improve the segmentation quality and provide a semantic segmentation. The last section describe the adaptive segmentation of new images based on the learnt segmentation knowledge.

4.2 Learning for Segmentation Parameter Tuning

In this section, we detail our parameter optimization procedure. The goal is to optimize the parametrization of segmentation algorithms according to ground truth segmentations of training images. For this task, the user must provide:

1. Manual segmentations of the training images with closed outlined regions.

2. Segmentation algorithms with their free parameters, i.e. the sensitive parameters to be tuned, as well as their range values. This kind of knowledge is often given by the algorithm's author.

4.2.1 Formalization of the Optimization Problem

Let I be an image of the training image set $\mathcal{I}$, G_I be its ground truth (e.g. manual segmentation), A be a segmentation algorithm and $\mathbf{p}^A$ a vector of parameters for the algorithm A. The segmentation of I with algorithm A is defined as $A(I, \mathbf{p}^A)$. We define the segmentation quality E_I^A with the assessment function ρ as follows:

$$E_I^A = \rho\left(A(I, \mathbf{p}^A), G_I\right) \tag{4.1}$$

The value E_I^A is an assessment value of the matching between the segmentation when using algorithm A and the ground truth. This can be a goodness measure or a discrepancy measure.

The purpose of our optimization procedure is to determine a set of parameter values $\hat{\mathbf{p}}_I^A$ which minimizes/maximizes ρ:

$$\hat{\mathbf{p}}_I^A = \arg\min_{\mathbf{p}^A} / \max_{\mathbf{p}^A} \ \rho\left(A(I, \mathbf{p}^A), G_I\right) \tag{4.2}$$

The final assessment value $\hat{E}_I^A$ and the optimal parameter set $\hat{\mathbf{p}}_I^A$ make a pair sample noted $\left(\hat{\mathbf{p}}_I^A, \hat{E}_I^A\right)$. This pair forms the segmentation knowledge for the image I and the algorithm A. The set of all collected pairs constitutes the segmentation knowledge set $\mathcal{S}$ such that:

$$\mathcal{S} = \bigcup_{I \in \mathcal{I}} \left(\hat{\mathbf{p}}_I^A, \hat{E}_I^A\right) \tag{4.3}$$

One key-point of this optimization procedure is the definition of the assessment function ρ. The quality of the final result varies according to this fitness function. So the choice of a segmentation performance evaluation metric is fundamental. It is discussed in the next section.

4.2.2 Definition of the Segmentation Performance Evaluation Metric

As stated in section 2.3, it is not obvious to select a performance evaluation metric because no single metric can cover all aspects of segmentation algorithms. We propose to use a boundary-based metric and evaluates the segmentation in terms of both localization accuracy and the shape accuracy of the extracted regions. The biggest advantage of boundary-based metrics against region-based metrics is their lower computational cost. It is always faster to count and compare some boundary pixels than a lot of region pixels.

The region boundary set for the ground truth and for the segmentation result are noted B_I^G and B_I^A respectively. Two types of errors are considered: missing boundary rate e_m^B and false boundary rate e_f^B. The former, e_m^B, specifies the percentage of the points on B_I^G that are mistakenly classified as non-boundary

points; while the latter, e_f^B, indicates the percentage of the points in B_I^A that are actually false alarms. Therefore,

$$e_m^B = \frac{|T_1|}{|B_I^G|} \quad \text{and} \quad e_f^B = \frac{|T_2|}{|B_I^A|} \tag{4.4}$$

where

$$\begin{aligned} T_1 &= \{x \mid (x \in B_I^G) \wedge (x \notin B_I^A)\} \\ \text{and} \quad T_2 &= \{x \mid (x \in B_I^A) \wedge (x \notin B_I^G)\} \end{aligned} \tag{4.5}$$

and $|.|$ is the cardinal operator. We define the segmentation quality E_I^A with the assessment function ρ as follows:

$$E_I^A = \rho\big(B_I^A, B_I^G\big) = \frac{1}{2}\big(e_m^B + e_f^B\big) \tag{4.6}$$

with $E_I^A \in [0,1]$. The value $E_I^A = 0$ indicates perfect boundary pixel matching between the segmentation result and the ground truth when using algorithm A. The value $E_I^A = 1$ indicates that all pixels are misclassified. However, it is easy to show that this metric comes up against unadapted response to under-segmented results, as illustrated in Figure 4.1. Segmentation in panel (a) shows two regions with a quite good ground truth overlap, only three pixels are misclassified. In the panel (b), the segmentation shows only one region and the quality score is logically less than in (a). In the last panel (c), two regions are present but the center region badly overlaps the corresponding ground truth center region. In opposition with visual assessment, the segmentation quality is worst than in Figure 4.1(c).

The metric is improved by introducing two weighting terms w_f^B and w_m^B which quantify the average distance between misclassified points to the ground truth boundary such that:

$$w_m^B = \frac{1}{|T_1|} \sum_{x \in T_1} dist\left(x, \check{x}_I^A\right) \tag{4.7}$$

with $\check{x}_I^A$ the closest pixel to x belonging to B_I^A, and

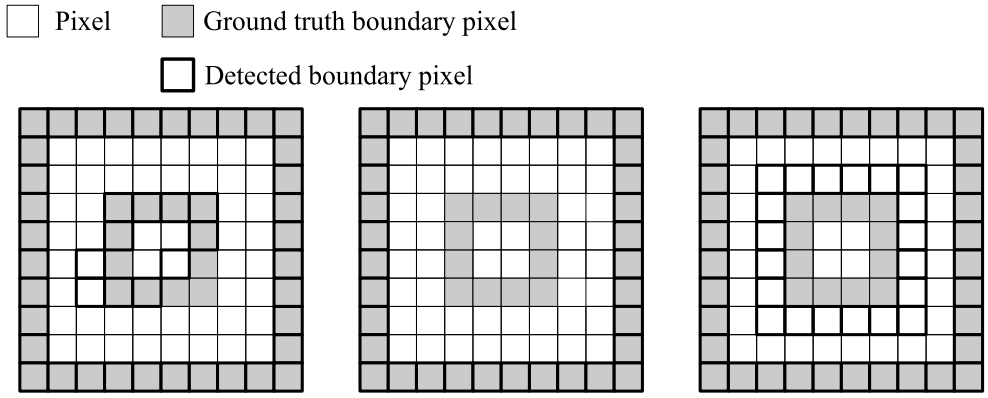
$$w_f^B = \frac{1}{|T_2|} \sum_{x \in T_2} dist\left(x, \check{x}_I^G\right) \tag{4.8}$$

with $\check{x}_I^G$ the closest pixel to x belonging to B_I^G. $dist(x_1, x_2)$ is the euclidean distance between two pixels $x_1(u,v)$ and $x_2(u,v)$ in a 4-neighborhood such that:

$$dist(x_1, x_2) = \sqrt{\big(x_1(u) - x_2(u)\big)^2 + \big(x_1(v) - x_2(v)\big)^2} \tag{4.9}$$

Since w_f^B and w_m^B have no fixed upper bounds, the normalization factor is useless and the segmentation quality measure becomes:

$$\begin{aligned} E_I^A &= w_m^B \times e_m^B + w_f^B \times e_f^B \\ &= \frac{1}{|B_I^G|} \sum_{x \in T_1} dist\left(x, \check{x}_I^A\right) + \frac{1}{|B_I^A|} \sum_{x \in T_2} dist\left(x, \check{x}_I^G\right) \end{aligned} \tag{4.10}$$



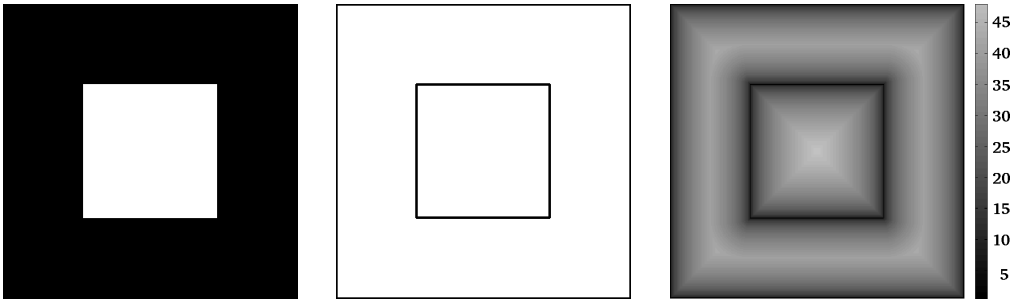
(a) $e_m^B = \frac{3}{48}$, $e_f^B = \frac{3}{47}$, $E_I^A = 0.063$

(b) $e_m^B = \frac{12}{48}$, $e_f^B = \frac{0}{36}$, $E_I^A = 0.125$

(c) $e_m^B = \frac{12}{48}$, $e_f^B = \frac{20}{56}$, $E_I^A = 0.304$

Figure 4.1: Limitation of the segmentation evaluation metric when weighting terms (w_m^B and w_f^B) are not used.

The search of $\check{x}_I^A$ (resp. $\check{x}_I^G$) is made easier by the use of a distance map [Maurer et al., 2003] computed from B_I^A (resp. B_I^G). This operation is exemplified in Figure 4.2.



(a) region-based segmentation composed of 2 regions

(b) region boundary representation of the segmentation in (a)

(c) distance map of (b), the gray-level value of a pixel represents the euclidean distance to the nearest boundary pixel

Figure 4.2: An example of a distance map from a binary contour segmentation.

By taking back the example in Figure 4.1 with the new definition of the evaluation metric, the values of E_i^A for the cases (a), (b), and (c) are respectively 0.168, 0.75, and 0.679, yielding a good correlation with a visual assessment.

Once our performance evaluation metric is defined, the goal is now to minimize the segmentation error E_I^A in order to learn optimal segmentation parameters.

This is the role of our closed-loop global optimization procedure.

4.2.3 Choice of the Optimization Algorithm

Of primary importance in this optimization procedure is finding an optimal segmentation parameter setting $\hat{\mathbf{p}}_I^A$ for each $I \in \mathcal{I}$. We also aims at providing a good evaluation study of the tested optimization techniques in terms of performance versus computational cost and parameter setting. In the family of free derivative techniques, we propose the following criteria to assess the optimization algorithms:

1. Since the segmentation of an image is the most expensive process in the optimization loop, the number of maximum segmentation algorithm calls might be set as a parameter. Indeed, even if the ultimate goal of an optimization procedureis to find a global optimum, the computational cost should remain realistic.

2. The optimization algorithm must be able to converge whatever the evaluation profile, i.e. robust enough to find (quasi-)global optimum of various non-smooth functions.

3. The final quality of the optimization procedure should no be too dependent of the tuning of the optimization algorithm parameters, whatever the segmentationalgorithm.

We have seen in our survey (see section 2.4.2) that several optimization techniques have been applied to tackle the segmentation optimization problem. Although all of them are suitable with our problem, no comparative study exists to help us in our choice. Thus, we have decided to focus on two techniques which are worth being compared. The first one is the Simplex algorithm [Nelder and Mead, 1965] and the second is a standard genetic algorithm [Goldberg, 1989] using non-overlapping populations and optional elitism. In one hand, simplex is easy to use, fast to converge, but requires to define a initializing strategy (starting point(s) and starting step) and do not guarantee to find a global optimum. In an other hand, genetic algorithms are robust but are slower to converge and their parameters must be set carefully. Table 4.1 summarizes the set up of these two algorithms.

4.2.4 Discussion

In this section, we have presented our approach for learning the parametrization of segmentation algorithms. Our optimization procedure relies on three independent components: a segmentation algorithm, a performance evaluation metric, and an optimization algorithm. The goal is to find a parameter set to achieve a segmentation as closed as possible to the ground truth segmentation. We have defined a supervised quantitative evaluation metric assessing the matching between the segmentation result and the manual segmentation. This metric is broadly usable since it mainly relies on generic concepts (false and missed boundary pixel

Most significant parameters to tune

Simplex (see Figure ??)	Genetic Algorithm
• starting values: $\mathbf{p}^A(t_0)$	• initial population size
• starting step values: $\mathbf{p}^A(t_0+1)$	• initial number of generations
• ending criteria = $f(\textit{max nb of calls}, T)$	• number of generations to convergence
• simplex coefficients (α, β, γ, σ)	• cross-over probability rate
• maximum number of calls	• mutation probability rate

Table 4.1: Optimization algorithm parameters.

rates). The simplex algorithm and a genetic algorithm are preferred to solve the optimization problem for their ability to optimize a large spectrum of non-smooth functions. The main difficulty lies in finding the right parametrization of these algorithms to prevent from excessive computation time or weak performance. This point will be a part of our evaluation study in the next chapter.

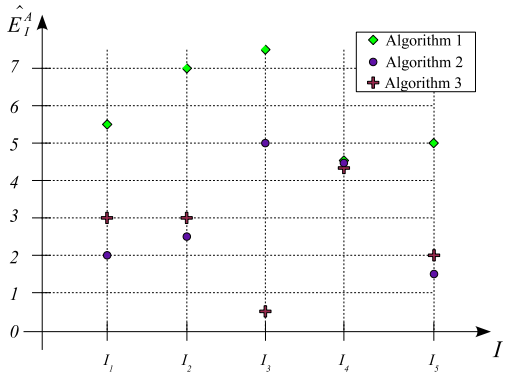
After all pair samples $\left(\hat{\mathbf{p}}_I^A, \hat{E}_I^A\right)$ have been extracted for all segmentation algorithms to test, the next step is to select and tune the one(s) which will be learnt. The following section discusses our selection strategy.

4.3 Learning to Select a Segmentation Algorithm

The previous parameter optimization step allows us to objectively compare the segmentation algorithms with regards to their best performance scores $\hat{E}_I^A$. A straightforward strategy for the selection of an algorithm is thus to take the first best. Nevertheless, the problem becomes more difficult when the training images are heterogeneous, due for instance to global or local variations in the background. In this case, one segmentation algorithm could be the best adapted for the segmentation of a training image subset and another one for another subset. We propose to tackle this problem by associating one algorithm per subset. More precisely, we first identify the different subsets from the whole training image set and then rank the segmentation algorithms for each identified subset. The next two sections details this twofold strategy based on algorithm ranking and image-content analysis.

4.3.1 A Selection Strategy Based on Algorithm Ranking

A first strategy to the algorithm selection problem is to perform a global ranking of the algorithms and to select the best one. Let us illustrate it with a toy example as in Figure 4.3. The graph on the left represents the performance of three optimized segmentation algorithms applied on five different images. The best segmentation quality corresponds to $E_I^A = 0$. In this example, the best algorithm is the one performing the best average performance on the image set, i.e. the algorithm 3 with a mean score of 2.6.



	$\hat{E}_I^A$		
	A|go 1	A|go 2	A|go 3
I_1	5.5	2.0	3.0
I_2	7.0	2.5	3.0
I_3	7.5	5.0	0.5
I_4	4.5	4.5	4.5
I_5	5.0	1.5	2.0
mean	5.9	3.1	**2.6**

Figure 4.3: Algorithm selection in a toy problem with five images and three segmentation algorithms. The values of the table correspond to the segmentation quality $\hat{E}_I^A$.

For each algorithm, a mean parameter set $\bar{\mathbf{p}}^A$ is computed as follows:

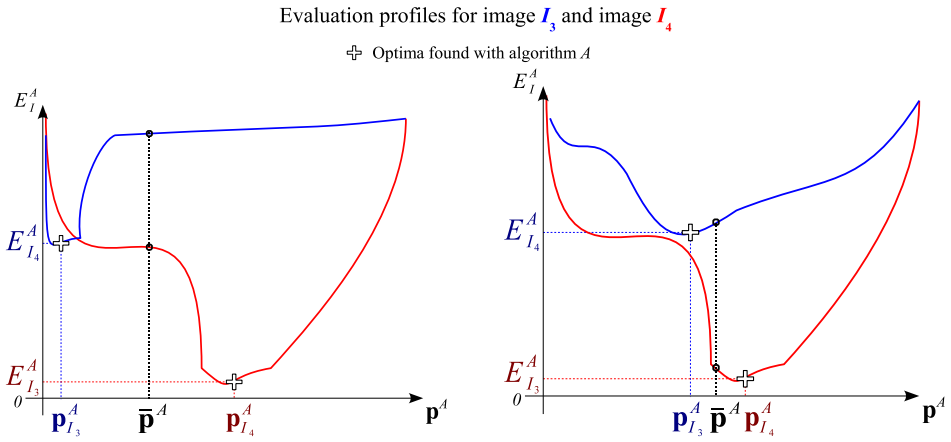
$$\bar{\mathbf{p}}^A = \frac{1}{|\mathcal{I}_A|} \sum_{I \in \mathcal{I}_A} \hat{\mathbf{p}}_I^A \tag{4.11}$$

where $\mathcal{I}_A$ is the set of training images for which the algorithm A has obtained the best evaluation results among the other algorithms. Then, for each training image and each algorithm A tuned with $\bar{\mathbf{p}}^A$, the segmentation quality is computed again. The algorithm having the best average performance on the training image set is finally selected.

This selection strategy comes to select the robustest algorithm based on objective comparisons, i.e. the algorithm which can deliver the best results for the training image set with a globally relevant parameter set. However, this straightforward ranking approach has two major drawbacks. First, by selecting only one algorithm and averaging its parameters, it reduces the previously extracted segmentation knowledge amount to one mean case. Second, even if the selected algorithm overperforms the others in most of the cases, the parameter averaging can have disastrous effects on the algorithm performance. Such a situation is illustrated in Figure 4.4. Let us consider two images composing $\mathcal{I}_A$, i.e. I_3 and I_4. On the two graphs, we show their evaluation profiles (i.e. the drawing of the evaluation fitness functions) with two different possible shapes for I_3. For simplicity, we suppose that $\mathbf{p}^A$ is reduced to one parameter (i.e. 1D profile). It is easy to understand that averaging optimized parameters in the left graph will weaken the algorithm performance for both images I_3 and I_4. In the graph on the right, the averaging is less problematic since the profile shapes are more correlated.

Finally, ranking algorithms and computing a mean parameter set is reliable under the following assumptions:

- The selected algorithm is robust enough to provide good results over the whole image set.



(a) non-correlated evaluation profiles: averaging parameters is unsuitable.

(b) roughly correlated evaluation profiles: averaging parameters is suitable.

Figure 4.4: Consequence of parameter averaging in different evaluation profile cases.

- The evaluation profiles of the images must be quite good correlated to make the parameter averaging plausible.

Depending on the application and the segmentation algorithms, these assumptions are more or less reasonable. That is why we also propose to select the algorithm depending on the image contents.

4.3.2 An Algorithm Selection Approach Based on Image-Content Analysis

The second part of our strategy for algorithm selection is to tackle the problem *a priori* of the segmentation. In this case, the goal is not to directly select the algorithm depending on its relative performance evaluation but depending on the image to segment. Usually, variations between images lead to a variability in the segmentation. As a consequence, similar images should be segmented with the same algorithm and different images should be segmented with different algorithms or different parameter setting. These variations can be induced by changes in background appearance, changes in illumination source, changes in imagery device configuration and so on. The goal is to identify the different situations leading to different segmentation configurations. To this end, we define the context of an image as the quantitative representation of its local and global characteristics. Practically, the context is described by a d-dimensional feature vector $\mathbf{v}(I)$ extracted from the whole image (e.g., a color histograms). The algorithm selection problem can be formalized as follows:

$$\begin{aligned} f : \mathbb{R}^d &\longrightarrow \mathcal{S} \\ \mathbf{v}(I) &\longmapsto (A, \hat{\mathbf{p}}^A) \end{aligned} \tag{4.12}$$

However, it is impossible to continuously predict the algorithm behavior according to image variations and therefore the function f cannot be seen as a regression model. Our approach is to tackle this modelling problem by applying an unsupervised clustering of the training images to identify the different contexts, i.e. clusters of images having similar feature vectors.

In our experiments, we have used a Density-Based Spatial clustering algorithm called DBScan proposed by Ester et al. [Ester et al., 1996]. This algorithm is well-adapted for clustering noisy data of arbitrary shape in high-dimensional space as histograms. Starting from one point, the algorithm searches for similar points in its neighborhood based on a density criteria to manage noisy data. Non clustered points are considered as 'noise' points. The runtime of the algorithm is of the order $O(n \log n)$ with n the dimension of the input space. DBSCAN requires only one critical input parameter, the *Eps*-neighborhood, and supports the user in determining an appropriate value for it. A low value will raises to many small clusters and may also classify a lot of points as noisy points, a high value prevents from noisy point detection but produces few clusters. A good value would be the density of the least dense cluster. But it is very hard to get this information on advance. Normally one does not know the distribution of the points in the space. If no cluster is found, all points are marked as noise. In our approach, we set this parameter so as to have at the most 15% of the training images classified as 'noise' data.

We denote κ a cluster of training images belonging to the same context θ. The set of the n clusters is noted $\mathcal{K} = \{\kappa_1, \ldots, \kappa_n\}$ and the corresponding context set $\Theta = \{\theta_1, \ldots, \theta_n\}$. Once the clustering is done, clusters are learnt. Then, for each cluster (i.e. images of the same context), segmentation algorithms are ranked and the best one is learnt by following the same strategy as described in section 4.3.1. We obtain a discrete function f taking a context identifier θ as input and returning an algorithm A with a mean parameter setting $\bar{\mathbf{p}}^A$ such as:

$$\begin{aligned} \mathrm{f} : \Theta &\longrightarrow \mathcal{S} \\ \theta &\longmapsto (A, \bar{\mathbf{p}}^A) \end{aligned} \tag{4.13}$$

The principal purpose of this strategy is to overcome the drawbacks of the pure global ranking strategy by dividing the solution space $\mathcal{S}$ and by restricting the ranking process onto each subspace. The main advantage on ranking algorithms inside a subspace is that evaluation profiles are likely more correlated.

4.3.3 Summary

In this section, we have shown that the algorithm selection problem cannot be separated from the parameter tuning problem. This statement means that a solution to the algorithm selection issue is composed of both an algorithm and a parameter setting. We have described our twofold strategy for learning the algorithm selection based on algorithm ranking and image-content analysis. Starting from a training image set and segmentation algorithms, our approach first identifies different situations based on image-content analysis, then select the best

algorithm with a mean parameter set for each identified context based on optimized parameter values. At the end of the learning process, contexts are learnt with their associated pairs $(A, \bar{\mathbf{p}}^A)$.

The next step is devoted to semantic image segmentation.

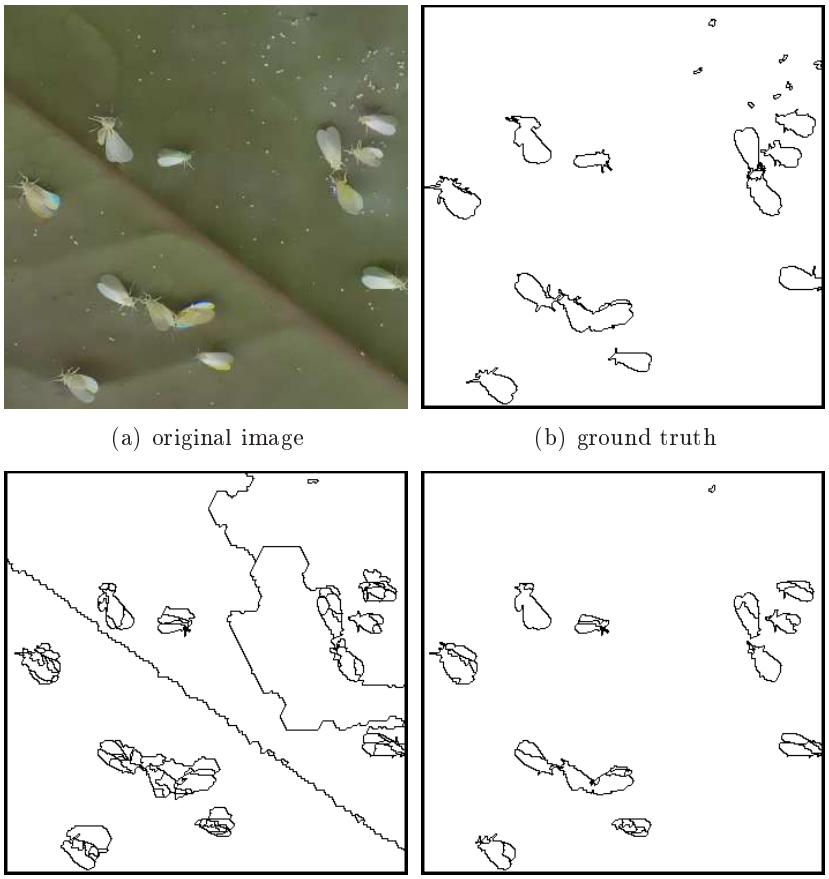
4.4 Learning for Semantic Image Segmentation

In this section, we propose an approach for semantic image segmentation based on high-level knowledge acquisition and learning. Even if the segmentation is optimized, low-level segmentation algorithms cannot reach a semantic partitioning of the image. Thus, compared to the ground truth, some regions remain over-segmented, as illustrated in Figure 4.5. If we can assign the right label to each region, neighboring regions with similar labels are merged and, as a consequence, the residual over-segmentation becomes invisible. This means to be able to map region features onto a symbolic concept, i.e. a class label. We use the example-based modelling approach as an implicit representation of the low-level knowledge. This approach has been applied successfully in many applications such as detection and segmentation of objects from specific classes (e.g., [Schnitman et al., 2006, Borenstein and Malik, 2006]). Starting from representative patch-based samples of objects (e.g., fragments), modelling techniques (e.g., mixture of Gaussian, neural networks, naive Bayes classifiers) are implemented to obtain codebooks or class-specific detectors for the segmentation of images. Our strategy follows this implicit knowledge representation and associates it with machine learning techniques to train *region classifiers*. The following sub-sections describe this stage in details.

4.4.1 Class Knowledge Acquisition by Region Annotations

In our case, region annotations represent the high-level information. This approach assumes that the user is able to gather, in a first step, a representative set of manually segmented training images, i.e. a set that illustrates the variability of object characteristics which may be found. Then, the user must define a domain class dictionary composed of k classes as $\mathcal{Y} = \{y_1, \ldots, y_k\}$. This dictionary must be designed according to the problem objectives. For instance, y_1= background class, y_2= object class #1, etc. Once $\mathcal{Y}$ is defined, the user is invited, in a supervised stage, to label the regions of the manually segmented images with respect to $\mathcal{Y}$. From a practical point of view, an annotation is done with the help of a graphical user interface we have developed. This tool allows to interact with a region-based segmentation of an image by clicking into a region r and by selecting the desired class label y (see Figure 4.6).

At the end of the annotation task, we obtain a list of labeled ground truth regions which belong to classes defined by the user. Since the segmentation result is not exactly the same than the manual segmentation, the next step is to map, for each training image, the labels of ground truth regions onto the regions of the



(a) original image

(b) ground truth

(c) segmentation with default parameters

(d) segmentation with optimized parameters

Figure 4.5: An example of a parameter optimization loop. The final result (d) is not perfect since some regions are over-segmented with respect to the ground truth (b).

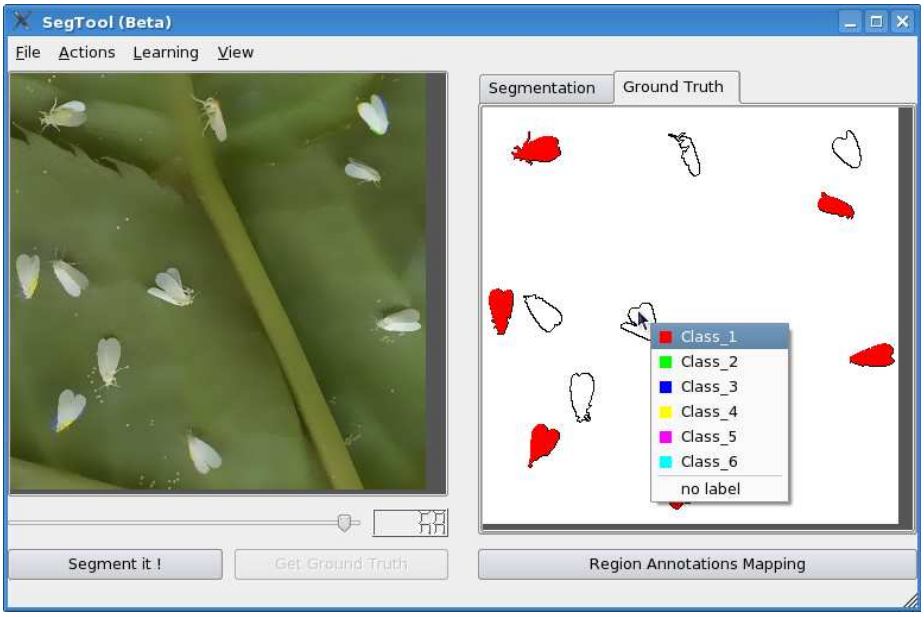


Figure 4.6: Region annotations with the developed graphical tool.

region map R_I^A resulting from the segmentation of the image I with the selected algorithm A tuned with the parameter set $\bar{\mathbf{p}}^A$, as described in section 4.3. The mapping is done by majority overlap such as for each region $r \in R_I^A$,

$$y(r) = \begin{cases} y_i | i = \arg\max \mathbf{h}_r, \text{ if } \frac{\max \mathbf{h}(r)}{|r|} > T \\ y_0, \text{ else} \end{cases} \tag{4.14}$$

with $|r|$ the number of pixels of the region r, T a threshold, and $\mathbf{h}(r) = \{h_1(r), \ldots, h_i(r), \ldots, h_k(r)\}$ the label histogram of the region r such that for a pixel u and a label y_i, $h_i(r) = card\{u \in r \mid y(u) = y_i\}$, $i \in 1, \ldots, k$.

If the ratio of the most represented class in r does not reach the threshold T (here fixed at 0.8), the region label is set to $y_0 \notin \mathcal{Y}$. This prevents from labeling badly segmented region as sketched in Figure 4.7.

We also denote the set of all region annotations $\mathcal{RA}_\mathcal{I}$ such as:

$$\mathcal{RA}_\mathcal{I} = \bigcup_{I \in \mathcal{I}} \bigcup_{r \in R_I^A} \{y(r) \mid y(r) \neq y_0\} \tag{4.15}$$

and the set of all annotated regions $\mathcal{R}_\mathcal{I}$ such as:

$$\mathcal{R}_\mathcal{I} = \bigcup_{I \in \mathcal{I}} \bigcup_{r \in R_I^A} \{r \mid y(r) \neq y_0\} \tag{4.16}$$

For each region, a feature vector $\mathbf{x}(r)$ is extracted and makes with the label a pair sample noted $(\mathbf{x}(r), y(r))$. The set of all collected pair samples from $\mathcal{I}$ constitute

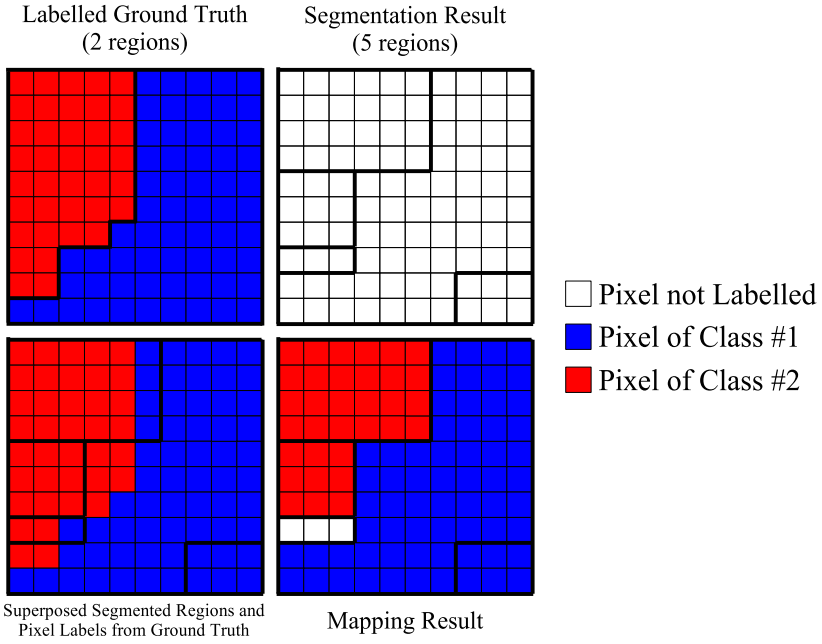


Figure 4.7: Example of the mapping between a labelled ground truth regions and segmented regions.

the training data set $\mathcal{T}_{\mathcal{I}}$ (see Algorithm 3) such as:

$$\mathcal{T}_{\mathcal{I}} = \bigcup_{I \in \mathcal{I}} \bigcup_{r \in R_I^A} \{(\mathbf{x}(r), y(r)) \mid y(r) \neq y_0\} \tag{4.17}$$

$\mathcal{T}_{\mathcal{I}}$ represents the knowledge of the segmentation task and is composed, at this time, of raw information. In the following section, we address the problem of knowledge modelling by statistical analysis.

4.4.2 Segmentation Knowledge Modelling

The first step towards learning statistical models from an image partition is to extract a feature vector from each region. But which low-level features the most representative for a specific region labeling problem? In more general terms, which features are useful to build a good model predictor? This fundamental question, referring to the feature selection problem, is a key issue for most of the class-based segmentation approaches.

Feature Extraction

When defining a set of features for classification problems, two approaches can be considered: a first approach aims at building relevant feature sets, while a

second approach more focus on the usefulness of each feature. In the first case, the choice of relevant features mostly relies on knowledge of the domain. In the second case, the goal is clearly to select features useful for building a good predictor, even if some relevant features may be excluded. We propose a trade-off approach: starting from heuristically selected features we aim at training robust region classifiers. To this end, we combine generic features, such as color and texture and apply a feature selection algorithm.

In our approach, color histograms represent the color information of each segmented region. Two parameters must be set: the color space (cs) as RGB, HSV, or XYZ, on which the histograming is applied, and the quantization parameter q which defines the number of bins. Each color space has its specificity according to the color model it is based on. However, it is not obvious to select the best appropriate color space for a specific problem, and most of the time, several experiments have to be conducted. In our approach, we do not state *a priori* the relevance of one color space against others. We rather consider a color space as a parameter of the feature selection problem. The same statement of fact can be done for the setting of the quantization parameter: if it is easy to discriminate data (e.g., two classes represented by respectively red and green objects), q could be set to a low value (i.e. few bins). On the contrary, if the color distributions of different object classes are mixed, a higher quantization value might be used.

Texture feature extraction techniques have received considerable attention during the past decades and numerous approaches and comparative studies have been presented [Reed and du Buf, 1993]. The most commonly used are the gray-level coocurence matrices introduced by Haralick [Haralick, 1979], the Law's texture energy [Laws, 1980], and the Gabor multi-channel filtering [Jain and Farrokhnia, 1991]. Two surveys on texture feature extraction techniques can be found in [Reed and du Buf, 1993] and [Randen and Husoy, 1999]. For the characterization of texture, we use oriented Gaussian derivatives (OGD) to generate rotation invariant feature vectors. OGD are equivalent to the Gabor features but are computationally simpler. The basic idea is to compute the "energy" of a region as a steerable function. This energy is computed for different "power" channel, which are the result of convolving the region pixels with OGD filters of a specific order. In some way, the first order OGD computes some edge energy while the second order OGD compute some line energy and then produce a strong correlation with the human vision theory. As color histograms, texture feature vectors depend on the q parameter.

