Segmentation interactive multiclasse d'images par classification de superpixels et optimisation dans un graphe de facteurs

Bérengère Mathieu

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *15 novembre 2017* **par :**

BÉRENGÈRE MATHIEU

Segmentation interactive multiclasse d'images par classification de superpixels et optimisation dans un graphe de facteurs

Interactive multi-class image segmentation using superpixel classification and factor graph-based optimisation

---

JURY

PHILIPPE JOLY	Professeur, Université Toulouse 3	Examinateur
LAURENT MASCARILLA	Maître de Conférences HDR, Université de La Rochelle	Rapporteur
LAURE TOUGNE	Professeur, Université Lumière – Lyon 2	Rapporteur
ALAIN CROUZIL	Maître de Conférences, Université Toulouse 3	Directeur de thèse
JEAN-BAPTISTE PUEL	Maître de Conférences, ENSFEA Toulouse-Auzeville	Co-directeur de thèse

---

École doctorale et spécialité :
MITT : Image, Information, Hypermédia
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)
Directeur(s) de Thèse :
Alain CROUZIL et *Jean-Baptiste PUEL*
Rapporteurs :
Laurent MASCARILLA et *Laure TOUGNE*

Avatar hocha la tête.

- Ainsi, dit-il, vous avez travaillé treize longues années à rendre un idiot immortel ?
- La Science, dirent-ils, est au-dessus des jugements humains.
- En ce cas devrait-elle être aux ordres de qui la nourrit ? dit Avatar. Posez ça sur la table et aller me chercher la quadrature du cercle. Comme ça, ajouta-t-il, une fois les savants partis tout perplexes, pendant ce temps-là ils ne feront pas de mal.

  Et qu'est-ce que je vais en faire, dit-il en tournant dans sa main le flacon. Car enfin, qui voudra sauver sa vie ne la perdra-t-il pas ?

  Et d'un bras ferme il expédia à travers la fenêtre la sombre fiole, qui avec un joyeux plouc tomba au fond des douves.

Christiane Rochefort, *Archaos ou le jardin étincelant*

Remerciements

En premier lieu je souhaite remercier Alain et Jean-Baptiste.

Alain, c'est ta réputation d'enseignant qui m'a conduite à aller te trouver pour te proposer d'encadrer mon stage de fin de licence : sans doute le début de tout. Depuis, je n'ai eu cesse de constater ce sens du contact, cet altruisme et cette générosité dont tes anciens étudiants me parlaient. Tu m'as transmis goût de la rencontre, et avec, le plaisir d'enseigner comme le désir de parcourir le monde.

Jean-Baptiste, je regrette que nous n'ayons pas eu davantage le temps de mettre en place les nombreux projets dont nous avons parlés. Travailler sur l'application Android pour les Observatoires Photographiques du Paysage est l'un de mes meilleurs souvenirs de cette thèse. Je te souhaite d'aller loin dans ce jeu d'équilibriste où tu maries avec brio géographie, pédagogie et informatique pour donner corps à des réalisations atypiques.

Je remercie mes collègues de TCI : Adrian, Christophe et Denis pour leur gentillesse, leur accueil, leurs suggestions, nos discussions. Il en va de même pour toutes les personnes de l'IRIT, que nous nous soyons rencontrées lors des cours donnés ou au travers du conseil de Laboratoire, ainsi que pour les membres du Labex CIMI. J'ai une pensée toute spéciale pour les membres du groupe Filles et Informatiques (même lorsqu'ils étaient garçons et mathématiciens) qui ont été une véritable bouffée d'air. Enfin, il y a ces doctorants et stagiaires avec qui j'ai partagé un peu plus que des échanges scientifiques, ceux qui au fil du temps sont devenus des amis. Arturo, Anahid, Camille, Clément, Éléonore, Florent, Damien, Lisa, Ophélie, Roman, Romain, Sandra et Tom, lorsque je me retourne sur cette période, vous faites partie de ces personnes épatantes avec qui je suis heureuse d'avoir échangé. De vous j'emporte le souvenir de nos soirées, de nos projets. Je sais qu'un chapitre de mon existence se clôture : vous avez contribué à le rendre passionnant.

Je remercie ma famille. Ma mère et mon père, qui m'ont soutenue, écouté, épaulé. Ce que je leur dois remonte bien plus loin que ma thèse. Tout enfant mérite d'être accueilli comme je l'ai été : par des personnes bienveillantes capables de croire en lui et de l'accompagner sans jamais le juger.

Merci également à Églantine. En tant que sœur tu es l'une des personnes qui me connaît le mieux. Et comme amie, tu fais partie de celles dont les remarques me laissent le plus à penser.

Merci à Corentin. C'est chouette de se découvrir un cousin de dix-huit ans, passionné et

passionnant. Quoi que tu fasses, j'ai confiance en ton intelligence et ton jugement. Fonce : tu es parti pour de grandes aventures.

Merci à mes oncles et tantes, et puis à ma grand-mère. Vous avez suivi l'affaire de loin mais vous avez été les temps de repos nécessaires à la partition.

Je remercie mes collègues actuels qui apportent Makina Corpus comme réponse éclatante à la question humaine. Ça fait plaisir de constater chaque jour que je suis allée taper à la bonne porte.

Je remercie Nadia pour nos tasses de cafés et nos discussions. J'espère que la prochaine nous la prendrons au bord de l'océan.

Je remercie Bérénice artiste, boulangère et informaticienne, parce que certaines choses ne bougent pas, qu'il suffit d'un peu de peinture ou d'une ballade pour que nous retrouvions plus de dix ans en arrière dans ce qui nous a liées adolescente.

Je remercie Anne. Je ne sais où j'en serais si je ne t'avais pas eu près de moi pour me comprendre. Nos réflexions m'apaisent, parce que nous arrivons sans difficultés à nommer puis décortiquer les choses, pour ne pas en rester prisonnières. Tu es une amie prodigieuse.

Je remercie Yohan que j'aime comme les peupliers aiment la source qui baigne leurs racines. Je ne sais où tu la puises cette douceur que tu me donnes. Simplement chaque fois que tu me souris, j'ai la certitude d'un don précieux que me fait la vie.

Pour conclure, je dédicace ce mémoire à Shaé, fauve diabolique. À te voir allongée sur mon clavier, je ne doute pas que les algorithmes ne perturbent guère ta vie féline. Mais comme l'écrit Prévert :

« Un chat, c'est quelque chose tout de même », dit une voix douce...

Resumé

La segmentation est l'un des principaux thèmes du domaine de l'analyse d'images. Segmenter une image consiste à trouver une partition constituée de régions, c'est-à-dire d'ensembles de pixels connexes homogènes selon un critère choisi. L'objectif de la segmentation consiste à obtenir des régions correspondant aux objets ou aux parties des objets qui sont présents dans l'image et dont la nature dépend de l'application visée.

Même s'il peut être très fastidieux, un tel découpage de l'image peut être facilement obtenu par un être humain. Il n'en est pas de même quand il s'agit de créer un programme informatique dont l'objectif est de segmenter les images de manière entièrement automatique.

La segmentation interactive est une approche semi-automatique où l'utilisateur guide la segmentation d'une image en donnant des indications. Les méthodes qui s'inscrivent dans cette approche se divisent en deux catégories en fonction de ce qui est recherché : les contours ou les régions.

Les méthodes qui recherchent des contours permettent d'extraire un unique objet correspondant à une région sans trou. L'utilisateur vient guider la méthode en lui indiquant quelques points sur le contour de l'objet. L'algorithme se charge de relier chacun des points par une courbe qui respecte les caractéristiques de l'image (les pixels de part et d'autre de la courbe sont aussi dissemblables que possible), les indications données par l'utilisateur (la courbe passe par chacun des points désignés) et quelques propriétés intrinsèques (les courbes régulières sont favorisées).

Les méthodes qui recherchent les régions groupent les pixels de l'image en des ensembles, de manière à maximiser la similarité en leur sein et la dissemblance entre les différents ensembles. Chaque ensemble correspond à une ou plusieurs composantes connexes et peut contenir des trous. L'utilisateur guide la méthode en traçant des traits de couleur qui désignent quelques pixels appartenant à chacun des ensembles Si la majorité des méthodes ont été conçues pour extraire un objet principal du fond, les travaux menés durant la dernière décennie ont permis de proposer des méthodes dites multiclasses, capables de produire une partition de l'image en un nombre arbitraire d'ensembles.

La contribution principale de ce travail de recherche est la conception d'une nouvelle méthode de segmentation interactive multiclasse par recherche des régions. Elle repose sur la modélisation du problème comme la minimisation d'une fonction de coût pouvant être représentée par

un graphe de facteurs. Elle intègre une méthode de classification par apprentissage supervisé assurant l'adéquation entre la segmentation produite et les indications données par l'utilisateur, l'utilisation d'un nouveau terme de régularisation et la réalisation d'un prétraitement consistant à regrouper les pixels en petites régions cohérentes : les superpixels.

L'utilisation d'une méthode de sur-segmentation produisant des superpixels est une étape clé de la méthode que nous proposons : elle réduit considérablement la complexité algorithmique et permet de traiter des images contenant plusieurs millions de pixels, tout en garantissant un temps interactif.

La seconde contribution de ce travail est une évaluation des algorithmes permettant de grouper les pixels en superpixels, à partir d'un nouvel ensemble de données de référence que nous mettons à disposition et dont la particularité est de contenir des images de tailles différentes : de quelques milliers à plusieurs millions de pixels. Cette étude nous a également permis de concevoir et d'évaluer une nouvelle méthode de production de superpixels.

Mots-clés : segmentation, segmentation interactive, sur-segmentation, superpixels, SVM, graphe de facteurs.

Abstract

Image segmentation is one of the main research topics in image analysis. It is the task of researching a partition into regions, *i.e.*, into sets of connected pixels, meeting a given uniformity criterion. The goal of image segmentation is to find regions corresponding to the objects or the object parts appearing in the image. The choice of what objects are relevant depends on the application context.

Manually locating these objects is a tedious but quite simple task. Designing an automatic algorithm able to achieve the same result is, on the contrary, a difficult problem.

Interactive segmentation methods are semi-automatic approaches where a user guide the search of a specific segmentation of an image by giving some indications. There are two kinds of methods : boundary-based and region-based interactive segmentation methods.

Boundary-based methods extract a single object corresponding to a unique region without any holes. The user guides the method by selecting some boundary points of the object. The algorithm search for a curve linking all the points given by the user, following the boundary of the object and having some intrinsic properties (regular curves are encouraged).

Region-based methods group the pixels of an image into sets, by maximizing the similarity of pixels inside each set and the dissimilarity between pixels belonging to different sets. Each set can be composed of one or several connected components and can contain holes. The user guides the method by drawing colored strokes, giving, for each set, some pixels belonging to it. If the majority of region-based methods extract a single object from the background, some algorithms, proposed during the last decade, are able to solve multi-class interactive segmentation problems, *i.e.*, to extract more than two sets of pixels.

The main contribution of this work is the design of a new multi-class interactive segmentation method. This algorithm is based on the minimization of a cost function that can be represented by a factor graph. It integrates a supervised learning classification method checking that the produced segmentation is consistent with the indications given by the user, a new regularization term, and a preprocessing step grouping pixels into small homogeneous regions called superpixels.

The use of an over-segmentation method to produce these superpixels is a key step in the proposed interactive segmentation method : it significantly reduces the computational complexity and handles the segmentation of images containing several millions of pixels, by keeping the

execution time small enough to ensure comfortable use of the method.

The second contribution of our work is an evaluation of over-segmentation algorithms. We provide a new dataset, with images of different sizes with a majority of big images. This review has also allowed us to design a new over-segmentation algorithm and to evaluate it.

Keywords : segmentation, interactive segmentation, over-segmentation, superpixels, SVM, factor graph.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

Le point de départ des travaux exposés dans ce mémoire est l'adoption par le Conseil de l'Europe, en octobre 2000, de la convention européenne du paysage. Ce traité invite chaque état signataire à considérer le paysage comme un patrimoine commun. Il affirme l'utilité sociale de ce dernier pour le bien-être individuel comme collectif et aborde sa place dans la qualité du cadre de vie, que celui-ci soit urbain ou rural, remarquable ou anodin, préservé ou dégradé.

La validation de la convention européenne du paysage par le Conseil de l'Europe s'est accompagné de la mise en place d'observatoires photographiques du paysage, afin de disposer d'un ensemble de données permettant de réfléchir sur l'intégration du paysage dans les politiques d'aménagement du territoire. En France, ces observatoires prennent la forme de fonds photographiques numérisées, organisés en séries, chaque série étant rattachée à un point de vue geo-référencé qui est re-photographié régulièrement. Un protocole strict assure que les photographies d'une même série pourront être facilement comparées. Un intervalle de temps relativement long (quelques mois) sépare chaque prise de vue.

L'examen de ces fonds par des experts (géographes, personnes en charge de l'aménagement du territoire, etc.) permet d'analyser l'évolution d'un paysage lors d'un événement spécifique (construction d'un pont, chantier d'une autoroute) ou d'étudier la dynamique de zones particulières (les territoires périurbains par exemple). Leur mise à disposition pour le grand public participe à la valorisation du patrimoine paysager et offre un outil aux citoyens pour s'impliquer davantage dans les actions publiques pouvant influencer leur cadre de vie.

La création, l'enrichissement et la consultation des données d'un observatoire photographique du paysage fait intervenir de nombreux outils informatiques. A minima, un observatoire photographique du paysage donne lieu à la mise en place d'un site web, par le biais duquel l'ensemble des citoyens peuvent accéder aux différentes séries de photographies. La mise en place de traite-

FIGURE 1.1 – Photographies extraites de l'observatoire photographique du paysage mis en place lors de la construction de l'autoroute A89.

ments plus fins, capables d'extraire des informations pertinentes de ce catalogue d'images offrent de nombreux défis dans le domaine de la vision par ordinateur.

1.1 Vision par ordinateur

Olivier Faugeras [12] définit la vision par ordinateur comme la discipline se focalisant sur les quatre problèmes suivants :

- quelles informations peuvent être extraites à partir des données enregistrées par un capteur visuel (caméra, appareil photographique, etc.) ?
- comment ces informations sont-elle extraites ?
- comment doivent-elles être représentées ?
- à quel point peuvent-elles être utilisées par un système robotique pour réaliser une tâche donnée ?

Actuellement les applications de ce domaine débordent largement de la robotique et la vision par ordinateur peut être vue comme la construction de modèles algorithmiques possédant des propriétés semblables à celle de la vision humaine. Ainsi, à l'aide d'un ensemble de connaissances *a priori*, l'enjeu consiste à fournir une interprétation de l'information contenu dans une ou plusieurs images. Concrètement, il s'agit d'extraire des primitives visuelles de l'image, de proposer une représentation des connaissances et de mettre l'une et l'autre en correspondance afin de fournir une interprétation fiable et rapide de la scène.

Les problèmes classiques en vision par ordinateur sont, par exemple, les procédés de contrôle de pièce dans la robotique industrielle, la navigation pour les véhicules autonomes, la détection d'événements, la modélisation 3D d'objets ou d'environnements, la reconnaissance d'objets, etc.