The final feature vector representing a region is a concatenation of the feature vectors extracted from each cue. The feature extraction process is applied on each region of the annotated regions set $\mathcal{R}_{\mathcal{I}}$ so as to build the training data set $\mathcal{T}_{\mathcal{I}}$, as depicted in algorithm 3.

Following our cognitive approach of the segmentation problem, we need to avoid manually selected and tuned algorithms. At the feature selection level, this means to be able to automatically select and tune the feature extraction algorithms.

Algorithm 3: Algorithm pseudo-code for the training data set building

inputs : $\mathcal{I}$, $\mathcal{R}_\mathcal{I}$, $\mathcal{RA}_\mathcal{I}$, q, cs
outputs: $\mathcal{T}_\mathcal{I}$

$\mathbf{X}_\mathcal{I} \leftarrow \{\}$;
$\mathcal{T}_\mathcal{I} \leftarrow \{\}$;
foreach $I \in \mathcal{I}$ **do**
  foreach $r \in \hat{R}_I^A \subset \mathcal{R}_\mathcal{I}$ **do**
    $\mathbf{x}(r) \leftarrow$ `regionFeatureExtractor(`I, r, q, cs`)` ;
    $\mathbf{X}_\mathcal{I} \leftarrow \mathbf{X}_\mathcal{I} \cup \mathbf{x}(r)$;
$\mathcal{T}_\mathcal{I} \leftarrow \{\mathbf{X}_\mathcal{I}, \mathcal{RA}_\mathcal{I}\}$;
return $\mathcal{T}_\mathcal{I}$

Feature Selection

The feature selection is used to reduce the number of features, remove irrelevant, redundant, or noisy data, and it brings the immediate effects of speeding up and improving the prediction performance of learning models. Since feature selection is a fertile field of research, we refer the reader to surveys [Guyon and Elisseeff, 2003, Kohavi and John, 1997, Blum and Langley, 1997] as good starting literatures. The optimality of a feature subset is measured by an evaluation criterion. Feature selection algorithms designed with different evaluation criteria broadly fall into two categories: the *filters* and the *wrappers*.

Filters select subsets of features as a pre-processing step, independently of the chosen predictor. Well-known methods dedicated to this purpose are basic linear transforms of the input features like Principal Component Analysis (PCA) and Fisher Linear Discriminant Analysis (LDA). PCA is an unsupervised technique useful for data set dimensionality reduction. For supervised feature selection, i.e. when feature samples are labelled, LDA is more appropriated. This technique selects features that maximize the ratio of the *between-class scatter* to the *within-class* scatter. Techniques based on iterative search are also widespread as sequential forward/backward algorithms (e.g. SFFS, SBS, ReliefF).

Wrappers utilize the learning machine of interest (e.g., SVM, neural networks) as a black box to score subsets of features according to their predictive power. Consequently, wrappers are remarkably universal and simple. An interesting comparative study of such feature selection algorithms can be found in [Molina et al., 2002].

The feature selection approach we propose is derived from wrappers. Our goal is to find the best feature extractor configuration which minimizes the joint classification errors of the class predictors applied on the training data set $\mathcal{T}_\mathcal{I}$. Unlike classical approaches, we act on the feature extractor parameters to generate different feature vectors, instead of reducing the feature vector itself. This approach

is sketched in Figure 4.8.

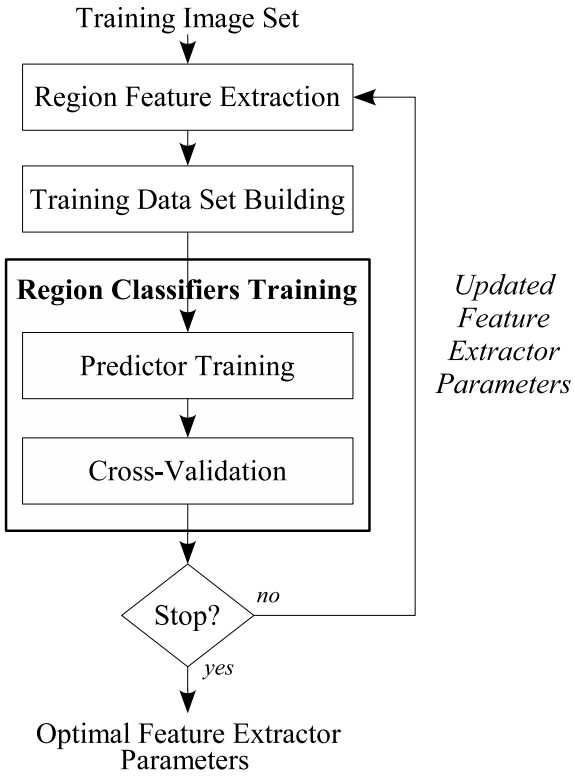


Figure 4.8: Feature selection schema based on tuning of the feature extractor parameters.

The two free parameters of our selected feature extractors are the color space encoder for color feature extractor, and the quantization level for both color and texture feature extractors. The goal is to find the best combination able to induce the minimum region classification errors. The quality estimation is conducted via a cross-validation procedure which gives, for each region classifier c_i, the classification Mean Square Error (MSE), noted $\epsilon(c_i) \in [0, 1]$. A global MSE, noted $\overline{\epsilon}$ in Algorithm 4, is then computed by averaging all the $\epsilon(c_y)$.

We use an iterative search strategy to cover the value spaces of the two parameters q and cs. This technique guarantees to find a global optimal solution but is computationally expensive: first, it requires to run $M \times N \times O$ region classifier training procedures, with M the number of quantization levels (typically equals to 256), N the number of color spaces, and O the number of classifiers to train; second, when the value of q increases, so does the size of the feature vector $\mathbf{v}$. So, to avoid an unreasonable computational time, the choice of the training algorithm must take into account this computational constraint.

Algorithm 4: Algorithm pseudo-code for the color feature selection

inputs : $\mathcal{R}_\mathcal{I}$, $\mathcal{RA}_\mathcal{I}$
outputs: q (quantization level), cs (color space)

$\hat{\epsilon} \leftarrow 1$;
for $q \leftarrow q_{min}$ **to** q_{max} **do**
  foreach $cs \in CS$ **do**
    $\mathcal{T}_\mathcal{I} \leftarrow$`trainingDataSetBuilding`$(\mathcal{R}_\mathcal{I}, \mathcal{RL}_\mathcal{I}, q, cs)$;
    foreach $y \in \mathcal{Y}$ **do**
      $c_y \leftarrow$ `regionClassifierTraining`$(y, \mathcal{T}_\mathcal{I})$;
      $\epsilon(c_y) \leftarrow$ `crossValidator`$(c_y, \mathcal{T}_\mathcal{I})$;
    if $\overline{\epsilon} < \hat{\epsilon}$ **then**
      $\hat{\epsilon} \leftarrow \overline{\epsilon}$;
      $\hat{q} \leftarrow q$;
      $\hat{cs} \leftarrow cs$;
return $(\hat{q}, \hat{cs})$

Training Algorithm for Class modelling

After extracting a feature vector for each region of the training data set, the next step is to model the knowledge in order to produce region classifiers (one classifier per class). For a feature vector $\mathbf{x}(r)$ and a class y_i,

$$c_i(r) = p\big(\, y(r) = y_i \mid \mathbf{x}(r) \,\big) \tag{4.18}$$

with $c_i(r) \in [0, 1]$, is the estimated probability associated with the hypothesis: "feature vector $\mathbf{x}(r)$ extracted from region r is a representative sample of the class y_i". The set of these trained region classifiers is noted $\mathcal{C} = \{c_1, \ldots, c_k\}$.

A variety of techniques have been successfully employed to tackle the problem of knowledge modelling such as naives Bayes networks, decision trees or support vector machine (SVM). We propose to use SVM [Burges, 1998] as a template-based approach. SVM are known to be efficient discriminative strategies for large-scale classification problems such as in image categorization [Chen and Wang, 2004] or object categorization [Huang and LeCun, 2006]. SVM yields also state-of-the-art performance at very low computational cost. SVM training consists of finding an hyper-surface in the space of possible inputs (i.e. feature vectors labeled by +1 or -1). This hyper-surface will attempt to split the positive samples from the negative samples. This split will be chosen to have the largest distance from the hyper-surface to the nearest of the positive and negative samples.

Given training vectors $\mathbf{x}_i \in \mathbb{R}^n, i = 1, \ldots, n$ and a vector $\mathbf{y}_i \in -1, +1$, a C-support vector classification [Vapnik, 1995] (C-SVC) solves the following primal

problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{4.19}$$

Its dual is:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - \mathbf{e}^T\alpha \\ \text{subject to} \quad & \mathbf{y}^T\alpha = 0, \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, l \end{aligned} \tag{4.20}$$

where $\mathbf{e}$ is the vector of unity, $C > 0$ is the penalty parameter of the error term, Q is an l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is the kernel. Here training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function ϕ.

For any testing instance $\mathbf{x}$, the decision function f (predictor) is:

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} y_i\alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{4.21}$$

The most commonly used kernels are the following:

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = \gamma(\mathbf{x}_i^T\mathbf{x}_j + r)^d,\ \gamma > 0$
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right),\ \gamma > 0$
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma\mathbf{x}_i^T\mathbf{x}_j + r)$

Here, γ, r, and d are kernel parameters to be *a priori* defined.

We adopt a *one-vs-rest* multiclass scheme with probability information [Wu et al., 2004] to train one region evaluator c per class y. We use SVM with radial basis function as region classifiers. There are two parameters while using RBF kernels: C (penalty parameter of the error term) and γ (kernel parameter). It is not known beforehand which C and γ are the best for one problem; consequently some kind of model selection (parameter search) must be done. To fit the C and γ parameters, we adopt a grid-search method using 5-fold cross-validation on training data. Basically, pairs of (C, γ) are tried and the one with the best cross-validation accuracy is picked (see algorithm 5). This straightforward model selection efficiently prevents overfitting problems. As seen in Figure 4.9, the model selection is wrapped in the feature selection schema with whom it shares the cross-validation step.

The training stage ends up when all combinations of $\{(q, cs), (C, \gamma)\}$ have been tested. The one giving the lowest global classification error is picked and the region classifiers are trained a last time with this configuration.

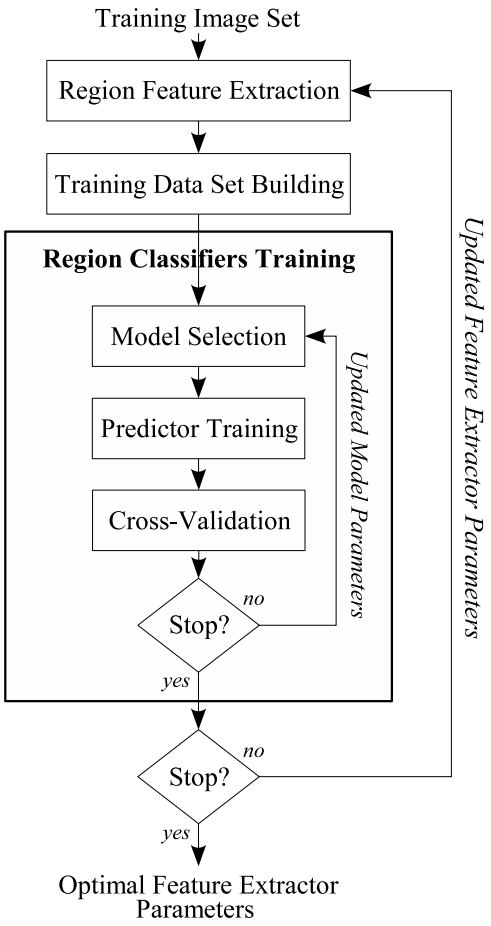


Figure 4.9: Model selection schema based on tuning of the predictor parameters.

Algorithm 5: Model selection algorithm pseudo-code

inputs : $y_i \in \mathcal{Y}$, $\mathcal{T}_\mathcal{I}$
outputs: c_y,$\hat{\epsilon}$

$\mathcal{T}'_\mathcal{I} \leftarrow \mathcal{T}_\mathcal{I}$;
foreach $(\mathbf{x}(r), y(r)) \in \mathcal{T}'_\mathcal{I}$ **do**
  if $y(r) = y_i$ **then**
    $y(r) \leftarrow +1$;
  else
    $y(r) \leftarrow -1$
$\hat{\epsilon} \leftarrow 1$;
for $C \leftarrow C_{min}$ **to** C_{max} **do**
  for $\gamma \leftarrow \gamma_{min}$ **to** γ_{max} **do**
    $c_y \leftarrow$ `predictorTraining`$(\mathcal{T}'_\mathcal{I}, C, \gamma)$;
    $\epsilon(c_y) \leftarrow$ `crossValidation`$(c_y, \mathcal{T}'_\mathcal{I})$;
    if $\epsilon(c_y) < \hat{\epsilon}$ **then**
      $\hat{C} \leftarrow C$;
      $\hat{\gamma} \leftarrow \gamma$;
      $\hat{\epsilon} \leftarrow \epsilon$;
$c_y \leftarrow$ `predictorTraining`$(\mathcal{T}_\mathcal{I}{}', \hat{C}, \hat{\gamma})$;
return c_y,$\hat{\epsilon}$

4.5 Adaptive Image Segmentation

For a new incoming image I not belonging to the training set, a feature vector is first extracted then classified into a cluster. The classification is based on the minimization of the distance between the feature vector and the cluster set $\{\kappa_i\}$ as follows:

$$I \in \theta_i \Leftrightarrow \mathbf{v}(I) \in \kappa_i \mid i = \arg\min_{i \in [1,n]} dist\left(\mathbf{v}(I), \kappa_i\right) \tag{4.22}$$

The pair $(A, \bar{\mathbf{p}}^A)$ associated with the detected context θ_i, is returned.

Once the algorithm is selected and tuned, the image is segmented. For each region, a feature vector is extracted using the $(\hat{q}, \hat{cs})$ parameter set and given as input to each trained region classifiers c_i. Classes are scored according to the classifier responses $\{c_i(r)\}$ and finally, the returned label $y(r)$ is such as:

$$y(r) = \arg\max_i c_i(r) \tag{4.23}$$

When all regions are labelled, neighboring regions with the same label are merged to form a semantic partitioning of the image. This final segmentation is returned to the user, as described in Algorithm 6.

Algorithm 6: Algorithm pseudo-code for adaptive image segmentation

inputs: $I \notin \mathcal{I}$, $\mathcal{C}$, $\hat{cs}$, $\hat{q}$

```
1  v(I) ← GlobalFeatureExtractor(I) ;
2  θ_I ← ContextClassification(v(I)) ;
3  (A, p̄^A) ← θ_I ;
4  R_I^A ← A(I, p̄^A) ;
5  foreach r ∈ R_I^A do
6  |   x(r) ← regionFeatureExtractor(r,ĉs,q̂) ;
7  |   y(r) ← RegionClassification(x(r),C) ;
8  forall (r_i, r_j) ∈ R_I^A, i ≠ j do
9  |   if (r_i IsNextTo r_j) ∧ (y(r_i) = y(r_j)) then
10 |   |   RegionMerger(r_i,r_j) ;
11 return semantic segmentation of I
```

4.6 Framework Conclusion

In this chapter, we have presented our framework for adaptive image segmentation. We have detailed each step of the learning module for algorithm parameter tuning, algorithm selection, and semantic image segmentation. The algorithm parametrization issue is tackled with a generic optimization procedure based on three independent components. We have designed our performance evaluation metric to be broadly applicable and with a low computational cost. It allows to assess a large variety of segmentation algorithms and only relies on manual segmentations. However, further experiments need to be done to assess the performances and the accuracy of the two optimization algorithms (the Simplex algorithm and a Genetic Algorithm). Our strategy for algorithm selection can be summarized as follows:

- The user is assumed to provide a training image set representative of the different situations.
- The training image set is clustered in order to divide the algorithm selection problem into sub-problems more tractable, each sub-problem representing an image context. To this end, an unsupervised clustering algorithm is used to cluster feature vectors extracted from the training image set. This strategy assumes, in a way, the existence of a link between a quantitative image representation and a tuned segmentation algorithm.
- For each identified cluster, one algorithm is selected based on performance ranking. A mean parameter set is computed. This ranking strategy reduces the number of acceptable solutions to one mean solution.

The final step of the learning module is to train region classifiers to refine the segmentation according to semantic region labelling. In this task, the user must

annotate the regions of the manually segmented images with class labels. Our approach is based on the discriminative power of the SVM Classifiers to ground low-level region features into symbolic classes. We have also proposed an unsupervised method for the learning of SVM and region feature extractor parameters. The goal is to optimize the performance of the classifiers without the help of the user.

The module for adaptive image segmentation makes use of the learnt segmentation knowledge. For a new image, the algorithm selection and tuning is fast since it only relies on the computation of a global feature vector. Then, each region is labelled according to the region classifiers responses and the final semantic image is returned to the user.

The next two chapters are dedicated to the validation of this framework on real-world applications.

Chapter 5

Experiment and Evaluation for Image Segmentation

This chapter is dedicated to the validation of the framework presented in the previous chapter for image segmentation in real world applications. In particular, we are interested in the segmentation step of a cognitive vision system dedicated to the recognition of biological organisms. We first present the biological problem and the experimental protocol. Then we give a brief description of the cognitive vision system used to solve the biological problem. The last section is dedicated to the detailed assessment of the vision system with a particular focus on the segmentation level. We also give some evaluation results on a public image database at the end of the chapter.

5.1 A Major Biological Challenge: the Early Detection of Plant Diseases

5.1.1 A Major Challenge for Integrated Pest Management

Integrated Pest Management (IPM) is a knowledge-based approach to crop protection. It is an important tool for the management of insects, pathogens, weeds, and cultural problems in greenhouse. The goal of IPM is to integrate cultural, physical, biological, and chemical practices to grow crops with minimal use of pesticides. This approach is particularly promising in the context of ornamental crops in greenhouses because of the high level of control needed in such agrosystem. Indeed, early detection of plant diseases makes it possible to operate efficiently at the beginning of an infection to limit the plant damage. Thereby, it can reduce the amount of pesticide applications and thus reduce the control cost. However, no automatic methods are available to precisely and periodically evaluate the biotic status of plants. In fact, the detection of biological objects as small as such insects (dimensions are about 2 mm) is a real challenge, especially when considering greenhouses dimensions (10 to 100 meters long). Traditionally in production conditions, visual observations are made each week by human experts (greenhouse

staff), often on colored sticky traps. Since this technique does not allow to precisely study the epidemic spatial model, observations on natural support (i.e. on leaves) are preferred. But it is difficult or even not possible to perform a continuous (typically daily) human control and to examine every leaf in the greenhouse. Moreover the accuracy of the observations depends on the human eye resolution, even if magnification tools are used.

5.1.2 Context of the Experiment

This part of the work consists in a research cooperation between the Orion team of INRIA Sophia Antipolis and the Integrated Research in Horticulture Unit (URIH) of INRA Sophia Antipolis (National Institute for Agricultural Research). The context of this work is also the region PACA (Provence Alpes Côte d'Azur) which is the leading horticultural region of France[1].

5.1.3 Choosing a crop and a bioagressor as a model study

For this study, we first chose a model "crop × bioagressor". On the one hand, rose, an ornamental crop, was chosen because it attracts various bioagressors and it requires high level standard quality for flowers as well as leaves. On the other hand, white fly *Trialeurodes Vaporariorum* was chosen because this bioagressor requires early detection and treatment to prevent durable infestation. Eggs and larvae identification and counting by vision techniques are difficult because of critical dimension (eggs) and weak contrast between object and image background (larvae). For these reasons we decided to focus first on adults. Eggs, larvae and adults are present on back faces of leaves.

5.2 Experimental Protocol

5.2.1 Greenhouse experiment

The agrosystem was a 256 m^2 plastic twin-tunnel greenhouse planted with roses. The management of climate, fertilization and irrigation was carried out by a control/command computer system designed at INRA. Two rose cultivars (Suella^{TM}, a yellow one, and Miss Paris^{TM}, a red one) were planted. They are known for their different resistance to powdery mildew and attractive powers to insects. The total cultivation corresponds to 1200 plants. A map of the greenhouse is shown on Figure 5.1.

5.2.2 Sampling strategy

We chose our sampling strategy based on the following requirements:

[1] Roses are widely produced in PACA and early disease detection is classified as a major challenge.

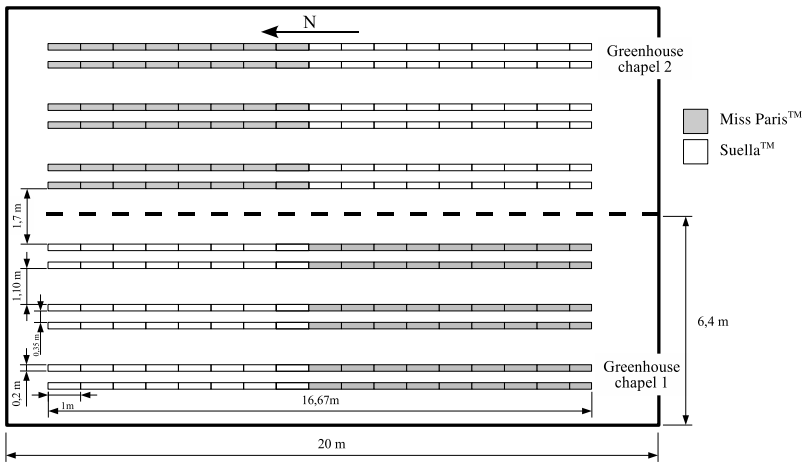


Figure 5.1: Greenhouse map showing two chapels of 128 m^2 each.

- Spatially, data should be uniformly distributed, thus samples were randomized (according to a grid) over the whole greenhouse.

- Temporal sampling should be realistic, i.e. provide a good ratio data relevance versus sampling duration.

The spatial sampling strategy consists in a randomized sampling in the horizontal plane and optimized sampling along the vertical axis. Since it is convenient to consider 12 plants or 2 m rock wool slab as a standard visual observation unit, it was decided to take a leaf sample (5 or 7 leaflets) every second meter along plantation lines. We have done a pre-study on sample cuts at various heights of plant canopy to decide the optimal localization of samples: for early detection of mature white flies, growing stems have been preferred. Hence, 100 samples were taken corresponding to 1200 plants. Samples are rose leaves, each leaf being made up of 5 or 7 leaflets, cut in the central part of the canopy where growing stems are the most numerous. Both sides of leaves were scanned individually and 200 images were recorded (see Figure 5.2 for an example). If we assume a LAI (Leaf Area Index) of 3 for rose crop [Raviv and Blom, 2001], and with an effective crop surface of 100 m^2, it means that around 0.36% of LAI (for one face) is analyzed at each survey by using the above sampling strategy[2].

Concerning the temporal sampling strategy, the time required to perform an automatic survey is of the same order of magnitude as the time necessary to make a chemical treatment on an equivalent surface. Thus, this quick delivery of results, i.e. within half a day, is compatible with rapid decision.

[2]the survey ratio (SR) is computed as follows: $SR = (N_{scan} * S_{scan})/(LAI * S_{crop})$, with N_{scan} the number of total scanned leaves (200), S_{scan} the effective scanned surface per acquisition (0.0054 m^2) and S_{crop} the effective crop surface (100 m^2)

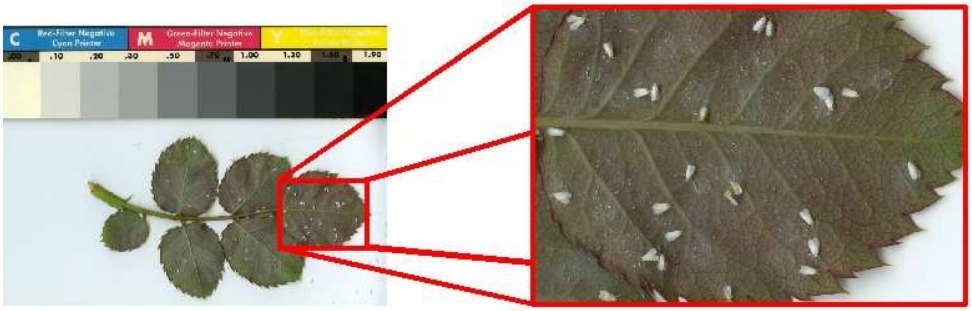


Figure 5.2: Example of a scanned rose leaf infested by white flies.

For this study, samples were manually cut and scanned directly in the greenhouse by using a consumer electronics flatbed A4 scanner. This allowed a high image quality and a short scanning/transfer time. A resolution of 600 dpi was chosen. This corresponds to theoretical square pixel dimensions of 42 μm $\times$ 42 μm. Such a resolution is a good compromise: it is precise enough to digitize objects as little as mature white flies (500 pixels of area) and compatible with data acquisition/storage constraints.

Once the data acquisition conditions fixed, the next step is to provide a system that automatically identifies and counts white flies on the scanned images. This system is presented in the next section.

5.3 The Cognitive Vision System for Pest Detection and Counting

Following a cognitive vision approach, we propose to use an automatic image interpretation system that combines image processing, learning and knowledge-based techniques for the detection and the counting of mature white flies. Our approach follows previous work presented in [Hudelot and Thonnat, 2003] and [Boissard et al., 2003] and enriches it with learning and control abilities at the segmentation level.

5.3.1 System Overview

As human biologists do, a cognitive vision system has to analyze raw images and to label interesting regions that correspond to objects of interest (e.g., insects). To recognize a region as an insect, a human expert relies on (biological and contextual) domain knowledge about insects (e.g., species features, life cycle, host plant) as well as visual data that can be extracted from images (color, texture, shape, and size). A software system must take into account both kinds of knowledge. To separate the different types of knowledge and the different reasoning strategies, the cognitive vision platform proposes an architecture based on specialized mod-

ules, as shown in Figure 5.3. It consists of two knowledge-based systems (KBS), a set of image processing (IP) algorithms, and a learning module.

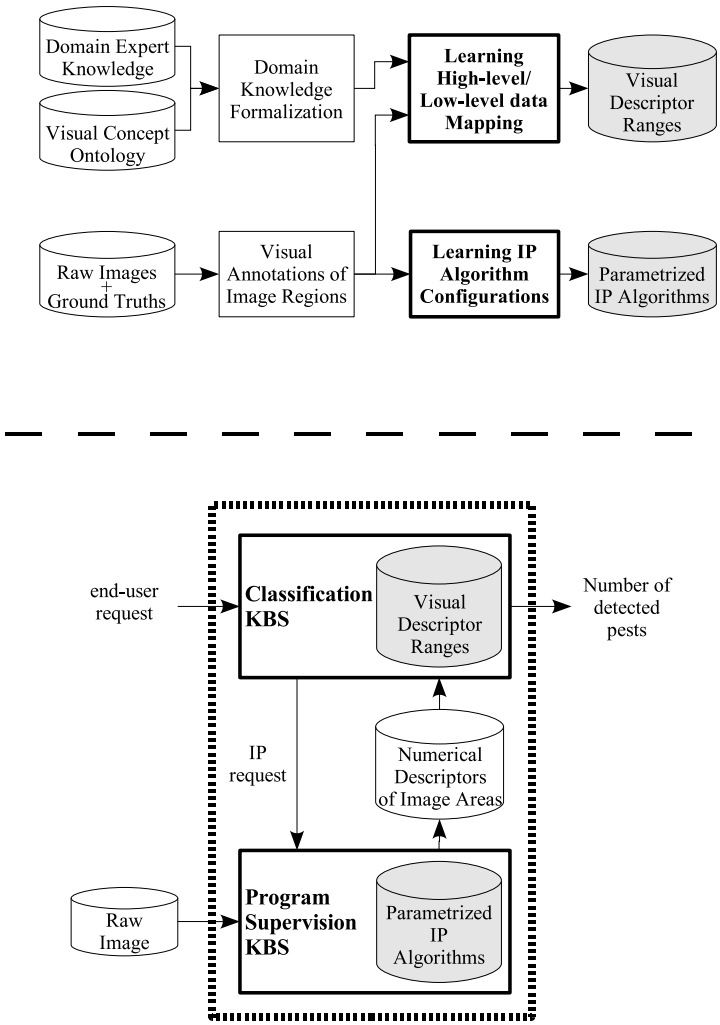


Figure 5.3: Cognitive vision system. The top part corresponds to the initial learning module and the bottom part to the automatic system for routine execution.

Before routine execution, a *learning* stage (Figure 5.3 top) is performed once on a training image sub-set. This preliminary stage is used to complement the knowledge necessary to run the two following KBSs.

The *classification* KBS (Figure 5.3 bottom) aims at selecting interesting regions in images. To this end it triggers image processing requests and interprets the numerical results into higher level concepts, i.e. (parts of) objects of interest. It only retains the regions corresponding to target insects and returns their number to the user.

The *supervision* KBS (Figure 5.3 bottom) is used to monitor the execution of

the image processing requests. It selects and plans the best programs with the best parameter values for each image. From raw images provided by the end-user, the goal is to extract numerical values needed by the classification KBS.

5.3.2 Learning Stage

Learning techniques are used for two purposes: to learn how to map low-level features to high-level domain concepts and how to tune parameters of image processing programs.

5.3.2.1 Learning Visual Concepts

The goal is to map object descriptions in the knowledge base of the classification system to numerical values. Most of real objects can be described in terms of concepts, such as their shape, color or texture. We call them *visual concepts*. Visual concepts are an intermediate level that helps mapping low-level numerical value to a domain class description. For instance, a length in millimeters and/or a color in RGB values may be mapped to an insect body. Thus, each biological class description appearing in the classification knowledge base must be precisely specified in terms of visual concepts.

We refer to a general ontology proposed by N. Maillot et al. in [Maillot et al., 2004] and in [Maillot, 2005], which is a hierarchy divided into three parts: spatio-temporal, color and texture concepts. For instance, spatio-temporal concepts include shape, size, and spatio-temporal relations. The main advantage in using this ontology is to provide domain experts with a vocabulary for describing domain classes in visual terms (as shown in Figure 5.4) by means of *numerical descriptors*. The role of these descriptors is to bridge the semantic gap between low-level numerical values and visual concepts. A descriptor has an attached numerical value that can be computed by vision programs; for instance, a program can compute an area length in millimeters. It also corresponds to a visual feature meaningful for a human expert to describe an object; for instance, an expert may select the *length* descriptor of the *size* concept as relevant to describe an insect body.

We refer also to [Maillot, 2005] for the learning of visual concept. Based on a training set of images and for each visual concept used by the expert, the learning module learns semi-automatically the range of possible values for all the numerical descriptors necessary to recognize the concept.

5.3.2.2 Learning Image Processing Parameters

Our goal is to provide a meaningful segmentation of white flies for further interpretation purposes. Since no dedicated segmentation algorithm exists for this specific task, we started with a set of state-of-the-art region-based algorithms and then, following our approach described in chapter 4, we optimized their free parameters on a training image sub-set. At the end of the optimization process,

each algorithm was tuned with its optimal parameter setting. Then, a clustering decomposition was performed on the training image sub-set to identify the different situations. The algorithm achieving the best segmentation results (according to ground truth) for each identified cluster is retained. Two region-classifiers were trained to reach a goal-oriented segmentation: one for the mature white fly class and one for the rose leaf class. This part of the work is more detailed in section 5.4.

5.3.2.3 Learning Issues

At the end of the learning stage, we get image processing algorithm with fine tuned parameters and descriptor ranges for all relevant numerical descriptors of the application.

Note that this learning stage is done only once, for one (set of) insect(s) to detect and one set of programs to run. Provided that the acquisition conditions do not change drastically, routine execution only involves running the two following knowledge-based systems, i.e. the classification system and the image processing supervision system.

5.3.3 Classification System

The role of the classification system is to recognize and to count white flies on an image. To this end its relies on knowledge about insect descriptions and on numerical descriptor values provided by image processing programs.

The knowledge in this KBS consists of descriptions of domain classes and class hierarchies, provided by biologists. We propose a dedicated expert language to describe these hierarchies. In the case of white fly detection, the knowledge base mainly contains knowledge of these insects (see Figure 5.4). The *WhiteFly* domain class describes how such an insect may be recognized thanks to different visual concepts selected by the expert, namely its shape, size and color. These concepts refer to general ones (such as *ColorConcept*) defined in the general ontology.

Each visual concept is in turn described by a relevant set of numerical descriptors and their associated fuzzy ranges of possible values, as learnt on the training set with the help of a domain expert during the initial learning stage (see section 5.3.2.1). For example, some values of the HSV color of image areas are linked with the white fly color. Experts use the vocabulary defined by the general visual concept ontology such as the term "circularity" to characterize the shape of a white fly.

It should be noted that we do not need to manage a complete biological hierarchy of insects (i.e. with all sub-species), but only the parts that are useful for the recognition task. Indeed, it is useless (and often impossible with the currently available vision techniques) to precisely recognize the sub-species of an insect, because we know that not all sub-species will infest a type of plant.

To summarize, the classification KBS provides class hierarchies and a description of each class in terms of numerical descriptors. To get the real values of

DOMAIN CLASS *WhiteFly*	SUPERCLASS: *Bioagressor*				
SHAPE: ShapeConcept					
Descriptors:					
circularity	[0.05	0.2	0.5	0.6	]
excentricity	[0.1	0.2	0.4	0.5	]
rectangularity	[0.5	0.6	0.8	0.85	]
elongation	[0.3	0.35	0.7	0.8	]
convexity	[0.7	0.75	1.0	1.1	]
compacity	[0.1	0.25	0.9	1.0	]
COLOR: ColorConcept					
Descriptors:					
saturation	[0.0	0.0	0.2	0.3	]
lightness	[120	130	240	260	]
hue	[80	90	170	180	]
SIZE: SizeConcept					
Descriptors:					
area	[0.5	0.6	1.2	1.3	]
length	[0.6	0.8	2.5	3.5	]
width	[0.2	0.3	1.0	1.3	]

Figure 5.4: High-level description of a domain class (white fly). Visual concepts are in SMALL CAPS. learnt fuzzy ranges are shown on the right. They are composed of four numbers, corresponding respectively to the minimum admissible value, the minimum and maximum most probable values, and the maximum admissible value.

numerical descriptors, the correct image processing algorithms must be launched and controlled: this is the role of the image processing supervision KBS.