1.2 Application dans le cadre des observatoires photographiques du paysage

Dans le contexte des observatoires du paysage, le capteur visuel peut être un smartphone, un appareil photographique grand public ou un appareil photographique professionnel. Les données obtenues sont des photographies couleur de grande taille. Ces images sont généralement de bonne qualité : elles sont peu bruitées et les objets qu'elles contiennent sont nets. La figure 1.1 montre trois images extraites d'une même série.

Nos travaux concernent l'identification et la localisation des objets présents dans ces photographies. Nous souhaitons attribuer à chaque pixel un label indiquant la catégorie d'objet à laquelle il appartient (ciel, herbe, route, etc.). Parce qu'ils permettent d'extraire les régions correspondant aux éléments d'une image et de leur attribuer une signification, les algorithmes visant à réaliser de tels traitements appartiennent au domaine de *la segmentation sémantique*. Les méthodes produites au sein de cette thématique de recherche peuvent être entièrement automatiques ou bien semi-automatiques.

Nous avons choisi de nous intéresser aux méthodes semi-automatiques, dans la perspective de créer un outil permettant d'enrichir les données d'un observatoire photographique du paysage. Cet outil intervient au sein d'une application plus complexe, capable de guider un utilisateur vers le point à re-photographier le plus proche puis de l'aider à reprendre une photographie à l'identique. Avant d'être envoyé à un serveur stockant les séries de photographies, cette dernière peut alors être segmentée de manière à ce que les éléments importants qu'elle contient soit documentés et localisés précisément dans l'image.

Formulé différemment, nous cherchons un moyen efficace de sélectionner chacun des objets visibles et de lui attribuer un label correspondant à des catégories sémantiques propres à l'observatoire. Ce type de logiciel est appelé *outil de segmentation interactive*.

1.3 Segmentation interactive

Une méthode de segmentation interactive est un outil informatique capable de sélectionner un ou plusieurs objets au sein d'une image à partir d'indications données par un utilisateur. Le domaine d'application le plus courant de ce type d'algorithme est celui des logiciels de manipulation d'images, dont les deux plus connus sont sans doute Gimp [1] et Photoshop [2].

Ce type d'algorithme se situe au croisement de deux thématiques : celle de la vision par ordinateur et celle de l'étude des interactions entre l'homme et la machine.

Du point de vue de la vision par ordinateur, un accent particulier sera mis sur la formulation du problème de la segmentation interactive et sur les hypothèses qu'elle émet vis-à-vis du résultat recherché. Par exemple, certaines méthodes de segmentation interactive sont conçues pour extraire un unique objet du fond. Au contraire, d'autres algorithmes s'intéressent à la sélection simultanée de plusieurs objets, ce qui implique une augmentation significative de l'espace des

1. http://gimp.org
2. http://www.photoshop.com

solutions possibles. D'autres problématiques interviennent également de manière récurrente :

- quelles primitives visuelles utiliser ? Par exemple, faut-il travailler directement au niveau du pixel ou rechercher des primitives visuelles plus complexes, telles que des groupes de pixels ?
- les indications données par l'utilisateur doivent-elles être considérées comme fiables ou faut-il envisager qu'elles contiennent des erreurs ?
- quel est le coût pour extraire un type d'information par rapport à son bénéfice dans la recherche d'un résultat ? Par exemple, si l'accès à la couleur d'un pixel est immédiat, analyser la texture présente dans son voisinage est une opération plus complexe, dont nous pouvons interroger la pertinence.
- est-il possible de trouver une solution exacte au problème posé ou faut-il se contenter d'une approximation ?
- quelles approches permettent une recherche rapide de cette solution, qu'elle soit exacte ou approchée ?

L'étude des interactions entre l'utilisateur et le programme mettra l'accent sur l'ergonomie de la méthode de segmentation interactive, se demandant par exemple :

- quelles modalités permettent à l'utilisateur de donner les indications nécessaires à la méthode ?
- pour une modalité donnée, quel est son degré de pénibilité ?
- l'impact de ces indications sur le résultat obtenu est-il facilement prévisible par l'utilisateur ?
- est-il facile d'apprendre à se servir de la méthode proposée ?
- le temps nécessaire à la méthode pour obtenir un résultat à partir des indications en permet-il une utilisation fluide ?

1.4 Problématique

Le problème que nous cherchons à résoudre est le suivant : soit une photographie couleur et un ensemble de pixels attribués à différentes catégories par un utilisateur ; comment associer de manière fiable et efficace chaque pixel de l'image à l'une des catégories demandées par l'utilisateur et ce, de façon à ce que la segmentation sémantique produite soit cohérente tant vis-à-vis du contenu de l'image que des indications fournies par l'utilisateur ?

Nous ne souhaitons pas nous limiter aux seules photographies des observatoires du paysage et cherchons à concevoir un outil de segmentation interactive qui pourrait constituer une amélioration intéressante par rapport à ceux utilisés actuellement dans les logiciels de manipulation d'images. Ainsi, nous n'avons aucune connaissance *a priori* sur le contenu des photographies ni sur les objets que l'utilisateur souhaitera sélectionner. En d'autres termes, seules les indications données par l'utilisateur permettent d'apprendre les caractéristiques des catégories d'objets. Cet

apprentissage est propre à chaque photographie et doit être recommencé pour chaque nouvelle image.

Nous ne nous donnons pas de limite sur le nombre d'objets à sélectionner et nous considérons que ce dernier varie d'une image à l'autre. Nous nous intéressons à la conception d'un algorithme itératif, où l'utilisateur peut corriger de manière intuitive les erreurs contenues dans le résultat obtenu, afin de converger rapidement vers la sélection qu'il désire. Nous supposons également que les indications données par l'utilisateur sont majoritairement fiables.

Enfin, l'une des contraintes importantes de nos travaux concerne la rapidité de la méthode que nous proposons. Dans le contexte des observatoires photographiques du paysage comme dans celui d'un logiciel de manipulation d'images, les photographies à traiter contiennent plusieurs millions de pixels. Le caractère interactif des algorithmes de segmentation interactive impose qu'un résultat soit fournit rapidement à l'utilisateur. Notre choix des primitives visuelles et de la manière dont nous les utilisons doit garantir une faible complexité algorithmique.

1.5 Organisation du mémoire

Ce mémoire comprend sept chapitres.

Chapitre 1 – Introduction –

Nous présentons les motivations à l'origine des travaux décrits dans ce mémoire.

Chapitre 2 – Segmentation interactive –

Nous présentons le domaine de la segmentation interactive et nous proposons une catégorisation des méthodes appartenant à cette thématique : les méthodes de binarisation interactive par recherche des contours, les méthodes de binarisation interactive par recherche des régions et les méthodes de segmentation interactive multiclasse. Pour chaque catégorie, nous présentons quelques algorithmes de référence.

Chapitre 3 – Algorithme de segmentation interactive Superpixel α-Fusion –

Nous proposons une nouvelle méthode de segmentation interactive multiclasse $S\alpha F$, de l'anglais « *Superpixel α-Fusion* ». Nous commençons par formuler le problème de la segmentation interactive comme la minimisation d'une fonction de coût. Cette fonction pouvant être factorisée, nous nous ramènerons à un problème d'optimisation dans un graphe de facteurs. Un graphe de facteurs est un graphe biparti où les sommets sont séparés en deux ensembles : les variables et les facteurs. Dans notre cas, les variables correspondent à de petits groupes connexes et homogènes de pixels : les superpixels. Les facteurs correspondent aux sous-fonctions. Nous verrons comment les valeurs numériques de ces sous-fonctions peuvent être obtenues à l'aide d'une méthode de classification.

Chapitre 4 – Évaluation des méthodes de sur-segmentation –

Produire une sur-segmentation d'une image consiste à trouver une partition de ses pixels en petits ensembles connexes (les superpixels), dont la taille est nettement inférieure à celle des objets. Pour la méthode que nous proposons, l'obtention d'une sur-segmentation est une étape clé. Nous avons donc réalisé une évaluation rigoureuse des algorithmes de sur-segmentation. Si la principale motivation de ce travail reste de sélectionner l'algorithme le plus approprié pour la méthode $S\alpha F$, les conclusions que permettent de tirer cette évaluation dépassent largement le cadre de la segmentation interactive.

Chapitre 5 – Évaluation de l'algorithme $S\alpha F$ –

Ce chapitre concerne l'évaluation de $S\alpha F$. Nous comparons ses performances à celles des algorithmes de l'état de l'art décrits dans le chapitre 2. Puis nous nous intéressons aux propriétés de $S\alpha F$, notamment celles concernant son ergonomie. Nous concluons par deux applications de $S\alpha F$: comme greffon pour le logiciel Gimp et comme outil d'annotation pour l'enrichissement d'un observatoire photographique du paysage.

Chapitre 6 – Sur-segmentation adaptative par fusion de superpixels –

Nous terminons ce mémoire par la proposition d'un nouvel algorithme de sur-segmentation, $ASARI$, de l'anglais « *Adaptive Superpixel Algorithm with Rich Information* ». Ce dernier permet d'obtenir une précision supérieure ou égale à celles des méthodes de l'état de l'art, avec un nombre inférieur de superpixels. Par ailleurs, les superpixels produits sont homogènes au sens de la couleur et au sens de la texture.

L'intégration de cet algorithme au sein de $S\alpha F$ constitue une perspective intéressante pour améliorer ses résultats. Elle requiert cependant :

- d'optimiser l'implémentation d'$ASARI$, les temps d'exécution de ce dernier étant encore trop importants pour l'intégrer de manière bénéfique dans une méthode de segmentation interactive ;
- de réaliser des tests approfondis. L'une des particularités des superpixels produits par $ASARI$ étant d'intégrer une information de texture, au delà d'une simple évaluation montrant qu'$ASARI$ permet d'améliorer les performances de $S\alpha F$, il convient de vérifier si cette information de texture peut ou non avoir une influence positive sur la qualité des segmentations réalisées par $S\alpha F$.

Ces travaux restent à mener.

Chapitre 7 – Conclusion –

Nous terminons par un résumé des contributions présentées et par quelques prolongements possibles de nos travaux.

Chapitre 2

Segmentation interactive

2.1 Introduction

Ce chapitre traite de la segmentation interactive. Comme son nom l'indique, ce domaine de recherche est rattaché à celui de la segmentation. Il comprend un ensemble de méthodes semi-automatiques, où l'utilisateur fournit les indications nécessaires à l'obtention d'un résultat.

Afin de bien en comprendre les enjeux, nous commencerons par quelques rappels sur la segmentation qui nous permettront de définir la segmentation interactive. Nous nous intéresserons ensuite aux différentes pistes explorées jusqu'à présent et nous présenterons quelques algorithmes clés de l'état de l'art.

2.1.1 Segmentation

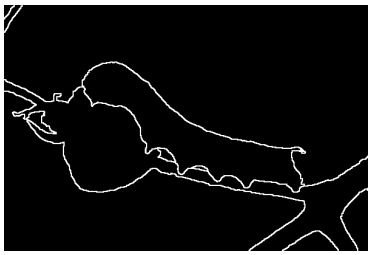
La segmentation est une étape clé pour un grand nombre de méthodes d'analyse d'images. Son but consiste à produire une partition des pixels de l'image en composantes connexes, selon des critères prédéfinis (homogénéité en termes de couleurs, de texture, etc.). Ces composantes connexes sont nommées régions et correspondent à des objets ou à des parties d'objets, formant un pavage de l'image en primitives visuelles de haut niveau. La figure 2.1 contient un exemple de segmentation. Les pixels de l'image originale (figure 2.1a) sont groupés en une dizaine de régions (figure 2.1b). La figure 2.1c représente les contours de ces régions, où apparaissent en blanc les pixels ayant un de leurs voisins qui appartient à une autre région.

Trois hypothèses guident les travaux menés en segmentation :

- l'homogénéité à l'intérieur de chaque région en termes de couleur ou de texture doit être maximisée ;

(a) Image originale.

(b) Segmentation de cette image : une couleur par région.



(c) Contours des régions de cette segmentation.

FIGURE 2.1 – Exemple de segmentation.

- les similitudes entre les différentes régions doivent être minimisées ;
- les pixels voisins appartenant à des régions différentes doivent être aussi dissemblables que possible.

2.1.2 Segmentation sémantique

Tandis que la segmentation se contente de produire un pavage de l'image en régions, la segmentation sémantique réalise une partition des pixels en ensembles qui correspondent à des catégories d'objets. Comme un même objet peut correspondre à plusieurs régions ou plusieurs objets de la même catégorie être présents dans l'image, les ensembles ne sont pas nécessairement des composantes connexes. Si le problème de la segmentation peut être vu comme la recherche des frontières des objets présents dans une photographie, celui de la segmentation sémantique consiste à localiser et identifier les objets qui la composent c'est-à-dire comme la résolution d'un problème de classification, avec une classe par catégorie d'objets. La figure 2.2b contient une segmentation sémantique associée à la figure 2.2a. Les pixels se répartissent en trois ensembles : le premier correspondant à une table en bois (en jaune), le second à la chenille (en rose) et le troisième à la branche de tomate (en bleu).

(a) Image originale.

(b) Segmentation sémantique de cette image. Trois classes sont identifiées : le bois (en jaune), la chenille (en rose) et la tomate (en bleu).

FIGURE 2.2 – Exemple de segmentation sémantique.

L'utilisation des réseaux de neurones convolutifs (CNN, de l'anglais « *Convolutional Neural Networks* ») et de l'apprentissage profond ont conduit à des avancées significatives dans le domaine la segmentation sémantique [14, 16, 28]. Toutefois, malgré la qualité impressionnante des résultats obtenus, ce type de méthode ne peut être guidée que lors de l'étape d'apprentissage [25].

L'un des inconvénients les plus évidents de cette contrainte réside dans le fait que lorsque la méthode produit une segmentation erronée, il n'est pas possible de la corriger sans recommencer l'apprentissage du CNN, lequel nécessite une quantité de données d'apprentissage importante et de nombreuses heures de calculs. Par ailleurs, le CNN est entraîné à reconnaître un nombre fini de classes, définies lors de l'apprentissage. Pour introduire une nouvelle classe, il faut recommencer l'étape d'apprentissage.

La résolution de ces deux problèmes peut passer par la recherche de méthodes semi-automatiques, intégrant de manière plus souple l'intervention d'un utilisateur. Elle correspond au domaine de la segmentation interactive.

2.1.3 Segmentation interactive

Le principe de la segmentation interactive consiste guider la recherche d'une segmentation particulière en intégrant quelques indications fournies par un utilisateur. Le fonctionnement général d'une méthode de segmentation interactive est illustré par la figure 2.3. Une photographie est donnée à l'utilisateur qui décide des entités qu'il souhaite sélectionner. Notons que l'utilisateur peut choisir ou non de produire une segmentation sémantique. Dans certains cas, il souhaitera sélectionner ensemble tous les objets d'une même catégorie (par exemples, tous les arbres). Dans d'autres, il pourra ne vouloir sélectionner qu'un seul objet, appartenant à une catégorie représentée plusieurs fois dans l'image (par exemple, un arbre spécifique).