5.3.4 Image Processing Supervision System

The image processing task itself is achieved by a program supervision [Thonnat et al., 1998] knowledge-based system. Program supervision techniques make it possible to automate the use of complex programs, i.e. to plan and control processing activities. In cognitive vision systems, the supervision system controls the use of the image processing sub-tasks, such as image segmentation and features extraction. It is based on knowledge about the programs, their input/output data, their applicability conditions, their possible combinations, and the necessary information to run them in various situations. This knowledge is given by image processing experts in the form of *operators* and decision *rules* to guide the supervision engine reasoning (e.g., to select programs, initialize their parameters, or assess their results). Suitable parameter initialization values have been learnt during the previous learning step, but more tuning is possible dynamically depending on the input image specificity. Operators and rules are formally described in the knowledge base using an expert language (see an example Figure 5.5).

```
Composite Operator { name Segmentation
     comment "image segmentation operator"
     Input Data
          Image input_image
     Output Data
          Image output_image
          ...
     Body
          RegionBased | EdgeBased
     Choice Rule { name RegionChoice
          If concept == ShapeConcept
          Then useOperator RegionBased }
}
```

Figure 5.5: An example from the program supervision knowledge base. A composite operator describes an alternative decomposition (denoted by a |) into two sub-operators: region or edge-based segmentation, and a rule selects the first one if the concept to recognize (as indicated by the classification KBS) is Shape.

Once the objects have been extracted, the second step of the image processing task, feature extraction, computes the attributes corresponding to each region, according to the domain feature concepts (e.g., color, shape and size descriptors). Finally, the supervision KBS returns a set of candidate areas to the classification KBS. Each area has an attached set of computed numerical descriptor values. The classification KBS can use these descriptors in conjunction with its own knowledge to select areas corresponding to white flies and to return the number of recognized

flies to the user.

5.4 Approach Assessment

This section is dedicated to the performance evaluation of the segmentation step and of the cognitive vision system.

5.4.1 Segmentation Algorithms

In this section, we briefly describe the segmentation algorithms we used for our experiment. Our set is composed of algorithms reflecting different segmentation strategies as developed in section 2.2.2 namely region growing, split-and-merge, watershed, or thresholding techniques.

CSC: The Color Structure Code [Priese and Sturm, 2002] is a method combining the advantages of local (simplicity and fastness) and global (robustness and accuracy) techniques. It is a hierarchical region growing method that is inherently parallel and therefore independent of the choice of the starting point and the order of processing. The generation of the CSC operates essentially in three phases. In an initialization phase the image is partitioned into small, atomic color regions. These small color regions are growing in the linking phase in a hierarchical manner to complete color segments. Within the linking phase it is possible to detect that color regions connected by a chain of smoothly changing colors have to be split again. This is done in the splitting phase. The most important parameter to set is the linking threshold based on the quadratic color distance between two pixels.

SRM: The SRM algorithm, for Statistical Region Merging [Nock and Nielsen, 2004] is a region-based segmentation algorithm following a particular order in the choice of regions. The merging criteria is based on an adaptive statistical threshold merging predicate on color channels that does not require to maintain dynamically the region adjacency graph. The algorithm is able to cope with hard noise corruption, handle occlusion, authorize the control of the segmentation scale through a free parameter Q.

EGBIS: The Efficient Graph-Based Image Segmentation algorithm [Felzenszwalb and Huttenlocher, 2004] is based on the definition of a predicate for measuring the evidence for a boundary between two regions using a graph-based representation of the image. The pairwise region comparison predicate considers the minimum weight edge between two regions in measuring the difference between them. An important characteristic of the method is its ability to preserve detail in low-variability image regions while ignoring detail in high-variability regions. A function parameter k controls the degree of difference between two regions with respect to their

internal differences. The parameter σ control the (optional) smoothing of the input image.

Hysteresis thresholding: This straightforward algorithm considers any pixel above the upper threshold T_{high} and under the lower threshold T_{low} as a background pixel. It does not take into account any spatial coherency.

CWAGM: CWAGM means Color Watershed - Adjacency Graph Merge. The algorithm [Alvarado Moya, 2004b] computes an over-segmentation with the watershed transformation and merge the regions to minimize the mean square error of a piece-wise constant image approximation. To compute which threshold should be used in the watershed segmentation, the accumulative histogram of gradient values will be used as a probability distribution. It will be assumed that the probability for a gradient value to be relevant must be greater than the given value p. The merge threshold m indicates a square distance between mean values in the color space weighted by the area of the regions.

The Table 5.1 summarizes these algorithms and gives important information concerning their free parameter with their ranges and default values provided by the algorithm's authors.

Algorithm	Free Parameter	Range	Default Value
CSC	t: region merging threshold	5.0-255.0	20.0
SRM	Q: coarse-to-fine scale control	1.0-255.0	32.0
EGBIS	σ: smooth control on input image	0.0-1.0	0.50
	k: color space threshold	0.0-2000.0	500.0
Hysteresis thresholding	T_{low}: low threshold	0.0-1.0	-
	T_{high}: high threshold	0.0-1.0	-
CWAGM	m: region merging threshold	0.0-200.0	100.0
	n: min. region number	1.0-100.0	10.0
	p: min. probability for watershed threshold	0.0-1.0	0.45

Table 5.1: Components of the segmentation algorithm bank, their names, parameters to tune with range and author's default values.

5.4.2 Parameter optimization Assessment

Before assessing the optimization procedure, we illustrate the optimization problem with some examples of evaluation profiles. We present 1D and 2D profiles for the different segmentation algorithms (except the EGBIS which has a parameter space in $\mathbb{R}^3$) for the four training images of the Figure 5.6. The best segmentation quality correspond to assessment value $E_I^A = 0$.

Concerning the CSC algorithm (see Figure 5.7), the shapes of the curves are similar for the four images and present a global minimum which falls in the same part of the parameter space. The global optima for the SRM algorithm (see Figure 5.8) are found in a very narrow band of the parameter space. Many local optima characterizes the curves of the EGBIS algorithm (see Figure 5.9). The thresholding algorithm behavior is more straightforward regarding to the obtained curves (see Figure 5.10). Globally, two performance levels are revealed where "good performances" are achieved for a large range of the parameter values. However, the global optima is more difficult to see since the difference between the "good" performance level (in blue) and its level is very thin. From these observations, we can conclude that the evaluation profiles are not always convex hulls and their granularity can depend on the image.

The set up of the Simplex algorithm and the Genetic algorithm used to find global minimum are described in Table 5.2.

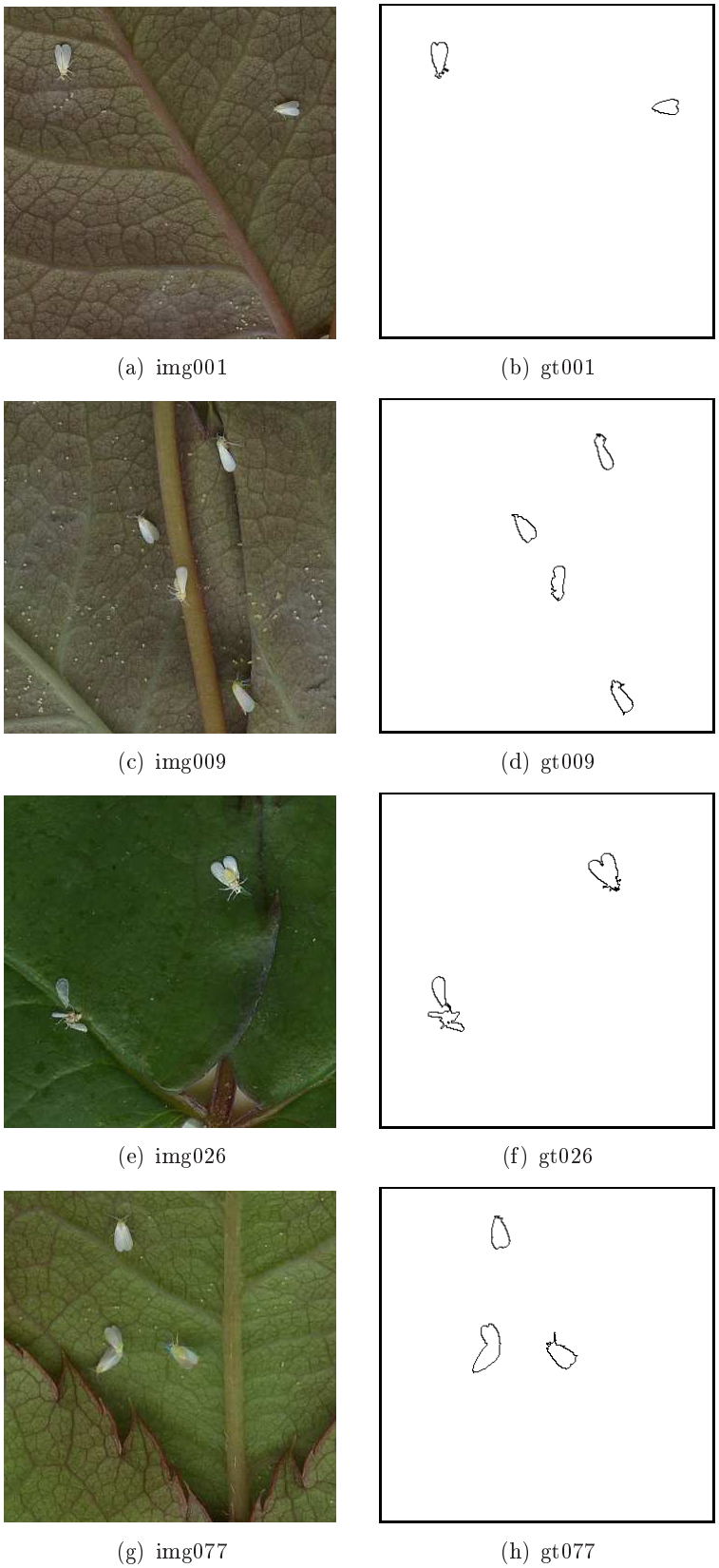
Simplex Algorithm	Genetic Algorithm
• starting step value = 0.5	• initial population size = 40
• ending criteria = 0.001	• initial number of generations = 20
• simplex coefficients (α=0.5, β=1.0,	• nb of generations to convergence = 5
γ=0.5, σ=2.0)	• cross-over probability rate = 0.7
• maximum number of calls = 80	• mutation probability rate = 0.05

Table 5.2: Set up of the optimization algorithms.

Since the Simplex algorithm does not guarantee to obtain a global optimum, we divide each parameter space into three sub-spaces and run an optimization on each sub-space. This means that 3^N optimization loops are run for a segmentation algorithm with N free-parameters.

Table 5.3, and 5.4 present the optimization results of the five segmentation algorithms in terms of segmentation performance. Globally, all the algorithms reach a good level except the EGBIS algorithm, as shown in Figure 5.11. This result is due to the fact that this algorithm is sensitive to small gradient variations. As expected, the EGBIS has a big standard deviation (due to the presence of many local optima) whereas the thresholding one is low (due to its straightforward behavior). We have also compared the performances of the optimization algorithms (the Simplex and the GA) with a systematic search method (see Table 5.5. By systematic, we means an iterative search throughout the whole parameter space with a fixed sampling rate. The sampling rate depends on the dimensionality of the parameter space. The global performance of the three methods are similar with a very little advantage to the Simplex.

To decide between the three different methods, we have compared them by considering their computational cost as described in Table 5.6. The systematic search is obviously the most costly method. The Simplex is the fastest method to converge apart from the CWAGM algorithm. According to the previous perfor-



(a) img001 (b) gt001

(c) img009 (d) gt009

(e) img026 (f) gt026

(g) img077 (h) gt077

Figure 5.6: Four representative training images and associated ground truth segmentations used in figure 5.7 to figure 5.10.

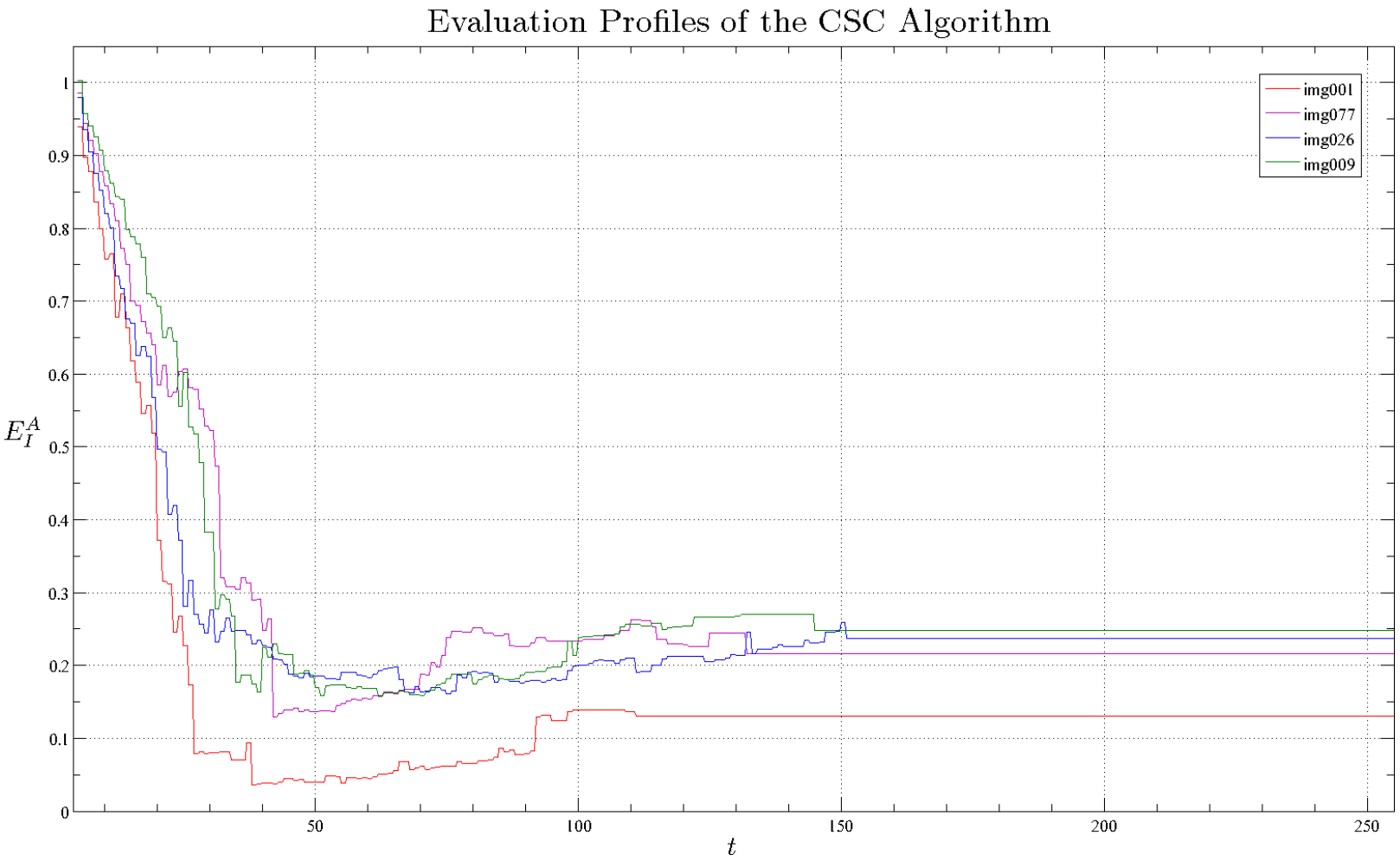


Figure 5.7: Evaluation profiles of the CSC algorithm applied on the four training images presented in Figure 5.6. $E_I^A = 0$ corresponds to the optimum.

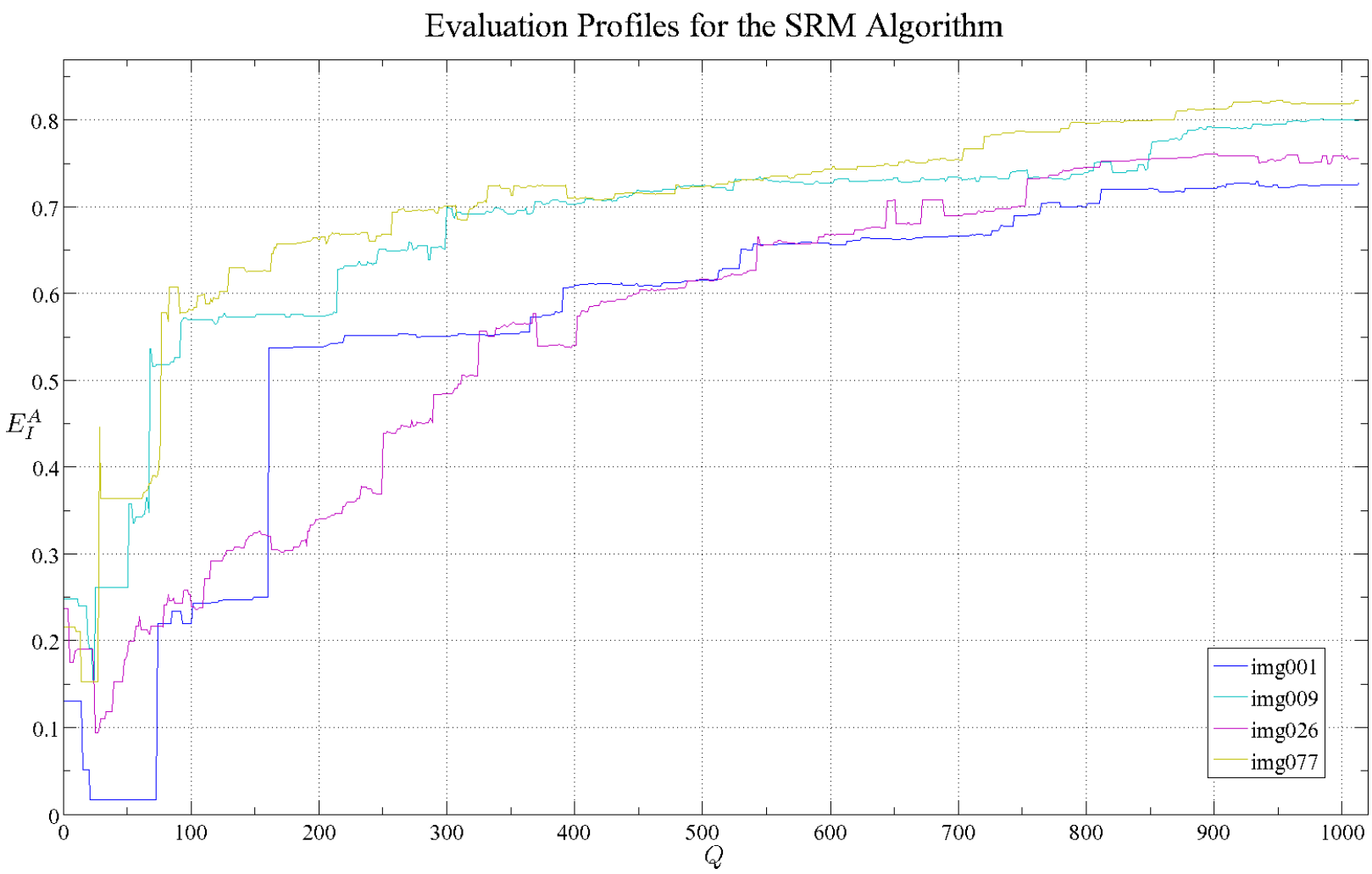


Figure 5.8: Evaluation profiles of the SRM algorithm applied on the four training images presented in Figure 5.6. $E_I^A = 0$ corresponds to the optimum.

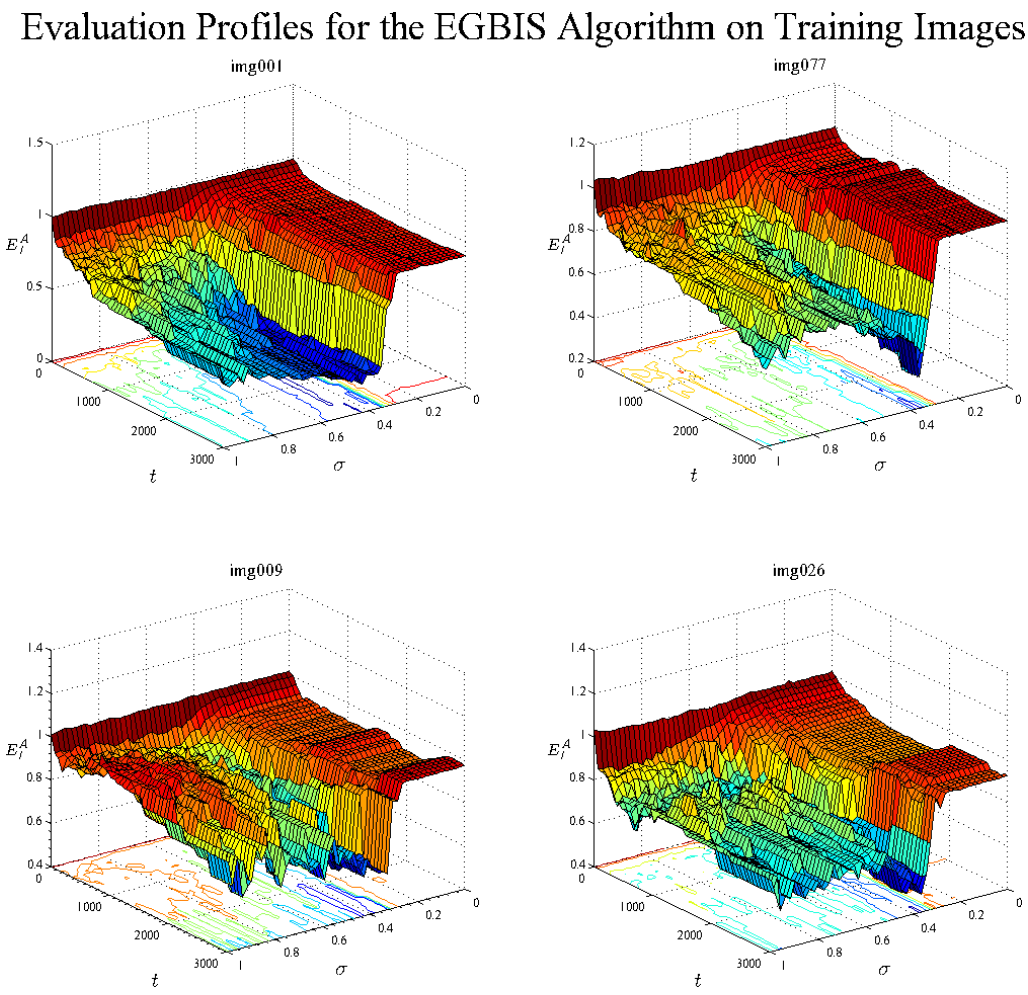


Figure 5.9: Different evaluation profiles of the EGBIS algorithm applied on the four training images presented in Figure 5.6. $E_I^A = 0$ corresponds to the optimum. t and σ are the two free parameters.

Algorithm	E_I^A using the simplex algorithm			
	min	max	mean	std
CSC	0.000	0.497	0.139	0.110
SRM	0.000	0.522	0.126	0.114
THRESH	0.000	0.351	0.113	0.092
EGBIS	0.0620	0.734	0.366	0.142
CWAGM	0.000	0.436	0.119	0.089

Table 5.3: Statistics on the optimization performances for the training image set using the Simplex algorithm.

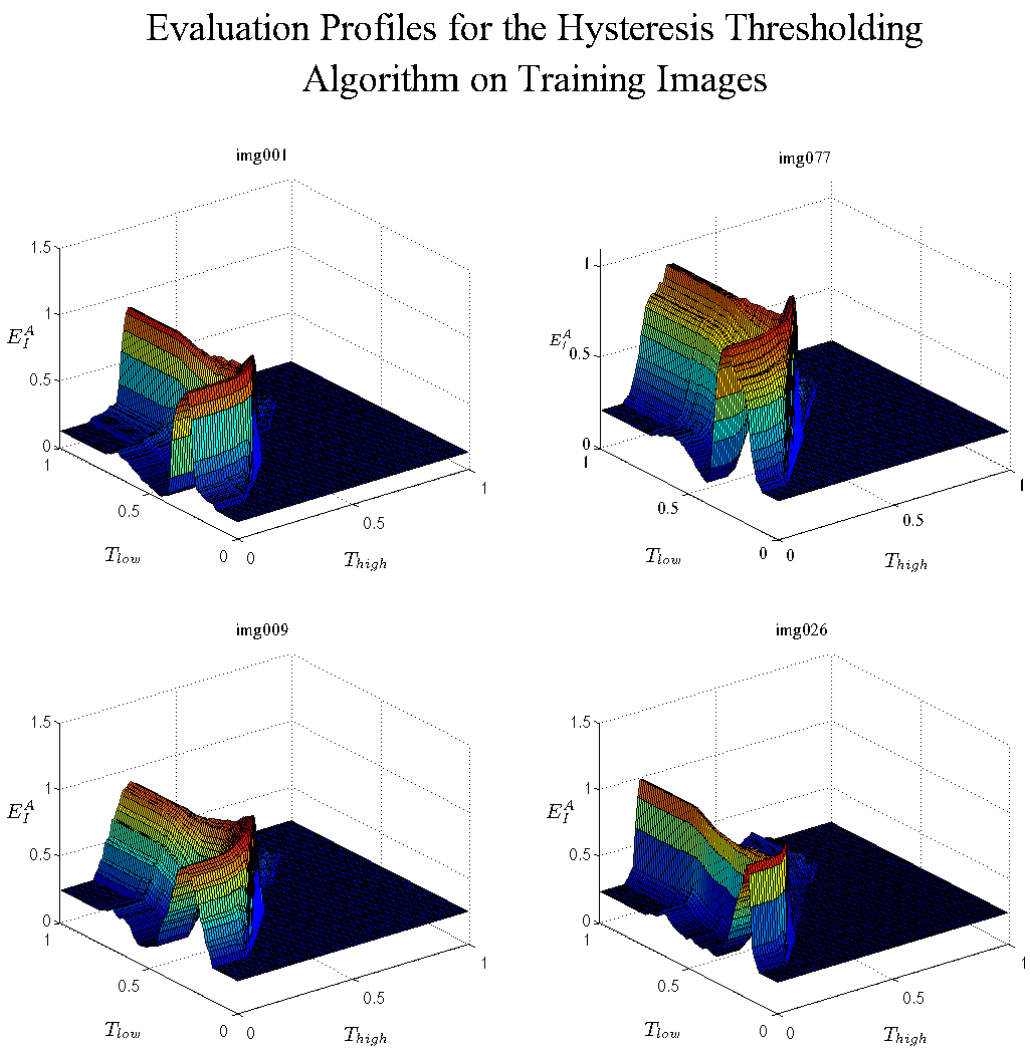
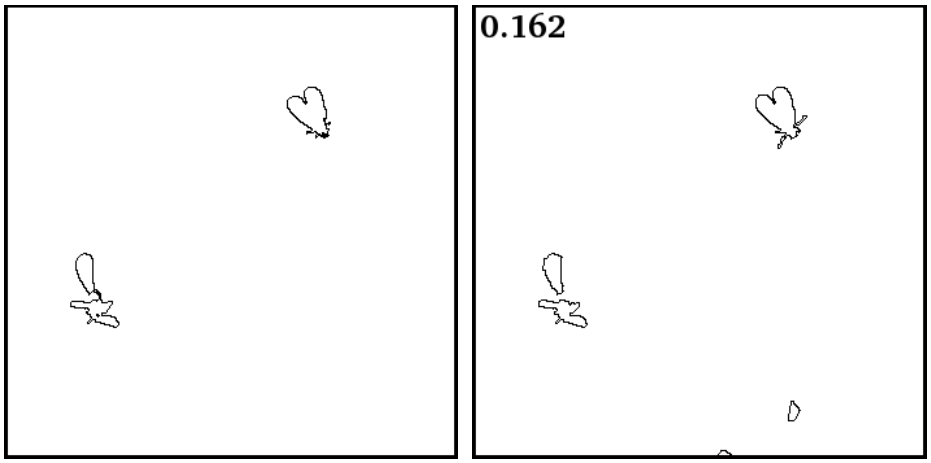


Figure 5.10: Different evaluation profiles of the hysteresis thresholding algorithm applied on the four training images presented in Figure 5.6. $E_I^A = 0$ corresponds to the optimum. T_{low} and T_{high} are the two free parameters.

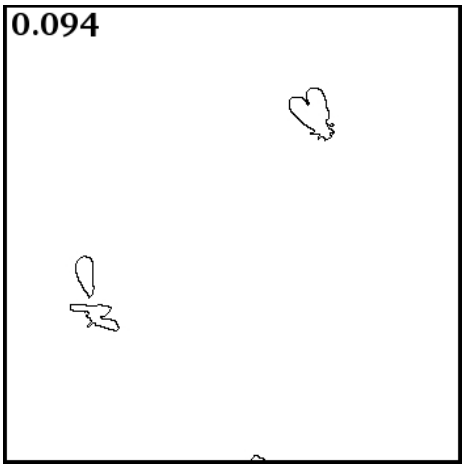
Algorithm	E_I^A using the genestic algorithm			
	min	max	mean	std
CSC	0.000	0.462	0.134	0.099
SRM	0.000	0.485	0.123	0.100
THRESH	0.000	0.348	0.114	0.091
EGBIS	0.118	0.708	0.371	0.140
CWAGM	0.000	0.436	0.118	0.090

Table 5.4: Statistics on the optimization performances for the training image set using the genetic algorithm.

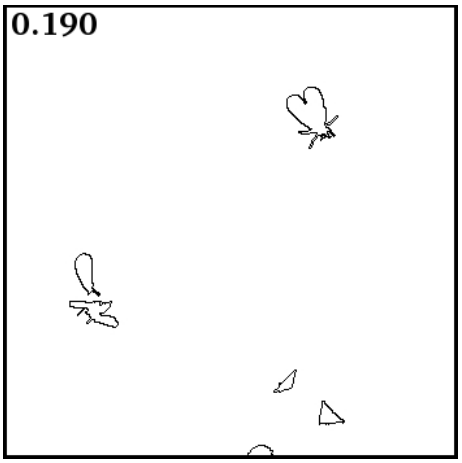


(a) ground truth

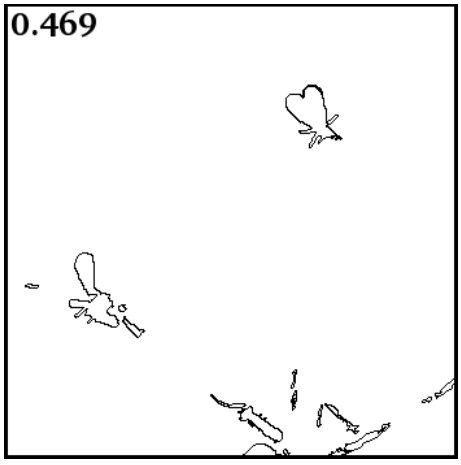
(b) CSC



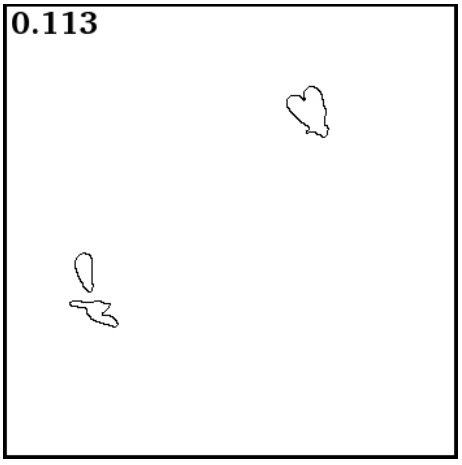
(c) SRM



(d) THRESH



(e) EGBIS



(f) CWAGM

Figure 5.11: Example of optimization results for the img026 compared to the ground truth with their performance scores (0 = no difference).

Algorithm	E_I^A using the systematic search			
	min	max	mean	std
CSC	0.000	0.462	0.129	0.097
SRM	0.000	0.479	0.116	0.099
THRESH	0.000	0.350	0.113	0.092
EGBIS	0.126	0.708	0.392	0.140
CWAGM	0.000	0.456	0.193	0.077

Table 5.5: Statistics on the optimization performances for the training image set using the systematic search.

mance score tables, the simplex is definitively the best algorithm to optimize low dimensional parameter spaces in a few number of iterations. For segmentation algorithms with more than two free-parameters, the Genetic Algorithm should be preferred, requiring less iterations for the same level of performance. Note that we have limited the number of iterations — mainly for computational cost reasons — for the systematic search method to 2550 for the EGBIS algorithm and to 1250 for the CWAGM algorithm, respectively. These two algorithms are relatively slow comparing to the others and the parameter space to explore is really huge, particularly for the CWAGM.

Algorithm	mean number of iterations		
	Systematic search	GA	Simplex
CSC	1000	733	83
SRM	1000	734	82
THRESH	10000	840	404
EGBIS	2550	840	497
CWAGM	1250	840	1821

Table 5.6: Computational cost of each optimization method.

The number of iterations is also dependent of the parametrization of the optimization algorithm. For the Simplex algorithm, it mainly depends on the $maxCalls$ parameters which specifies the maximum allowed number of calls of the fitness function in an optimization loop. A too low value will cause the algorithm to break before optimization is complete. A too high value will lead to needless calls of the fitness function. Figure 5.12 shows the influence of this parameter on the convergence accuracy. We start the test on the img001 with $maxCall$ set to 3 (minimum allowed by the algorithm) and increase it up to 80. For a one-dimensional parameter space, this means that the total number of iterations will be between 9 ($3\times$ 3) and 240 (3×80), for a two dimensional space between 27 ($3^2 \times 3$) and 720 ($3^2 \times 80$), and so on. The study of the graph brings us to several conclusions. First, for the CSC and SRM algorithms (one free-parameter), setting $maxCalls$ to 8 (i.e. a total number of iterations equals to 24) suffices to

reach the best performance at the convergence point. For the EGBIS and thresholding algorithms, the convergence points are reached after 45 and 117 iterations, corresponding to $maxCalls = 5$ and $maxCalls = 13$, respectively. 783 iterations are needed for the CWAGM algorithm ($maxCalls = 29$). As a consequence, the dimensionality of the parameter space to explore has to be taken into account for the setting of $maxCalls$ but excessive values are useless. This study also reveals that the parameter space is not explored in the same way, depending on the segmentation algorithm. Indeed, some algorithms, have parameter sub-spaces which induce flat evaluation profiles, as for instance the thresholding algorithm. In these sub-spaces, the Simplex converges in a few number of iterations. The same study is done for the GA and the results are graphically reported in Figure 5.13. We decide to assess the GA sensitivity to the initial population size. The number of initial points is here independent of the segmentation algorithm and varies between 20 and 840. The same conclusions can be drawn. We just can add that the EGBIS algorithm brings some problem to the GA which falls in many local optima (peaks of the EGBIS curve in Figure 5.13).