Les indications que l'utilisateur donne permettent à la méthode de produire un premier résultat. Le plus souvent ce dernier ne correspond pas exactement aux attentes de l'utilisateur, qui vient ajouter ou supprimer des indications, déclenchant la recherche d'une deuxième segmentation. Ce processus est répété jusqu'à satisfaction des objectifs de l'utilisateur.

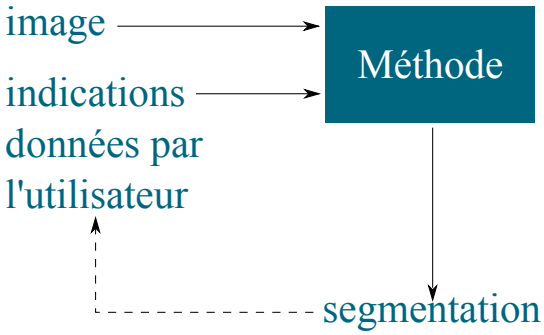


FIGURE 2.3 – Processus, pouvant être itératif, d'utilisation d'une méthode de segmentation interactive.

Soit I une image. Soit $\mathbb{P} = \{p_1, \cdots, p_{N_I}\}$ l'ensemble des N_I pixels de I. Soit $\Lambda = \{\lambda_1, \cdots, \lambda_{N_\Lambda}\}$

un ensemble de N_Λ labels indexant chaque classe, avec une classe par entité que l'utilisateur souhaite sélectionner. Les indications données par l'utilisateur prennent la forme d'un ensemble $G = \{g_1, \cdots, g_{N_G}\}$ dont chaque élément g_i est un couple de valeurs (p_i, λ_j), tel que $p_i \in \mathbb{P}$ et $\lambda_i \in \Lambda$. Réaliser une segmentation interactive de I consiste à associer à chaque élément de $\mathbb{P}$ un unique élément de Λ, en garantissant le maximum de cohérence :

- avec le contenu de l'image ;
- entre les pixels attribués à une classe et les indications données par l'utilisateur.

La communication de l'utilisateur vers la méthode s'effectue par le biais des indications (l'ensemble G). Certaines méthodes de segmentation interactive considèrent ces indications comme parfaitement fiables [4, 45], d'autres comme pouvant contenir des erreurs [6, 35]. La communication de la méthode vers l'utilisateur est assurée par l'affichage d'une segmentation de l'image.

Ces deux caractéristiques imposent, entre autres :

- que l'impact d'une indication sur le résultat soit facilement prévisible par l'utilisateur, afin qu'il puisse guider efficacement la méthode ;
- que les modalités d'interaction permettant d'ajouter ou de retirer une indication soient faciles d'utilisation ;
- que le temps nécessaire pour obtenir une segmentation à partir d'une image et d'un ensemble d'indications demeure suffisamment court pour permettre une interaction fluide (généralement nous considérerons que cette durée ne peut excéder quelques secondes).

Il s'avère intéressant de mettre ce dernier point en relation avec le contexte d'application des méthodes de segmentation interactive. Actuellement, ces dernières sont employées afin de sélectionner des objets au sein d'une photographie pour leur appliquer des traitements particuliers ou pour les intégrer au sein d'un collage, dans le domaine artistique ou dans celui du design. Les images à traiter comportent souvent plusieurs millions de pixels, ce qui soulève la question du passage à l'échelle des algorithmes de segmentation interactive. Enfin, si actuellement la majorité de ces méthodes sont implémentées pour des ordinateurs, le remplacement progressif de ce type de machine par des smartphones auprès du grand public requiert d'envisager des algorithmes capables de passer à l'échelle malgré des contraintes matérielles importantes (taille de la mémoire vive, capacité du processeur, puissance de la carte graphique, etc.).

2.1.4 Catégorisation

Durant ces deux dernières décennies, le domaine de la segmentation interactive s'est enrichi de nombreux algorithmes. Dans ce mémoire, nous proposons de les classer en trois catégories :

- **Binarisation interactive par recherche des contours -** Ces algorithmes déterminent pour chaque pixel s'il appartient ou non au contour de l'objet d'intérêt. Il s'agit donc d'un problème de classification à deux classes, λ_C pour les pixels du contour, $\lambda_{\neg C}$ pour les autres. L'utilisateur indique quelques pixels de la classe λ_C.
- **Binarisation interactive par recherche des régions -** Ces algorithmes groupent les pixels de manière à obtenir deux ensembles homogènes. Ces ensembles correspondent à une

ou plusieurs composantes connexes. Il s'agit là aussi d'un problème de classification à deux classes avec, d'une part, les pixels de l'objet principal (λ_O) et, d'autre part, ceux du fond (λ_F). L'utilisateur attribue quelques pixels à chacune des deux classes.

- **Segmentation interactive multiclasse -** Ces algorithmes correspondent à une généralisation de la catégorie précédente, où le nombre de classes n'est plus limité à deux.

2.2 Lexique et notations

2.2.1 Lexique

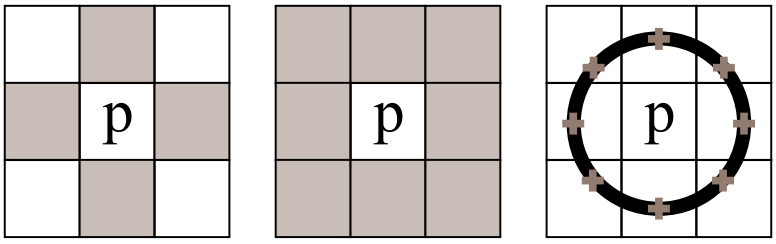
Par la suite, nous utiliserons les termes :

- **germe**, pour désigner le pixel attribué à une classe par l'utilisateur (un germe correspond à une indication) ;
- **classe**, pour désigner une entité (un objet ou une catégorie d'objets) à laquelle certains pixels seront assignés à l'issue du processus de segmentation ;
- **segmentation**, pour désigner le résultat produit par une méthode de segmentation interactive à chaque itération ;
- **région**, pour désigner un ensemble de pixels connexes appartenant à la même classe.

2.2.2 Notations

Soit I une image.

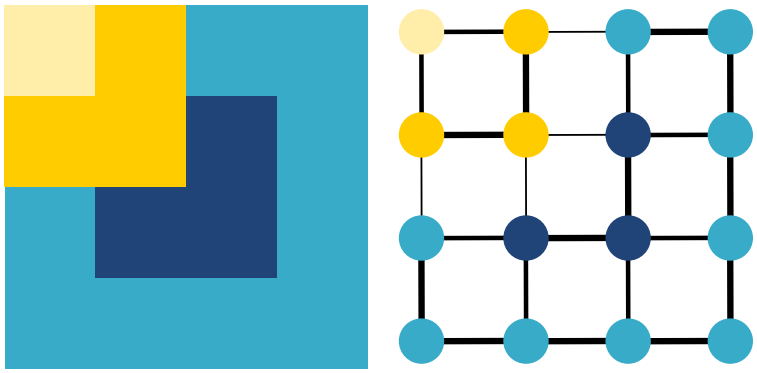
- **L'ensemble** $\mathbb{P} = \{p_1, \cdots, p_{N_I}\}$ contient les pixels de I.
- **La fonction** $\mathcal{F}_{\text{voi}}(p)$ donne l'ensemble des pixels voisins du pixel p. La figure 2.4 décrit les trois principaux systèmes de voisinage : les quatre voisins (nord, est, sud, ouest), les huit voisins (nord, nord-est, est, sud-est, sud, sud-ouest, ouest, nord-ouest) et le voisinage circulaire qui comprend les N_{voi} voisins répartis régulièrement sur le cercle de rayon r ayant pour centre le pixel p. Dans ce dernier cas, le niveau de gris d'un voisin est obtenu par interpolation des niveaux de gris environnants.



(a) 4-voisinage. (b) 8-voisinage. (c) Voisinage circulaire.

FIGURE 2.4 – Les principaux systèmes de voisinage pour un pixel.

- **Le graphe** $\mathcal{G} = < V, E >$ est une représentation de I, où $V = \{v_1, \cdots, v_{N_I}\}$ est un ensemble de N_I sommets correspondant aux pixels de I et E un ensemble d'arêtes reliant les sommets v_i et v_j si et seulement s'ils correspondent à des pixels voisins. Un exemple d'un tel graphe est donné sur la figure 2.5.
- **L'arête** $e_{i,j}$ relie les sommets v_i et v_j, correspondant respectivement aux i^{e} et j^{e} pixels.
- **Le poids** $w_{i,j}$ est le réel associé à l'arête $e_{i,j}$.
- **Une segmentation** $S = \{s_1, \cdots, s_{N_S}\}$ de I consiste en une partition de $\mathcal{G}$ en N_S composantes connexes.
- **L'ensemble** $\Lambda = \{\lambda_1, \cdots, \lambda_{N_\Lambda}\}$ contient les labels associés aux N_Λ classes.
- **Les germes donnés par l'utilisateur** sont représentés sous la forme d'un ensemble $G = \{g_1, \cdots, g_{N_G}\}$, où chaque élément g_i est un couple de valeurs (p_i, λ_j), tel que $p_i \in \mathbb{P}$ et $\lambda_i \in \Lambda$.
- **La fonction de coût** $\mathcal{F}(I, S, G)$ permet d'évaluer la pertinence d'une segmentation S de I, par rapport aux caractéristiques intrinsèques de I et aux germes G.



(a) Réprésentation d'une image couleur de 4×4 pixels.

(b) Graphe construit à partir de l'image. Nous considérons ici que les pixels sont voisins au sens du 4-voisinage. L'épaisseur des arêtes est proportionnelle au degré de similarité entre les pixels.

FIGURE 2.5 – Représentation d'une image par un graphe.

2.3 Méthodes de binarisation interactive par recherche des contours

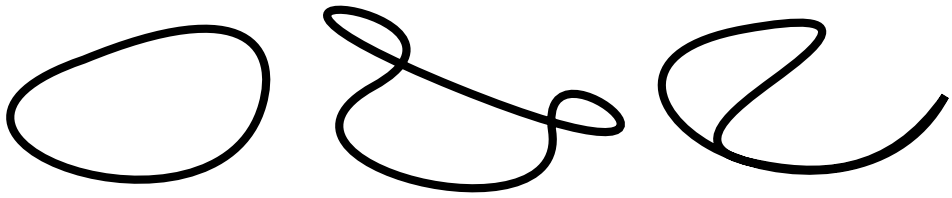
2.3.1 Formulation du problème

Les méthodes de binarisation interactive par recherche des contours séparent un objet du fond en retrouvant les pixels correspondant à son contour. Il s'agit d'un problème de classification à deux classes, λ_C si le pixel appartient au contour, $\lambda_{\neg C}$ sinon. Les germes sont quelques uns des pixels du contour. L'ensemble G correspond donc à des pixels de label λ_C.

Soit $\mathbb{P}_C$, l'ensemble des pixels de label λ_C dans le résultat donné par la méthode. Cet ensemble doit respecter les propriétés suivantes :

- il doit contenir l'ensemble des pixels de G ;
- il doit avoir une forte probabilité d'appartenir aux contours de I ;
- il doit former une courbe fermée (voir la figure 2.6a) ;
- cette courbe doit être aussi simple que possible et notamment éviter de se croiser (voir la figure 2.6b) ou de se chevaucher (voir la figure 2.6c).

Il n'existe pas à ce jour de méthode de binarisation interactive par recherche des contours résolvant des problèmes multiclasses : le chemin formé par les pixels de label λ_C étant fermé, la segmentation S obtenue contient uniquement deux régions, l'une correspondant à l'objet à extraire S_O et l'autre au fond S_F.



(a) Courbe simple. (b) Courbe se croisant. (c) Courbe se chevauchant.

FIGURE 2.6 – Quelques types de courbes.

2.3.2 Interaction avec l'utilisateur

La figure 2.7 illustre le type d'indications généralement associé aux méthodes de binarisation par recherche des contours. Les germes donnés par l'utilisateur sont représentés par de petits disques clairs. Le contour trouvé par la méthode est affiché sur la forme de traits blancs bordés de noir. Lorsque celui-ci s'avère erroné, l'utilisateur peut ajouter, déplacer ou supprimer certains germes.

Même s'il permet de modifier facilement la forme de la courbe, ce mode d'interaction présente deux désavantages. Tout d'abord, dans des zones où les contours n'apparaissent pas de manière nette (par exemple parce que l'objet est flou ou en raison d'un faible contraste), l'utilisateur peut être contraint de donner de très nombreux germes, ce qui rend rapidement le procédé fastidieux et frustrant. Ensuite, du fait qu'aucune erreur n'est tolérée dans les germes (ces derniers doivent être exactement sur le contour de l'objet), il demande à l'utilisateur de pouvoir les désigner avec précision. Si celle-ci est facilement obtenue à l'aide d'un périphérique tel qu'une souris ou une tablette graphique, elle devient problématique pour du matériel reposant sur un écran tactile, tel que les tablettes ou les smartphones.

2.3.3 Méthode de Mortensen *et al.*

Proposé par Mortensen *et al.*, l'algorithme des ciseaux intelligents [33] formule la détection

FIGURE 2.7 – Binarisation interactive par recherche des contours : exemple d'interaction avec l'utilisateur. Les germes donnés par l'utilisateur apparaissent sous la forme de disques blancs. Le contour trouvé par la méthode est indiqué par un trait blanc bordé de noir.

du contour d'un objet (donc des pixels de label λ_C) comme un problème de recherche du plus court chemin au sein d'un graphe. Chaque arête $e_{i,j} \in E$ est pondérée à l'aide de la mesure suivante :

$$\mathcal{F}_{sim}(i,j) = 0{,}43 f_1(j) + 0{,}43 f_2(j) + 0{,}14 f_3(i,j) \tag{2.1}$$

où

- la fonction binaire f_1 correspond à la réponse d'un détecteur de contour par passage à 0 du laplacien ;
- f_2 renvoie une valeur inversement proportionnelle à celle de la norme du gradient de I ;
- f_3 est une fonction de régularisation, favorisant les contours réguliers, en pénalisant les variations brusques de la direction du contour.

Soit I_{Lap} le résultat de la convolution d'une image I par le masque laplacien K_{Lap} :

$$K_{Lap} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.2}$$

La matrice I_{Lap} correspond à une approximation du laplacien de I dont les passages à 0 coïncident théoriquement avec des points de contour dans l'image. Mortensen *et al.* détectent un élément $I_{Lap}(i)$ comme un point de contour si au moins l'un de ses voisins est de signe différent et si sa valeur absolue est inférieure ou égale à la valeur absolue de chacun de ses voisins. Dans ce cas, f_1 retourne 0, sinon elle renvoie la valeur 1.

Soit $I_x(j)$ (respectivement $I_y(j)$) l'approximation discrète de la composante horizontale (respectivement verticale) du vecteur gradient de la fonction niveau de gris de l'image I pour le j^{e} pixel. La norme du vecteur gradient de I est obtenue en calculant :

$$Grad(j) = \sqrt{I_x^2(j) + I_y^2(j)} \tag{2.3}$$

et

$$f_2(j) = \frac{\max(Grad) - Grad(j)}{\max(Grad)}. \tag{2.4}$$

Ainsi, les valeurs retournées par les fonctions f_1 et f_2 sont d'autant plus petites que le pixel a une forte probabilité d'appartenir à l'un des contours de l'image.