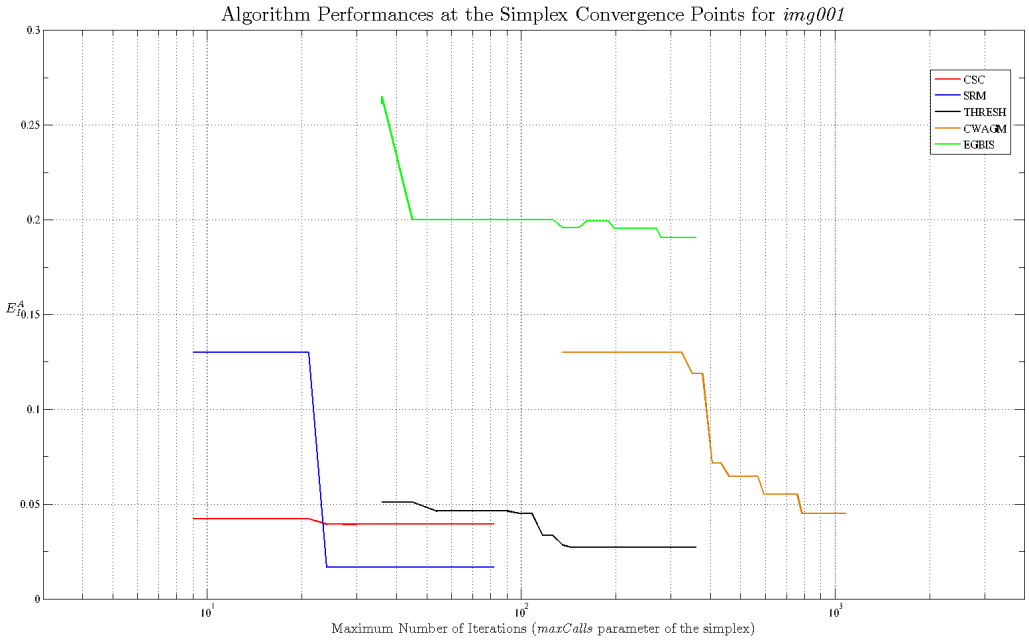


Figure 5.12: Convergence accuracy of the Simplex algorithm by varying the $maxCalls$ parameter.

5.4.3 Algorithm Selection

We applied the DBSCAN [Ester et al., 1996] algorithm to cluster the 20 training images as described in section 4.3.2. We obtain two clusters of 10 images (see Figure 5.14 for examples). Visually, the first cluster corresponds to the back side

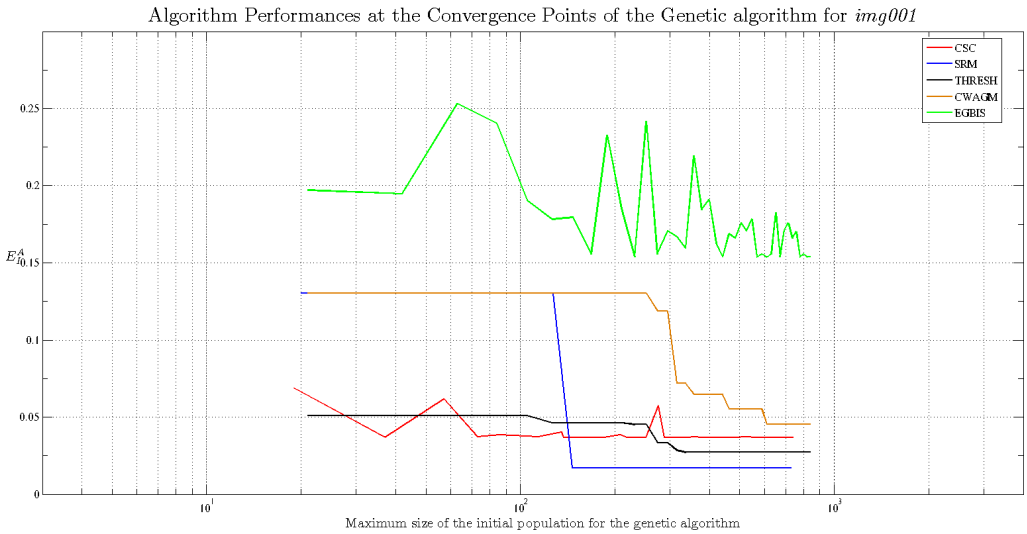


Figure 5.13: Convergence accuracy of the GA by varying the initial population size.

images of the scanned rose leaves and the second cluster to the front side images. For each cluster, mean parameter sets of the five segmentation algorithms

Figure 5.14: Examples of images for the two identified clusters. Left = cluster 1 (front side of the leaves), right = cluster 2 (back side of the leaves).

are computed w.r.t. their performance scores. The segmentation performances of the tuned algorithms are evaluated on each training image sub-set. The tuned algorithm which gets the best mean performance score for each cluster is elected. Before the last ranking step, the best algorithm for the first cluster was the hysteresis thresholding algorithm and the best for the second cluster was the CSC algorithm. After the last ranking step, the CSC algorithm was found as the best

one for the two clusters but with different parameter set. This means that even if the thresholding algorithm performs better in individual cases, the CSC algorithm is more robust than the thresholding algorithm when tuned with a mean parameter set.

5.4.4 Region-Classifier Performance Assessment

For each identified image cluster, region labels of annotated manual segmentations are mapped into regions of the segmented image following the method described in Chapter 4.4.1. Then, for each region class, a region classifier is trained with region features as input. We used our wrapper scheme detailed in Chapter 4.4.2 to optimize the classifier performances. Three color space are used in this experiment: RGB, HSV, and XYZ. The performances of the classifiers trained respectively on the whole training set, the cluster 1, and the cluster 2 are plotted in Figure 5.15, Figure 5.16, and Figure 5.17. The study of these three graphs leads to some conclusions. First, HSV is the more discriminative color space in this problem except for the cluster 2 where better results are achieved with the XYZ color space. The CIE XYZ color space was deliberately designed so that the Y parameter was a measure of the brightness or luminance of a color. For the context 2, the brightnesses of the classes are very different. That's why the XYZ color space is here well-adapted. Second, the optimization of the SVM parameters increases the classifier performances of 5-10%. The best cross-validation rates are reached with q (quantization level) values superior to 50.

We have also tested texture features but their performances are 10% inferior in mean than with the color features as shown in Figure 5.18.

Finally, the classifiers are trained a last time with the configurations giving the best cross-validation rates. The final set up of the different algorithms is then as follows (Table 5.7):

Context	Seg. Alg. (param)	Class	Feature extractor param.		SVM param.	
			Color space	q	C	γ
context 1	CSC	rose leaf	HSV	112	4	1
(light green leaves)	(41.9)	white fly	HSV	112	1	4
context 2	CSC	rose leaf	XYZ	21	64	4
(dark green leaves)	(48.74)	white fly	XYZ	21	256	0.25

Table 5.7: Set up of the segmentation, the feature extractors, and the classifiers.

5.4.5 Final Segmentation Quality Assessment

In this section, we present the segmentation results on the test set. We compare six different methods, comprising (parts of) our approach and a pure top-down segmentation.

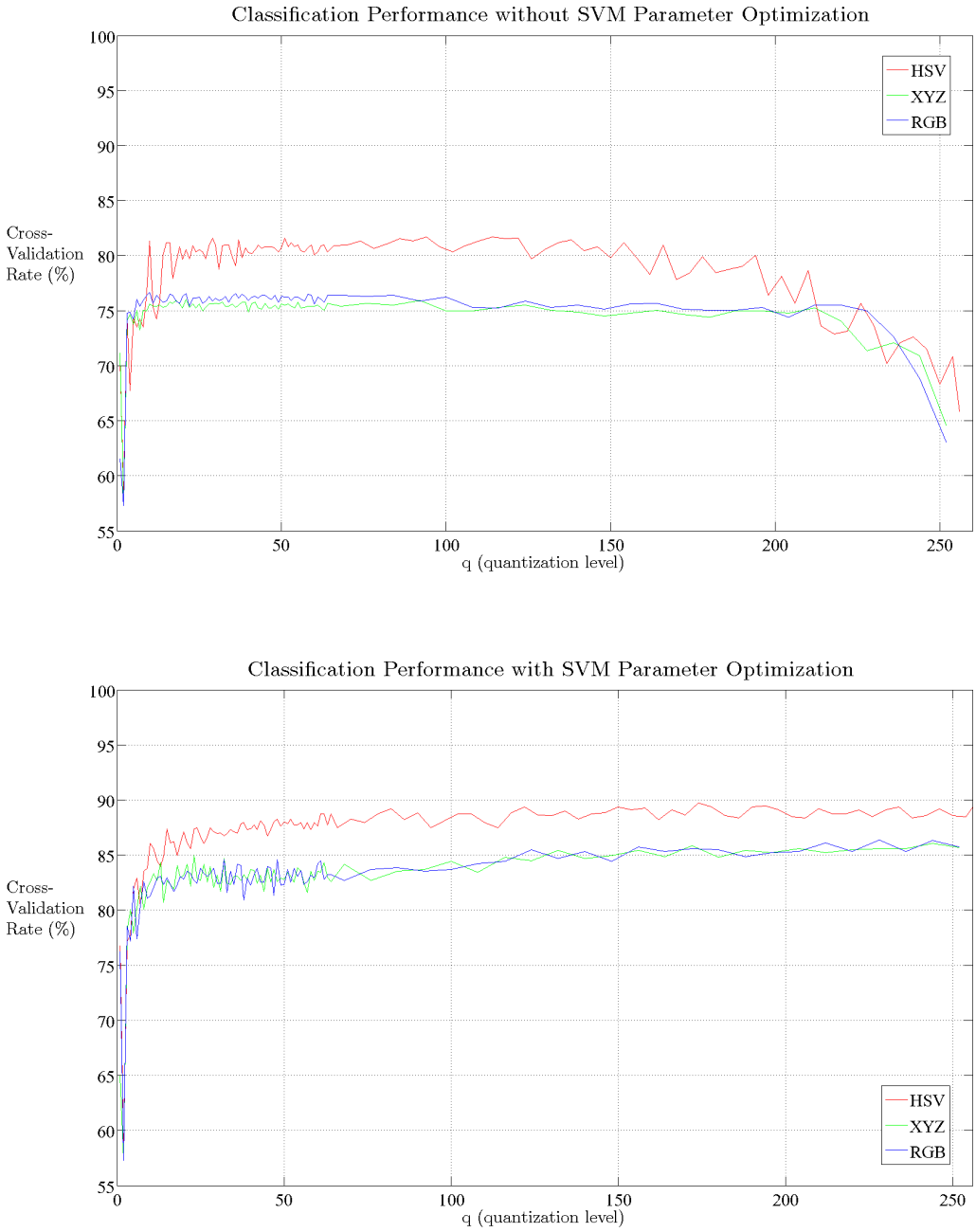


Figure 5.15: Performances of the region classifiers trained on the whole training set and different color features.

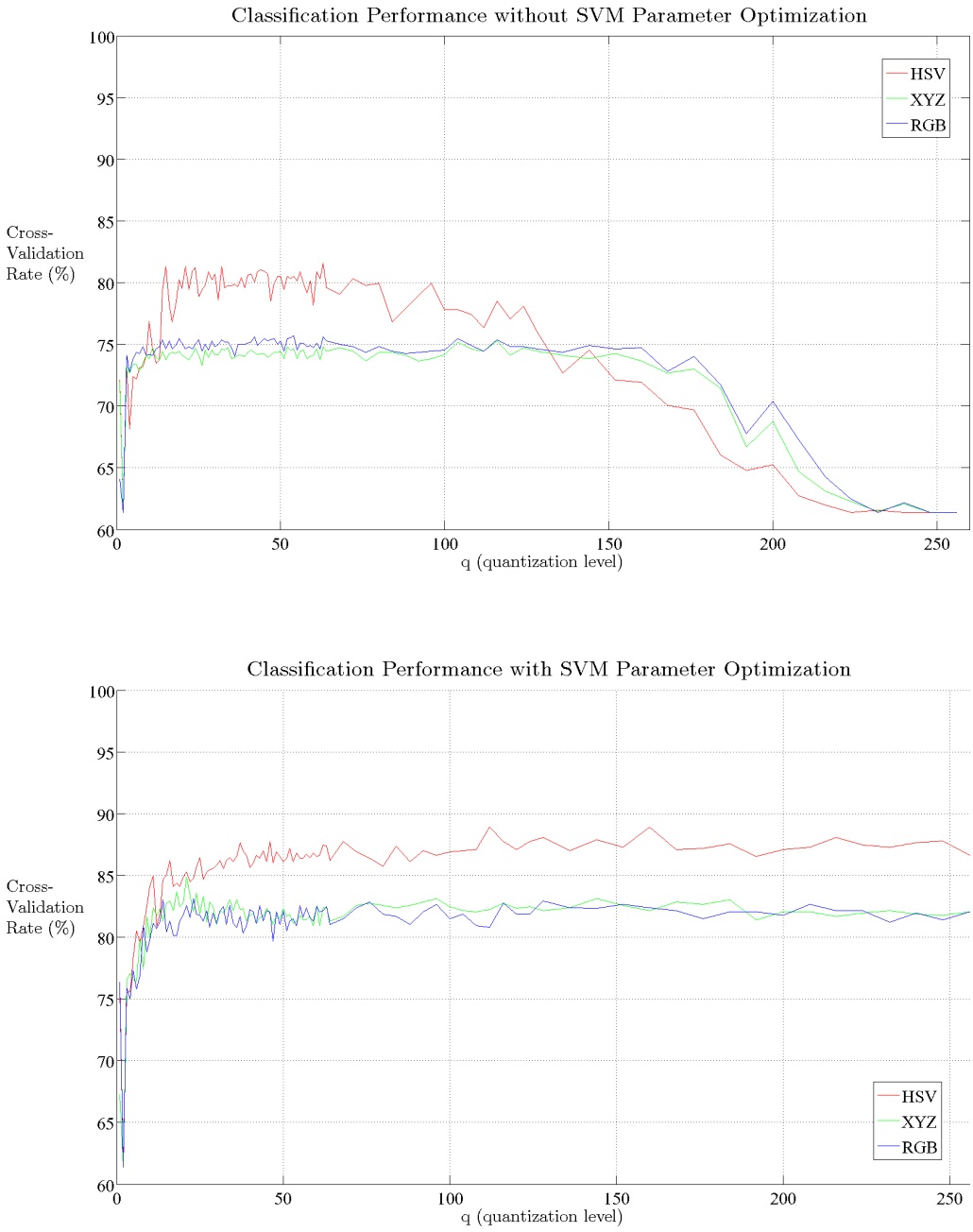


Figure 5.16: Performances of the region classifiers trained with the ten images of the cluster 1 (light green rose leaves) and different color features.

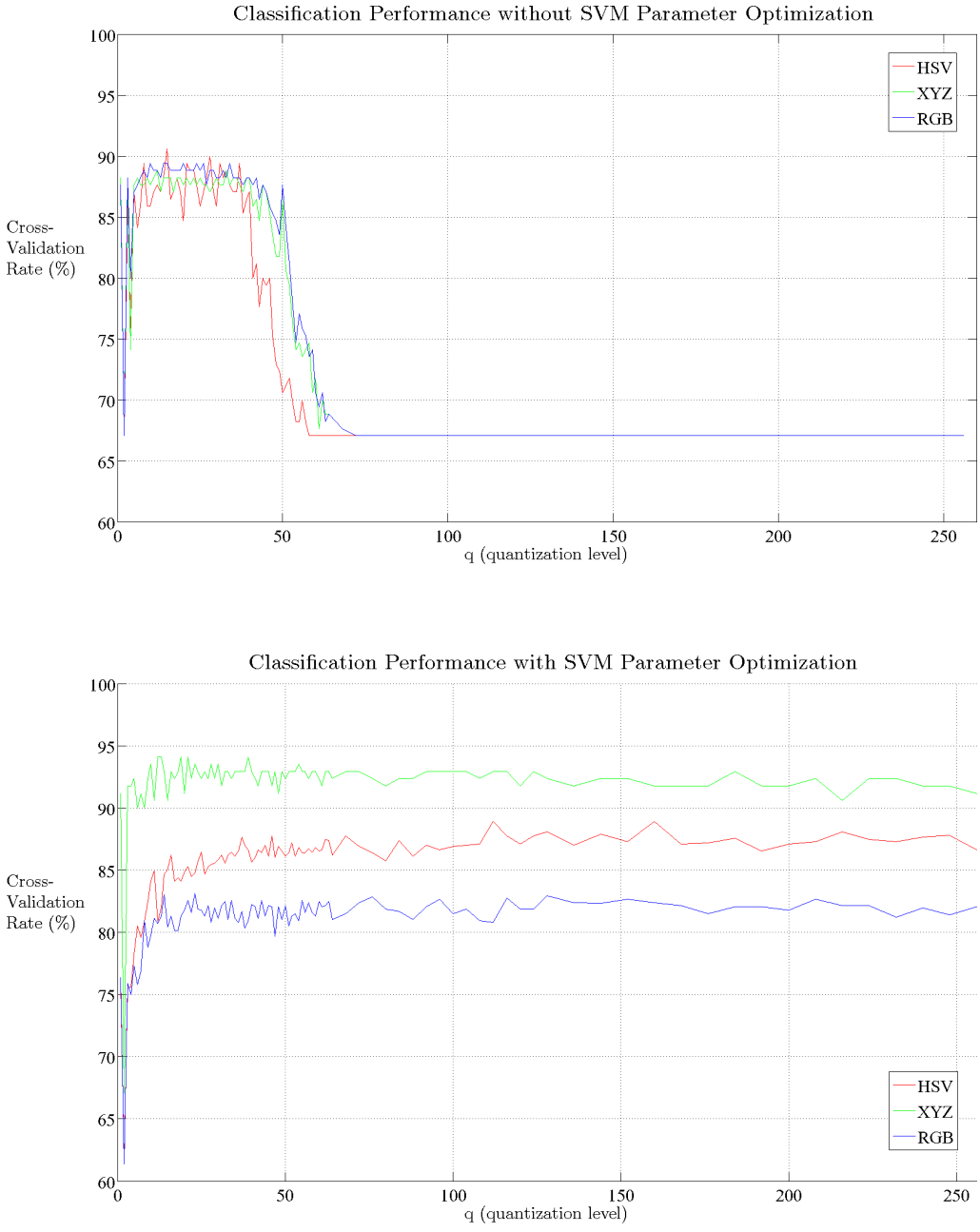


Figure 5.17: Performances of the region classifiers trained with the ten images of the cluster 2 (dark green rose leaves) and different color features.

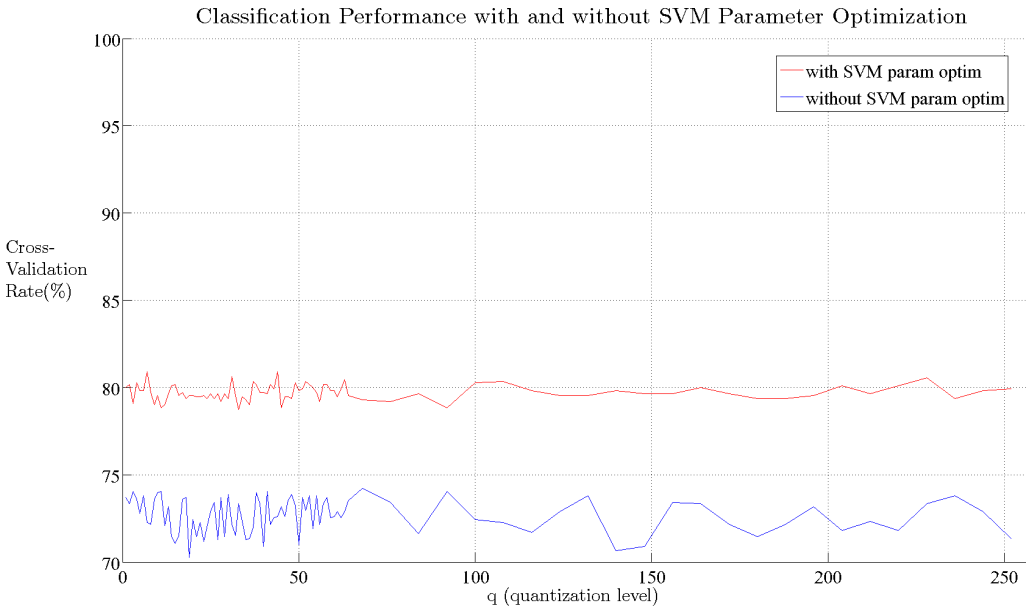


Figure 5.18: Performances of the region classifiers trained with the whole training set and texture features.

- method 1: ad hoc segmentation, with the Hysteresis thresholding algorithm tuned with $T_{low} = 0.45$ and $T_{high} = 1.0$.
- method 2: algorithm selection and tuning based on the learnt parameters from the whole training set (CSC is the best algorithm),
- method 3: method 2 + semantic segmentation (region labelling),
- method 4: algorithm selection and tuning based on image content analysis (one algorithm with learnt parameters per context),
- method 5: method 4 + semantic segmentation,
- method 6: over-segmentation + semantic segmentation

The over-segmentation used in method 6 is performed with the CWAGM algorithm manually tuned with a very low region merging threshold (see Figure 5.19).

Performance scores of the test set are summarized in Table 5.8 and some examples of results for four test images are presented in Figure 5.20, Figure 5.21, Figure 5.22, and Figure 5.23. Methods 3 and 5 gives the best result. This result is predictable since the segmentation algorithm used for the method 5 is the same (CSC) and the parameter settings for the context 1 is close to the one for the context 2. The small difference between the performance scores of methods 3 and 5 is at the method 3 advantage. The white fly region classifier for the context 2 has been trained on few samples since there are not many white flies on the front side of rose leaves. Consequently, the classification errors for the white fly class are higher for the method 5 context 2 than for the method 3. In a biological point

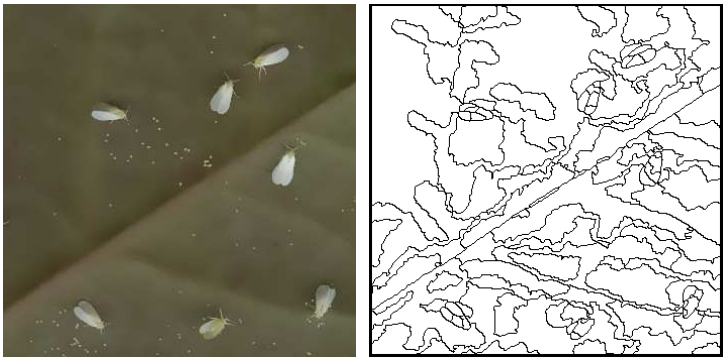


Figure 5.19: Example of an initial over-segmented image used in method 6.

of view, insects prefer to live hided on the back side of the leaves, where they are better camouflaged (low contrast, not visible, etc.). Method 6 does not perform better results even if its initial over-segmentation is more precise (i.e. less missed boundary pixels) than with the CSC algorithm in methods 2 to 5.

method	Performance scores of the segmentation			
	min	max	mean	std
1	0.000	0.351	0.095	0.080
2	0.000	0.779	0.213	0.164
3	0.000	0.654	0.122	0.139
4	0.000	0.832	0.234	0.170
5	0.058	0.617	0.123	0.140
6	0.000	0.668	0.153	0.144

Table 5.8: Statistics on the segmentation performances for the test set using different segmentation strategies.

5.4.6 Overall System Assessment

To assess the quality of the cognitive system, the results have been compared with ground truth. Three human operators (one expert in agronomy, one expert in image processing and one non-expert neither in agronomy nor in image processing) have manually counted the white flies on 180 images. Each operator has a different point of view when counting. The expert in image processing focuses on pure visual characteristics while the expert in agronomy focuses more on the semantic meaning of images. This can lead to different counting results as illustrated in Figure 5.24: the expert in agronomy counts three white flies, the expert in image processing only one (because only one object matches the visual criteria), and the non-expert two. This explains the size of some error bars on ground truths in Figure 5.25 (e.g., samples 142 and 148).

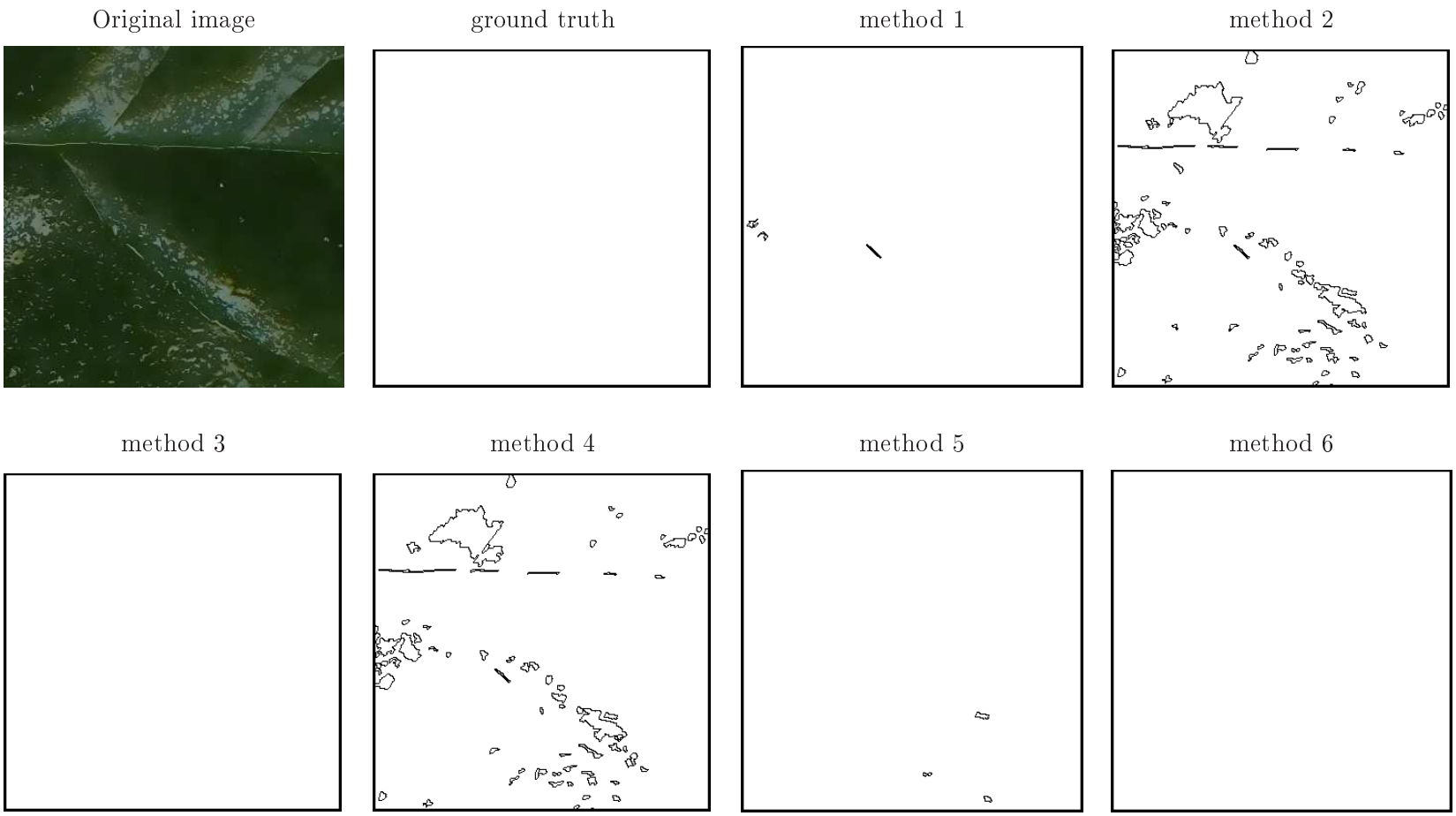


Figure 5.20: Examples of results on a test image for different segmentation configurations (1).

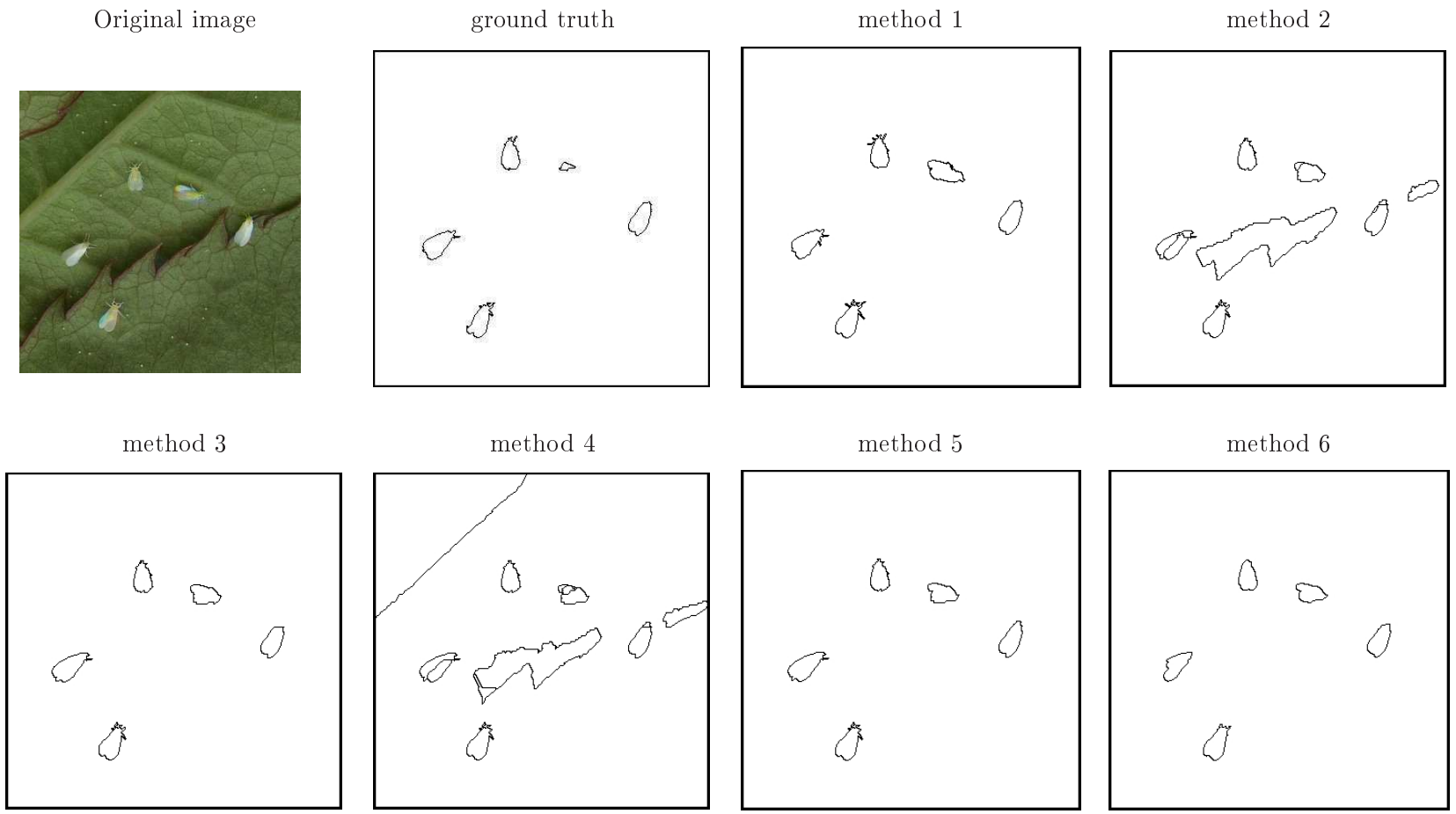


Figure 5.21: Examples of results on a test image for different segmentation configurations (2).

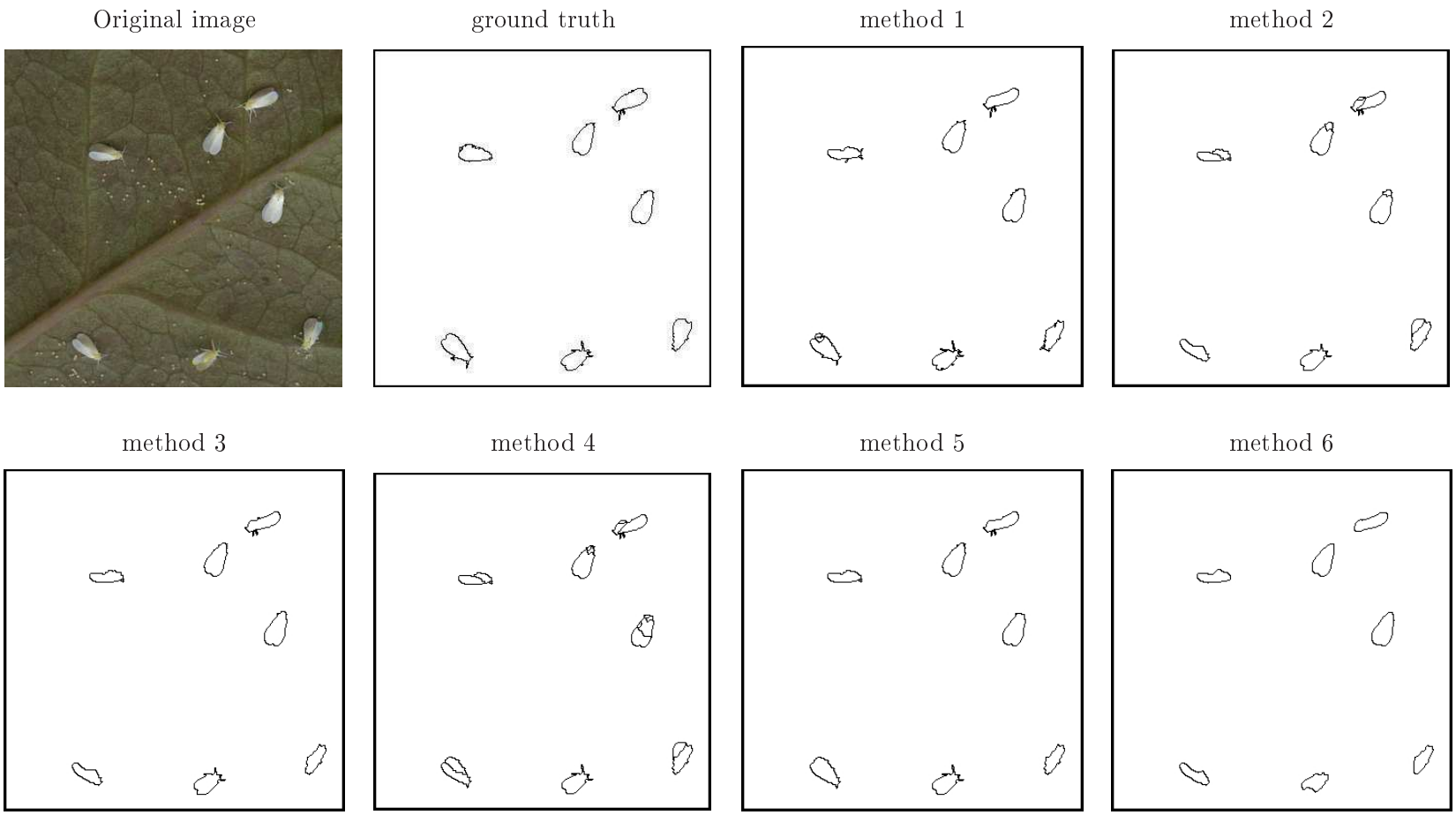


Figure 5.22: Examples of results on a test image for different segmentation configurations (3).