La fonction f_3, quant à elle, pénalise les changements brusques au niveau de la courbe.

Soit

$$\overrightarrow{p}_{\perp} = \frac{\begin{bmatrix} I_y & -I_x \end{bmatrix}^{\top}}{\| \begin{bmatrix} I_y & -I_x \end{bmatrix}^{\top} \|} \tag{2.5}$$

le vecteur unitaire perpendiculaire à la direction du gradient. Soit $\overrightarrow{p_i}_{\perp}$ ce vecteur pour le i^{e} pixel. Soit $\overrightarrow{p_{i,j}}$ le vecteur unitaire pointant du i^{e} pixel vers le j^{e} pixel. Soit le lien bidirectionnel entre les pixels p_i et p_j :

$$f_d(i,j) = \begin{cases} \overrightarrow{p_{i,j}} & \text{si } \overrightarrow{p_i}_{\perp} \cdot \overrightarrow{p_{i,j}} \geqslant 0 \\ \overrightarrow{p_{j,i}} & \text{sinon.} \end{cases} \tag{2.6}$$

Le terme de lissage de Mortensen *et al.* est calculé de la manière suivante :

$$f_3(i,j) = \frac{2}{3\pi}\left(\operatorname{acos}\left(\overrightarrow{p_i}_{\perp} \cdot f_d(i,j)\right) + \operatorname{acos}(f_d(i,j) \cdot \overrightarrow{p_j}_{\perp}) \right). \tag{2.7}$$

Le poids de chaque fonction a été déterminé de manière empirique par Mortensen *et al.* Chaque fois que l'utilisateur indique qu'un pixel appartient au contour de l'objet, le chemin le plus court entre ce pixel et le précédent pixel sélectionné est calculé, grâce à l'algorithme de Dijkstra [10]. La courbe est donc construite au fur et à mesure.

Mortensen *et al.* proposent une variante de l'algorithme des ciseaux intelligents [34], avec une fonction de similarité plus complexe, dont les poids sont mis à jour à la volée, à partir des caractéristiques des pixels du contour précédemment détectés. Cette variante s'avère pertinente dans le cas où la différence entre l'objet à extraire et le fond est moins marquée que les différences entre certaines composantes au sein de l'une ou l'autre de ces deux classes. La figure 2.8 montre un exemple de ce cas de figure. L'objet à segmenter est le ventricule gauche, dont le contour est bien moins marqué que celui du cœur.

Une autre solution apportée à ce même problème consiste à ne chercher le plus court chemin que dans une partie du graphe. Le plus souvent, cette partie correspond à un rectangle englobant les points de départ et d'arrivé du chemin.

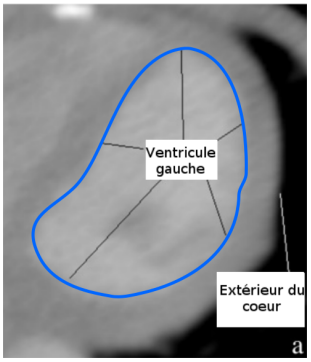


FIGURE 2.8 – Exemple d'image pour laquelle le contour de l'objet à extraire (en bleu) est moins marqué que le contour entre différents éléments du fond.

2.3.4 Méthode de Mille *et al.*

L'algorithme par combinaison de chemins géodésiques [32] est l'une des plus récentes contributions dans le domaine de la binarisation interactive par recherche des contours. Conçu par Mille *et al.*, il repose sur la recherche d'un chemin fermé C minimisant

$$\mathcal{F}_{CCG}(I, C) = f_1(C) + w_2 f_2(I, C) + w_3 f_3(I, C) \tag{2.8}$$

avec :

- f_1 une fonction de lissage, dont le résultat est un réel positif qui augmente lorsque certaines portions de la courbe se chevauchent et lorsque la courbe contient des boucles ;
- f_2 une fonction favorisant le passage de la courbe par des pixels appartenant aux contours de l'image, en effectuant la somme de l'inverse des normes des vecteurs gradients des points de la courbe ;
- f_3 une fonction qui, en utilisant les coefficients de Bhattacharyya, favorise la production de régions à l'intérieur et à l'extérieur de la courbe dont les couleurs correspondent à des distributions statistiques différentes.

À l'instar de l'algorithme des ciseaux intelligents, les germes donnés par l'utilisateur sont ordonnés et forment l'ensemble $G = \{g_0, \cdots, g_i, g_{i+1}, \cdots, g_{N_G-1}\}$, où g_{i-1} est le i^{e} germe donné par l'utilisateur. Tandis que l'algorithme de Mortensen *et al.* repose sur une approche locale, se contentant de rechercher un chemin optimal entre chaque couple $(g_i, g_{((i+1) \mod N_G)})$, la méthode de Milles *et al.* prend en compte la totalité de la courbe et recherche la *combinaison de chemins* qui minimise $\mathcal{F}_{CCG}$.

Un parcours exhaustif de l'ensemble de ces combinaisons n'étant pas possible, l'algorithme produit une approximation de la solution en réalisant une présélection de chemins probables pour chaque couple $(g_i, g_{((i+1) \mod N_G)})$. Soit $C_i^* = \{c_i^1, \cdots, c_i^m\}$ l'ensemble de ces chemins probables reliant g_i à $g_{((i+1) \mod N_G)}$. Soit une fonction $f_D(c_i^j)$ renvoyant une valeur d'autant moins élevée que le chemin passe par peu de pixels et que ces pixels ont une forte probabilité d'appartenir à un contour, une approximation de la probabilité d'appartenir à un contour étant calculée à

partir de la norme du vecteur gradient de la fonction niveau de gris de l'image I. Le chemin c_i^1 est le chemin entre g_i et$g_{((i+1) \mod N_G)}$, pour lequel la valeur de f_D est minimale. Le chemin suivant, c_i^2, est obtenu en recherchant à nouveau le plus court chemin, mais sous la contrainte que c_i^1 et c_i^2 soient parfaitement distincts, c'est-à-dire que leurs seuls pixels communs soient g_i et $g_{((i+1) \mod N_G)}$.

Même en se limitant à un ensemble discret de chemins possibles entre chaque paire de germes consécutifs, une recherche directe de la combinaison minimisant $\mathcal{F}_{CCG}$ se révèle trop coûteuse en termes de temps de calcul. Milles *et al.* proposent une heuristique où les chemins entre deux germes sont ordonnés en fonction de leur aire signée, calculée à l'aide du théorème de Green. Si nous considérons la droite D_i passant par deux germes g_i et $g_{((i+1) \mod N_G)}$ ainsi que la surface S_i^j délimitée par cette dernière et un chemin donné c_i^j, l'aire signée est calculée en soustrayant l'aire des parties de S_i^j situées au dessous de D_i à l'aire des parties de S_i^j situées au dessus. L'algorithme est initialisé avec les chemins ayant la plus faible aire signée. À chaque itération, une série de combinaisons est produite à partir de la combinaison précédente, en ne changeant qu'un seul chemin. La combinaison minimisant $\mathcal{F}_{CCG}$ est sélectionnée et le processus répété jusqu'à convergence.

2.4 Méthodes de binarisation interactive par recherche des régions

2.4.1 Formulation du problème

Les algorithmes de binarisation interactive orientés régions cherchent à produire une classification des pixels de l'image en deux catégories : l'objet à extraire (associé au label λ_O) et le fond (associé au label λ_F). L'utilisateur guide la méthode en associant quelques pixels à chacune des deux classes. Il existe donc une partition de G en deux sous-ensembles, G_O pour les germes de label λ_O et G_F pour ceux de label λ_F.

La segmentation produite par ce type d'algorithme correspond à une partition de l'image en N_R régions, N_R pouvant être supérieur à deux. Les solutions proposées pour ce problème s'articulent autour de deux grands types d'approches : la minimisation d'une fonction de coût et la fusion de régions.

Minimisation d'une fonction de coût

Les algorithmes par minimisation d'une fonction de coût reposent sur la définition d'une fonction de la forme :

$$\mathcal{F}_C(I, S, G) = \mathcal{F}_D(I, S, G) + \mathcal{F}_R(S) \tag{2.9}$$

avec :

- $\mathcal{F}_D$ un terme d'attache aux données, qui évalue l'homogénéité à l'intérieur de chaque région, les dissemblances entre les régions attribuées à des classes différentes et la cohérence entre les indications données par l'utilisateur et les pixels attribués à chacune des classes ;

- $\mathcal{F}_R$ un terme de régularisation qui permet d'éliminer les partitions en de très nombreuses régions et les régions avec des contours irréguliers.

Les termes $\mathcal{F}_D$ et $\mathcal{F}_R$ sont définis de manière à ce qu'une segmentation optimale vis-à-vis des caractéristiques de l'image et des germes corresponde à une valeur minimale de $\mathcal{F}_C$. La recherche exacte de ce minimum n'est souvent pas envisageable pour des raisons de temps de calcul et la segmentation obtenue est le résultat d'une approximation.

Les méthodes de Boykov *et al.* [4] et de Jian *et al.* [18], respectivement décrites dans les sections 2.4.3 et 2.4.4, constituent deux exemples de ce type de démarche.

Fusion de régions

Les algorithmes par fusion de régions produisent une segmentation d'une image en groupant petit à petit les pixels voisins et similaires, de manière à obtenir des ensembles cohérents.

Ce type d'algorithme nécessite une étape d'initialisation où le point de départ de chaque ensemble est donné. Dans le cas de la segmentation interactive, les germes constituent ces embryons de régions qui seront par la suite agrandis.

Soit $R = \{r_1, \cdots, r_{N_R}\}$ ces régions initiales, correspondant aux ensembles connexes de germes appartenant à la même classe. À chaque itération de l'algorithme, chacune des N_R régions est agrandie en lui ajoutant un ou plusieurs pixels voisins qui :

- n'ont été attribués à aucune région ;
- satisfont un critère de similarité avec la région.

À chaque fois qu'une région est agrandie, ses caractéristiques sont mises à jour. Lorsque deux régions assignées à la même classe comprennent des pixels adjacents, elles sont également fusionnées. Le processus s'arrête lorsque chaque pixel est attribué à une et une seule région.

L'algorithme de Salembier *et al.* [45], décrit dans la section 2.4.5, constitue un bon exemple d'une méthode de binarisation interactive par fusion de régions.

2.4.2 Interaction avec l'utilisateur

Le mode d'interaction le plus courant consiste à demander à l'utilisateur de venir tracer des traits de couleur sur chaque région, comme sur la figure 2.9 où les pixels attribués à l'objet ont été désignés par un trait jaune, tandis que ceux associés au fond sont représentés par un trait bleu.

Notons que le fait de tracer des traits n'est pas le seul moyen pouvant être utilisé pour donner les germes [15, 43].

Le principal défaut de ce type d'indications est qu'il nécessite que l'utilisateur sache quels pixels permettront à la méthode de binarisation interactive de distinguer au mieux la forme du fond. Par exemple, sur la figure 2.9, le fait qu'aucun germe n'ait été placé sur les cheveux de la personne en arrière plan risque de produire une segmentation erronée, où ces cheveux seront confondus avec le pelage de l'ornithorynque.

FIGURE 2.9 – Binarisation interactive par recherche des régions : exemple de germes donnés par l'utilisateur.

2.4.3 Méthode de Boykov *et al.*

L'algorithme par coupure de graphe de Boykov *et al.* [4] cherche la segmentation S^* qui minimise $\mathcal{F}_C$ en utilisant un algorithme de maximisation de flot sur un graphe $\mathcal{G}' = < V', E' >$, créé à partir de $\mathcal{G}$, avec :

- l'ensemble de sommets $V' = V \cup \{v_S, v_P\}$, où v_S et v_P sont deux sommets spéciaux, la source et le puits ;
- l'ensemble d'arêtes $E^{'} = E \cup E_S \cup E_P$, où E_S est composé des arêtes reliant chaque sommet de V à v_S et E_P est composé des arêtes reliant chaque sommet de V à v_P.

Soit $I(p_i)$ le niveau de gris du i^{e} pixel et $P(i, \lambda)$, la probabilité pour ce pixel d'appartenir à la classe de label λ. Chaque arête $e_{i,j} \in E$ reçoit une pondération inversement proportionnelle à la distance euclidienne entre les positions des i^{e} et j^{e} pixels ainsi qu'à la différence de leurs niveaux de gris, ce qui permet de s'assurer que des pixels voisins et similaires soient regroupés dans la même région. La pondération de chaque arête $e_{i,P} \in E_S$ (respectivement $e_{i,S} \in E_P$) est inversement proportionnelle à la probabilité que le i^{e} pixel appartienne à l'objet (respectivement au fond) sachant son niveau de gris. Ces probabilités sont calculées à partir de l'analyse des distributions des niveaux de gris des germes attribués à chacune des classes. Le tableau 2.1 récapitule, pour chaque type d'arête, la fonction permettant de calculer sa pondération.

Une manière intuitive de produire une binarisation de I à partir de sa représentation $\mathcal{G}'$ consiste à retirer les arêtes de plus faibles pondérations jusqu'à obtenir deux composantes connexes, l'une contenant la source et l'autre le puits : dans ce cas, il s'agit de trouver une coupe minimale de $\mathcal{G}$. Or ce problème est équivalent à la recherche d'un flot maximum, problème pour lequel Boykov *et al.* proposent un algorithme permettant de trouver efficacement une solu-

Arête	**Type d'arête**	**Pondération**
$e_{i,j}$	$e_{i,j} \in E$	$\exp\left(\frac{-(I(p_i)-I(p_j))^2}{2\sigma^2}\right)$
$e_{i,S}$	le i^{e} pixel n'est pas un germe	$-\ln(P(i,\lambda_F))$
$e_{i,S}$	le i^{e} pixel est un germe de label λ_O	$+\infty$
$e_{i,S}$	le i^{e} pixel est un germe de label λ_F	0
$e_{i,P}$	le i^{e} pixel n'est pas un germe	$-\ln(P(i,\lambda_O))$
$e_{i,P}$	le i^{e} pixel est un germe de label λ_O	0
$e_{i,P}$	le i^{e} pixel est un germe de label λ_F	$+\infty$

TABLEAU 2.1 – Pondération des arêtes dans l'algorithme de Boykov *et al.* [4].

tion. Afin d'obtenir une binarisation de l'image, il suffit alors d'attribuer le label λ_O à tous les pixels rattachés à la source et le label λ_F à ceux rattachés au puits.

2.4.4 Méthode de Jian *et al.*

Contrairement aux approches précédemment décrites qui travaillent à l'échelle du pixel, l'algorithme de Jian *et al.* [18] utilise une sur-segmentation de l'image, où les pixels sont groupés en petites régions homogènes, nommées *superpixels*. Ces superpixels sont produits grâce à l'algorithme mean-shift [7], qui permet de réduire considérablement le nombre de primitives visuelles à manipuler, en créant des ensembles de pixels adjacents dont les couleurs sont similaires. Un exemple visuel du résultat de cette compression est donné sur la figure 2.10.

(a) Image originale.

(b) Superpixels produits par l'algorithme mean-shift.