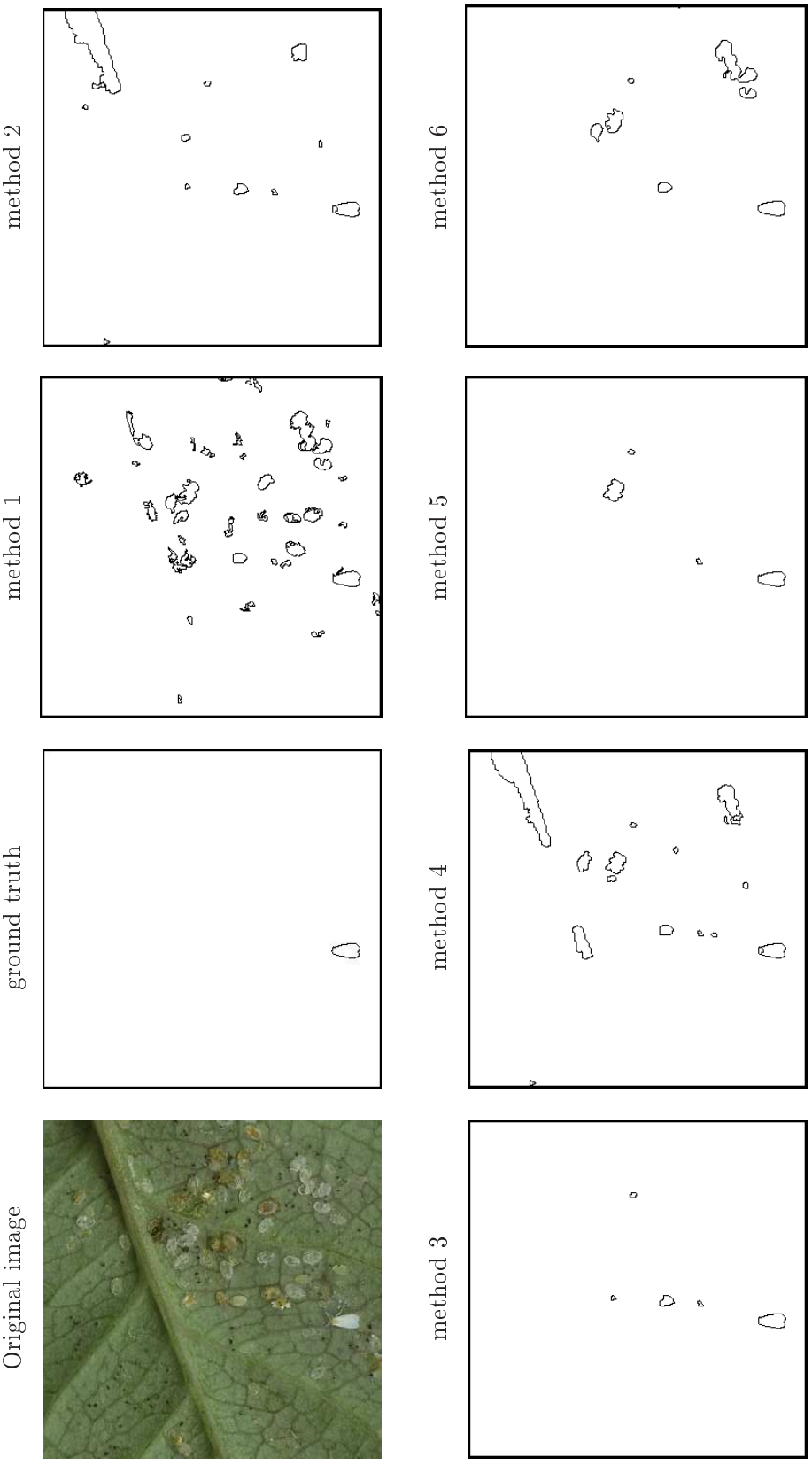


Figure 5.23: Examples of results on a test image for different segmentation configurations (4).

Figure 5.24: Example of an ambiguous image sample for ground truth estimation. The two white flies on the top have moved during the scanning. This leads to color flickering which do not correspond to the normal white fly color.

We detail hereafter the result evaluation for early detection of mature white flies. From the 180 images composing the test set, 162 contain between zero and five white flies. Figure 5.25 presents the detailed results for the whole test set. For each image the average ground truth value (blue circle) is reported with its associated error bar. Red crosses represent the values found by the system. To prove the reliability of our learning approach, we have tested it against an *ad hoc* segmentation (i.e. a manually tuned algorithm): a hysteresis thresholding segmentation on gray-scaled normalized image (i.e. pixel values in $[0, 1]$) with low threshold (T_{low}) fixed to 0.45 and high threshold (T_{high}) fixed to 1.0. The two graphs present the results of mature white fly counting. The top graph corresponds to the system configured with *ad hoc* segmentation and the bottom graph corresponds to the system configured with our learning approach.

Globally, the detection rate is satisfactory. To fully make use of the results, we can separate the test samples into two classes depicting the most relevant situations. The first class (C_1) represents images without any mature white fly (i.e. images for which the ground truth error bar maximum is strictly inferior to 1.0) and the second class (C_2) represents images with at least one white fly detected (i.e. images for which the ground truth error bar maximum is equal or superior to 1.0). We define the False Positive Rate (FPR) as the rate of over-detection (i.e. images for which the number of detected white flies is greater than the ground truth error bar) and the False Negative Rate (FNR) as the rate of under-detection (i.e. images for which the number of detected white flies is less than the ground truth error bar). Table 5.9 summarizes the detection results. The figures represent the mean values of FNR and FPR for class C_1, C_2, and for the whole image test set.

The FNRs are roughly similar for the two configurations. In fact, this reveals confusing situations as the one presented in Figure 5.26: two overlapping white flies have been segmented into one region which has obviously not the shape of a single white fly. Hence, the system counts one white fly instead of three. This highlights the scale issue of our problem for which highly variable small objects are expected to be detected in a complex natural environment. Concerning the FPRs, they are up to four times smaller with the learnt segmentation than with

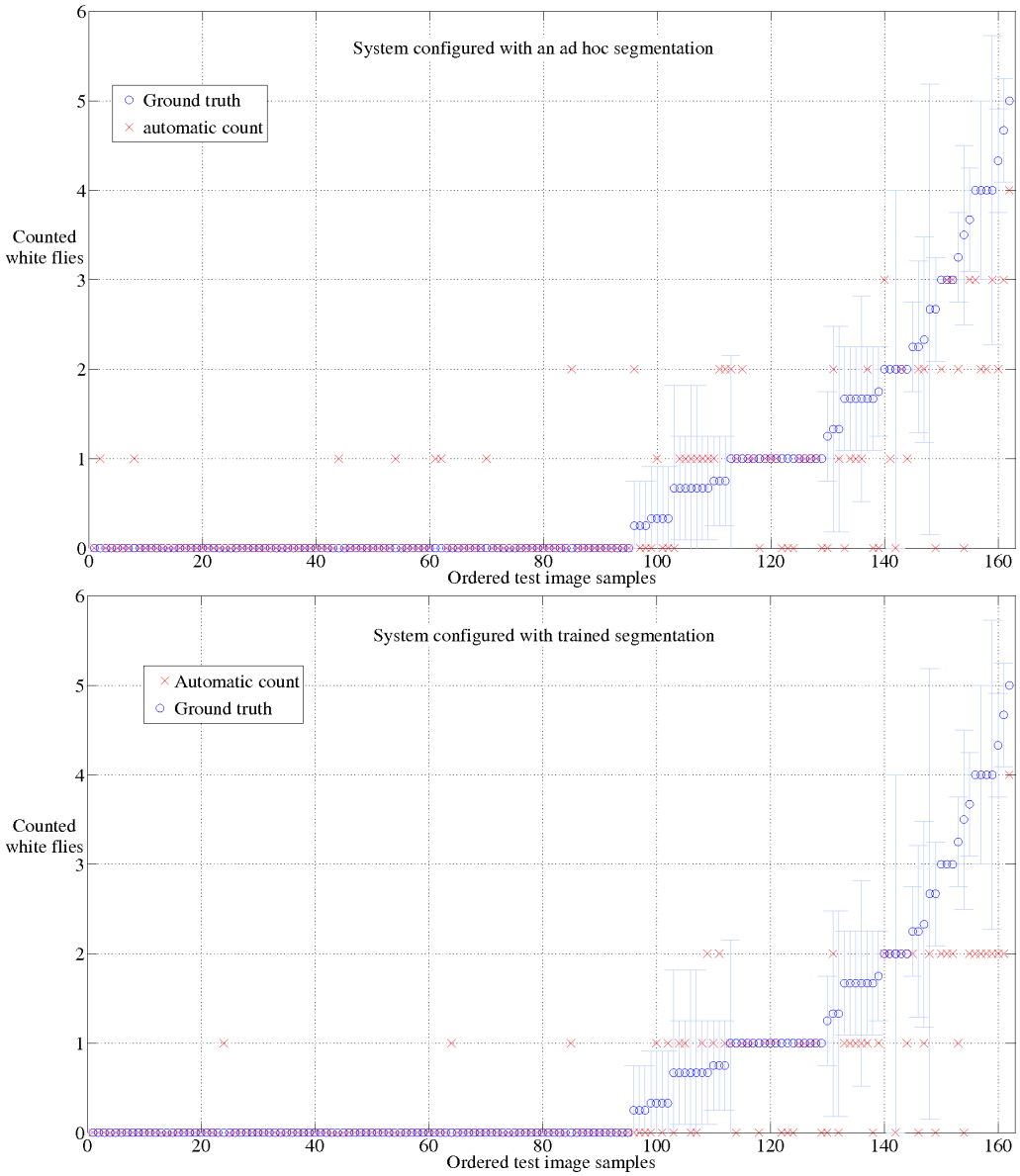


Figure 5.25: Evaluation of mature white fly counting results in early detection cases (i.e. between 0 and 5 flies per leaf). The upper graph presents the results for the system configured with trained segmentation parameters, the lower one presents the results for the system configured with an *ad hoc* segmentation.

	Results for early detection of mature white flies			
	With ad hoc segmentation		With learnt segmentation	
Samples	FNR (%)	FPR (%)	FNR (%)	FPR (%)
C_1 (102)	-	9.6	-	3.1
C_2 (60)	24.7	9.0	29.6	2.0
Whole set (162)	9.1	9.4	11.0	2.7

Table 5.9: False Negative Rate (FNR) and False Positive Rate (FPR) for test images with no white flies (class C_1), at least one white fly (class C_2) and for the whole test set.

the *ad hoc* one. This is due to our adaptive segmentation approach that allows to efficiently tune algorithm parameters with respect to variations in leaf color and contrast.

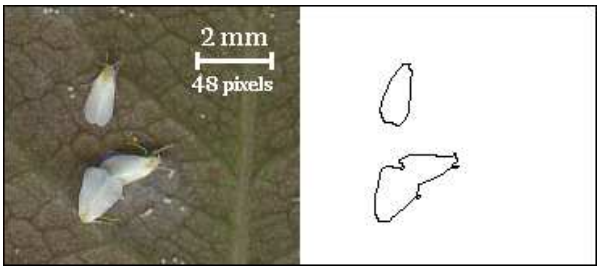


Figure 5.26: Example of an ambiguous situation leading to a wrong interpretation result.

5.5 Evaluation on a Public Image Database

In this section, we present evaluation results of the parameter optimization step on a public image database.

The goal of the Berkeley Segmentation Dataset and Benchmark (BSDB) image database [Fowlkes and Martin, 2007] is to provide an empirical basis for research on image segmentation and boundary detection. To this end, the authors have collected 6000 hand-labeled segmentations of 500 Corel dataset color images from 30 human subjects. The images depictes natural scenes with at least one foreground objet (e.g., an animal, a plant, a person, etc.). The public benchmark based on this data consists of all of the segmentations for 300 images. The images are divided into a training set of 200 images, and a test set of 100 images. The ground truth are not labelled and the possible semantic classes are too numerous. Consequently, we do not assess the semantic segmentation part of our framework on this image database.

The evaluation metric proposed in this image database for the benchmarking cannot be used with region-based segmentation algorithms since it relies on soft

boundary maps of edge-based segmentation results (e.g. maps of gradient magnitude). We thus prefer our segmentation performance metric. For each image, several human segmentation exists (from three to eight) with different levels of refinements. We have decided to take into account the finest ones. Then, for each segmentation algorithm of our algorithm bank and for each image, algorithm parameters are optimized thanks to the manual segmentation. As previously done in section 5.4.2, we have compared the optimized segmentation achieved with the three optimization algorithm based on: the Simplex algorithm (Table 5.10), the Genetic Algorithm (Table 5.11), and a systematic search (Table 5.12). Globally the three optimization algorithms performs in mean comparable results. This confirm the reliability of our parameter tuning approach for this image database.

Algorithm	E_I^A using the Simplex algorithm			
	min	max	mean	std
CSC	0.285	0.768	0.639	0.079
SRM	0.246	0.702	0.529	0.079
THRESH	0.193	0.680	0.510	0.081
EGBIS	0.207	0.632	0.495	0.077
CWAGM	0.224	0.691	0.530	0.081

Table 5.10: Statistics on the optimization performances using the Simplex algorithm.

Algorithm	E_I^A using the genetic algorithm			
	min	max	mean	std
CSC	0.373	0.756	0.633	0.086
SRM	0.232	0.665	0.532	0.088
THRESH	0.192	0.682	0.514	0.092
EGBIS	0.202	0.618	0.499	0.081
CWAGM	0.224	0.675	0.530	0.090

Table 5.11: Statistics on the optimization performances using the genetic algorithm.

Algorithm	E_I^A using the systematic search			
	min	max	mean	std
CSC	0.252	0.762	0.594	0.087
SRM	0.234	0.696	0.528	0.079
THRESH	0.376	0.648	0.521	0.064
EGBIS	0.337	0.600	0.509	0.066
CWAGM	0.496	0.0677	0.589	0.091

Table 5.12: Statistics on the optimization performances using the systematic search.

5.6 Conclusion

In this chapter, we have presented the validation of our framework for adaptive image segmentation on a biological application. We have paid a special attention to the assessment of each step of our learning module. We have seen that our optimization procedure is able to extract optimal parameters of different segmentation algorithms. The optimization is reasonable in terms of computational cost and delivers compatible results with the application needs. Then, region classifiers have been trained on a relative small set of training images (50) for which the user has provided manual segmentations and, regions annotations for two object classes. The qualitative and quantitative evaluations of the results demonstrate the potential of our method for this application. On the test image set, our adaptive segmentation outperforms an *ad hoc* segmentation with in mean 50% less segmentation errors according to our performance evaluation metric.

We have also shown how our framework can be used into a cognitive vision system dedicated to the detection and counting of insects on rose leaves to enrich the segmentation task with learning and adaptability faculties. Global performance of the system has been improved thanks to our adaptive segmentation and decrease the false rate detection in a factor three.

We have however limited our experiment to a figure-ground segmentation problem. Further experiments with more training data (i.e. more semantic classes) are necessary to fully validate our framework.

Chapter 6

Adaptive Figure-Ground Segmentation in Video Surveillance Applications

6.1 Introduction

Figure-ground segmentation of videos consists in separating the foreground pixels of the background pixels. In video applications, the variability of the two classes makes the detection of foreground pixels fairly impossible to predict without motion information. A widely used method to tackle this problem is to model the background in order to detect only moving pixels. If some techniques (e.g., median filtering [Prati et al., 2003], mixture of Gaussian [Stauffer and Grimson, 1999], kernel density estimator [Elgammal et al., 2000], codebooks [Kim et al., 2005]) have proved to be efficient in specific situations, the maintenance of background models in long-term videos of changing environment is still a real challenge. More precisely, these techniques are still not able to cope with both gradual changes (e.g., due to the change of the location of the sun) and sudden changes (e.g., due to the passage of clouds in front of the sun). In video surveillance applications, such situations are common, for instance in outdoor site or building entrance surveillance. Therefore, a need exists in improving the segmentation step to achieve a robust detection of moving objects of interest in every expected situations.

In this chapter, our objective is to remove the restrictions on the use of video segmentation algorithms. More precisely, our aim is to show how our algorithm selection method based on image context analysis can be used for the dynamic selection of background model. The goal is to divide the background modelling problem into more tractable sub-problems, each of them being associated with a specific context.

6.2 Meta-Learning for video segmentation algorithms

6.2.1 Targeted Applications

We consider the problem of the figure-ground segmentation task in video surveillance applications where both quick-illumination changes and long term changes are present. In this context, the major difficulty at the segmentation level is to deliver robust results whatever lighting changes occur in the scene. These lighting effects can be induced by weather conditions changes in outdoor scenes, by the switching of an artificial lighting source in indoor scenes, or by a combination of changes of different natures. The consequences at the pixel level are variations of intensity, color saturation, or inter-pixel contrast. At the image level, these changes can affect just a local area or the whole image. Another source of problems arises from the presence of non-static objects in the background as swaying trees or mobile objects as chairs. All these background variations make the foreground detection problem very difficult.

6.2.2 Targeted Algorithms

To estimate the motion, a basic approach is to compute the difference between a background image, called the reference image, and the current frame. The result is then thresholded to get a binary image of moving pixels. The result is obviously very sensitive to the threshold. Most of the time, the user must tune this threshold in a trial-and-error process. One difficulty arises when the background pixels are varying along the time. In this case, more elaborated approaches build a background model for each pixel based on the pixel's recent history by using, for instance a chronological average or median of the n previous frames. More recently, Mixture of Gaussian (MoG), Kernel Density Estimator (KDE), and codebook models have been proposed to cope with multiple modal background distributions. These algorithms are based on a training stage to estimate the Gaussian parameters (for MoG), to compute the probability density functions (for KDE), or to construct the codebooks. Each of these techniques can provide acceptable accuracy in specific applications: MoG are adapted to multi-modal background distributions but fail to provide sensitive detection when background has fast variations. KDE overcomes this problem but are limited to short-term videos due mostly to memory constraints. Codebooks alleviate this computation limitation by constructing a highly compressed background model but produce too wide background models when the environment is highly variable as in long-term videos.

Our approach focuses on algorithms assuming a training stage of the background model representation on background samples, i.e. a set of frames without any moving objects of interest. In particular, we focus on two algorithms of this family: the generalized Mixture of Gaussian (MoG) model [Stauffer and Grimson, 1999] and the codebook model [Kim et al., 2005]. The training stage of the Mog model consists in estimating k Gaussian parameters

set (ω, μ, Σ) for each pixel using an expectation-minimization algorithm, where k is the number of gaussians in the mixture. For the codebook model, the learning stage consists in constructing the set of codewords (i.e. a codebook) for each pixel. A codework is composed of a vector of mean RGB values and of a five-tuple vector containing intensity (brightness) minimum and maximum values, the frequency with which the codeword has occurred with its first and last access time. We have chosen these two algorithms for the differences they exhibit in their model representation and their training process, and for their comparable performances in the targeted applications.

6.2.3 Hypothesis

Our assumptions are the following:

1. We suppose that the user is able to collect a set of background samples for the training of the background models. In a practical point of view, the collection can be achieved by a manual selection of frame sequences where no object of interestis present.

2. We suppose that this set is large enough to illustrate the different variations of the scene (the contexts), e.g. the different illuminations changes that could be encountered in real-time use.

These two assumptions fit quite good with the targeted applications where videos can be acquired continuously, typically 24 hours per day and seven days per week. The quick availability of data allows to build hence huge training image set.

6.2.4 Experiment

The experimental conditions are the followings: the video data are taken during a period of 24 hours, at eight frames per second, from a video surveillance camera fixed above an outdoor cash desk of a car park. The video camera parameters are set in automatic mode. The size of the images is 352×288 pixels and are stored in JPEG format. For the experiment, we have taken one frame on five which correspond to 138000 frames in total. Six samples picked from the image set are shown in Figure 6.1. They have been chosen to illustrate the background modelling problem. In the learning stage, we have manually defined a training image set I composed of 5962 background frames (i.e. without foreground objects) along the sequence. This corresponds to pick one frame every 15 seconds in mean and represents 4.3% of the whole image set.

6.3 Context Analysis by Image Sequence Clustering

This step slightly differs from the one presented for static image segmentation for two reasons. First, the training image set is exclusively composed of background images and second, the features used to characterize the images variations are not

(a) (b)

(c) (d)

(e) (f)

Figure 6.1: Six frames representative of the background modelling problem.

the same (color histograms in static segmentation). Here, the field of view of the video camera is fixed. This configuration allows a more local analysis of the image variations. To this end, a straightforward approach, based on a global histograming of pixel intensity as in [Georis, 2006] is not fully adapted. Actually, such histograms lack spatial information, and images with different appearances can have similar histograms. To overcome this limitation, we use an histogram-based method that incorporates spatial information [Pass et al., 1997]. This approach consists in building a coherent color histogram based on pixel membership to large similarly-colored regions. For instance, an image presenting red pixels forming a single coherent region will have a color coherence histogram with a peak at the level of red color. An image with the same quantity of red pixels but widely scattered, will not have this peak. This is particularly significant for outdoor scene with changing lighting conditions due to the sun rotation, as in Figure 6.1(a,b).

Figure 6.2 gives a quick overview of the feature distribution along the sequence. In this figure, each X-Z slice is an histogram which represents the percentage of the number of pixels (Z axis) belonging to a given color coherent feature (X axis). The coherent color feature scale has been divided into 3 intervals for the three HSV channels. Histograms are ordered along the Y axis which represents the time in the course of a day. Several clusters of histograms can be easily visually discriminated as notified for cluster number 1, 14 and 2. Others clusters not represented here are intermediate ones and mainly correspond to transitions states between the three main clusters. Sixteen clusters are found (see Figure 6.3 for context class distribution). Three major clusters can be identified (number 1, 2 and 14). The order of class representation does not necessary correspond to consecutive time instants. Cluster 1 corresponds to noon (sunny context), cluster 2 correspond to the morning (lower contrast) and cluster 14 to the night.

Then, for each identified cluster, the corresponding training frames are put together and used to train a background model (i.e. codebooks). The next step is the real-time adaptive segmentation of the video using a dynamic selection of trained background models.

6.4 Real-Time Adaptive Figure-ground Segmentation

This task begins similarly to the one presented for the static segmentation task. For a new image I, a global feature vector $\mathbf{v}(I)$, here a coherent color histogram in the HSV color space, is extracted and classified as a context. We also use a temporal filtering step to reduce instability of the clustering algorithm. Indeed, in cluttered scenes, foreground objects can strongly interact with the environment (e.g. light reflections, projection of shadows) and then add a bias to the context analysis. So, it is important to smooth the analysis by ponderating the current result with respect to previous ones. Our temporal filtering criterion is defined as follows

Let us define θ the context cluster identifier (the buffered context), θ_I the cluster identifier for the incoming image I, and μ_θ the square mean of cluster

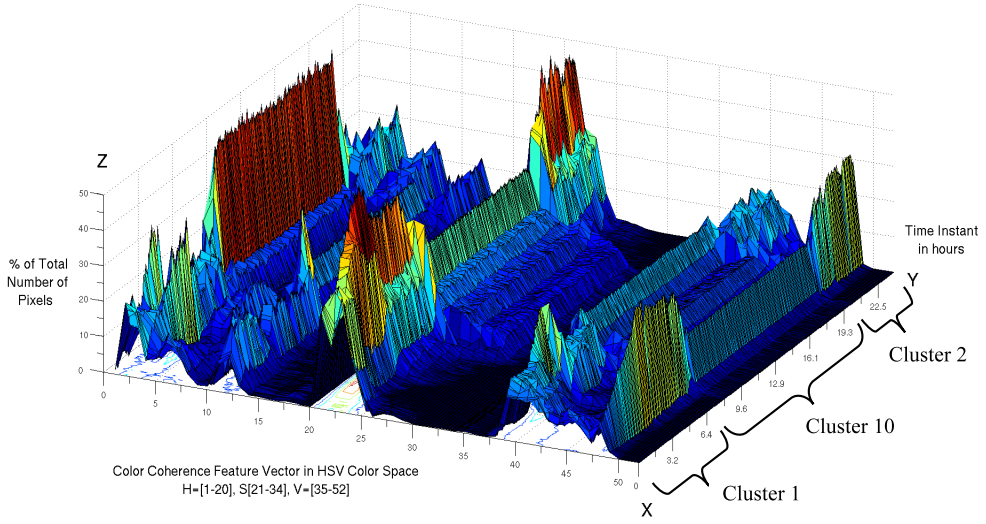


Figure 6.2: 3-D histogram of the image sequence used during the experiment (see Figure 6.1 for samples).

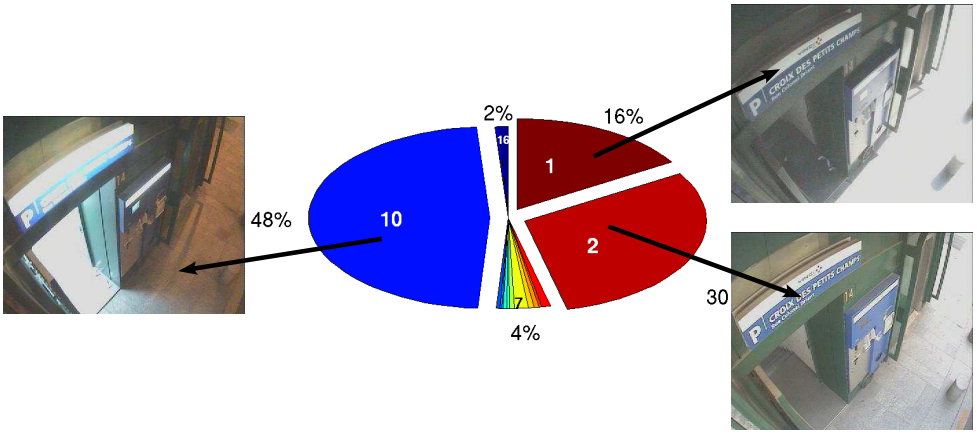


Figure 6.3: Pie chart of the context class distribution for the image sequence used for the experiments.

probability computed on a temporal window. α is a ponderating coefficient related to the width w of the temporal filtering window. To decide if θ_I is the adequate cluster for an incoming image I, we compare it with μ_θ as in Algorithm 7. In this algorithm, two cases are investigated. If θ_I is equal to θ, μ_θ is updated based on the context probability $p(\theta_I)$ and α. Else if θ_I is different to θ, the current $p(\theta_I)$ is tested against μ_θ. The square value of $p(\theta_I)$ is used to raise the sensibility of the temporal filtering to large variations of $p(\theta_I)$.

Algorithm 7: Context Temporal Filtering Algorithm

inputs : I
outputs: θ (the buffered context identifier)

```
   /*initializations for the first frame only                          */
 1 θ ← 0 ;                  /*set current context identifier to 'noise'*/
 2 μ_θ ← 0 ;                    /*set square mean of θ probability to 0*/
 3 α ← 0 ;                                  /*set weight parameter to 0*/

 4 [θ_I, p(θ_I)] ← contextAnalysis(I) ;     /*θ_I = context ident.  of I*/
 5 if θ = θ_I or θ = 0 then
 6 |   θ ← θ_I ;
 7 |   μ_θ ← (α×μ_θ + p²(θ_I)) / (α+1) ;          /*update the value of μ_θ*/
 8 |   if α < w then
 9 |   |   α ← α + 1 ;
10 else if p²(θ_I) ≥ μ_θ then
11 |   θ ← θ_I ;
12 |   μ_θ ← p²(θ_I) ;                           /*update the value of μ_θ*/
13 |   α ← 1 ;                                 /*reinitialize the weight α*/
14 return θ
```

When the context is identified, the corresponding background models are selected and the figure-ground segmentation of I is performed.

6.5 Experimental Results

In this section, we present experimental results of real-time figure-ground segmentation. Since no ground truth are available for the car park video, we are only able to present qualitative results. We compare the results obtained with different segmentation settings (with or without the context adaption, etc.) at different moments of the day and in several difficult situations.

Boundaries of the detected regions (in green) have been dilated for a better visualization. We remember that we took one frame every five seconds in our experiment. `context ID` is the identifier of the detected context and `prob` is the estimate probability of the identified context.

6.5.1 Model Selection Effect

In this section, we show some examples Figure 6.4 where the selection of the background model helps to improve the segmentation. In this figure, the left column corresponds to the codebook segmentation when trained on the whole training image set. The right column corresponds to the codebook segmentation results thank to our context adaptation method, i.e. with a dynamic selection of a background model. We can see that our approach achieves a better detection rate without adding false detection.

6.5.2 Temporal Filtering Effects

In this section, we present some situations where the temporal filtering algorithm can help to correct classification mistakes. The columns of Figure 6.5 and Figure 6.6 corresponds to the segmentation result with the codebook algorithm based on respectively one background model (left column), dynamic selection of the background model (middle column), and dynamic selection of the background plus temporal filtering (right column).

When a foreground object crosses the scene

The presence of a person modify the pixel distribution of the scene and then perturbs the context classification. Consequentely, a 'noise' context (`ID:0`) is often detected as shown in Figure 6.5. The temporal filtering algorithm smooths the context analysis by integrating the results of the previous frames, and then helps in keeping a correct context classification in such cases. We can also see on the second row that the man's shadow is not detected. In fact, context number 1 gathers frames from sunny and shaded illumination conditions of this scene part. The corresponding background model has thus integrated these values during the training.

When an unadapted context is detected

When the lighting condition suddenly changes due to incoming reflections on shiny surfaces for instance, the context classification is biased and returns an unadapted context identifier. Once more, the used of the temporal filtering is well-adapted for these -not so rare- situations as seen in Figure 6.6.

6.5.3 Borderline and Bad Results

In this section, we give some examples where the results are not the expected ones. In particular, we try to exhibit the limits of our approach at both model selection level and context filtering level.

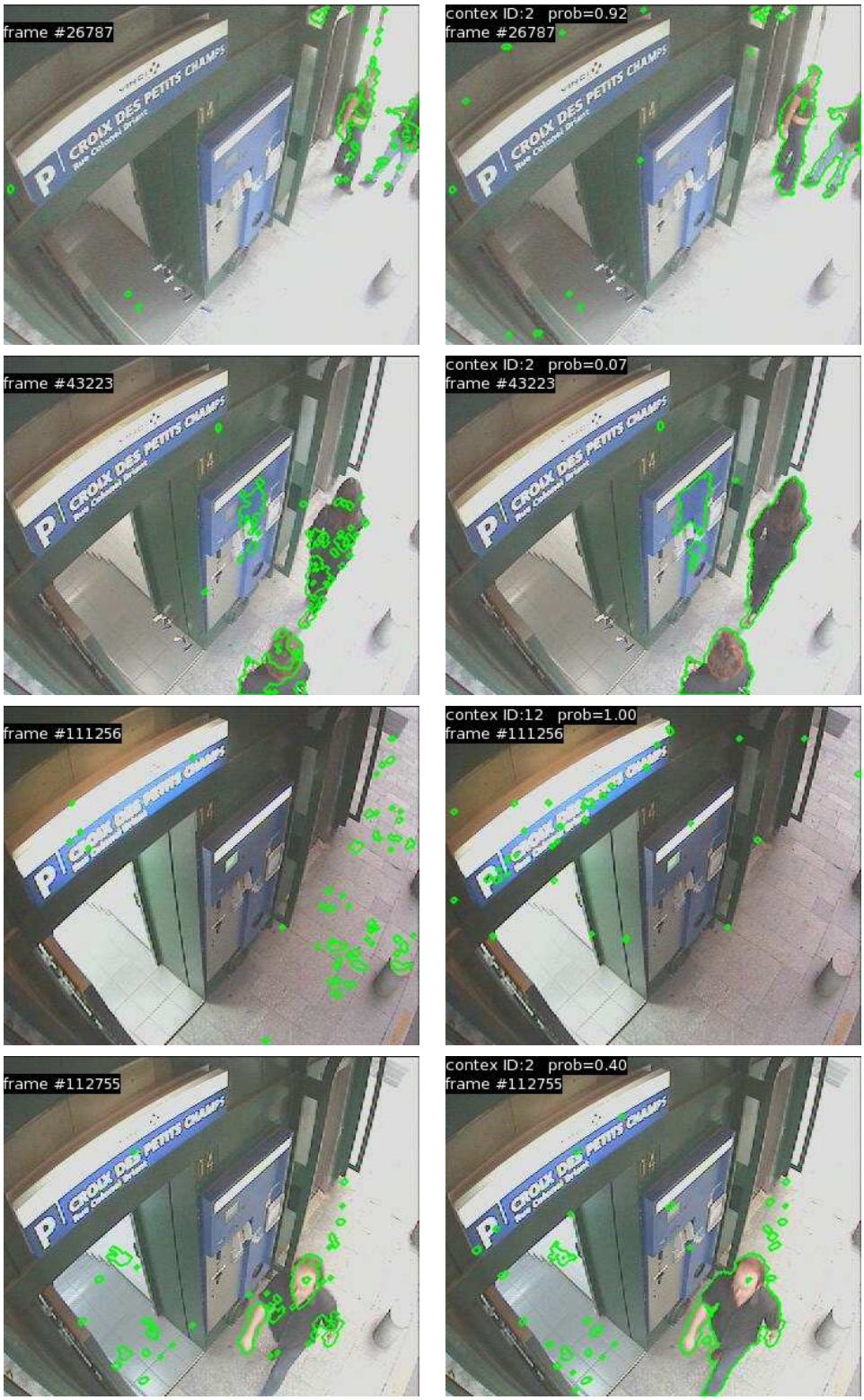


Figure 6.4: Illustration of the segmentation improvement when a dynamic selection of a background model is applied (right column).

Figure 6.5: Illustration of the temporal filtering effect on the context analysis (1). Columns are, from left to right: without context adaptation, with context adaptation, with filtered context adaptation. Rows are frame at time t and $t + 1,87s$.

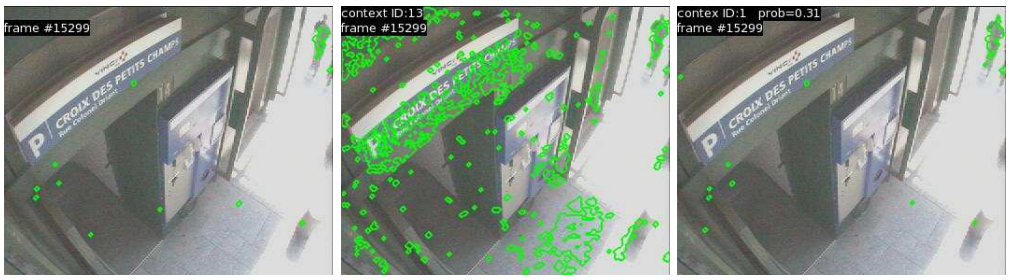


Figure 6.6: Illustration of the temporal filtering effect on the context analysis (2). Columns are, from left to right: without context adaptation, with context adaptation, with filtered context adaptation.

Shadow removal

Unlike Figure 6.5, Figure 6.7 shows a case where shadows are not correctly integrated into the background model. The context number 10 corresponds to the night and the only possible shadows are coming from people coming out the pedestrian entrance of the car park. This situation has not been learnt during the training stage of the codebook models.

Figure 6.7: Illustration of the shadow removal problem when the background model is not trained to such situations.

Noise sensitivity of poorly trained background models

The problem with intermediate contexts (i.e. representing a short period) is their brittleness to noise. Their associated background model has not been trained enough and the detection result has a greater false positive rate then wider clusters. This is the case of the context number 4 as in Figure 6.8.

Figure 6.8: Illustration of the noise sensitivity of a poorly trained background model.

Limitation in the quick adaptation to complex changes

At the end of the night, the street lighting is switched off. If the appearance of the scene is instantly modified, the video camera needs several couple of seconds to completely adapt its gain to the new illumination conditions. The modelling of this event is very difficult because it is a succession of small changes: shadows

vanish, color changes, and the contrast decreases. At the scale of 24 hours, this step is shown as a noise context, since it finally involves a short period (about half a minute). Figure 6.9 shows what is happening at the segmentation level with or without context adaptation and temporal filtering. When the street light switches off (second row), many false positive pixels are detected and the context analysis returns a noise context. The context analysis becomes correct again only three seconds later (third row). Concerning the temporal filtering of the context, the necessary time to find back a correct context adaptation is greater $(7,5s)$. This is due to the time lag added by the temporal window of the filtering.

6.5.4 Comparison with Mixture of Gaussian

In this section, we compare our approach with the MoG approach. We use an implementation of the algorithm proposed in [Stauffer and Grimson, 1999]. We use the default parameter setting. A MoG background model is trained for each identified cluster then dynamically selected during the real-time segmentation.

Figure 6.10 shows that MoG are more sensitive to shadows than codebooks.

Figure 6.11 shows the high sensitivity of Mog to global changes (first row) and the effects of a too large learning rate: foreground pixels from the first row still remains on second row. We also see that the same false detection problem occurs with Mog when models are not enough trained (third row). The last row shows the difficulty of the model to integrate noisy pixel value induces by the high gain level of the video camera.

Figure 6.12 shows the same frames than the ones in Figure 6.9. We can see that the MoG model encounters the same problem than the codebook and fails to quickly adapt to the background variations.

6.6 Conclusion

In this chapter, we have presented the validation of our adaptive segmentation approach for video surveillance applications. We have focused on a difficult long-term video surveillance application (outdoor car park entrance surveillance) where both gradual and sudden changes occur. In this application, a huge amount of data are easily available since images can be acquired continuously. In a weakly supervised learning stage, the user's task is to collect background samples illustrating the different situations. The unsupervised clustering algorithm has successfully identified meaningful clusters of training images like sunny context, night context, or dawn context. For each identified image cluster, a background model has been trained using the codebook model [Kim et al., 2005]. This approach, consisting in generating sub-goals and training learning-based algorithms on each sub-goal is similar to a meta-learning approach. In real-time figure-ground segmentation, the different contexts are successfully retrieved thanks to the temporal filtering algorithm. However, some problems remain in the context adaptation especially when unforeseen changes occur.

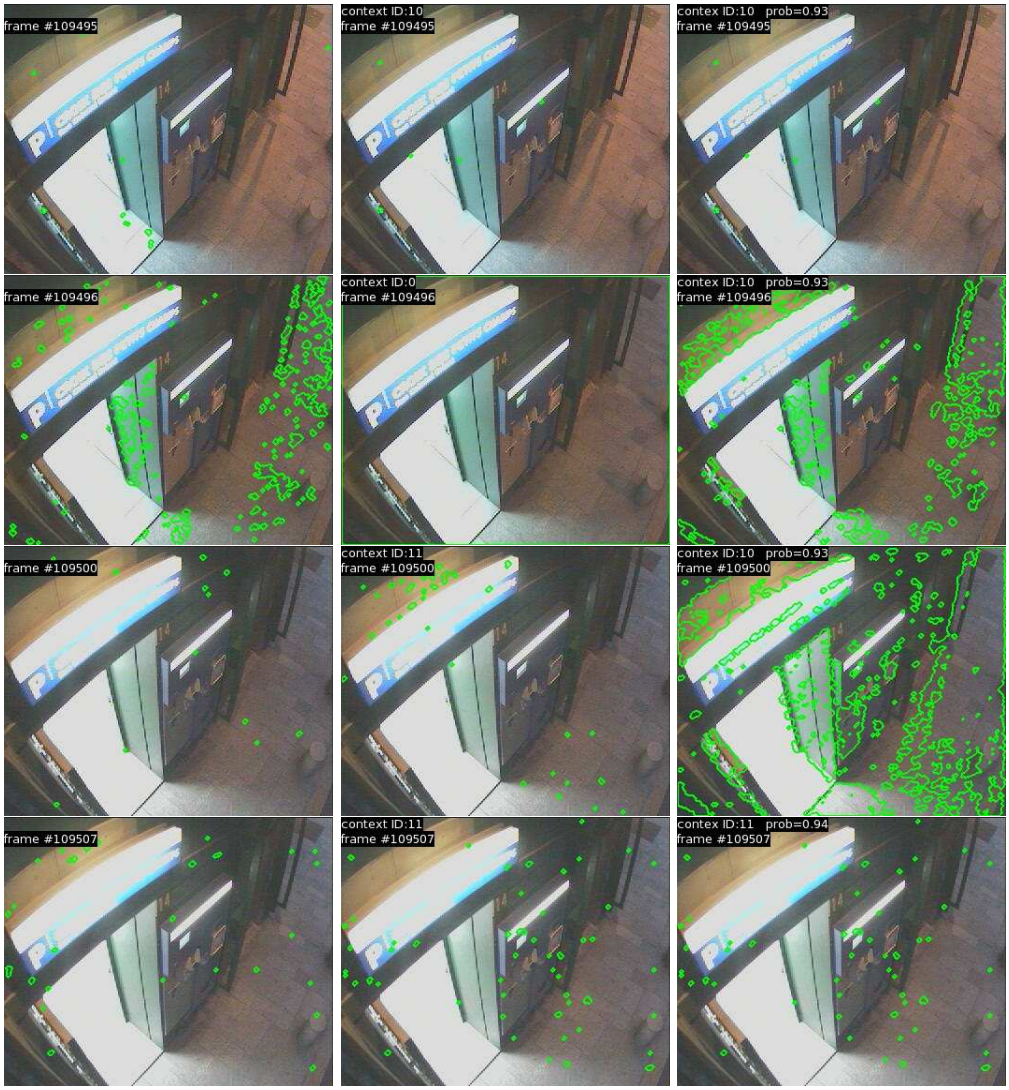


Figure 6.9: Illustration of the limitation to quick adaptation of the context adaptation and temporal filtering. Columns are, from left to right: without context adaptation, with context adaptation, with filtered context adaptation. Rows are frame at time t, $t+0.62s$, $t+3.12s$, and $t+7.5s$.

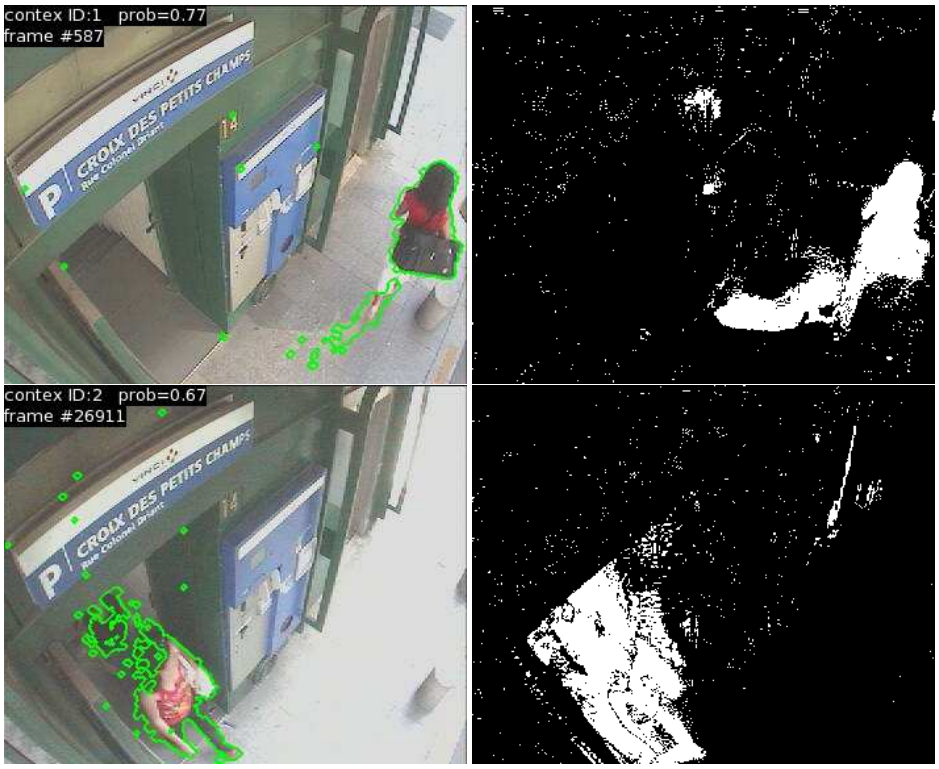


Figure 6.10: Comparison between the proposed approach (left column) with the codebook model [Kim et al., 2005] and the MoG model [Stauffer and Grimson, 1999] (right column) (1).

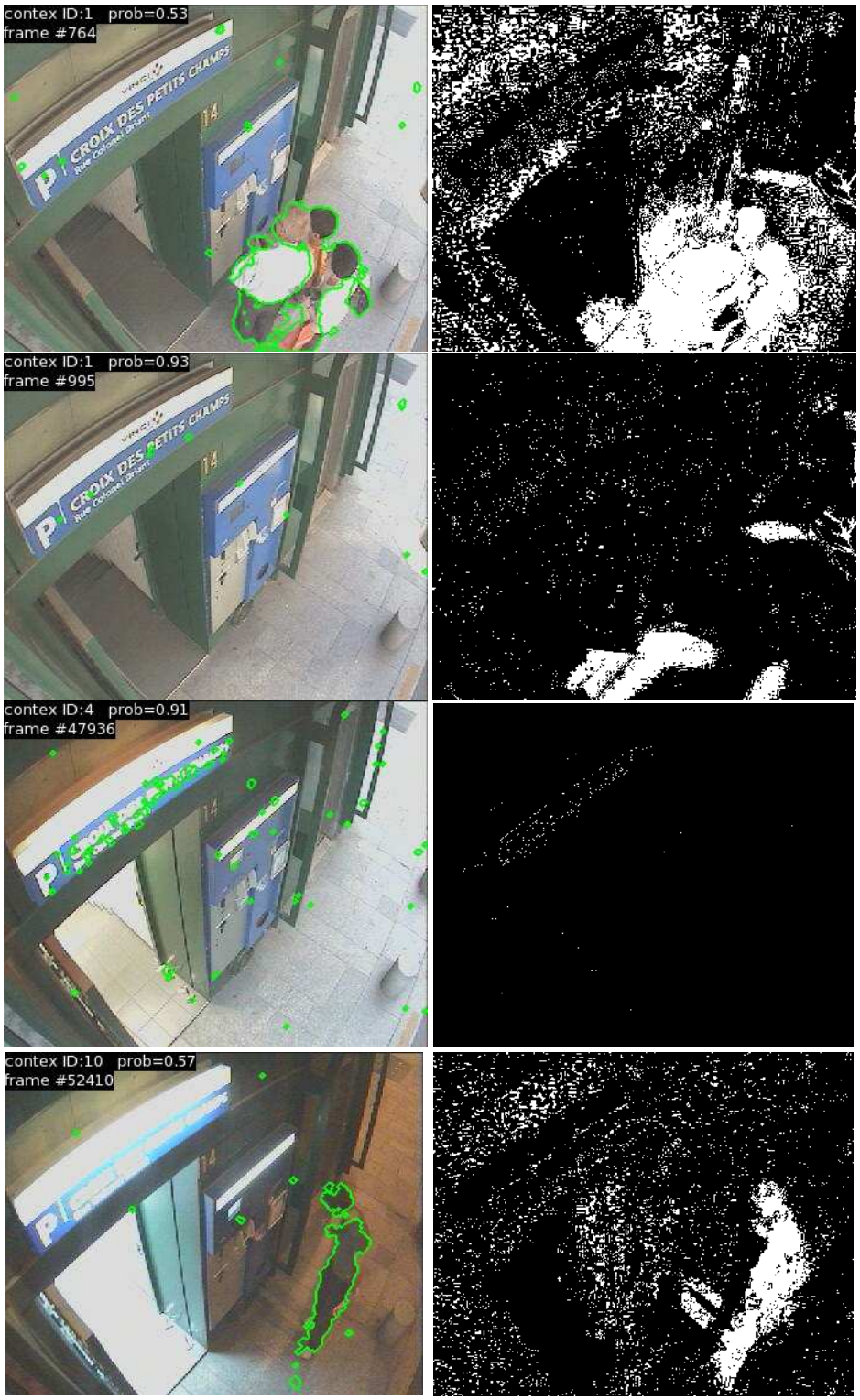


Figure 6.11: Comparison between the proposed approach (left column) with the codebook model [Kim et al., 2005] and the MoG model [Stauffer and Grimson, 1999] (right column) (2).

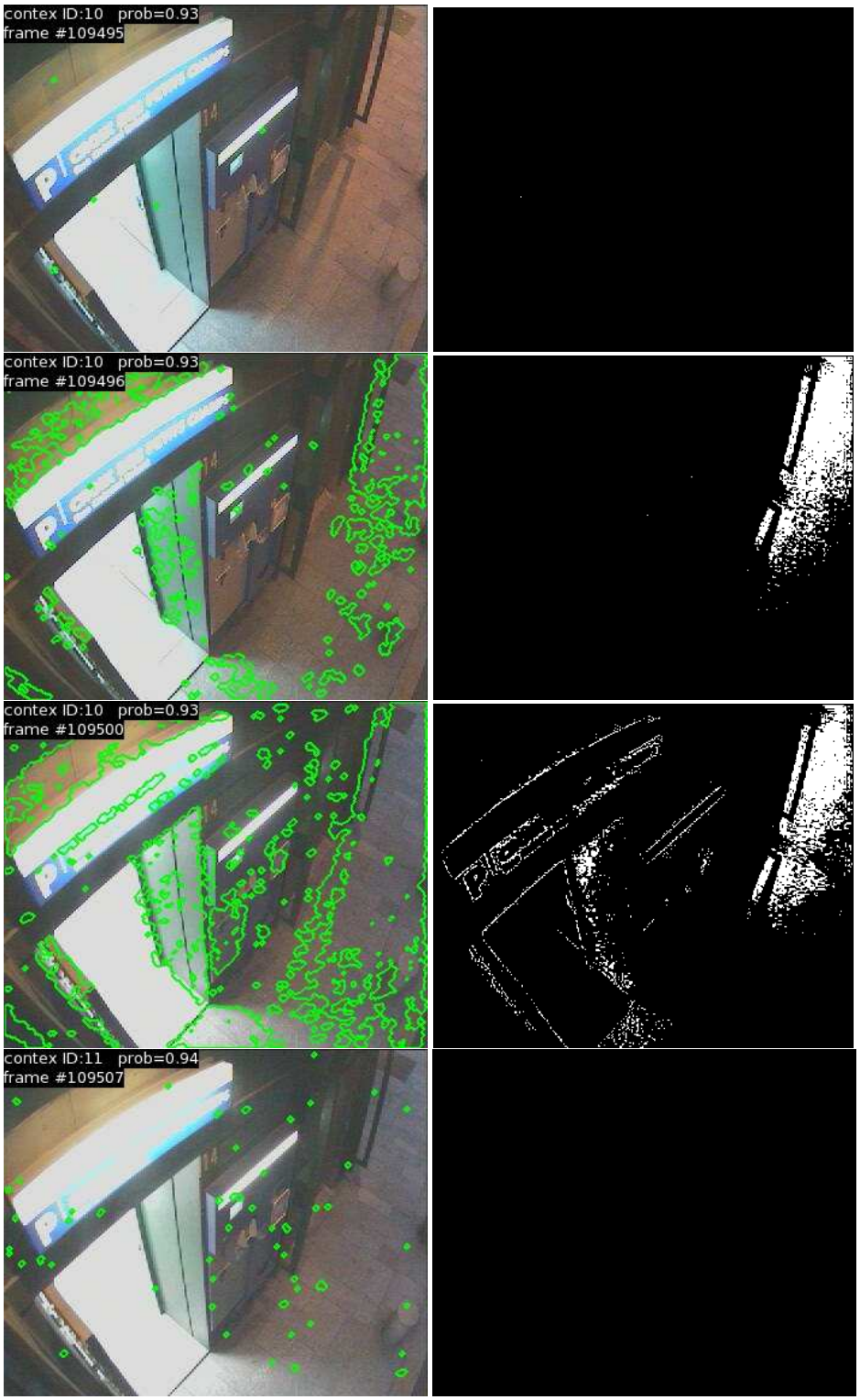


Figure 6.12: Comparison between the proposed approach (left column) with the codebook model [Kim et al., 2005] and the MoG model [Stauffer and Grimson, 1999] (right column) on the sequence of Figure 6.9.

In this problem, our context analysis uses pixel values in the HSV color space. The transformation of the pixel values into this color space is costly and then degrades the frame-rate in real-time segmentation. Nevertheless, the motivation in using this color space to discriminate image context is based on the fact that illumination changes are more visible in the HSV space than in the RGB space. Indeed, the saturation channel is very sensitive to changes induced by shadows or sun illumination.

If the temporal filtering of the context has the expected smoothing effects when spurious contexts are detected, the algorithm mainly relies on the α parameter setting. A too large value will add a time lag in the context adaptation whereas a too small value will make the algorithm too sensitive to spurious context detection. So, depending on the application needs and the frame rate, a trade-off value might be set.

The codebook model has shown to be well-adapted to deal with this experiment. Comparisons with the MoG model reveal its robustness in different situations as quick illuminations changes variations or shadows removal. Nevertheless, a quantitative evaluation study remains to be done to objectively assess our approach against other algorithms.

Chapter 7

Conclusion and Perspectives

In this thesis, I address the problem of image and video segmentation with a cognitive vision approach. More precisely, I study two major issues of the segmentation task in vision systems: selection of an algorithm and tuning of its free parameters, according to the image content and to the application needs. Most of the time, this tedious and time-consumming task is achieved by an expert in image processing using a manual trial-and-error process. Recently, some attempts at automating the extraction of optimal parameters of segmentation have been made but they are still too application-dependent. The re-usability of such methods is still an open problem. We have chosen to handle this issue with a cognitive vision approach. Cognitive vision is a recent research field which proposes to enrich computer vision systems with *cognitive* capabilities, e.g., to reason from *a priori* knowledge, to learn from perceptual information, or to adapt its strategy to different problems.

In this thesis, I propose a supervised learning-based methodology for off-line configuration and on-line adaptation of the segmentation task in vision systems. The off-line configuration stage requires minimal knowledge to learn the optimal selection and tuning of segmentation algorithms. In an on-line stage, the learned segmentation knowledge is used to perform an adaptive segmentation of images or videos. This cognitive vision approach to segmentation is thus a contribution for the research in cognitive vision. Indeed, it enables robustness, adaptation, and re-usability faculties to be fullfiled.

The proposed approach has been implemented and validated on two types of real-world applications: adaptive static image segmentation in a biological application and figure-ground segmentation in a video surveillance application.

The first part of this chapter reviews my approach and discusses its contributions and its limitations. The second part presents perspectives to improve the method, in particular concerning the learning topic.

7.1 Review of the Proposed Approach and Contributions

7.1.1 A Generic Optimization Procedure

Our optimization procedure automatically extracts the optimal parameters of segmentation algorithms based on a quantitative evaluation of the segmented image quality w.r.t. manual segmentations. The method is independent of the application and of the segmentation algorithms. Only the free parameters to tune with their range values are required. This kind of knowledge is usually provided by the algorithms' authors. The criterion used to evaluate the segmentation quality makes no assumptions on the application nor on the algorithm behaviors. It has been applied to assess segmentation tasks in two applications (a biological application and a video surveillance one). It has also been applied to the Berkeley public segmentation database [Fowlkes and Martin, 2007]. Two free-derivative optimization algorithms (a direct search method and a genetic algorithm) have been successfully used to minimize the criteria. In this field, my contribution is a comparative study of the two optimization algorithm performances. Thanks to this study, we have identified two situations: when the number of parameters is up to two, the Simplex provides good results in a minimal number of iterations. When the number of parameters is greater than two, the genetic algorithm should be preferred.

The main difficulty of this supervised learning approach is the manual segmentation of images. This task is tedious, subjective, and time-consumming. User-friendly annotation tools should be used to alleviate users' efforts. The strength of this approach is also dependent on the intrinsic performance of the segmentation algorithms. As a consequence, this approach supposes that at least one algorithm is able to perform good results for the target segmentation purpose.

7.1.2 A Strategy for the Algorithm Selection

After that several segmentation algorithms have been optimized on a training image set by using the proposed generic optimization procedure, the next issue is algorithm selection. The goal of this step is to answer to the user's question: "which algorithm is the best adapted to segment my image?". The first part of my strategy consists in identifying different situations in the training image set. A situation, also called a *context*, is represented by a sub-set of images sharing the same global characteristics, such as color distributions. First, an unsupervised clustering algorithm is used to identify these contexts. The second part uses the results of the previous optimization stage to perform a local ranking of the optimized algorithms for each context according to their performance values.

This strategy allows a dynamic control of the segmentation task (i.e. algorithm selection plus optimal parameter setting) without the need of explicit *a priori* knowledge of the application domain or the segmentation algorithms themselves.

It should be noted that this strategy makes several assumptions. First, it sup-

poses that all possible contexts are illustrated in the training image set. Second, this strategy argues that for each identified context, a mean parameter set of the best ranked algorithm exists to deliver good segmentation results.

7.1.3 A Semantic Approach to Image Segmentation

Most of the time, segmentation results provided by bottom-up algorithms are semantically meaningless. I propose a semantic approach to image segmentation where high level region labels help to validate region segmentation results. The region labelling algorithm relies on three steps and makes use of the results of the previous stages (parameter optimization and algorithm selection). In a first step, for each training image, the user is invited to affect semantic labels to regions of manual segmentations according to the application needs. Then, an automatic region label matching is achieved between the regions of the manual segmentation and the regions of the optimized segmentation. Finally, a set of classifiers (SVMs) are trained for each label based on numerical features of regions. The originality of the approach is that each step of the learning process, i.e. feature extraction and SVM training, is optimized in a wrapper scheme so as to maximize the classification performance of the algorithm.

Currently, region features are limited to color and texture information. The method could be improved by also taking into account spatial information, such as the relationships between the different semantic classes of regions.

7.1.4 A Software Implementation of the Methodology

A software implementing this methodology for off-line configuration and on-line adaptation of the segmentation is proposed. Starting from a training image set with the corresponding manual segmentations, the system, via a graphical user interface, is able to:

- extract optimal parameters for six segmentation algorithms (four for static image segmentation and two for video segmentation),
- perform the image cluster decomposition,
- select the best performing algorithm for each identified context,
- annotate the regions with respect to predefined class labels,
- train region classifiers,
- control the segmentation of new images with respect to the learned segmentation knowledge,
- visualize segmentation results.

More development on the implementation in C++ code is given in appendix B.

Finally, by addressing the problem of adaptive image segmentation, we have also addressed underlying problems, such as feature extraction and selection, and segmentation evaluation and mapping between low-level and high-level knowledge. Each of these well-known challenging problems is not easily tractable and still

demands to be intensively considered. We have designed our approach (and our software) to be modular and upgradable so as to take advantage of new progresses in these topics.

7.1.5 Contributions for the Cognitive Vision Platform

My approach has enriched the platform by enabling learning faculties at the segmentation level. Previously, segmentation algorithms were manually tuned by an expert in image processing and the dynamic selection relied on a knowledge base written by hand. The same algorithms are now automatically tuned and thus allow an adaptive segmentation of different images, thanks to a training stage. The gain obtained at the segmentation level benefits to the higher level modules of the platform.

7.1.6 Contributions for the Biological Application

Despite the appearance, robust segmentation of mature white flies on rose leaves is not a trivial task. The variability of leaves color and texture combined with the semi-transparent nature of the white fly wings and the presence of number of other objects (e.g., white fly eggs, larvae, chemical treatments traces, water drop, etc.) makes the segmentation not so easy. Compared to an *ad hoc* segmentation tuned by hand, our adaptive segmentation achieves better results and thus leads to a better counting precision. Moreover, the semantic segmentation drastically reduces the number of regions by merging the subparts of objects. This technique decreases the computational cost of the system since less regions have to be processed at the interpretation level.

At present, the platform is able to manage the detection and the counting of only one biological object. Other bioagressors (e.g., greenfly, aphids, etc.) or other stages of development of white flies (e.g., larvae, eggs) should be treated in order to assess both our adaptive segmentation approach and the platform to a multi-class problem (more than two). To this end, we need to acquire new data (i.e. images with manual segmentations and region annotations) as well as high level knowledge (i.e. descriptions of the objects in terms of visual concepts) for the correct descriptions of the new objects.

Finally, the platform is currently also limited by its acquisition system (a flatbed scanner). We plan to overcome such a limitation by using video cameras. Another advantage of video cameras is that they provide temporal information which is of great interest to disambiguate occlusion situations, for instance.

7.1.7 Contributions for Video Surveillance Applications

In this application, my main contribution is the dynamic background model selection based on context analysis. This approach fits particularly well to applications where both short-term and long-term illumination changes may occur. The unsupervised clustering algorithm uses image global characteristics integrating spatial

information so as to take into account not only global changes but also local ones. We have also proposed an algorithm for temporal context filtering.

The first experiments have proved that the dynamic selection of background models is a good approach to deal with adaptation facilities. Nevertheless, it is clear that our approach is still unable to manage unforeseen situations, i.e. new contexts. An extension of this approach to enable continuous learning facility is thus actively needed.

Finally, in this application, we do not completely follow up our strategy for algorithm selection. It should be interesting to see how different figure-ground segmentation approaches could cooperate together.

7.2 Future Work

7.2.1 Short-Term Perspectives

Incremental learning for unforeseen situations

The brittleness of our approach to unknown situations is currently its major drawback. This concerns the context analysis level as well as the segmentation level. The concerned algorithms are the DBSCAN algorithm for image-content clustering and the SVMs for the semantic segmentation. Currently, neither the clustering algorithm nor the SVMS are able to adapt dynamically to new training data: the learning process must be run again on the whole training data set. The use of incremental machine learning techniques should be useful to fulfill the property of continuous learning. The main idea of incremental learning for unforeseen situations is to dynamically adapt the clustering/classification method w.r.t. to the classification error of new input data. In our problem, unexpected situations can be identified thanks to the estimates of the context probability and the estimates of the SVM classification probabilities. For instance, in video surveillance, when the 'noise context' is detected during several frames, an alarm could raise and the concerned frames could be considered as new training images. In a supervised process, a user could be asked to validate the new training images by checking whether each frame is a background frame or not. In an unsupervised process, the validation of the new training images could be based on a spatial analysis of the detected moving pixels. Usually, when a 'noise context' is detected, many meaningless moving pixels are detected all over the image. The use of an adaptive classification algorithm using robust incremental clustering as proposed in [Prehn and Sommer, 2006] will then allow to dynamically update the cluster and create new ones if necessary.

Meta-Evaluation of Image Segmentation

The assessment function for the evaluation of segmentation results we have proposed in chapter 4 is based on two fundamental criteria (counting of miss- and over-detected boundary pixels). It makes the metric re-usable for a large set of

segmentation tasks. Nevertheless, the way of weighting each criteria is a key-point element of the metric. For instance, in some applications, it should be better to give more weight to the miss-detection rate than to the over-detection rate. It should be also more adequate to use or to combine n different base evaluators ρ, e.g., boundary-based and region-based evaluators, depending on the application needs such as: $E_I^A = \alpha_1\rho_1 + \alpha_2\rho_2 + \ldots + \alpha_n\rho_n$. We believe that in general it is difficult to specify and exact form of the fitness function since this requires defining exact trade-offs to be done between the different measures. To this end, meta-heuristic for the weighting of functions as the Pareto front could be investigated as in [Everingham et al., 2002]. For each segmentation algorithm and for all training images, the parameter space is explored and gives a fitness function graph for each base evaluator and for each training image. The goal is then to estimate the combination of the different base evaluators which gives globally the best performance scores (i.e. the Pareto front). To this end, a global optimization algorithm, as a genetic algorithm is used to find the optimal configuration. Another possibility is to use machine learning meta-algorithms as in [Zhang et al., 2006]. The idea is to train a learning algorithm (e.g. a decision tree) that determines how to coalesce the results from the different base evaluators applied on the training image set. The main advantage is to obtain a tuned evaluation metric for the type of images upon which it is trained.

Local Tuning of the Parameters of Video Segmentation Algorithms

The goal of video segmentation algorithms as mixture of Gaussian, kernel density estimators, and codebooks is to learn the possible range values of background models for each pixel. During the learning process, some thresholds are set to define the bounds of the models. Usually, the values of these sensitive parameters are the same for all the pixels. In the case of video surveillance applications with a fixed video camera, the parameters should be optimized for each pixel. For instance, the detection thresholds for pixels in a zone where a moving object never passes through should be set to produce a low false detection rate. In the contrary, the detection thresholds for the pixels of zone(s) of interest should be set to produce sensitive models. This is examplified in Figure B.1 where z_1 is the zone where objects of interest (people) never comes and z_2 where they are expected to be visible. The detection thresholds for each pixel in z_1 should be set to a lower value to the ones for z_2.

Spatio-Temporal Video Segmentation

The major problem of pixel-based approaches for video segmentation is that no spatial coherency is taken into account. To overcome this limitation, a solution is to compute in parallel a region-based image segmentation. The objective is to refine the segmentation obtained with a pixel-based motion segmentation. This technique is illustrated in Figure 7.2. An input image (a) is segmented using a region-based segmentation algorithm. The result is presented in (b). In paral-

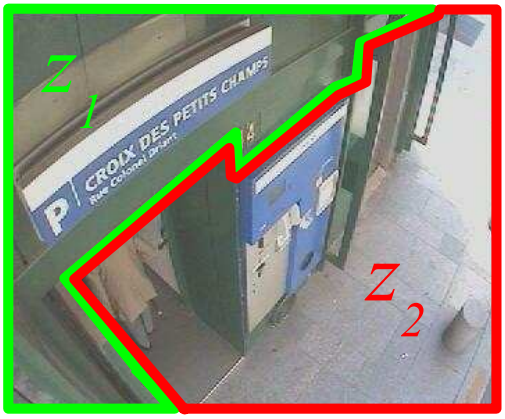


Figure 7.1: Example of a local tuning based on *a priori* knowledge of the scene. The tuning of the detection thresholds for pixels in z_1 should be less sensitive to variations than for z_2.

lel, a figure-ground segmentation (c) is computed using the codebook model for instance. The final result (d) is a combination of the two segmentations with respect to a majority overlap criteria. In this example, the region-based segmen-

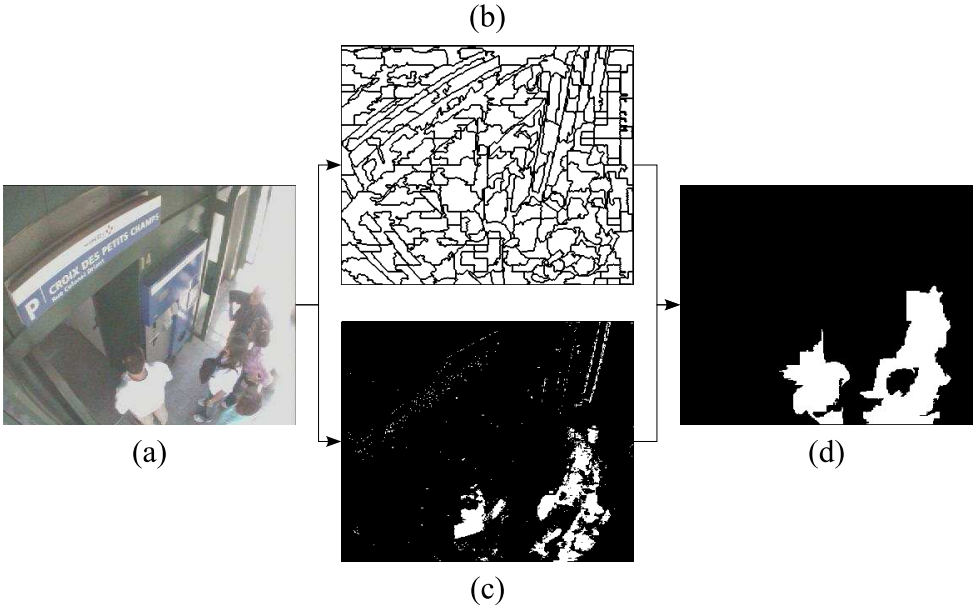


Figure 7.2: Illustration of a spatio-temporal segmentation (d) combining the results of a background subtraction algorithm (c) with a region-based algorithm (b).

tation algorithm has been manually tuned to produce an over-segmentation.

7.2.2 Long-Term Perspectives

Use of a Visual Concept Ontology for Semantic Segmentation

Currently, the semantic segmentation is based on numerical features describing independently each region. Complex objects like a person have several meaningful subparts (e.g., head, legs) which cannot be described with the same low-level features as color or texture. Moreover, some of these subparts can have an infinite space of color variations, depending on the clothes for instance. These subparts, belonging to the same semantic object, can yet be linked together by spatial relations and hierarchical decompositions. Hence, several different visual concepts should be used to achieve a semantic segmentation even if the object is difficult to model. To this end, we could take advantage of the visual concept ontology proposed by Maillot et al. in [Maillot and Thonnat, 2008] by mixing, in a structured way, *a priori* knowledge of different concepts (e.g., color, texture, geometric, and spatial relation features). The goal should be to assess the membership of each segmented region to a semantic object according to trained visual concept detectors.

Use of Shared Visual Feature

For very difficult cases where intra-class information (i.e. object appearance) is very heterogeneous and/or inter-class information is poorly discriminative, the selection of representative features is tricky and leads to poor performances. In this case, approaches based on shared visual features [Torralba and Murphy, 2007] across the classes as boosted decision stumps should be more appropriated and effective. Boosted decision stumps reduce the computational and the sample complexity by finding common features that can be shared across the classes. The detectors for each class are trained jointly, rather than independently. This approach is then particularly efficient for multi-class problems with few training examples.