FIGURE 2.10 – Compression des pixels en superpixels avec l'algorithme mean-shift.

Jian *et al.* choisissent de décrire chaque superpixel par son histogramme de couleurs normalisé. Afin de réduire la dimension du descripteur correspondant à cet histogramme, chaque canal de couleur est quantifié en 16 ensembles de taille identique.

Soit $\mathbb{S} = \{\mathbf{s}_1, \cdots \mathbf{s}_{N_\mathbb{S}}\}$ l'ensemble des descripteurs de ces superpixels. À partir de G, l'ensemble G' est obtenu. Ses éléments sont des couples de la forme $(\mathbf{s}_i, \lambda_j)$ associant aux superpixels conte-

nant des germes, le label correspondant. Les superpixels contenant à la fois des germes de label λ_O et de label λ_F sont écartés. Soit $N_{G'}$ le nombre de superpixels contenant des germes pour une seule classe : l'ensemble $\mathbb{S}$ peut alors être ordonné, de manière à ce que les $N_{G'}$ superpixels de l'ensemble G' constituent ses $N_{G'}$ premiers éléments.

Soit $\mathbb{S}_M$ l'ensemble des couples de superpixels rattachés à la même classe, c'est-à-dire que :

$$(\mathbf{s}_{i1}, \mathbf{s}_{i2}) \in \mathbb{S}_M \Rightarrow (\mathbf{s}_{i1}, \lambda_{j1}) \in G' \wedge (\mathbf{s}_{i2}, \lambda_{j2}) \in G' \wedge \lambda_{i1} = \lambda_{i2}. \tag{2.10}$$

Soit $\mathbb{S}_C$ l'ensemble des couples de superpixels rattachés à des classes différentes, c'est-à-dire que :

$$(\mathbf{s}_{i1}, \mathbf{s}_{i2}) \in \mathbb{S}_C \Rightarrow (\mathbf{s}_{i1}, \lambda_{j1}) \in G' \wedge (\mathbf{s}_{i2}, \lambda_{j2}) \in G' \wedge \lambda_{i1} \neq \lambda_{i2}. \tag{2.11}$$

Les éléments de la matrice de contraintes M_M sont définis de la manière suivante :

$$M_M(i,j) = \begin{cases} (\mathbf{s}_i - \mathbf{s}_j)(\mathbf{s}_i - \mathbf{s}_j)^\top & \text{si } (\mathbf{s}_i, \mathbf{s}_j) \in \mathbb{S}_M \\ 0 & \text{sinon.} \end{cases} \tag{2.12}$$

De même, les éléments de la matrice de contrainte M_C sont définis par :

$$M_C(i,j) = \begin{cases} (\mathbf{s}_i - \mathbf{s}_j)(\mathbf{s}_i - \mathbf{s}_j)^\top & \text{si } (\mathbf{s}_i, \mathbf{s}_j) \in \mathbb{S}_C \\ 0 & \text{sinon.} \end{cases} \tag{2.13}$$

Soient $N_\mathbb{S}$ le nombre de superpixels et M_{Id} la matrice identité de dimension $N_\mathbb{S} \times N_\mathbb{S}$. La matrice d'affinité entre les superpixels est une matrice carrée M_A de dimension $N_\mathbb{S} \times N_\mathbb{S}$, dont l'élément $M_A(i,j)$ correspond à une mesure de la similarité entre les i^e et j^e superpixels. Cette mesure est obtenue en utilisant les coefficients de Bhattacharyya :

$$M_A(i,j) = \mathcal{F}_{bha}(\mathbf{s}_i, \mathbf{s}_j) = \sum_{u=1}^{3} \sqrt{H_i(u) \cdot H_j(u)} \tag{2.14}$$

avec H_i l'histogramme de couleur normalisé pour le i^e superpixel et 3 le nombre de canaux de couleur.

Soit la matrice diagonale M_D, de dimension $N_\mathbb{S} \times N_\mathbb{S}$, telle que :

$$M_D(i,i) = \sum_{j=0}^{N_\mathbb{S}} M_A(i,j). \tag{2.15}$$

Soit la matrice :

$$A = \omega_L(M_{Id} - M_D^{-1/2} M_A M_D^{-1/2}) + \omega_\alpha M_M - \omega_\beta M_C \tag{2.16}$$

avec ω_L, ω_α, ω_β des pondérations.

La matrice A est décomposée en :

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \tag{2.17}$$

avec A_1 de dimension $N_{G'} \times N_{G'}$, A_2 de dimension $N_{G'} \times (N_{\mathbb{S}} - N_{G'})$, A_3 de dimension $(N_{\mathbb{S}} - N_{G'}) \times N_{G'}$ et A_4 de dimension $(N_{\mathbb{S}} - N_{G'}) \times (N_{\mathbb{S}} - N_{G'})$. À partir de ces contraintes et de la matrice d'affinité M_A, la structure discriminative globale de l'image est apprise et les germes sont propagés suivant l'algorithme 1.

Algorithme 1 Calcul de la structure discriminative globale d'une image

1: Calculer la matrice laplacienne $L = M_{Id} - M_D^{-1/2} M_A M_D^{-1/2}$
2: Créer la matrice laplacienne contrainte A
3: Séparer A en 4 sous-matrices
4: Trouver la matrice K_1^* qui minimise $\mathrm{tr}\left((A_1 - A_2 A_4^{-1} A_3^\top) K_1\right)$ sous la contrainte : $K_1(i,i) = 1$, avec $\mathrm{tr}(M)$ la trace de la matrice M.
5: Propager K_1^* pour obtenir $K^* = \begin{bmatrix} K_1^* & -K_1^* A_3 A_1^{-1} \\ -A_1^{-1} A_3^\top K_1^* & -A_1^{-1} A_3^\top K_1^* A_3 - A_1^{-1} \end{bmatrix}$

Les colonnes de la matrice K^* à l'issue de cette étape correspondent à de nouveaux descripteurs, avec un descripteur pour chaque superpixel. Afin de produire la segmentation finale, ils sont groupés en deux ensembles grâce à l'algorithme des k-moyennes.

2.4.5 Méthode de Salembier *et al.*

L'algorithme de Salembier *et al.* [45] utilise une segmentation hiérarchique de l'image, représentée sous la forme d'un arbre binaire de partition (ABP).

Soit $R^0 = \{r_0^0, \cdots, r_{N_I}^0\}$ une partition initiale de l'image, où chaque pixel correspond à une composante connexe r_i^0. Soit $f_d(r_i^m, r_j^n)$, une fonction calculant le degré de dissimilarité entre deux régions r_i^m et r_j^n. Les indices m et n représentent le niveau de chaque région dans l'arbre. La segmentation hiérarchique utilisée par l'algorithme de Salembier *et al.* est obtenue en fusionnant les deux régions qui minimisent f_d, jusqu'à ce qu'un critère de terminaison (par exemple un nombre de composantes connexes) soit atteint.

Un ABP est un arbre dont les feuilles correspondent aux composantes connexes de R^0 et où un nœud est ajouté dès qu'une fusion est réalisée. La figure 2.11 donne un exemple d'ABP obtenu à partir de l'image de la figure 2.5b. Dans cet exemple, chaque composante connexe est décrite par la couleur moyenne de ses pixels. La fonction f_s utilisée est la distance euclidienne entre les couleurs associées aux deux composantes connexes. Le critère d'arrêt est l'obtention d'une composante connexe regroupant tous les pixels de l'image.

Salembier *et al.* décrivent chaque région r_i^m par sa couleur c_i^m, exprimée dans l'espace CIELuv. La conversion de la couleur d'un pixel depuis l'espace RVB vers l'espace CIELuv est donnée par :

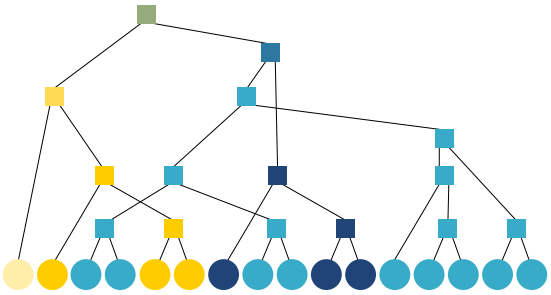


FIGURE 2.11 – ABP obtenu après fusion des pixels de l'image de la figure 2.5b.

$$\begin{bmatrix} L \\ u \\ v \end{bmatrix} = \begin{bmatrix} 116 f_{luv}(Y/Y_n) - 16 \\ 13L(\frac{4X}{X+15Y+3Z} - u_n) \\ 13L(\frac{9Y}{X+15Y+3Z} - v_n) \end{bmatrix} \tag{2.18}$$

avec Y_n, u_n et v_n les valeurs pour le blanc de référence. Les variables X, Y et Z correspondent à la conversion depuis l'espace RVB vers l'espace de CIEXYZ :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 2,7689 & 1,7517 & 1,1302 \\ 1 & 4,5907 & 0,601 \\ 0 & 0,056508 & 5,5943 \end{bmatrix} \begin{bmatrix} R \\ V \\ B \end{bmatrix} \tag{2.19}$$

et

$$f_{luv}(t) = \begin{cases} t^{1/3} & \text{si } t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{sinon.} \end{cases} \tag{2.20}$$

Lorsque deux régions r_i^m et r_j^n sont fusionnées, elles forment une nouvelle région $r_{i\cup j}^{\max(m,n)+1}$ dont le niveau dans l'arbre est égal au niveau le plus élevé des deux anciennes régions augmenté de 1. La couleur de cette nouvelle région est égale à :

$$c_{i\cup j}^{\max(m,n)+1} = \begin{cases} c_i^m & \text{si } |r_i^m| > |r_j^n| \\ c_j^n & \text{si } |r_j^n| > |r_i^m| \\ \dfrac{c_i^m + c_j^n}{2} & \text{si } |r_i^m| = |r_j^n| \end{cases} \tag{2.21}$$

avec $|r_i^m|$ la cardinalité de la région r_i^m. La fonction de dissimilarité entre deux régions est égale à :

$$f_d(r_i^m, r_j^n) = |r_i^m| \times ||c_i^m - c_{i\cup j}^{\max(m,n)+1}|| + |r_j^n| \times ||c_j^n - c_{i\cup j}^{\max(m,n)+1}|| \tag{2.22}$$

où $||.||$ désigne la norme euclidienne.

L'algorithme de Salembier *et al.* commence par propager les germes donnés par l'utilisateur des feuilles de l'ABP jusqu'à sa racine. Si deux labels différents remontent vers un même nœud, celui-ci est noté comme *instable*. Dans une seconde étape, les labels sont à nouveau propagés, de la racine vers les feuilles, en partant des nœuds qui ne sont pas marqués comme *instables*. À

l'issue de ces deux étapes, certains nœuds ou feuilles peuvent demeurer sans label. Les régions correspondant à ces nœuds reçoivent le label d'une des régions qui leur est adjacente. Si la région non classée comprend plusieurs régions adjacentes avec des labels différents, le label de la région la plus proche au sens d'une mesure de similarité lui est attribué. Salembier *et al.* n'indiquant pas de mesure de similarité particulière, McGuinness *et al.* [31], dans leurs travaux visant à évaluer différentes méthodes de binarisation interactive par recherche des régions, proposent d'utiliser la distance euclidienne entre les couleurs moyennes des régions, exprimées dans l'espace CIELuv.

2.4.6 Méthode de Friedland *et al.*

L'algorithme SIOX [15], proposé par de Friedland *et al.*, sépare un objet du fond en analysant leurs signatures de couleurs.

Soient $X = \{x_1, \cdots x_{N_X}\}$ un ensemble discret de valeurs possibles et $A = \{a_1, \cdots a_{N_A}\}$ un ensemble discret de N_A éléments, tel que $\forall i \in [1, N_A]$, $a_i \in X$. Nous notons $H_{X,A}$, l'histogramme des valeurs prisent par les éléments de l'ensemble A. Cet histogramme est une liste ordonnée de N_X entiers naturels qui sont le nombre d'occurrences de chaque valeur $x_i \in X$ dans l'ensemble A. Soit $H_{X,A}(x_i)$, ce nombre d'occurrences pour la valeur $x_i \in X$.

La fonction signature($H_{X,A}$) donne la signature de l'histogramme $H_{X,A}$, c'est-à-dire un ensemble de N_{sign} paires (w_i, m_i) telles que :

- $N_{sign} \leq N_X$;
- $w_i = x_i \Rightarrow H_{X,A}(x_i) > 0$;
- $m_i = H_{X,A}(w_i)$.

La signature peut donc être vue comme une version compressée d'un histogramme où les valeurs de X non représentées sont absentes.

Soit I une image couleur. À chaque pixel est associé un vecteur de trois réels qui correspondent aux valeurs de ce pixel pour chaque canal de couleur $[c_1, c_2, c_3]$. Soit A', un sous-ensemble des pixels appartenant à I. L'histogramme $H_{c_k,A'}$ correspond à l'histogramme des valeurs prises par les pixels de A', pour le canal de couleur c_k. Le nombre de classes pour cet histogramme est égal à N_{c_k}, le nombre de valeurs possibles pour le canal de couleur c_k. Plus ce nombre est important, plus la distinction entre les différentes couleurs est précise. Nous obtenons donc trois histogrammes ($H_{c_1,A'}$, $H_{c_2,A'}$, $H_{c_3,A'}$), avec chacun un nombre de classes qui lui est propre et qui dépend de la précision souhaitée sur chaque canal de couleur. Ces histogrammes et les signatures qui leur sont associées sont calculés pour le fond et pour la forme, à partir des germes donnés par l'utilisateur.

Chaque pixel de l'image est ensuite attribué au fond ou à l'objet à extraire en fonction de la signature à laquelle sa couleur appartient. Si la couleur du pixel n'appartient à aucune des signatures, il est attribué à la classe dont la signature contient la couleur la plus proche, au sens de la distance euclidienne.

Le nombre de classes pour les histogrammes est un paramètre fixé par l'utilisateur. Il peut varier d'une composante colorimétrique à l'autre et selon la complexité de l'image.

2.5 Méthodes de segmentation interactive multiclasse

2.5.1 Formulation du problème

Le problème de la segmentation interactive multiclasse peut se voir comme une généralisation de celui de la binarisation interactive par recherche des régions, la principale différence résidant dans le fait que le nombre de classes est variable d'une image à l'autre. L'ensemble des labels possibles devient donc : $\Lambda = \{\lambda_1, \cdots \lambda_{N_\Lambda}\}$. La segmentation obtenue correspond toujours à une partition S de l'image en régions, avec une ou plusieurs régions pouvant être attribuées à chaque classe.

L'une des approches les plus courantes pour résoudre ce type de problème consiste à trouver la segmentation S minimisant la fonction de Potts :

$$\mathcal{F}_{Potts}(I, G, S) = \omega \sum_{i=1}^{N_I} \mathcal{F}_D(p_i, \lambda_j) + \frac{1}{2} \sum_{i=1}^{N_S} \mathcal{F}_R(I, S_i) \tag{2.23}$$

où :

- F_R est un terme de régularisation
  - pénalisant les segmentations S comprenant des régions aux contours sinueux ;
  - favorisant l'alignement des contours des régions sur les contours de l'image ;
- F_D est le terme d'attache aux données évaluant la pertinence d'attribuer le label λ_j au pixel p_i, en fonction des caractéristiques de l'image I et des germes G ;
- ω est un paramètre pondérant l'influence du terme d'attache aux données.