Video Segmentation Benchmarking

Databases of videos for vision systems benchmarking exist but rarely refers to the detection level. Most of the ground truth data consists in bounding box surrounding the moving object(s). The building of bounding box relies on merging blobs and thus requires some *a priori* knowledge of the object to detect. Moroever, common metrics for segmentation performance evaluation are based on true and false detection rates and/or boundary pixel accuracy. Hence, bounding box are definitively not suited to evaluate detection level results such as region-based segmentations. The best solution consists in drawing, for example, the silhouette of a person in each frame of a video sequence. Obviously, this task requires a huge effort and cannot be achieved by only one user since videos are usually composed of several thousand of frames. This problem has yet been tackled for static image databases as the Berkeley Segmentation

Dataset and Benchmark [Fowlkes and Martin, 2007] (6000 hand-labeled segmentations of 500 Corel dataset images from 30 human subjects) and the MIT LabelMe database [Russell et al., 2005] (more than 41300 annotated images). The strengths of these databases are their open access to the scientific community and the tools they provide to facilitate the manual segmentations and annotations of images. We believe that such a strategy should be extended to annotate the content of video sequences.

Appendix A

Publications of the Author

- **Book Chapter:**

  [1] Martin, V. and Thonnat, M., A Learning Approach for Adaptive Image Segmentation. in *Scene Reconstruction, Pose Estimation and Tracking*, edited by Rustam Stolkin, chap. 23, pp. 431-454, June, 2007, I-Tech Publication

- **International Journal with Peer-review:**

  [2] Boissard, P. and Martin, V. and Moisan, S., A Cognitive Vision Approach to Early Pest Detection in Greenhouse Crops. *International Journal of Computer and Electronics in Agriculture*, ed. Elsevier, 2007, in Press

- **International Conferences with Peer-review:**

  [3] Martin, V. and Maillot, N. and Thonnat, M., A Learning Approach for Adaptive Image Segmentation. In Proceedings of *the IEEE International Conference on Computer Vision Systems (ICVS'06)*, pp. 40-48, New York, NY, January, 2006, IEEE Computer Society

  [4] Martin, V. and Thonnat, M., A Cognitive Vision Approach to Image Segmentation. In Proceedings of *the 19th IEEE International Conference on Tools for Artificial Intelligence (ICTAI'07)*, Vol. 1, pp. 480-487, Patras, Greece, October, 2007, IEEE Computer Society

  [5] Martin, V. and Thonnat, M., Learning Contextual Variations for Video Segmentation. Submitted to *the IEEE International Conference on Computer Vision Systems (ICVS'08)*, Santorini, Greece, May, 2008, IEEE Computer Society

Appendix B

Implementation

B.1 A Library for Adaptive Image and Video Segmentation

The goal of this appendix is not to provide a fully documented class description of our implementation but rather to describe the main components and to give pointers to the different libraries which have been used.

B.1.1 Main Class Descriptions

The architecture of our library (see Figure B.1) relies on the LTI-Lib (`http://ltilib.sourceforge.net/doc/html/index.shtml`) which is an object oriented library written in C++ with algorithms and data structures frequently used in image processing and computer vision

Environment	Linux Fedora Core 5 (kernel v2.6.18)
Compiler	g++ v3.4.6
Graphical library	QT v3
Hardware system	Intel Xeon bi-processor double core at 2.33GHz with 4 Go of RAM

Table B.1: Configuration set up for the implementation and the tests.

B.1.1.1 Segmentation Algorithms

The implementations of EGBIS, SRM, CSC, JSEG, SMG, KDE and MoG are provided from their authors. Some of the algorithms (hysteresis thresholding, CWAGM, region growing, meanshift, edge segmentation) are implemented in LTI-Lib. The Codebook Model algorithm has been re-implemented.

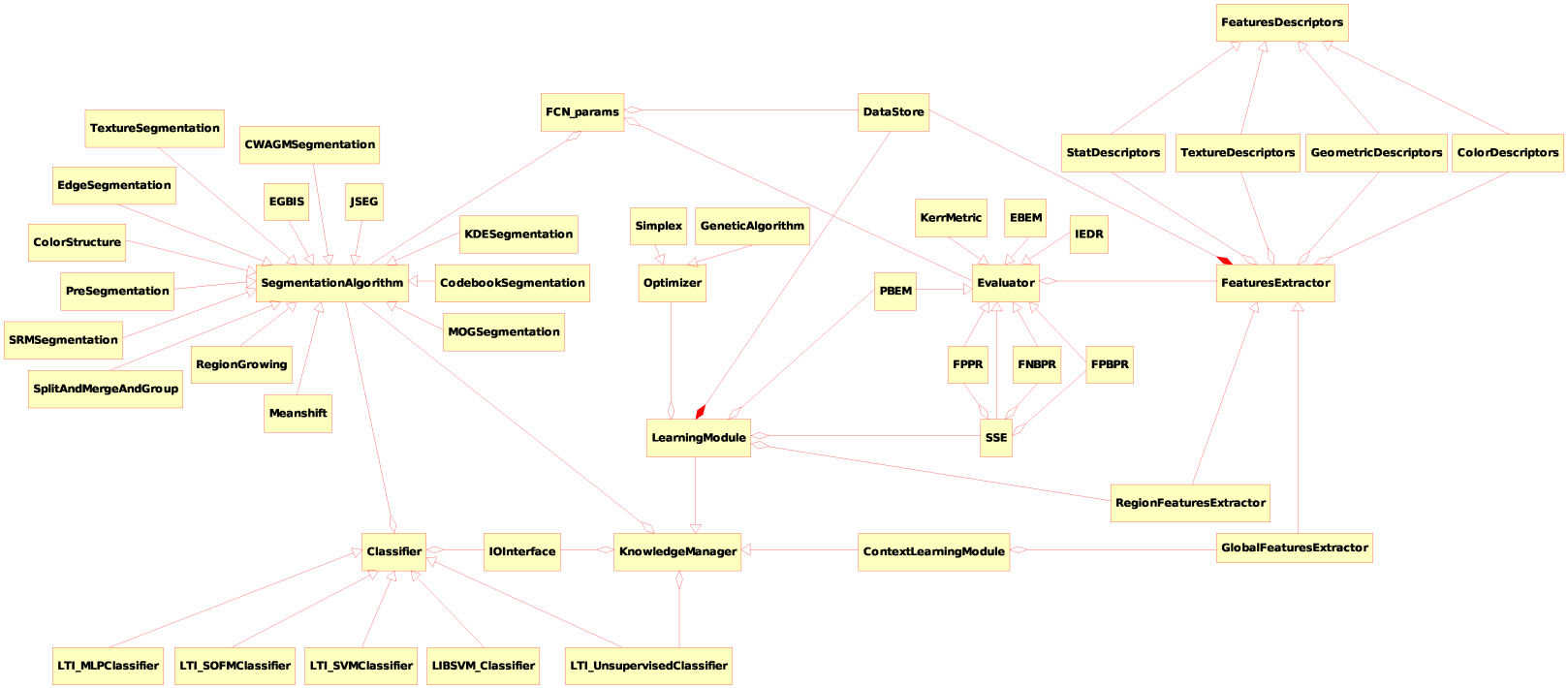


Figure B.1: Simplified UML diagram of the developped segmentation library.

B.1.1.2 Learning Algorithms

We have used LIBSVM (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`), a LIBrary for Support Vector Machines developed by Chih-Chung Chang and Chih-Jen Lin. We also used some classifiers implemented in LTI-Lib as the DBScan algorithm for unsupervised clustering.

B.1.1.3 Optimization Algorithms

We have used the C++ Direct Searches library developped by Liz Dolan et al. (`http://www.cs.wm.edu/~va/software/DirectSearch/direct_code/`).

For the genetic algorithm, we have used GALib (`http://lancet.mit.edu/ga/`), a C++ Library of Genetic Algorithm Components developped by Matthew Wall.

B.1.1.4 Data manipulation

Pixel data (e.g. segmentation results) are stored into *DataStore* objects. We mainly used the LTI-Lib matrix format to manipulate pixel data. Other data (e.g. segmentation knowledge) are stored into binary files for faster I/O manipulations.

B.2 A Graphical Tool for Adaptive Image and Video Segmentation

The developped graphical tool has the following functionnalities, based on our library for adaptive segmentation:

- display an image or a sequence of image, with ground truth if available
- label regions of ground truth with different colors (one color per class),
- select a segmentation algorithm and change the parameter setting
- display the result of the segmentation in different manners (e.g., region boundaries, colored regions, and so on.)
- optimize the parameterization of a segmentation algorithm w.r.t. a manual segmentation,
- automate some actions for image sequences as segmentation, parameter optimization, feature extraction, region labelling,
- train classifiers as SVM and DBScan,
- store and use segmentation knowledge which has been extracted,
- I/O actions (e.g., loading, saving) on images and segmentation knowledge.

The tool relies on a parameter file which gather all the information concerning data paths and settings for the main algoritms (e.g., choice of the optimization algorithm, features to extract, etc.).

We invite the reader to contact us if interested in using this implementation.

Appendix C

French Introduction

C.1 Motivations

Cette thèse traite de la segmentation d'images dans les systèmes de vision. La segmentation d'image consiste à grouper des pixels partageant des caractéristiques communes. Dans les systèmes de vision, la couche de segmentation précède habituellement l'analyse sémantique d'une image. Ainsi, afin d'être utile pour les tâches de haut-niveau, la segmentation doit être adaptée au but, c'est-à-dire être capable de segmenter efficacement les objets d'intérêt. Le tout premier problème est qu'une méthode générale et unique n'existe pas : en fonction de l'application, les performances de l'algorithme de segmentation varient. Ceci est illustré dans la Figure C.1 où deux algorithmes différents sont appliqués sur la même image. Le premier semble être visuellement plus efficace pour séparer la coccinelle de la feuille. Le second produit trop de régions faiblement significatives.

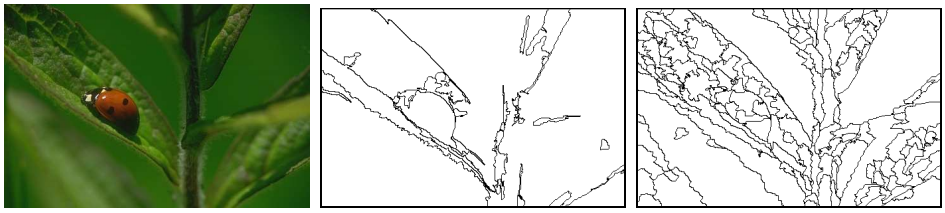


Figure C.1: Un exemple de la segmentation d'une image avec deux algorithmes différents. Le premier algorithme construit les régions en fonction d'un critère couleur multi-échelle alors que le second utilise un critère local d'homogénéité couleur.

De manière générale, il existe deux approches populaires pour configurer la tâche de segmentation dans un système de vision. La première approche est de développer un nouvel algorithme de segmentation dédié à l'application. Une seconde approche est de choisir de manière empirique un algorithme existant, par exemple dans un processus d'essai-erreur. La première approche conduit à développer un algorithme *ad hoc* à partir de rien et pour chaque nouvelle application. La deuxième approche ne garantie pas des résultats adaptés et la robustesse.

Ainsi, un besoin existe pour le développement d'une nouvelle approche du problème de la **sélection d'algorithme**. Face à différents algorithmes, cette approche doit être capable de choisir automatiquement le plus adapté à un but donné de segmentation.

Lors de l'élaboration d'un algorithme de segmentation, des paramètres internes (par exemple des seuils de tolérance couleur ou des tailles minimales de région) sont réglés avec des valeurs par défaut fournies par les auteurs de l'algorithme. En pratique, il revient souvent à l'expert en traitement d'images de superviser le réglage de ces paramètres libres afin d'obtenir des résultats cohérents. Comme il est montré en Figure C.2, il n'est pas évident de choisir le bon jeu de paramètres au regard des images segmentées : la première est assez bien segmentée mais de nombreuses parties de l'insecte sont manquantes; la seconde est Ãl'galement correcte avec une bonne délimitation de l'insecte bien que trop de régions insignifiantes soient présentes. Cependant, les interactions complexes entre les paramètres libres rendent le comportement de l'algorithme presque impossible à prédire. De plus, cette tâche délicate est fastidieuse et longue pour l'utilisateur. De ce fait, le **réglage des paramètres** des algorithmes est un réel défi. Pour résoudre ce problème, des méthodes d'optimisation doivent être examinées dans le but d'extraire automatiquement les valeurs de paramètres optimales.

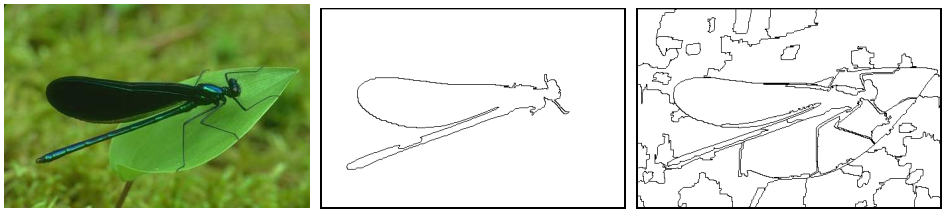


Figure C.2: Illustration du problème de réglage des paramètres. Une image est segmentée avec un même algorithme (basé sur un critère d'homogénéité couleur) réglé avec deux jeux de paramètres différents.

Dans les applications du monde réel, l'apparence des images change lorsque le contexte change. Ceci est particulièrement vrai pour les applications vidéo où les conditions d'éclairage sont continuellement en train de varier. Cela peut être dû à des changements locaux (projections d'ombres, réflections) et/ou des changements globaux de l'illumination due aux conditions météorologiques, comme illustré dans la Figure C.3 où les images sont extraites de la même scène à différents moments de la journée. Les conséquences au niveau de la segmentation peuvent être dramatiques. Ce problème de l'**adaptation au contexte** souligne le besoin d'automatismes pour l'adaptation.

C.2 Objectifs

Mon objectif est de proposer une approche de la segmentation d'image dans le cadre de la vision cognitive. Plus précisément, nous visons à introduire la capac-

Figure C.3: Illustration du problème de variations du contexte pour une application de vidéo-surveillance

ité d'apprentissage et d'adaptation dans la tâche de segmentation. Traditionnellement, la connaissance explicite est utilisée pour configurer cette tâche dans les systèmes de vision. Cette connaissance est principalement composée de programmes en traitement d'images (par exemple des algorithmes de segmentation spécialisés et des post-traitements) et de programmes sur l'utilisation des algorithmes afin de contrôler la segmentation (sélection et réglage d'algorithmes). Pour ce faire, trois problèmes majeurs de la tâche de segmentation dans les systèmes de vision doivent être résolus :

- Le premier point est d'extraire les paramètres optimaux des algorithmes de segmentation dans le but d'obtenir une segmentation adaptée à la tâche de segmentation; c'est-à-dire une segmentation orientée par le but. Le réglage des paramètres est connu pour être une tâche délicate qui requiert souvent des compétences en traitement d'images. Ainsi, notre objectif est triple : premièrement nous voulons automatiser cette tâche dans le but de diminuer l'effort demandé à l'utilisateur et d'éviter des résultats trop subjectifs. Deuxièmement, la fonction de coût utilisée pour évaluer la qualité de la segmentation doit être générique; c'est-à-dire non dépendante de l'application. Troisièmement, aucune connaissance *a priori* sur le comportement des algorithmes n'est requise, uniquement des vérités terrain doivent être fournies par l'utilisateur.

- Une fois que les algorithmes ont été optimisés, un second point est de sélectionner le meilleur. La stratégie de sélection doit être basée sur une évaluation quantitative de la performance de chaque algorithme. Cependant, quand les images de l'application sont fortement variables, il est pratiquement impossible d'obtenir de bons résultats de segmentation avec un seul et unique algorithme. Dans ce cas, une stratégie de sélection dépendante du contenu de l'image doit être préférée.

- Dans de nombreux systèmes de vision, à l'étape de détection, le but est de séparer les objets d'intérêt du fond de l'image. Quand les objets d'intérêt et/ou le fond de l'image sont complexes (par exemple composés de plusieurs sous-parties), un algorithme de bas niveau ne peut pas produire une segmentation sémantique, même si il est optimisé. Pour cette raison, un troisième

point est de raffiner la segmentation (optimisée) pour fournir une segmentation sémantiquement significative aux modules de vision de plus haut niveau.

Notre objectif final est de montrer le potentiel de notre approche au travers de deux tâches de segmentation différentes dans des applications du monde réel.

- La première tâche de segmentation à laquelle nous nous intéressons est la segmentation d'images statiques dans une application biologique pour la détection précoce et le comptage d'insectes nuisibles. Cela implique de séparer de manière robuste les objets d'intérêt (mouches blanches adultes) du fond de l'image (feuilles de rose). Notre but est de démontrer que la plate-forme de vision cognitive développée dans l'équipe couplée avec notre approche de segmentation adaptative permet d'obtenir un meilleur taux de détection des mouches blanches que lorsque la plate-forme est configurée avec une segmentation *ad hoc*.

- La deuxième tâche de segmentation à laquelle nous nous intéressons est la segmentation d'objets en mouvement dans une application de vidéosurveillance. Le but est de détecter des objets tels que des personnes marchant dans la rue dans le champ de vue d'une caméra fixe. La détection est habituellement effectuée en utilisant des méthodes de soustraction de fond. Notre objectif est de montrer qu'une sélection dynamique du modèle de fond permet d'élargir la portée des applications de vidéosurveillance aux environnements fortement variables.

C.3 Contexte de l'étude

Ce travail prend place au sein de l'équipe-projet INRIA ORION à Sophia Antipolis. Orion est une équipe phare dans le domaine de l'analyse de scènes, à la frontière entre la vision par ordinateur, les systèmes à base de connaissance et l'ingénierie des connaissances. Orion a une approche cognitive de la vision. Cette approche vise à concevoir des systèmes de vision robustes et adaptables en les dotant d'une faculté cognitive. Cela signifie la capacité d'apprendre, d'adapter et de pondérer des solutions alternatives, et également de développer de nouvelles stratégies pour la détection, la reconnaissance et l'interprétation. Récemment, Hudelot [Hudelot, 2005] a proposé une plate-forme de vision cognitive pour l'interprétation sémantique d'images statiques. Cette plate-forme est basée sur la coopération de trois systèmes à base de connaissance dont un est dédié à la gestion intelligente des programmes de traitement d'images. Maillot [Maillot, 2005] a enrichi cette plate-forme avec des facultés d'apprentissage et une représentation sémantique de la connaissance basée sur une ontologie. Actuellement, la couche de détection de la plate-forme repose sur une segmentation *ad hoc*. Cela signifie que tous les opérateurs de segmentation on été configurés dans le code une fois pour toute. Dans ce contexte, mon travail vise à enrichir cette plate-forme de vision

cognitive au niveau de la segmentation d'image pour permettre une segmentation plus adaptative.

C.4 Contributions

Ma principale contribution est de proposer une approche cognitive du problème de la segmentation en résolvant les problèmes cités ci-dessous :

- Je propose une procédure d'optimisation générique afin d'extraire automatiquement les paramètres optimaux des algorithmes de segmentation. Cette procédure est basée sur trois composantes indépendantes : un algorithme de segmentation avec un ou plusieurs paramètres libres à régler, une métrique d'évaluation de la performance et un algorithme d'optimisation globale. L'évaluation de la qualité de la segmentation est faite selon une segmentation de référence (par exemple une segmentation manuelle). La métrique d'évaluation est générique, a un faible coût de calcul et peut être utilisée pour de nombreux problèmes de segmentation. De cette façon, la tâche de l'utilisateur est réduite à fournir des données de référence comme des segmentations manuelles d'images d'apprentissage.

- Je propose deux stratégies pour le problème de sélection de l'algorithme de segmentation. Ces stratégies utilisent les résultats de la phase d'optimisation appliquée sur un ensemble d'images d'apprentissage représentatif du problème. La première est basée sur le classement des valeurs de performance des algorithmes. La deuxième stratégie est d'identifier les différentes situations, appelées contextes, à partir du jeu d'apprentissage, et d'associer un algorithme de segmentation configuré pour chaque contexte.

- Je propose également une approche sémantique pour la segmentation d'images. Dans cette approche, nous abordons le problème du rafinement de la segmentation comme un problème d'étiquetage de régions. Cette approche est par conséquent élaborée pour les algorithmes de segmentation basés sur les régions uniquement. Le but est d'évaluer l'appartenance de chaque région à un ensemble prédéfini de régions partageant la même étiquette. L'évaluation repose sur une étape préliminaire d'apprentissage supervisé durant laquelle des classifieurs de régions sont entraînés sur des échantillons. Le rôle de l'utilisateur est d'étiqueter les régions des segmentations manuelles. L'originalité de cette approche est double. Premièrement, nous utilisons les segmentations optimisées comme données d'entrée des classifieurs de régions. Deuxièmement, les tâches sous-jacentes du processus d'apprentissage, à savoir l'extraction de caractéristiques des régions, la sélection de ces caractéristiques et l'apprentissage des classifieurs sont automatiquement optimisées dans un schéma de type *wrapper* afin d'obtenir les meilleures performances de classification.

Concernant les deux tâches de segmentation précédemment décrites, mes contributions sont les suivantes :

- Pour la tâche de segmentation dans l'application biologique, l'approche proposée dépasse la segmentation *ad hoc* en termes de qualité de la segmentation et permet ainsi au système de compter les insectes avec une meilleure précision.
- Pour la tâche de segmentation vidéo, ma principale contribution se situe au niveau de la modélisation du contexte. En accomplissant une sélection dynamique du modèle de fond basée sur l'analyse du contexte, mon approche permet d'élargir le champ d'application des systèmes de vidéosurveillance aux environnements fortement variables.

Chaque étape de l'approche proposée a été testée et évaluée sur plusieurs jeux de données. Ceci nous aide à montre les forces et les limitations de notre approche en terme de performance, de coût de calcul et de sensibilité aux paramètres clés.

C.5 Plan

Ce manuscrit est structuré comme suit. Le chapitre 2 présente au lecteur la segmentation d'images dans le cadre des systèmes de vision par ordinateur. Nous proposons une vue d'ensemble autour de quatre thèmes reliés à notre problème : la segmentation d'image dans les systèmes de vision, les différentes approches de segmentation d'images et de vidéos, les techniques d'évaluation de la performance de la segmentation et les techniques d'optimisation. Le chapitre 3 introduit l'approche proposée et donne nos objectifs et nos hypothèses pour les différents problèmes de la segmentation. Le chapitre 4 détaille chaque étape de notre approche : l'optimisation des paramètres des algorithmes, la sélection de l'algorithme et l'étiquetage sémantique des régions. Le chapitre 5 est dédié à la validation de l'approche pour une application réelle. En particulier, nous nous sommes intéressés à l'étape de segmentation d'un système de vision cognitive dédié à la reconnaissance d'organismes biologiques. Dans le chapitre 6, nous présentons comment notre approche peut être utilisée pour la segmentation adaptative dans des applications de vidéosurveillance. Une conclusion ainsi que des discussions sur les travaux futurs sont exposées dans le chapitre 7.

Appendix D

French Conclusion

Dans cette thèse, j'aborde le problème de la segmentation d'image et de vidéo avec une approche cognitive de la vision par ordinateur. Plus précisément, j'étudie deux problèmes majeurs de la tâche de segmentation dans les systèmes de vision : la sélection d'un algorithme et le réglage de ses paramètres libres, suivant le contenu de l'image et les besoins de l'application. La plupart du temps, cette tâche longue et fastidieuse est réalisée par un expert en traitement d'image dans un processus d'essais successifs. Récemment, quelques tentatives pour automatiser l'extraction des paramètres optimaux de segmentation ont été faites mais sont toujours trop dépendantes de l'application. La réutilisabilité de telles méthodes reste un problème ouvert. Nous avons choisi de gérer ce problème dans le cadre de la vision cognitive. La vision cognitive est un récent champ de recherche qui propose d'enrichir les systèmes de vision avec des capacités cognitives, c'est-à-dire de raisonner à partir de connaissances *a priori*, d'apprendre à partir d'information perceptuelles ou d'adapter leurs stratégies aux différents problèmes.

Dans cette thèse, je propose une méthodologie d'apprentissage supervisé pour la configuration hors ligne et l'adaptation en ligne de la tâche de segmentation dans les systèmes de vision. L'étape de configuration hors ligne requiert une connaissance minimale pour apprendre la sélection et le réglage optimal des algorithmes de segmentation. Cette connaissance est ensuite utilisée en temps-réel pour la segmentation adaptative d'images et de vidéos. Cette approche cognitive de la segmentation est donc une contribution pour la recherche en vision cognitive. En effet, cette approche satisfait les critères de robustesse, d'adaptation et de réutilisabilité qui définissent un système de vision cognitif.

L'approche proposée a été implémentée et validée sur deux types d'applications du monde réel : la segmentation adaptative d'image statique dans une application biologique et la segmentation vidéo dans une application de vidéo surveillance.

La première partie de ce chapitre dresse le bilan de mon approche et discute de ses contributions et de ses limitations. La seconde partie présente les perspectives pour améliorer la méthode, en particulier au sujet de l'apprentissage.

D.1 Bilan de l'approche proposée et de ses contributions

D.1.1 Une procédure d'optimisation générique

Notre procédure d'optimisation extrait automatiquement les paramètres optimaux des algorithmes de segmentation en se basant sur une évaluation quantitative de la qualité de l'image segmentée en fonction des segmentations manuelles. La méthode est indépendante de l'application et des algorithmes de segmentation testés. Seuls les paramètres libres à régler avec leur fourchettes de variations sont requis. Ce type de connaissance est habituellement fourni par les auteurs des algorithmes de segmentation. Le critère utilisé pour évaluer la qualité de la segmentation ne fait aucune hypothèse sur l'application ni sur les comportements des algorithmes. Il a été utilisé pour juger la tâche de segmentation dans deux applications. Il a aussi été appliqué pour l'évaluation de la segmentation de la banque d'images de Berkeley [Fowlkes and Martin, 2007]. Deux algorithmes d'optimisation globale sans calcul de dérivée (une méthode de recherche directe un algorithme génétique) ont été utilisé avec succès afin de minimiser ce critère. Dans ce domaine, ma contribution est une étude comparative des performances des deux algorithmes d'optimisation. Grâce à cette étude, nous avons identifié deux situations : quand le nombre de paramètres libres à optimiser est inférieur ou égale à deux, l'algorithme du Simplex donne de bons résultats avec un nombre minimal d'itérations. Quand le nombre de paramètres est supérieur à deux, l'algorithme génétique doit être préféré.

La principale difficulté de cette approche d'apprentissage supervisé de la segmentation est la segmentation manuelle des images d'apprentissage. Cette tâche est longue, fastidieuse et subjective. Des outils d'aide à l'annotation doivent être utilisés afin d'alléger la charge des utilisateurs. La force de cette appoche est aussi dépendante de la performance intrinsèque des algorithmes de segmentation. En conséquent, cette approche suppose qu'au moins un algorithme soit capable de fournir de bons résultats pour le problème de segmentation en question.

D.1.2 Une stratégie pour la sélection d'algorithme

Après l'optimisation de plusieurs algorithmes de segmentation sur un jeu d'images d'apprentissage, le problème qui se pose est la sélection d'un d'entre eux. Le but de cette étape est de répondre à la question de l'utilisateur : "Quel algorithme est le plus adaptée pour segmenter mon image ?". La première partie de ma stratégie consiste à identifier différentes situations dans le jeu d'images d'apprentissage. Une situation, appelée contexte, est représentée par un sous-ensemble d'images partageant les mêmes caractéristiques globales, comme la distributions des couleurs. Premièrement, un algorithme de classification non supervisé est utilisé pour identifier ces contextes. La seconde partie utilise les résultats de la phase d'optimisation pour accomplir un classement local (pour chaque contexte) des algorithmes optimisés en fonction de leurs performances moyennes.

Cette stratégie donne la possibilité de contrôler dynamiquement la segmentation (sélection et paramétrage d'un algorithme) sans la nécessité d'une connaissance *a priori* explicite du domaine d'application ou des algorithme de segmentation eux-mêmes.

Il faut noter que cette stratégie fait plusieurs hypothèses. Premièrement, cela suppose que tous les contextes possibles sont illustrés dans le jeu d'images d'apprentissage. Deuxièmement, cette stratégie soutient que pour chaque contexte identifié un jeu de paramètres moyens de l'algorithme le mieux classé existe et permet de fournir des bons résultats de segmentation.

D.1.3 Une approche sémantique de la segmentation d'image

La pluspart du temps, les résultats de segmentation issus d'algorithmes bas niveau sont dénués de sens sémantique. Je propose une approche sémantique de la segmentation d'image où l'étiquetage des régions aide à valider les résultats de segmentation d'un point de vue sémantique. L'algorithme d'étiquetage sémantique des régions repose sur trois étapes et utilise les résultats de segmentation précédemment optimisés. Dans un premier temps, pour chaque image d'apprentissage, l'utilisateur est invité à affecter des étiquettes sémantiques aux régions des segmentations manuelles en fonction des besoins de l'application. Ensuite, une correspondance entre les régions étiquetées des segmentations manuelles et les régions issues des segmentations optimisées est réalisée automatiquement. Enfin, des classifieurs (SVM) sont entrainés pour chaque étiquette en se basant sur des caractéristiques numériques extraites des régions. L'originalité de cette approche est que chaque étape du processus d'apprentissage, c'est-à-dire extraction de caractéristiques, et entrainement des SVM, est optimisée dans une schéma de *wrapper* de manière à maximiser la performance de classification de l'algorithme.

Actuellement, les caractéristiques des régions sont limitées à de l'information couleur et texture. La méthode pourrait être améliorée en prenant aussi en compte l'information spatiale comme celle spécifique aux relations entre les différentes classes sémantiques de régions.

D.1.4 Une implémentation logicielle de la méthodologie

Une implémentation logicielle de la méthodologie pour la configuration hors ligne et la segmentation adaptative temps-réel est proposée. Disposant d'un jeu d'images d'apprentissage avec leurs segmentations manuelles, le système, au travers une interface utilisateur graphique est capable de :

- extraire les paramètres optimaux pour six algorithmes de segmentation (quatre statique et deux vidéo),
- identifier les différents contextes du jeu d'apprentissage,
- sélectionner l'algorithme le plus performant pour chaque contexte identifié,
- entrainer les classifieurs de régions,

- contrôler la segmentation de nouvelles images en fonction de la connaissance apprise (identification du contexte, choix et paramétrage de l'algorithme),
- visualiser les résultats sous différentes formes.

Plus de détails sur l'implémentation C++ de l'approche sont donnée en annexe B.

Finalement, en adressant le problème de l'adaptation de la segmentation, nous avons aussi soulevé d'autres problèmes sous-jacents tels l'évaluation de la segmentation, la sélection et l'extraction de caractéristiques et la mise en correspondance entre connaissance bas niveau (numérique) et connaissance haut niveau (symbolique). Chacun de ces problèmes bien connus n'est pas facilement soluble et demande toujours à être considéré à part entière. Nous avons conçu notre approche (et notre logiciel) de manière modulaire de manière à prendre avantage des nouveaux progrès dans chacun de ces sujets de recherche.

D.1.5 Contributions pour la plate-forme de vision cognitive

Mon approche a enrichi la plate-forme en lui ajoutant une couche d'apprentissage pour la segmentation. Précédemment, les algorithmes de segmentation étaient réglés manuellement par un expert en traitements d'image et la sélection dynamique reposée sur une base de connaissance écrite à la main. Les mêmes algorithmes sont, après une phase d'apprentissage, automatiquement réglés ce qui permet une segmentation adaptée de différentes images. Le gain obtenu au niveau de la segmentation est bénéfique pour les modules de haut niveau de la plate-forme.

D.1.6 Contributions pour l'application biologique

Malgré les apparences, la segmentation robuste des mouches blanches adultes sur des feuilles de rosier n'est pas une tâche aisée. La variabilité de la couleur et de la texture des feuilles combinée avec la semi-transparence des ailes des mouches ainsi que la présence d'un grand nombre d'autres objets (œufs de mouches blanches, larves, traces de traitement chimiques, goutte d'eau, etc.) font que la segmentation n'est pas si facile. Comparée à une segmentation *ad hoc* réglée à la main, notre segmentation adaptative obtient de meilleurs résultats et permet ainsi d'obtenir une meilleure précision de comptage des mouches. De plus, la segmentation sémantique réduit drastiquement le nombre de région en fusionnant les sous-parties des classes d'objets. Cette technique diminue le coût de calcul du système puisque moins de régions doivent être traitées au niveau de la couche d'interprétation.

Actuellement, la plate-forme est capable de gérer la détection et le comptage de seulement un objet biologique (les mouches blanches adultes). D'autres bio-agresseurs (par exemple les pucerons, les aphids, etc.) ou d'autres stades de développement des mouches blanches (larves, œufs), devraient être traités dans le but de pouvoir évaluer notre segmentation adaptative et la plate-forme sur un

problème multi-classes (supérieur à deux). Pour cela, nous devons acquérir de nouvelles données (images et segmentations manuelles annotées) ainsi que de la connaissance haut niveau (description des objets en termes de concepts visuels).

En fin de compte, la plate-forme est actuellement aussi limitée par le système d'acquisition (un scanner à plat). Nous envisageons de maîtriser ce problème en utilisant des caméras vidéo. Un autre avantage des caméras vidéo est qu'elles fournissent une information temporelle de grand intérêt pour désambiguïser, par exemple, les situations d'occultations.

D.1.7 Contributions pour l'application vidéo

Dans cette application, ma contribution principale est la sélection du modèle de fond dynamique basée sur l'analyse du contexte. Cette approche est particulièrement bien adaptée aux applications où des changements court et long termes se produisent. L'algorithme de partionnement non-supervisé utilise des caractéristiques globales de l'image intégrant une information spatiale de manière à prendre en compte non seulement les changements globaux mais aussi les changements locaux. Nous avons proposé une algorithme de filtrage temporel du contexte.

Les premières expérimentations ont prouvé que la sélection dynamique de modèle de fond est une bonne approche pour traiter les problèmes d'adaptation de la segmentation. Néanmoins, il est clair que notre approche est toujours incapable de gérer les situations inconnues ou imprévues comme des nouveaux contextes. Une extension de cette approche pour permettre d'apprendre en continu est donc nécessaire.

Enfin, dans cette application, nous n'avons pas complètement suivi notre stratégie pour la sélection d'algorithme. Il pourrait être intéressant de voir comment différents algorithmes de segmentation vidéo pourraient coopérer entre eux.