2.5.2 Interaction avec l'utilisateur

Comme le montre la figure 2.12, le mode d'interaction avec l'utilisateur est proche de celui employé pour les méthodes de binarisation interactive par recherche des régions, seul change le nombre de couleurs disponibles, lequel n'est plus limité à deux.

2.5.3 Méthode d'Adams *et al.*

L'algorithme d'Adams *et al.* [2] fait partie des méthodes de segmentation interactive les plus anciennes. Toutefois, sa grande simplicité en fait une méthode de segmentation interactive multiclasse parmi les plus rapides.

Soit $V_G \subset V$, tel que $v_i \in V_G \Leftrightarrow \exists (p_i, \lambda_j) \in G$, l'ensemble des sommets correspondant à des germes. Soit $A = \{A_1, \cdots, A_{N_{CC}}\}$ une partition de V_G en N_{CC} composantes connexes, de manière à ce que tous les sommets rattachés à une même composante connexe correspondent à des germes de même label. Soit $\overline{V_G}$ l'ensemble complémentaire de V_G.

Adams *et al.* proposent un algorithme itératif où, à chaque étape, un sommet de $\overline{V_G}$ est attribué à l'une des composantes connexes de A, jusqu'à que l'ensemble $\overline{V_G}$ soit vide. Le sommet sélectionné doit correspondre à un pixel voisin de l'un des pixels de V_G et satisfaire un critère

FIGURE 2.12 – Segmentation interactive multiclasse par recherche des régions : exemple d'indications données par l'utilisateur.

de similarité avec l'une des composantes connexes. Lorsqu'il est attribué, il est retiré de $\overline{V_G}$ et ajouté à V_G.

La similarité entre un pixel p_i et une composante connexe A_j est estimée à partir de la distance euclidienne entre la couleur de p_i et la couleur moyenne des pixels attribués à A_j, les couleurs étant exprimées dans l'espace CIELuv.

2.5.4 Méthode de Santner *et al.*

Dans leur article [46], Santner *et al.* proposent de trouver une segmentation optimale de l'image en minimisant la fonction de Potts, définie par l'équation 2.23.

Le terme d'attache aux données, $\mathcal{F}_D$ correspond à la probabilité de chaque pixel d'appartenir à chacune des classes. Cette probabilité est obtenue en utilisant une forêt aléatoire (FA) [5], qui est une méthode d'apprentissage consistant à combiner les résultats de plusieurs arbres décisionnels.

Un arbre décisionnel est un modèle prédictif reposant sur un ensemble de règles binaires. Présentée à un nœud, la donnée à classer est orientée vers le fils droit ou le fils gauche, en fonction de ses caractéristiques. L'arrivée sur une feuille de l'arbre correspond à une prise de décision. Afin de pouvoir classer correctement une donnée, l'arbre doit être paramétré lors d'une étape d'apprentissage, à partir de données dont la classe est connue. La phase d'apprentissage consiste à déterminer pour chaque nœud un hyperplan dans un espace à m dimensions, m étant le nombre d'attributs utilisés par ce nœud, inférieur ou égal au nombre d'attributs caractérisant une donnée. Durant la phase d'exploitation, lorsqu'une donnée arrive sur un nœud, les valeurs des attributs pris en compte par ce nœud permettent de la positionner dans l'espace à m dimensions : selon sa position vis-à-vis de l'hyperplan, la donnée est envoyée vers le fils droit ou vers le fils gauche.

Dans le cas d'un classificateur de type FA, chaque arbre décisionnel est entraîné à partir d'une partie des données d'apprentissage, prélevée de manière aléatoire. En phase d'exploitation, lorsqu'une donnée est fournie à la FA, chaque arbre vote pour la classe qui lui semble la plus adéquate. Pour chacune des classes, il est alors possible de connaître le nombre d'arbres décisionnels qui ont voté en sa faveur et d'en déduire la probabilité que la donnée appartienne à cette classe.

L'algorithme de Santner *et al.* utilise les germes comme données d'apprentissage. Chaque pixel est décrit par sa couleur exprimée dans l'espace $CIELab$ et un identifiant caractérisant la texture en son voisinage, nommé motif binaire local, en anglais « *local binary pattern* » (LBP) [37].

Soit p_i un pixel de niveau de gris $I(p_i)$ et $\{I(p_i^0), \cdots, I(p_i^{P-1})\}$ les niveaux de gris des P voisins de p_i répartis régulièrement sur le cercle de centre p_i et de rayon R. L'identifiant LBP de p_i est égal à :

$$LBP_{R,P}(i) = \sum_{j=0}^{P-1} \tau(I(p_i) - I(p_i^j))2^j \tag{2.24}$$

avec τ une fonction seuil renvoyant 0 si son argument est strictement négatif, 1 sinon.

La conversion d'une couleur de l'espace RVB à l'espace CIELab est donnée par :

$$\begin{bmatrix} L \\ a \\ b \end{bmatrix} = \begin{bmatrix} 116 f_{lab}(Y/Y_n) - 16 \\ 500 f_{lab}(\frac{X}{X_n} - \frac{Y}{Y_n}) \\ 200 f_{lab}(\frac{Y}{Y_n} - \frac{Z}{Z_n}) \end{bmatrix} \tag{2.25}$$

avec X_n, Y_n et Z_n correspondant aux valeurs pour le blanc de référence, X, Y et Z le passage de l'espace RVB vers l'espace CIEXYZ, décrit dans la section 2.4.5, et

$$f_{lab} = \begin{cases} t^{1/3} & \text{si } t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{sinon.} \end{cases} \tag{2.26}$$

Pour le terme de régularisation $\mathcal{F}_R$, le calcul du gradient permet d'obtenir pour chaque pixel sa probabilité d'appartenir à un contour de l'image.

Une fois le problème de la minimisation de la fonction de Potts reformulé en problème d'optimisation convexe grâce à l'approche de relaxation convexe de Pock *et al.* [41], une approximation de la solution est trouvée en utilisant un algorithme de type *primal-dual* [3].

2.5.5 Méthode de Müller *et al.*

À l'instar de Santner *et al.* [46], Müller *et al.* [35] cherchent une segmentation de l'image par minimisation de la fonction de Potts, dont la forme générale est donnée par l'équation 2.23.

Soit f_c la fonction associant à chaque pixel p_i sa couleur dans un espace colorimétrique donné et f_l la fonction associant à chaque pixel ses coordonnées dans l'image. Le terme d'attache aux données $\mathcal{F}_D$ de Müller *et al.* [35] s'inspire des travaux de Nieuwenhuis *et al.* [36], avec :

$$\mathcal{F}_D(p_i, \lambda_j) = -\log(P(p_i|\lambda_j)) \tag{2.27}$$

où $P(p_i|\lambda_j)$ est la probabilité conjointe d'avoir un pixel de couleur $f_c(p_i)$ et de localisation $f_l(p_i)$ sachant que le label λ_j lui est attribué. Soit G^{λ_j} l'ensemble des germes pour la classe de label λ_j :

$$P(p_i|\lambda_j) = \frac{1}{N_{G^{\lambda_j}}} \sum_{k=1}^{|G^{\lambda_j}|} \mathcal{K}_l\Big(f_l(p_i) - f_l(p_k)\Big)\mathcal{K}_c\Big(f_c(p_i) - f_c(p_k)\Big) \tag{2.28}$$

avec $|G^{\lambda_j}|$ la cardinalité de l'ensemble G^{λ_j}, $\mathcal{K}_l$ et $\mathcal{K}_c$ deux noyaux gaussiens de paramètres différents. Le fait que les deux noyaux gaussiens soient de largeurs différentes permet de pondérer l'influence de chacune des deux informations (la couleur et la localisation).

Alors que Nieuwenhuis *et al.* [36] supposent que les germes sont corrects, Müller *et al.* introduisent dans le calcul de $\mathcal{F}_D$ la probabilité $P_{corr}((p_i, \lambda_j), k)$ qu'un germe appartienne bien à la classe k :

$$P_{corr}((p_i, \lambda_j), k) = \begin{cases} \zeta & \text{si } \lambda_j = k \\ \dfrac{(1-\zeta)}{N_S - 1} & \text{sinon.} \end{cases}$$

Le paramètre ζ est la probabilité *a priori* que le germe soit correct. Il est fixé de manière empirique par Müller *et al.*

Le terme de régularisation $\mathcal{F}_R$ assure que le périmètre des régions obtenues soit minimal, en intégrant une distance qui ne prend pas seulement en compte le périmètre en termes de nombre de pixels, mais également la probabilité des pixels d'appartenir aux contours de l'image. L'une des particularité des travaux de Müller *et al.* [35] consiste à utiliser le détecteur de contour de Dollár *et al.* [11], donnant un résultat plus précis qu'un simple calcul de gradient. Ce détecteur repose sur une FA entraînée à détecter, dans un carré de taille 32×32, les pixels appartenant à un contour.

Chaque pixel de ce carré est décrit par :

- sa couleur exprimée dans l'espace CIELuv, ce qui donne 3 caractéristiques ;
- l'amplitude de son gradient, à deux échelles différentes (échelle 1 et $\frac{1}{2}$), ce qui donne 2 caractéristiques ;
- la présence du gradient dans l'une des quatre directions (nord, sud, est, ouest) à deux échelles différentes, ce qui donne 8 caractéristiques.

Au total, pour un carré de 32×32 pixels, un descripteur de $32 \times 32 \times 13$ éléments est obtenu. Afin de réduire la taille de ce descripteur, seule une valeur sur 2 dans le sens de la hauteur et dans celui de la largeur, est conservée. Le nombre d'éléments est donc ramené à $\frac{32 \times 32 \times 13}{4} = 3328$.

Ce descripteur est réduit à un carré de 5×5 pixels et les différences paire à paire entre ces pixels sont également calculées, pour chacune des caractéristiques précédentes (couleur, amplitude et direction du gradient). Ainsi $\binom{5\times5}{2} = 300$ différences sont calculées, pour 13 caractéristiques, ajoutant 3900 éléments au descripteur. Le descripteur final contient donc 7228 éléments.

Müller *et al.* [35] recherchent une segmentation minimisant la fonction de Potts ainsi obtenue en suivant une approche similaire à celle de Santner *et al.*, grâce à un algorithme de type *primal-dual* [3].

2.5.6 Méthode de Changjae *et al.*

L'algorithme de Changjae *et al.* [6] résout le problème de la segmentation interactive en produisant un ensemble de N_Λ fonctions, de la forme $f_{\lambda_j}(p_i, I, G)$. Chaque fonction donne un réel positif proportionnel à la probabilité du pixel p_i d'appartenir à la classe de label λ_j.

Soit f_c la fonction associant à chaque pixel p_i sa couleur dans un espace colorimétrique donné et f_l la fonction associant à chaque pixel ses coordonnées dans l'image. Le degré de ressemblance entre deux pixels p_i et p_j est donné par :

$$w_{i,j} = \exp\left(-\frac{||f_c(p_i) - f_c(p_j)||}{2{\sigma_c}^2} - \frac{||f_l(p_i) - f_l(p_j)||}{2{\sigma_l}^2}\right) \tag{2.29}$$

avec σ_c et σ_l deux paramètres fixés de manière empirique et dosant l'influence, respectivement, de la couleur et de la localisation du pixel.

Soit G^{λ_j} l'ensemble des germes donnés pour la classe de label λ_j. Pour chaque canal de couleur c_k, Changjae *et al.* calculent $H^{c_k}_{\lambda_j}(m_1, m_2)$, l'histogramme normalisé de co-occurrence, indiquant la proportion de germes :

- appartenant à G^{λ_j} ;
- ayant une valeur m_1 pour le canal de couleur c_k ;
- ayant un de leurs voisins dont la valeur, pour le même canal, vaut m_2.

Cet histogramme permet d'obtenir :

$$P^{c_k}_{\lambda_j}(m_1, m_2) = \frac{H^{c_k}_{\lambda_j}(m_1, m_2)}{\sum_{m_3 \in M^{c_k}} \sum_{m_4 \in M^{c_k}} H^{c_k}_{\lambda_j}(m_3, m_4)} \tag{2.30}$$

la probabilité d'occurrence et de co-occurrence des valeurs m_1 et m_2 pour le canal de couleur c_k. L'ensemble M^{c_k} correspond aux valeurs possibles pour le canal de couleur c_k. Par exemple, si nous nous intéressons au canal rouge de l'espace RVB, nous aurons $M^{rouge} = [0, 255]$.

La probabilité d'occurrence et de co-occurrence P_{λ_j} pour la classe de label λ_j est obtenue en faisant la moyenne des probabilités d'occurrence et de co-occurrence pour chacun des canaux de couleur. Cette probabilité permet d'obtenir pour chaque germe (p_i, λ_j) :

$$r^{\lambda_j}(i) = \sum_{p_k \in \mathcal{F}_{\text{voi}}(i)} P_{\lambda_j}(f_c(p_i), f_c(p_k)), \tag{2.31}$$

qui correspond à un score de confiance. La valeur $r^{\lambda_j}(i)$ est donc d'autant plus faible que le germe a une probabilité élevée d'être erroné.

Chaque fonction f_{λ_j} prend alors la forme d'un vecteur de N_I éléments, associant à chaque pixel un réel indiquant son degré d'appartenance à la classe de label λ_j et défini par

$$f_{\lambda_j}(I, G) = (D - W + \Omega)^{-1} \Omega R_j Z_j \tag{2.32}$$

avec :

- D une matrice diagonale de taille $N_I \times N_I$, où le i^{e} élément de la diagonale vaut $\sum_{j=1}^{N_I} w_{i,j}$;
- W une matrice carrée de taille $N_I \times N_I$, où l'élément situé sur la ligne i et la colonne j vaut $w_{i,j}$ si le pixel p_j fait partie des K plus proches voisins du pixel p_i (au sens de la couleur et de la localisation) ou 0 sinon ;
- R_j une matrice de taille $N_I \times N_I$, où le i^{e} élément de la diagonale vaut $r^{\lambda_j}(i)$;
- Z_j un vecteur de N_I éléments, où le i^{e} élément vaut 1 si le pixel p_i est un germe pour la classe de label λ_j, 0 sinon ;
- Ω est une matrice diagonale, dont chaque élément de la diagonale vaut ω, et qui sert à introduire le paramètre ω régulant l'influence du terme d'attache aux données $\mathcal{F}_D$.

2.6 Conclusion

Dans ce chapitre, nous avons présenté le problème de la segmentation interactive. Nous avons notamment décrit une dizaine d'algorithmes proposés durant les deux dernières décennies. Ces méthodes permettent de prendre conscience de la diversité des approches envisagées.

Certaines méthodes, à l'instar de l'algorithme de Mortensen *et al.* [33] reposent une interactivité forte avec l'utilisateur : chaque fois que ce dernier fournit un nouveau germe, la portion de contour entre celui-ci et le germe précédent est recherchée puis affichée. Cette recherche s'effectue en moins d'une seconde : l'utilisateur, de manière quasi immédiate, voit l'influence du germe qu'il vient de donner. Si le résultat obtenu ne lui convient pas, il peut aisément le modifier. Les deux manipulations principales pour améliorer la courbe (déplacer un germe pour qu'il coïncide davantage avec un point du contour et ajouter un germe entre deux germes précédents pour faciliter la recherche du contour) sont faciles à comprendre.