D.2 Perspectives

D.2.1 Perspectives à court terme

Apprentissage incrémental pour les situations inconnues

La fragilité de notre approche face à des situations inconnues est actuellement son défaut majeur. Cela joue aussi bien sur l'analyse du contexte que sur la segmentation elle-même. Les algorithmes concernés sont l'algorithme DBScan pour l'analyse du contexte et les SVMs pour la segmentation sémantique. Actuellement, aucun de ces deux algorithmes n'est capable de s'adapter dynamiquement à de nouvelles données d'apprentissage : le processus doit être lancé de nouveau sur le jeu entier de données d'apprentissage. L'utilisation de techniques d'apprentissage incrémental pourrait être utile pour satisfaire à la propriété d'apprentissage continu. L'idée principale de l'apprentissage incrémental pour les situations inconnues est d'adapter dynamiquement la méthode de partionnement/classification en fonction de l'erreur de classification des nouvelles données d'entrée. Dans notre

problème, les situations inattendues peuvent être identifiées avec l'aide des estimations de la probabilité du contexte et de la probabilité de classification des rÃľgions. Par exemple, dans l'application de vidéosurveillance, quand le contexte de bruit est détecté durant plusieurs images, une alarme pourrait être levée et les images concernées pourraient être considérées comme étant de nouvelles images d'apprentissage. Dans un processus supervisé, l'utilisateur aurait alors à décider si chaque image est une image de fond (scène vide) ou non. Habituellement, losqu'un contexte de bruit est détecté, de nombreux pixels de mouvement sont détectés partout dans l'image. Dans un processus non-supervisé, la validation des nouvelles images d'apprentissage pourrait donc aussi être basée sur une analyse spatiale des pixels en mouvement. L'utilisation d'un algorithme de classification adaptative utilisant un partionnement incrémental robuste comme proposé dans [Prehn and Sommer, 2006] permettrait ainsi de mettre à jour dynamiquement les partitions et d'en créer de nouvelles si nécessaire.

Méta évaluation de la segmentation d'image

La fonction d'évaluation des résultats de segmentation que nous avons proposée dans le chapitre 4 est basé sur deux critères fondamentaux (comptage des pixels de contour sur- et sous-détectés). Cela permet de rendre la métrique réutilisable pour un large ensemble de problèmes de segmentation. Cependant, la manière de pondérer chaque critère est un élément clé de la métrique. Par exemple, pour certaines applications, il est préférable de donner plus de poids au taux de sous-détection qu'au taux de sur-détection. Il serait aussi plus adéquat d'utiliser ou de combiner n critères ρ différents selon les besoins de l'application, par exemple des critères basés sur les régions et sur les contours tel que : $E_I^A = \alpha_1\rho_1 + \alpha_2\rho_2 + \ldots + \alpha_n\rho_n$. Nous pensons qu'en général il est difficile de spécifier une forme exacte de la fonction de coût car cela requiert de définir précisément la part de chaque critère. Pour cela, des méta heuristiques pour la pondération de fonctions, comme le front de Pareto, pourraient être utilisées, comme dans [Everingham et al., 2002]. Pour chaque algorithme de segmentation et pour toutes les images d'apprentissage, l'espace des paramètres est exploré et produit un graphe de la fonction de coût pour chaque critère et pour chaque image d'apprentissage. Le but est alors d'estimer la combinaison des différents critères d'évaluation qui donne globalement les meilleurs scores de performance (i.e. le front de Pareto). Pour ce faire, un algorithme d'optimisation globale, comme un algorithme génétique est utilisé afin de trouver la configuration optimale. Une autre possibilité est d'utiliser des algorithmes de méta apprentissage comme dans [Zhang et al., 2006]. L'idée est d'entraîner un algorithme d'apprentissage (par exemple un arbre de décision) qui détermine comment unir les résultats des différents critères d'évaluation appliqués au jeu d'images d'apprentissage. Le principal avantage est d'obtenir une métrique d'évaluation adaptée au type d'images à traiter.

Paramétrage local des algorithmes de segmentation vidéo

Le but des algorithmes de segmentation vidéo comme ceux basés sur les mixtures de gaussiennes, des estimateurs de densité à noyau et les codebooks est d'apprendre les fourchettes de valeurs possibles des modèles de fond pour chaque pixel. Durant le processus d'apprentissage, quelques seuils sont définis pour fixer ces fourchettes de valeurs. Généralement, les valeurs de ces paramètres sensibles sont les même pour tous les pixels. Dans le cas des applications de vidéo surveillance avec une caméra fixe, les paramètres devraient être optimisés pour chaque pixel. Par exemple, les seuils de détection pour les pixels des zones d'intérêt devraient être réglés de manière à produire des modèles de fond sensibles aux variations. Ceci est illustré à la Figure D.1 où z_1 est la zone où les objets d'intérêt (personnes) ne passe jamais et z_2 la zone où ils sont attendus. Les seuils de détection pour chaque pixel dans z_1 devraient ainsi être plus faibles que ceux de z_2.

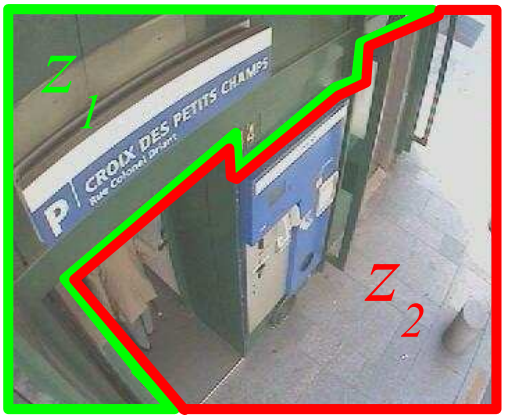


Figure D.1: Exemple de réglage local des paramètres basé sur une connaissance *a priori* de la scène filmée. La valeur du seuil de détection pour les pixels dans z_1 devrait être plus faible que celle pour les pixels dans z_2.

Segmentation vidéo spatio-temporelle

Le principale défaut des approches basées sur le pixel en segmentation vidéo est qu'aucune cohérence spatiale de détection de mouvement n'est prise en compte. Pour s'affranchir de ce problème, une solution est de calculer en parallèle une segmentation d'image basée région. L'objectif est de rafiner la segmentation obtenue avec l'algorithme de détection du mouvement. Cette technique est illustrée dans la Figure D.2. Une image d'entrée (a) est segmentée en utilisant un algorithme de segmentation basé région. Le résultat est présenté en (b). En parallèle, une segmentation basé mouvement (c) est calculé en utilisant le modèle de codebook par exemple. Le résultat final (d) est une combinaison des deux segmentations en respectant un critère de recouvrement. Dans cette exemple, la segmentation

basée région a été manuellement paramétrée pour produire une sur-segmentation.

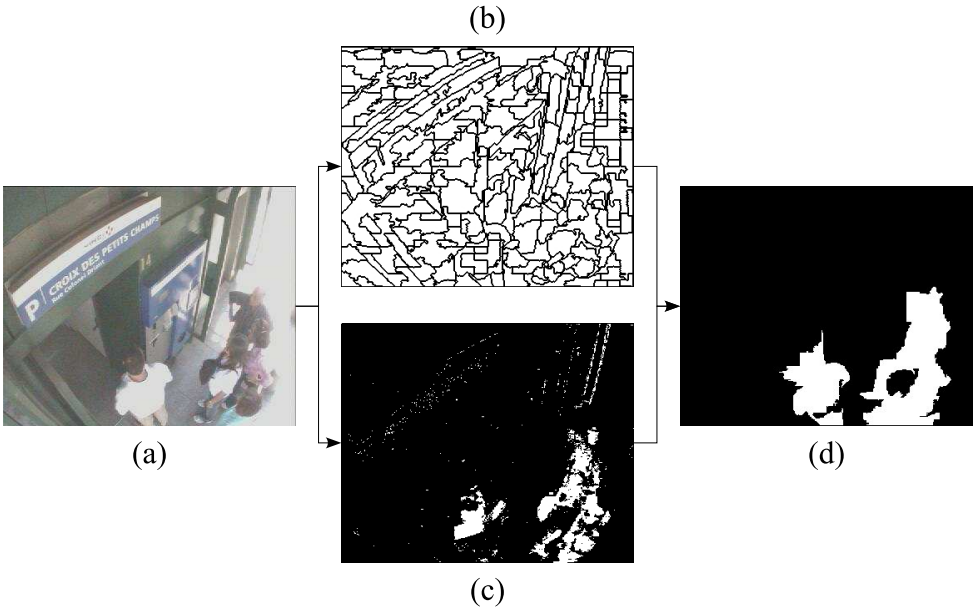


Figure D.2: Illustration d'une segmentation spatio-temporelle (d) combinant les résultats d'un algorithme de soustraction de fond (c) et d'un algorithme basé région (b) pour l'image d'entrée (a).

D.2.2 Perspectives à long terme

Utilisation d'une ontologie de concepts visuels pour la segmentation sémantique

Actuellement, la segmentation sémantique est basée sur des caractéristiques numériques décrivant indépendamment chaque région. Des objets complexes, comme une personne, sont composés de plusieurs sous-parties (tête, jambes, etc.) qui ne peuvent pas être décrites avec les même caractéristiques bas niveau comme la couleur ou la texture. De plus, certaines de ces sous-parties peuvent avoir une infinité de variations dans leur couleur selon les habits de la personne par exemple. Ces sous-parties, se rapportant au même objet sémantique peuvent cependant être liées entre elles par des relations spatiales et des décompositions hiérarchiques. Par conséquent, plusieurs concepts visuels différents devraient être utilisés pour atteindre une segmentation sémantique même si l'objet est difficile à modéliser. Pour ce faire, nous pourrions faire usage de l'ontologie de concepts visuels proposée par Maillot et al. [Maillot and Thonnat, 2008] en mixant, d'une manière structurée, la connaissance *a priori* des différents concepts (caractéristiques de couleur, de texture, géométriques et de relations spatiales). Le but

serait d'évaluer l'appartenance de chaque région segmentée à un objet sémantique selon les détecteurs de concepts visuels entrainés.

Utilisation de caractéristiques partagée

Pour les cas vraiment difficiles où l'information intraclasses (i.e. l'apparence d'un objet) est vraiment hétérogène et/ou l'information interclasses est faiblement discriminative, la sélection de caractéristiques représentatives est délicate et mène à de faibles performances de classification. Dans ce cas, les approches basées sur des caractéristiques visuelles partagées [Torralba and Murphy, 2007] entre les classes comme les *boosted decision stumps* seraient plus appropriées et efficaces. Les *bossted decision stumps* réduisent la complexité de calcul et d'échantillonnage en trouvant des caractéristiques qui peuvent être partagées parmi les classes. Cette approche est particulièrement efficace pour les problèmes de classification multi-classes avec peu d'exemple d'apprentissage.

Benchmarking de la segmentation vidéo

Des bases de données vidéos pour l'évaluation des systèmes de vision existent mais sont rarement adaptées à l'évaluation de la couche de détection. La plupart des vérités terrain fournies avec les vidéos sont des boîtes englobantes autour des objets d'intérêt mobiles. Le traçé des boîtes englobantes repose sur la fusion de blobs et requiert ainsi une certaine connaissance *a priori* sur les objets à détecter. De plus, les métriques usuelles pour l'évaluation de la performance de la segmentation sont basées sur les taux de vraie et fausse détection et/ou sur la précision des pixels de contours détectés. Par conséquent, les boîtes englobantes ne sont vraiment pas adaptées pour l'évaluation des résultats au niveau de la détection. La meilleure solution consiste à tracer, par exemple, la silhouette d'une personne pour chaque image d'une séquence vidéo. Evidemment, cette tâche requiert un énorme effort et ne peut pas être accomplie par une seul utilisateur puisque les vidéos sont composées de plusieurs milliers d'images. Ce problème a déjà été abordé pour les banques d'images statiques comme la banque de Berkeley [Fowlkes and Martin, 2007] pour l'évaluation de la segmentation basée contour (6000 segmentations manuelles de 500 images couleur de la collection Corel faites par 30 sujets humains) et la base du MIT LabelMe [Russell et al., 2005] (plus de 41300 images segmentées manuellement et annotées). La force de ces banques de données est leur libre accès à la communauté scientifique et les outils fournis pour faciliter la segmentation manuelle et les annotations des images. Nous pensons qu'une telle stratégie devrait être étendue pour l'annotation de contenu vidéo.

Bibliography

[Abdul-Karim et al., 2005] Abdul-Karim, M.-A., Roysam, B., Dowell-Mesfin, N. M., Jeromin, A., Yuksel, M., and Kalyanaraman, S. (2005). Automatic selection of parameters for vessel/neurite segmentation algorithms. *Transactions on Image Processings*, 14(9):1338–1350.

[Alvarado Moya, 2004a] Alvarado Moya, J. P. (2004a). *Segmentation of color images for interactive 3D object retrieval.* PhD thesis, Fakultat fur Elektrotechnik und Informationstechnik der Rheinisch-Westfalischen Technischen Hochschule Aachen.

[Alvarado Moya, 2004b] Alvarado Moya, J. P. (2004b). *Segmentation of color images for interactive 3D object retrieval.* PhD thesis, Technical University of Aachen.

[Aschlock, 2006] Aschlock, D. (2006). *Evolutionary Computation for Modeling and Optimization.* Springer.

[Bahnu et al., 1995] Bahnu, B., Lee, S., and Das, S. (1995). Adaptive image segmentation using genetic and hybrid search methods. *Transactions on Aerospace and Electronic Systems*, 31(4):1268–1291.

[Barron et al., 1994] Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performance of optical flow techniques. *Int. J. Comput. Vision*, 12(1):43–77.

[Bliek et al., 2001] Bliek, C., Spelucci, P., Vicente, L. N., Neumaier, A., Granvilliers, L., Monfroy, E., Benhamou, F., Huens, E., Van Hentenryck, P., Sam-Haroud, D., , and Faltings, B. (2001). Algorithms for solving nonlinear constrained and optimization problems: The state of the art. Technical report, COCONUT Project Report.

[Blum and Langley, 1997] Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271.

[Boissard et al., 2003] Boissard, P., Hudelot, C., Thonnat, M., Peŕez, G., Pyrrha, P., and Bertaux, F. (2003). An automated approach to monitoring the sanitary status and to detect early biological attacks on plants in greenhouses - example on flower crops. In *International Symposium on Greenhouse Tomato, Integrated Crop Protection and Organic Production*, Avignon, France.

[Borenstein and Malik, 2006] Borenstein, E. and Malik, J. (2006). Shape guided object segmentation. In *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 969–976. IEEE Computer Socciety.

[Borenstein and Ullman, 2004] Borenstein, E. and Ullman, S. (2004). Learning to segment. In LNCS, editor, *ECCV'04*, volume 30, pages 315–328.

[Borsotti et al., 1998] Borsotti, M., Campadelli, P., and Schettini, R. (1998). Quantitative evaluation of color image segmentation results. *Pattern Recognition Letters*, 19:741–747.

[Burges, 1998] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[C. A. and P. M., 1996] C. A., F. and P. M., P. (1996). *State of the Art in Global Optimization: Computational Methods and Applications*, volume 7 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698.

[Cardoso and Corte-Real, 2005] Cardoso, J. and Corte-Real, L. (2005). Toward a generic evaluation of image segmentation. *IEEE Trans. on Image Processing*, 14(11):1773–1782.

[Chang et al., 1994] Chang, M. M., Sezan, I. M., and Tekalp, M. A. (1994). Adaptive bayesian segmentation of color images. *Journal of Electronic Imaging*, 3(4):404–414.

[Chen and Wang, 2004] Chen, Y. and Wang, J. Z. (2004). Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939.

[Cheung and Kamath, 2004] Cheung, S.-c. S. and Kamath, C. (2004). Robust techniques for background subtraction in urban traffic video. In Panchanathan, S. and Vasudev, B., editors, *Proceedings of the SPIE Visual Communications and Image Processing*, volume 5308, pages 881–892.

[Cinque et al., 2002] Cinque, L., Corzani, F., Levialdi, S., Cucchiara, R., and Pignalberi, G. (2002). Improvement in range segmentation parameters tuning. In *Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02)*, volume 1, page 10176, Washington, DC, USA. IEEE Computer Society.

[Clément and Thonnat, 1993] Clément, V. and Thonnat, M. (1993). A knowledge-based approach to integration of image processing procedures. *Computer Vision, Graphics and Image Processing: Image Understanding*, 57(2):166–184.

[Clouard et al., 1999] Clouard, R., Elmoataz, A., Porquet, C., and Revenu, M. (1999). Borg : A knowledge-based system for automatic generation of image processing programs. 21(2):128–144.

[Coimbra et al., 2003] Coimbra, M., Davies, M., and Velastin, S. (2003). Pedestrian detection using mpeg-2 motion vectors. pages 164–169, London, UK. Izquierdo, E.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619.

[Correia and Pereira, 2000] Correia, P. and Pereira, F. (2000). Objective evaluation of relative segmentation quality. In *International Conference on Image Processing (ICIP)*.

[Correia and Pereira, 2003] Correia, P. and Pereira, F. (2003). Objective evaluation of video segmentation quality. *IEEE Trans. on Image Processing*, 12(2):186–200.

[Cristani et al., 2002] Cristani, M., Bicego, M., and Murino, V. (2002). Integrated region- and pixel-based approach to background modelling. In *Proceedings of the Workshop on Motion and Video Computing*, page 3, Washington, DC, USA. IEEE Computer Society.

[Crubézy, 1999] Crubézy, M. (1999). *Pilotage de programmes pour le traitement d'images médicales*. PhD thesis, Université de Nice Sophia Antipolis.

[Draper et al., 1996] Draper, B., Hanson, A., and Riseman, E. (1996). Knowledge-directed vision: Control, learning, and integration. In IEEE, editor, *Signals and Symbols*, volume 84, pages 1625–1637.

[ECVISION, 2005] ECVISION (2005). A research roadmap of cognitive vision. Technical report, Project IST-2001-35454.

[Elgammal et al., 2000] Elgammal, A. M., Harwood, D., and Davis, L. S. (2000). Non-parametric model for background subtraction. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 751–767, London, UK. Springer-Verlag.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, Portland.

[Everingham et al., 2002] Everingham, M., Muller, H., and Thomas, B. (2002). Evaluating image segmentation algorithms using the pareto front. In *7th Eur. Conf. Computer Vision, Part IV*, volume 2353, pages 34–38.

[Fan et al., 2005] Fan, J., Zeng, G., Body, M., and Hacid, M. (2005). Seeded region growing: an extensive and comparative study. *Pattern Recogn. Lett.*, 26(8):1139–1156.

[Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *IJCV*, 59(2).

[Ficet-Cauchard et al., 1999] Ficet-Cauchard, V., Porquet, C., and Revenu, M. (1999). Cbr for the management and reuse of image-processing expertise: a conventional system. *Engineering Applications of Artificial Intelligence*, 12(6):733–747.

[Fletcher, 1987] Fletcher, R. (1987). *Practical methods of optimization; (2nd ed.)*. Wiley-Interscience, New York, NY, USA.

[Fowlkes and Martin, 2007] Fowlkes, C. and Martin, D. (2007). The berkeley segmentation dataset and benchmark. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

[Frucci et al., 2007] Frucci, M., Perner, P., and Sanniti di Baja, G. (2007). *Case-based Reasoning for Image Segmentation by Watershed Transformation*. Springer Verlag.

[Gelasca et al., 2003] Gelasca, E., Salvador, E., and Ebrahimi, T. (2003). Intuitive strategy for parameter setting in video segmentation. In *Visual Communications and Image Processing*, volume 5150, pages 998–1008. Proc. of SPIE.

[Georis, 2006] Georis, B. (2006). *Program Supervision Techniques for Easy Configuration of Video Understanding Systems*. PhD thesis, Université Catholique de Louvain Faculté des Sciences Appliquées.

[Gevers and Smeulders, 1997] Gevers, T. and Smeulders, A. W. M. (1997). Combining region splitting and edge detection through guided delaunay image subdivision. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1021–1026, Los Alamitos, CA, USA. IEEE Computer Society.

[Gill et al., 1981] Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical Optimization*. Academic Press, San Diego, CA 92101.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

[Han et al., 2004] Han, B., Comaniciu, D., and Davis, L. (2004). Sequential kernel density approximation through mode propagation: Applications to background modeling. In *Proc. of Asian Conf. Computer Vision*, Jeju Island, Korea.

[Haralick, 1979] Haralick, R. (1979). Statistical and structural approaches to texture. In *Proceedings of the IEEE*, volume 67.

[He et al., 2006] He, X., Zemel, R., and Ray, D. (2006). Learning and incorporating top-down cues in image segmentation. *Lecture Note in Computer Science*, 1(3951):338–351.

[Herbulot et al., 2006] Herbulot, A., Jehan-Besson, S., Duffner, S., Barlaud, M., and Aubert, G. (2006). Segmentation of vectorial image features using shape gradients and information measures. *ournal of Mathematical Imaging and Vision*, 25(3):365–386.

[Hooke and Jeeves, 1961] Hooke, R. and Jeeves, T. A. (1961). " direct search" solution of numerical and statistical problems. *J. ACM*, 8(2):212–229.

[Horn and Schunck, 1992] Horn, B. K. P. and Schunck, B. G. (1992). *Determining optical flow*, pages 389–407. Jones and Bartlett Publishers, Inc., USA.

[Huang and LeCun, 2006] Huang, F. J. and LeCun, Y. (2006). Large-scale learning with svm and convolutional nets for generic object categorization. In *CVPR'06*, pages 284–291. IEEE Computer Society.

[Hudelot, 2005] Hudelot, C. (2005). *Towards a Cognitive Vision Platform for Semantic Image Interpretation; Application to the Recognition of Biological Organisms*. PhD thesis, Nice-Sophia Antipolis University.

[Hudelot and Thonnat, 2003] Hudelot, C. and Thonnat, M. (2003). A cognitive vision platform for automatic recognition of natural complex objects. In *International Conference on Tools with Artificial Intelligence*, Sacramento, USA. IEEE.

[Ignizio and Cavalier, 1994] Ignizio, J. P. and Cavalier, T. M. (1994). *Linear Programming*. Prentice Hall, Englewood Cliffs.

[Itoh and Matsuda, 1996] Itoh, S. and Matsuda, I. (1996). Segmentation of colour still images using voronoi diagrams. In *Proc. of the 8th European Signal Processing Conference*, volume 3, pages 1869–1872, Triestre, Italy.

[Jain and Farrokhnia, 1991] Jain, A. K. and Farrokhnia, F. (1991). Unsupervised texture segmentation using gabor filters. *Pattern Recogn.*, 24(12):1167–1186.

[Jehan-Besson et al., 2004] Jehan-Besson, S., Gastaud, M., Precioso, F., Barlaud, M., Aubert, G., and Debreuve, E. (2004). From snakes to region-based active contours defined by region-dependent parameters. *Journal of Applied Optics*, 43(2):247–256.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. P. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Kass et al., 1988] Kass, M., Wintkin, A., and Terzopoulos, D. (1988). Snakes : Active contour models. *International Journal of Computer Vision*, 1(4):321–332.

[Kim et al., 2005] Kim, K., Chalidabhongse, T. H., Harwood, D., and Davis, L. (2005). Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

[Kohonen, 1989] Kohonen, T. (1989). *Self-Organization and associative memory.*

[Laws, 1980] Laws, K. (1980). *Textured Image Segmentation.* PhD thesis, Univ. Southern California.

[Levine and Nazif, 1985] Levine, M. D. and Nazif, A. (1985). Dynamic measurement of computer generated image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 7(2):155–164.

[Lewis et al., 2000] Lewis, R., Torczon, V., and Trosset, M. (2000). Direct search methods: Then and now.

[Liu and Yang, 1994] Liu, J. and Yang, Y.-H. (1994). Multiresolution color image segmentation. *IEEE Trans. on PAMI*, 16(7):689–700.

[Lucchese and Mitra, 2001] Lucchese, L. and Mitra, S. (2001). Color image segmentation : A state-of-the-art survey. volume 67 of *A*, pages 207–221, New Dehli. Indian National Science Academy.

[Maillot, 2005] Maillot, N. (2005). *Ontology Based Object Learning and Recognition.* PhD thesis, Nice-Sophia Antipolis University.

[Maillot and Thonnat, 2008] Maillot, N. and Thonnat, M. (2008). Ontology based complex object recognition. *Image and Vision Computing*, 26(1):102–113.

[Maillot et al., 2004] Maillot, N., Thonnat, M., and Boucher, A. (2004). Towards ontology-based cognitive vision. *MVA*, 16:33–40.

[Mao and Kanungo, 2000] Mao, S. and Kanungo, T. (2000). Automatic training of page segmentation algorithms : An optimizatin approach. In *ICPR, Barcelona, Spain*, pages pp 531–534.

[Marr, 1982] Marr, D. (1982). *Vision.* W.H. Freeman and Compagny, New York.

[Martin et al., 2004] Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(1).

[Matsuyama and Hwang, 1990] Matsuyama, T. and Hwang, V. S. (1990). *SIGMA: A Knowledge-Based Aerial Image Understanding System*. Perseus Publishing.

[Maurer et al., 2003] Maurer, C. R. J., Qi, R., and Raghavan, V. (2003). A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):265–270.

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, M., and Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys*, 21(6):1087–1092.

[Mezaris et al., 2003] Mezaris, V., Kompatsiaris, I., and Strintzis, M. (2003). Still image objective segmentation evaluation using ground truth. In *Fifth COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, pages 9–14, Prague, Czech Republic.

[Mittal and Paragios, 2004] Mittal, A. and Paragios, N. (2004). Motion-based background subtraction using adaptive kernel density estimation. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 02:302–309.

[Moisan and Thonnat, 1995] Moisan, S. and Thonnat, M. (1995). Knowledge-based systems for program supervision. In *1st International Workshop on Knowledge Based systems for the (re)Use of Program Librairies*, pages 4–8.

[Molina et al., 2002] Molina, L. C., Belanche, L., and Nebot, A. (2002). Feature selection algorithms : A survey and experimental evaluation. In *ICDM'02*, pages 306–313, Japan.

[Munoz et al., 2003] Munoz, X., Freixenet, J., Cufi, X., and Marti, J. (2003). Strategies for image segmentation combining region and boundary information. *Pattern Recognition Letters*, 24:375–392.

[Nagel and Gehrke, 1998] Nagel, H. and Gehrke, A. (1998). Spatiotemporally adaptive estimation and segmentation of of-fields. volume 1407, pages 86–102, Freiburg, Germany. Springer-Verlag.

[Nazif and Levine, 1984] Nazif, A. M. and Levine, M. D. (1984). Low level image segmentation : An expert system. *Pattern Analysis and Machine Intelligence*, 6(5):555–577.

[Nelder and Mead, 1965] Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 15:1162–1173.

[Nock and Nielsen, 2004] Nock, R. and Nielsen, F. (2004). Statistical region merging. *Pattern Analysis and Machine Intelligence*, 26(11):1452–1458.

[Odet and Benoit-Cattin, 2002] Odet, C. andBelaroussi, B. and Benoit-Cattin, H. (2002). Scalable discrepancy measures for segmentation evaluation. 1.

[Ossola, 1996] Ossola, J.-C. (1996). *Coopï£¡ation de sytï£¡es ï£¡base de connaissance pour l'analyse et la reconnaissance d'objets naturels complexes : application au classement de galaxies ou de zooplanctons*. PhD thesis, Universitï£¡de Nice Sophia Antipolis.

[Pal and Pal, 1989] Pal, N. R. and Pal, S. (1989). Entropic thresholding. *Signal Processing*, 16:97–108.

[Pal and Pal, 1993] Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294.

[Panjwani and Healey, 1995] Panjwani, D. K. and Healey, G. (1995). Markov random field models for unsupervised segmentation of textured color images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 17, pages 939–954, Los Alamitos, CA, USA. IEEE Computer Society.

[Pass et al., 1997] Pass, G., Zabih, R., and Miller, J. (1997). Comparing images using color coherence vectors. In *ACM International Conference on Multimedia*, pages 65–73. ACM Press, New York, USA.

[Peng and Bahnu, 1998] Peng, J. and Bahnu, B. (1998). Delayed reinforcement learning for adaptive image segmentation and feature extraction. *Systems, Man, and Cybernetics*, 28(3).

[Pignalberi et al., 2003] Pignalberi, G., Cucchiara, R., Cinque, L., and Levialdi, S. (2003). Tuning range image segmentation by genetic algorithm. *EURASIP Journal on Applied Signal Processing*, 2003(8):780–790.

[Prati et al., 2003] Prati, A., Mikic, I., Trivedi, M., and Cucchiara, R. (2003). Detecting moving shadows: algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):918–923.

[Pratt et al., 1978] Pratt, W., Faugeras, O., and Gagalowicz, A. (1978). Visual discrimination of stochastic texture fields. *IEEE Trans. on Systems, Man, and Cybernetics*, 8(11):796–804.

[Prehn and Sommer, 2006] Prehn, H. and Sommer, G. (2006). An adaptive classification algorithm using robust incremental clustering. In *Proc. of the 18th International Conference on Pattern Recognition (ICPR'06)*, pages 896–899, Washington, DC, USA. IEEE Computer Society.

[Priese and Sturm, 2002] Priese, Lutz Rehrmann, V. and Sturm, P. (2002). Color structure code.

[Randen and Husoy, 1999] Randen, T. and Husoy, J. H. (1999). Filtering for texture classification: A comparative study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(4):291–310.

[Raviv and Blom, 2001] Raviv, M. and Blom, T. (2001). The effect of water availability and quality on photosynthesis and productivity of soilless-grown cut roses. *Scientia Horticulturae*, 88(4):257–276.

[Reed and du Buf, 1993] Reed, T. R. and du Buf, J. M. H. (1993). A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Underst.*, 57(3):359–372.

[Ronfard, 1994] Ronfard, R. (1994). Region-based strategies for active contour models. *Int. J. Comput. Vision*, 13(2):229–251.

[Rosenberger, 1999] Rosenberger, C. (1999). *Mise en oeuvre d'un système adaptatif de segmentation d'images*. PhD thesis, Université de Rennes 1.

[Rosenberger et al., 2005] Rosenberger, C., Chabrier, S., Laurent, H., and Emile, B. (2005). *Advances in Image and Video Segmentation*, chapter Unsupervised and Supervised Segmentation evaluation. Yu-Jin Zhang, Tsinghua University, Beijing, China.

[Russell et al., 2005] Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2005). Labelme: A database and web-based tool for image annotation. Technical report, Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology.

[Ruzon and Tomasi, 2001] Ruzon, M. A. and Tomasi, C. (2001). Edge, junction, and corner detection using color distributions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1281–1295.

[Sahoo et al., 1988] Sahoo, P. K. Soltani, S., Wong, A. K., and Chen, Y. C. (1988). A survey of thresholding techniques. *Comput. Vision Graph. Image Process.*, 41(2):233–260.

[Schnitman et al., 2006] Schnitman, Y., Caspi, Y., Cohen-Or, D., and Lischinski, D. (2006). Inducing semantic segmentation from an example. In *ACCV*, volume 3852, pages 393–384. Springer-Verlag.

[Seki et al., 2003] Seki, M., Wada, T., Fujiwara, H., and Sumi, K. (2003). Background subtraction based on cooccurrence of image variations. volume 02, page 65, Los Alamitos, CA, USA. IEEE Computer Society.

[Sharifi et al., 2002] Sharifi, M., Fathy, M., and Mahmoudi, M. T. (2002). A classified and comparative study of edge detection algorithms. pages 117–120, Los Alamitos, CA, USA. IEEE Computer Society.

[Skarbek and Koschan, 1994] Skarbek, W. and Koschan, A. (1994). Colour image sementation - a survey. Technical report, Technical University of Berlin.

[Smith and Brady, 1997] Smith, S. and Brady, J. (1997). Susan - a new approach to low level image processing. *Int. Journal of Computer Vision*, 23(1):45–78.

[Stauffer and Grimson, 1999] Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 246–252.

[Stiller and Konrad, 1999] Stiller, C. and Konrad, J. (1999). Estimating motion in image sequences, a tutorial on modeling and computation of 2d motion. *IEEE Signal Process. Mag.*, 16:70–91.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

[Thonnat, 2002] Thonnat, M. (2002). Knowledge-based techniques for image processing and for image understanding. *j. Phys. IV*, 12(1):189–236.

[Thonnat et al., 1998] Thonnat, M., Moisan, S., and Crubézy, M. (1998). Experience in integrating image processing programs. In Christensen, H., editor, *1st International Conference on Vision Systems. Lecture Notes in Computer Science*, Gran Canaria, Spain. Springer-Verlag.

[Torralba and Murphy, 2007] Torralba, A. and Murphy, K. P. (2007). Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869.

[Toyama et al., 1999] Toyama, K., Krumm, K., Brumitt, B., and Meyers, B. (1999). Wallflowers: principles and practice of background maintenance. pages 255–261.

[Vapnik, 1995] Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

[Vasaru and Hong, 1996] Vasaru, D. and Hong, H. (1996). Survey on nonlinear optimization. Technical Report 96–04, Johannes Kepler University, Linz, Austria.

[Vincent and Soille, 1991] Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598.

[Vincze et al., 2006] Vincze, M., Ponweiser, W., and Zillich, M. (2006). Contextual coordination in a cognitive vision system for symbolic activity interpretation. In *ICVS '06: Proceedings of the Fourth IEEE International Conference on Computer Vision Systems*, page 12, Washington, DC, USA. IEEE Computer Society.

[Vinet, 1991] Vinet, L. (1991). *Segmentation et mise en correspondance de régions de paires d'images stÁĺrÁĺoscopiques*. PhD thesis, Université de Paris IX Dauphine.

[Weszka and Rosenfeld, 1978] Weszka, J. and Rosenfeld, A. (1978). Threshold evaluation techniques. *IEEE Trans. on SMC*, 8:622–629.

[Wren et al., 1997] Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785.

[Wu et al., 2004] Wu, T., Lin, C., and Weng, R. (2004). Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005.

[Xia et al., 2005] Xia, Y., Feng, D., Rongchun, Z., and Petrou, M. (2005). Learning-based algorithm selection for image segmentation. *Pattern Recognition. Letters*, 26(8):1059–1068.

[Yasnoff et al., 1977] Yasnoff, W., Mui, J., and Bacus, J. (1977). Error measures for scene segmentation. *Pattern Recognition*, 9:217–231.

[Zeboudj, 1988] Zeboudj, R. (1988). *Filtrage, seuillage automatique, contraste et contours : du prÃĺ-traitement ï£¡ l'analyse d'images*. PhD thesis, UniversitÃĺ de Saint-Etienne.

[Zhang and Lu, 2001] Zhang, D. and Lu, G. (2001). Segmentation of moving objects in image sequence: A review. *Circuits, Systems, and Signal Processing*, 20(2):143–183.

[Zhang et al., 2006] Zhang, H., Cholleti, S., and Goldman, A. (2006). Meta-evaluation of image segmentation using machine learning. In *Proceedings of the 2006 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1138–1145.

[Zhang et al., 2004] Zhang, H., Fritts, J., and Goldman, S. (2004). An entropy-based objective evaluation method for image segmentation. In *Proc. of the SPIE Storage, and Retrieval Methods, and Applications for Multimedia*, volume 5307, pages 38–49.

[Zhang, 1996] Zhang, Y. (1996). A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346.

[Zhang and Luo, 2000] Zhang, Y. and Luo, H. (2000). Optimal selection of segmentation algorithms based on performance evaluation. *Optical Engineering*, 39:1450–1456.

[Zugaj and Lattuati, 1998] Zugaj, D. and Lattuati, V. (1998). A new approach of color images segmentation based on fusing region and edge segmentations outputs. *Pattern Recognition Letters*, 31(2):105–113.