D'autres algorithme, comme celui de Müller *et al.* [35], mettent en jeu des traitements plus complexes, tels que l'analyse des germes donnés par l'utilisateur afin de déterminer leur degré de fiabilité. La production d'une segmentation résultat demande des calculs plus nombreux et plus longs : sur des images de quelques milliers de pixels, le temps d'exécution moyen de cette méthode est de 2 secondes, sous réserve d'utiliser une version optimisée tirant profit d'une parallélisation des étapes clés de l'algorithme.

Si de telles méthodes résolvent indiscutablement des problèmes de vision par ordinateur plus complexes, leur pertinence dans les domaines d'applications qui nous concernent est moins évidente.

Tout d'abord, rappelons que le cadre des observatoires photographiques du paysage nous impose de réfléchir à des algorithmes capable de fonctionner sur des smartphones, qui disposent de moins de ressources mémoire qu'un ordinateur standard, tout en faisant intervenir des processeurs moins puissants. Ensuite, soulignons que dans le cadre d'un logiciel de manipulation d'image tel que Gimp ou Photoshop, les outils de segmentation interactive sont souvent utilisés sur des photographies comprenant au moins plusieurs millions de pixels. Dans ces deux contextes, la complexité d'un algorithme comme celui de Müller est un défaut important.

Par ailleurs, si le fait de permettre de sélectionner plusieurs objets à la fois (donc de réaliser une segmentation interactive multiclasse) est un avantage par rapport aux méthodes de binari-

sation interactive, le fait de remettre en cause les indications données par l'utilisateur peut se révéler plus gênant qu'avantageux. En effet, les tests avec des utilisateurs réalisés par McGuinness *et al.* [31] montrent l'importance du caractère prévisible du résultat à partir des germes : en d'autres termes, un algorithme pour lequel l'utilisateur comprendra rapidement l'influence des indications qu'il donne sera généralement mieux accepté qu'une méthode capable de produire, en moyenne, des résultats un peu plus précis, mais ne permettant pas une correction facile lorsqu'elle se trompe. Dans le cas de Müller *et al.* [35], le fait de ne pas tenir compte de certaines indications, même s'il permet à l'utilisateur d'être moins précis lorsqu'il donne des germes, peut induire des situations frustrantes où, malgré l'ajout de germes, le résultat n'est pas mis à jour. Ce dernier point est particulièrement vrai lorsque le contraste entre deux objets à sélectionner n'apparaît pas clairement, à cause d'une ombre, d'un problème d'éclairage, etc.

Tous ces éléments nous ont conduit à réfléchir sur un algorithme de segmentation interactive multiclasse qui conserve les deux avantages qui ont fait le succès de méthodes comme celles de Mortensen *et al.* : une faible complexité algorithmique garantissant la production rapide d'un résultat et l'utilisation de germes dont l'influence est facilement compréhensible par l'utilisateur. Contrairement aux dernières méthodes de segmentation interactive proposées [6, 35, 46], le fonctionnement de cette méthode est itératif, ce qui induit une communication beaucoup plus dynamique avec l'utilisateur.

Chapitre 3

Algorithme de segmentation interactive Superpixel α-Fusion

3.1 Introduction

L'une des principales contributions présentée dans ce mémoire est la proposition d'une nouvelle méthode de segmentation interactive, $S\alpha F$ (Superpixel α-Fusion) [**?**, **?**]. Cette méthode permet la sélection précise, intuitive et rapide de multiples objets au sein d'une même photographie. Elle appartient à la catégorie des algorithmes de segmentation interactive multiclasse et repose sur une approche par recherche des régions. Les germes donnés par l'utilisateur correspondent à des traits de couleur qui indiquent quelques pixels appartenant à chaque objet à extraire.

L'algorithme $S\alpha F$ repose sur la modélisation du problème de la segmentation interactive comme la minimisation d'une fonction de coût pouvant être représentée par un graphe de facteurs. Il intègre l'utilisation d'une méthode de classification par apprentissage supervisé assurant l'adéquation entre la segmentation produite et les germes donnés, l'introduction d'un nouveau terme de régularisation et la réalisation d'un pré-traitement consistant à regrouper les pixels en petits ensembles cohérents.

3.2 Modélisation du problème de segmentation interactive

Soit I une image, $\mathbb{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_{N_\mathbb{S}}\}$ un ensemble de $N_\mathbb{S}$ primitives visuelles, $\Lambda = \{\lambda_1, \cdots, \lambda_{N_\Lambda}\}$ un ensemble de N_Λ labels correspondant à N_Λ classes et $G = \{g_1, \cdots, g_{N_G}\}$ un ensemble de germes indiquant pour chaque classe quelques pixels lui appartenant. Nous nous intéressons

au problème consistant à attribuer, à chaque pixel de I, un élément de Λ sous les contraintes suivantes :

- les pixels possédant le même label doivent former un ensemble visuellement homogène ;
- la classification produite doit être cohérente vis-à-vis de G.

Dans le cadre de l'algorithme que nous proposons, chaque primitive visuelle $\mathbf{s}_i$ est un ensemble de pixels voisins et similaires, nommé *superpixel*. Ces ensembles sont construits de manière à ce que l'attribution d'une et d'une seule classe à chaque superpixel permette d'obtenir la segmentation recherchée par l'utilisateur.

Les superpixels forment une partition en composantes connexes des pixels de l'image I. Ils peuvent donc être représentés par un graphe $\mathcal{G} = < X, E >$, avec :

- $X = \{x_1, \cdots, x_{N_\mathbb{S}}\}$ un ensemble de sommets, où chaque sommet x_i représente un superpixel $\mathbf{s}_i$;
- E un ensemble d'arêtes non orientées, avec une arête $e_{i,j}$ pour chaque paire de superpixels $(\mathbf{s}_i, \mathbf{s}_j)$ ayant des pixels voisins.

Soit $C = \{c_1, \cdots, c_\mathbb{S}\}$, un ensemble de variables aléatoires indexées par les sommets de $\mathcal{G}$, tel que $c_i \in \Lambda$. Si nous admettons que la probabilité d'une variable c_i de recevoir le label λ_k ne dépend que des caractéristiques des sommets x_j voisins directs de x_i, alors C est un champ aléatoire de Markov (CAM) et une segmentation optimale C^* s'obtient en minimisant :

$$\mathcal{F}_{CAM}(I, C, G) = \sum_{i=1}^{N} \mathcal{F}_D(x_i, c_i) \; + \sum_{e_{i,j} \in E} \omega(x_i) \mathcal{F}_R(x_i, c_i, x_j, c_j) \tag{3.1}$$

avec :

- $\mathcal{F}_D$ un terme d'attache aux données, s'assurant de la cohérence avec les caractéristiques de l'image I et celles des germes G ;
- $\mathcal{F}_R$ un terme de régularisation vérifiant certaines propriétés propres à la segmentation, concernant par exemple la taille ou la forme des régions ;
- ω une fonction de pondération.

3.2.1 Terme d'attache aux données

Une segmentation optimale C^* de I étant obtenue par minimisation de $\mathcal{F}_{CAM}$, le terme d'attache aux données $\mathcal{F}_D$ vaut :

$$\mathcal{F}_D(x_i, c_i) = 1 - P(\lambda_j | x_i) \text{ avec } c_i = \lambda_j. \tag{3.2}$$

où $P(\lambda_j | x_i)$ est la probabilité que le sommet x_i appartiennent à la classe de label λ_j.

Approche de Wu

Comme nous nous intéressons à un problème de classification multiclasse, $P(\lambda_j | x_i)$ peut être estimée de manière précise grâce à la seconde des deux méthodes décrites par Wu *et al.*, dans

leur article *« Probability estimates for multi-class classification by pairwise coupling »* [56]. Elle s'appuie sur le résultat d'une méthode de classification supervisée résolvant des problèmes à deux classes. La fonction binaire associée à cette méthode est $F_{sup}(x_i, \lambda_j, \lambda_k)$ qui vaut :

- 0 si les caractéristiques de x_i sont plus proches de celles de la classe λ_j ;
- 1 si les caractéristiques de x_i sont plus proches de celles de la classe λ_k.

L'estimation de la probabilité que x_i appartienne à la classe de label λ_j plutôt qu'à la classe de label λ_k est obtenue en calculant :

$$F_{jk}(x_i, \lambda_j, \lambda_k) = \frac{1}{1 + \exp(\omega_A F_{sup}(x_i, \lambda_j, \lambda_k) + \omega_B)} \tag{3.3}$$

où ω_A et ω_B sont deux paramètres obtenus lors de la phase d'apprentissage de la méthode de classification supervisée. À partir de cette probabilité, Wu *et al.* montrent que la distribution de probabilité $\mathcal{P} = \{P(\lambda_1|x_i), \cdots, P(\lambda_{N_\Lambda}|x_i)\}$ s'estime en minimisant :

$$\sum_{j=1}^{N_\Lambda} \sum_{\substack{k=1 \\ j \neq k}}^{N_\Lambda} \Big(F_{jk}(x_i, \lambda_j, \lambda_k) P(\lambda_k|x_i) - F_{kj}(x_i, \lambda_j, \lambda_k) P(\lambda_j|x_i) \Big)^2 \tag{3.4}$$

sous les contraintes suivantes :

- $\sum_{j=1}^{N^\Lambda} P(\lambda_j|x_i) = 1$;
- $P(\lambda_j|x_i) \ > 0 \ \forall \ j \ \in [1, N_\Lambda]$.

Classification supervisée

L'utilisation d'une méthode de classification supervisée implique de disposer d'un ensemble de *données d'apprentissage*, permettant de paramétrer la méthode, afin que celle-ci puisse classer les données restantes, nommées *données d'exploitation*. Dans le cas de la méthode que nous proposons, il s'agit de produire une partition de l'ensemble des superpixels $\mathbb{S}$ associé à une image en deux ensembles :

- $\mathbb{S}_A$ pour les données d'apprentissage ;
- $\mathbb{S}_E$ pour les données d'exploitation.

La construction de ces deux ensembles s'effectue par analyse des indications données par l'utilisateur. L'ensemble des germes G étant constitué de couples (p_i, λ_j), associant un label à certains pixels de l'image I, il convient de trouver une procédure à la fois rapide et fiable pour transférer ces germes vers les superpixels.

Soit $\mathbf{s}_k$ un superpixel. Trois cas de figure peuvent se présenter :

1. $\mathbf{s}_k$ ne contient aucun pixel correspondant à un germe c'est-à-dire $\nexists (p_i, \lambda_j) \in G$, $p_i \in \mathbf{s}_k$;
2. $\mathbf{s}_k$ contient des germes de plusieurs classes, c'est-à-dire $\exists (p_{i1}, \lambda_{j1}) \in G \wedge (p_{i2}, \lambda_{j2}) \in G$, $p_{i1} \in \mathbf{s}_k \wedge p_{i2} \in \mathbf{s}_k \wedge \lambda_{j1} \neq \lambda_{j2}$;

3. $\mathbf{s}_k$ contient des germes d'une seule classe, de label λ_l, c'est-à-dire que $(p_i, \lambda_j) \in G \wedge p_i \in \mathbf{s}_k \implies \lambda_j = \lambda_l$.

Nous avons choisi d'utiliser uniquement les superpixels de la troisième catégorie pour constituer l'ensemble $\mathbb{S}_A$ et de considérer les superpixels du deuxième type comme étant des données erronées.

Les méthodes de classification supervisée de type séparateur à vaste marge (SVM) et de type FA sont traditionnellement utilisées avec l'approche décrite par Wu *et al.* [56]. La section 3.5 présente notre démarche pour sélectionner la plus pertinente dans le cadre d'une d'application de segmentation interactive.

3.2.2 Terme de régularisation

La segmentation obtenue par la seule minimisation du terme d'attache aux données $\mathcal{F}_D$ peut, dans certains cas, contenir de nombreuses erreurs qui se présentent sous la forme de superpixels attribués à la mauvaise classe, alors que l'ensemble de leurs voisins a été classé correctement. Nous parlons alors de segmentation *bruitée*, par analogie avec le bruit présent dans les images, au niveau des pixels.

Ce bruit complique considérablement la tâche de l'utilisateur, qui doit rajouter de nombreux germes pour le corriger. La méthode la plus courante pour s'en prémunir consiste à intégrer, comme nous l'avons fait, un terme de régularisation au sein de la fonction de coût à minimiser. Les termes de régularisation généralement utilisés favorisent la production d'une segmentation dont les régions sont grandes, de formes régulières, avec des contours suivant les bordures des objets présents dans l'image. La prise en compte de ce type d'*a priori* sur les régions composant une segmentation optimale présente cependant le désavantage d'augmenter significativement le temps d'exécution de la méthode [32, 46].

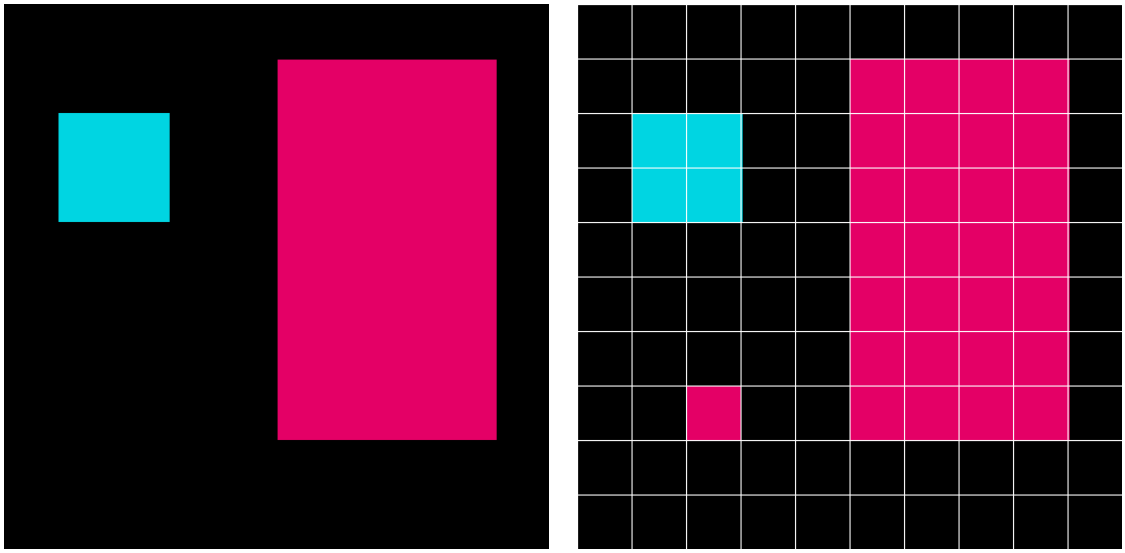
Nous proposons un nouveau terme de régularisation dont les objectifs sont les suivants :

- réduire le bruit au sein de la segmentation ;
- conserver une complexité algorithmique suffisamment faible pour garantir une méthode fonctionnant en temps interactif ;
- préserver les classes rares.

Par *temps interactif* nous entendons un temps d'exécution de quelques secondes maximum pour des images de plusieurs millions de pixels, prérequis nécessaire pour que la méthode soit adoptée par un utilisateur.

Une classe est dite *rare* si elle correspond à un petit objet, peu représenté en termes de nombre de superpixels. Les classes rares se révèlent problématiques car elles présentent de nombreux aspects communs avec le bruit présent dans la segmentation. Ces classes rares prennent la forme de quelques superpixels entourés de superpixels attribués à une autre classe. La figure 3.1 illustre notre propos. Sur l'image d'origine (voir la figure 3.1a), les objets visibles se répartissent en trois classes : deux classes majoritaires (en noir et rose) et une classe rare (en bleu). La figure 3.1b montre une segmentation obtenue. Cette segmentation est bruitée, l'un des superpixels de la classe noire ayant été attribué par erreur à la classe rose.

L'analyse de cet exemple nous donne une piste pour la résolution du problème qui nous occupe. En effet, même si la segmentation obtenue sur la figure 3.1b est bruitée, nous pouvons constater qu'il est plus fréquent pour un superpixel attribué à la classe bleue d'avoir comme voisin un superpixel de la classe noire, que pour un superpixel de la classe rose. Ainsi la probabilité $P_i(j)$ pour un superpixel de classe de label λ_i d'avoir comme voisin un superpixel attribué à la classe de label λ_j, fournit une information permettant de distinguer un superpixel appartenant à une classe rare d'un superpixel bruité.



(a) Image originale. Trois classes sont présentes : deux classes majoritaires (en noir et rose) et une classe rare (en bleu).

(b) Segmentation erronée : l'un des superpixels de la classe noire a été attribué par erreur à la classe rose. Les contours des superpixels sont tracés en blanc.

FIGURE 3.1 – Distinction entre classes rares et erreurs de segmentation.

Soit $\mathbf{s}_i$ un superpixel, x_i le sommet qui lui est associé et $\mathcal{P} = \{P(\lambda_1|x_i), \cdots, P(\lambda_{N_\Lambda}|x_i)\}$ l'ensemble des probabilités que ce superpixel appartienne à chacune des classes. La classe de label λ_j est attribuée à x_i de manière à ce que :

$$P(\lambda_j|x_i) \geqslant P(\lambda_k|x_i) \ \forall k \in [1, N_\Lambda]. \tag{3.5}$$

Après avoir ainsi obtenu une première segmentation, la probabilité $P_i(j)$ s'obtient rapidement en calculant l'histogramme normalisé des classes des superpixels voisins d'un superpixel de label λ_i. Soulignons que, du fait qu'une distribution de probabilité soit calculée pour chacune des classes $P_i(j) \neq P_j(i)$. Comme les arêtes de $\mathcal{G}$ sont non orientées, $P_i(j)$ et $P_j(i)$ doivent être fusionnées. Nos tests ont montré que :

$$\mathcal{F}_R(x_i, c_i, x_j, c_j) = 1 - \max(P_i(j), P_j(i)) \tag{3.6}$$

donne de bons résultats. L'utilisation de la fonction moyenne ou de la fonction minimum, à la

place de la fonction maximum, ne permet pas d'améliorer la qualité des segmentations produites par $S\alpha F$.

3.2.3 Fonction de pondération

La fonction de pondération ω dose l'influence du terme de régularisation par rapport au terme d'attache aux données. Dans la fonction de coût définie par l'équation 3.1, le terme d'attache aux données intervient une fois par sommet, tandis que le terme de régularisation contribue à augmenter la valeur de la fonction pour chaque arête, c'est-à-dire pour chaque couple de sommets voisins. Pour que la contribution de la somme des termes de régularisation rattachés à un même sommet soit similaire quel que soit le nombre de voisin d'un sommet, il est nécessaire que la fonction de pondération intègre le nombre de voisins de ce sommet. Soit $\mathcal{F}_{\text{voi}}(x_i)$, la fonction renvoyant l'ensemble des sommets voisins de x_i. Plusieurs tests nous ont conduit à sélectionner la fonction suivante :

$$\omega(x_i) = \frac{0{,}7}{|\,\mathcal{F}_{\text{voi}}(x_i)|}. \tag{3.7}$$

3.2.4 Algorithme $S\alpha F$

L'objectif de l'algorithme $S\alpha F$ consiste à minimiser la fonction $\mathcal{F}_{CAM}$. La figure 3.2 en schématise le fonctionnement général.

L'approche $S\alpha F$ comprend une étape d'*initialisation*, qui permet d'extraire les superpixels, et une étape de *segmentation*, où les germes donnés par l'utilisateur sont pris en compte. Tandis que l'étape d'initialisation ne nécessite d'être réalisée qu'une seule fois, l'étape de segmentation est répétée tant que l'utilisateur modifie les germes.

Lors de l'étape d'initialisation, les pixels sont d'abord groupés en superpixels. Les caractéristiques de chaque superpixel sont ensuite analysées et un descripteur sous la forme d'un vecteur à N_d dimensions est calculé. Ainsi, le nombre de primitives visuelles se voit considérablement réduit, chaque superpixel regroupant plusieurs centaines de pixels, ce qui accèlere les traitements effectués lors de l'étape de segmentation.

Cette dernière concerne la minimisation à proprement parler de $\mathcal{F}_{CAM}$ et se déroule de la manière suivante :

1. les germes donnés par l'utilisateur sont analysés afin d'obtenir :
   (a) le nombre de classes N_Λ ;
   (b) pour chaque classe, l'ensemble $\mathcal{S}_A^{\lambda_i}$ des superpixels nécessaires pour entraîner la méthode de classification à reconnaître les superpixels appartenant à cette classe ;
2. à l'aide de la méthode de classification, la probabilité de chaque superpixel d'appartenir à chacune des classes est calculée ;
3. l'obtention de ces probabilités permet de calculer, pour chaque superpixel, le terme d'attache aux données $\mathcal{F}_D$;

4. une première segmentation est produite en attribuant à chaque superpixel la classe la plus probable ;
5. cette segmentation est analysée afin d'obtenir pour chaque couple de classes (λ_i, λ_j) les probabilités $P_i(j)$ et $P_j(i)$;
6. l'obtention de ces probabilités permet de calculer pour chaque paire de superpixels voisins le terme de régularisation $\mathcal{F}_R$;
7. une approximation de la segmentation correspondant à la valeur minimale de $\mathcal{F}_{CAM}$ est recherchée ;
8. le label attribué à chaque superpixel est transféré à l'ensemble des pixels le constituant ;
9. la segmentation résultat est montrée à l'utilisateur qui peut éventuellement ajouter ou supprimer des germes.

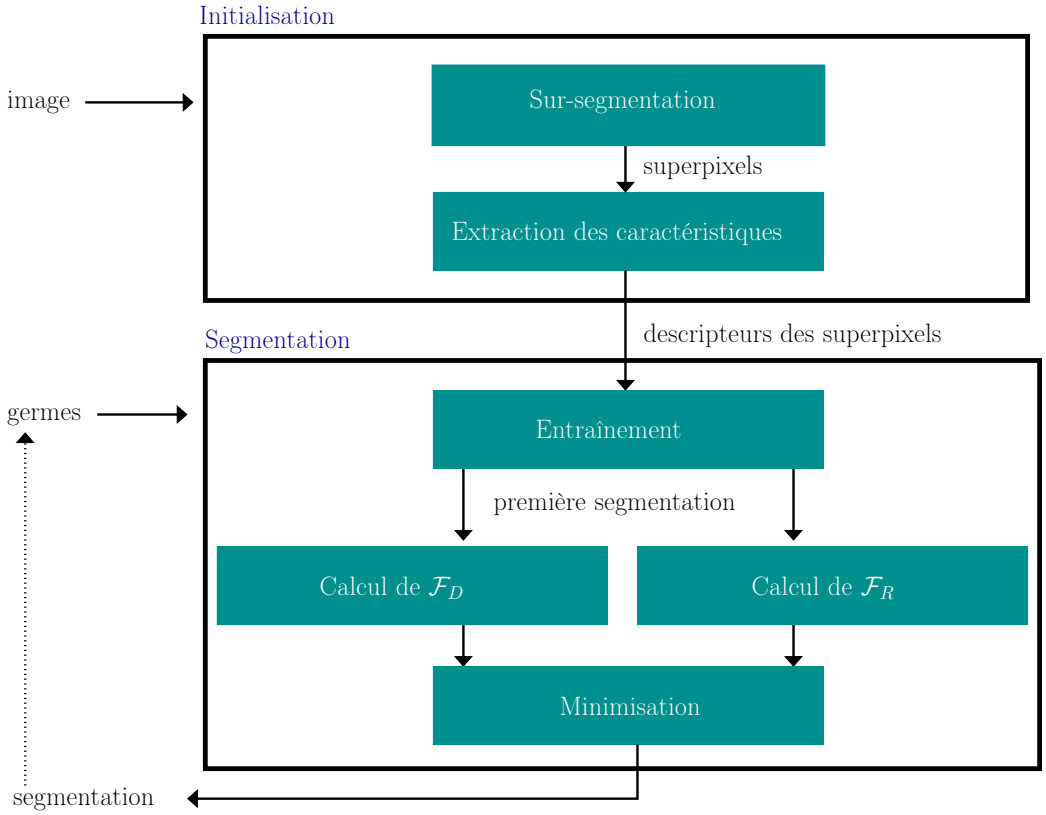


FIGURE 3.2 – Schéma général de l'algorithme $S\alpha F$.

À partir de cet algorithme général, toute une famille de méthodes peuvent être obtenues, dont les variations tiennent dans le choix de la méthode de sur-segmentation, des caractéristiques utilisées pour décrire chaque superpixel, de la méthode de classification par apprentissage supervisé et de l'algorithme d'optimisation permettant de minimiser la fonction de coût.

3.3 Algorithme de sur-segmentation

3.3.1 Problématique

L'algorithme de sur-segmentation utilisé doit permettre de regrouper rapidement les pixels de l'image en superpixels, de manière à minimiser la probabilité qu'un superpixel chevauche deux objets différents. L'obtention des superpixels ainsi que le calcul de leurs descripteurs n'est réalisé qu'une seule fois, lors de l'étape d'initialisation. Comme la même classe est attribuée à l'ensemble de ses pixels, un superpixel contenant des pixels appartenant à deux éléments que l'utilisateur souhaite séparer engendrera des erreurs dans la segmentation résultat. Quel que soit le nombre de germes ajoutés par l'utilisateur, cette erreur ne pourra pas être corrigée, puisqu'aucun mécanisme ne permet de séparer le superpixel en deux nouveaux superpixels.

Le choix d'un algorithme de sur-segmentation est donc une étape clé pour le bon fonctionnement de $S\alpha F$. Suite à une évaluation des principales méthodes existantes (qui fait l'objet du chapitre 4), nous avons choisi l'algorithme SLIC (« *Simple Linear Iterative Clustering* ») [1] qui fournit un bon compromis entre qualité du résultat et rapidité.

3.3.2 Principe général de l'algorithme SLIC et choix des paramètres

Proposé par Achanta *et al.* SLIC [1], est une version modifiée de l'algorithme de k-moyennes. Des superpixels initiaux sont produits en regroupant les pixels selon une grille régulière. Le choix de taille des cases est un paramètre qu'il convient de fixer, soit en précisant un nombre souhaité de superpixels (les pixels de l'image sont alors découpés en autant de case que nécessaire), soit en précisant l'aire moyenne désirée pour les superpixels (la hauteur et la largeur des cases sont alors déterminées pour s'approcher le plus possible de cette aire).

L'algorithme SLIC répète alors une dizaine de fois les deux étapes suivantes :

1. la couleur et la localisation moyennes de chaque superpixel sont calculées ;
2. chaque pixel est rattaché au superpixel maximisant une fonction de similarité.

L'une des contributions essentielles d'Achanta *et al.* concerne la fonction permettant de mesurer la similarité entre un pixel et un superpixel. Cette dernière fait intervenir à la fois la couleur et la localisation. Un paramètre, fixé par l'utilisateur, permet de pondérer l'influence de chacune de ces deux informations.

Une dernière étape permet d'uniformiser la taille des superpixels et de s'assurer que ces derniers correspondent bien à des composantes connexes.

Les deux paramètres à fixer pour cette méthode sont donc :

- le nombre de superpixels, que nous avons fixé à 3000, afin d'obtenir des superpixels suffisamment petits pour réduire les erreurs dans la sur-segmentation ;
- le poids pour la fonction de similarité, pour lequel Achanta *et al.* [1] recommandent une valeur de 10, ce qui s'est avéré pertinent dans le cas de $S\alpha F$.

Une description détaillée de l'algorithme SLIC est donnée dans la section 4.5.4. La section 4.6.5 contient, quant à elle, une analyse des qualités nous ayant conduit à le choisir.

3.4 Description des superpixels

3.4.1 Problématique

Les superpixels correspondant à des ensembles homogènes de pixels connexes, les descripteurs qui caractérisent leur apparence peuvent au choix, soit tirer partie de la diversité de l'information contenue dans un superpixel, soit chercher à la résumer. Les descripteurs de la première catégorie comprennent par exemple :

- les histogrammes des différents canaux de couleur ;
- l'histogramme d'un descripteur de texture tel que les textons ou les motifs binaires locaux [40] (LBP, de l'anglais « *Local Binary Patterns* »).

Pour la seconde catégorie, nous trouvons, entre autres :

- la moyenne de chaque canal de couleur ;
- l'écart type de chaque canal de couleur ;
- le coefficient de dissymétrie de chaque canal de couleur ;
- le kurtosis de chaque canal de couleur ;
- la position du barycentre ;
- l'aire en nombre de pixels ;
- le rapport entre l'aire du superpixel et celle du plus petit rectangle l'englobant.

Les descripteurs choisis doivent permettre de regrouper les superpixels appartenant à la même classe, tout en séparant ceux correspondant à des objets différents. Ne disposant d'aucune connaissance *a priori* sur l'image, ni sur les propriétés de la segmentation que l'utilisateur voudra obtenir, ils doivent également s'adapter à un large panel de situations. Enfin, ni leur extraction durant l'étape d'initialisation, ni leur utilisation durant l'étape de segmentation ne doivent se répercuter de manière trop importante sur les temps d'exécution.

Certaines caractéristiques, notamment celles associées à la texture, requièrent un nombre de calculs élevé lors de l'étape d'initialisation. Par ailleurs, plus le descripteur contient de caractéristiques, plus le temps nécessaire pour évaluer la probabilité d'un superpixel d'appartenir à chacune des classes est important.

3.4.2 Choix d'un descripteur

Nous avons choisi un descripteur minimaliste comprenant cinq caractéristiques :

- trois pour exprimer la couleur moyenne du superpixel dans l'espace colorimétrique RGB ;
- deux pour exprimer sa localisation dans l'image, par les coordonnées de son barycentre.

Ces cinq caractéristiques sont normalisées : pour la couleur, nous divisons chaque moyenne par 255 ; pour les coordonnées du barycentre, nous divisions l'abscisse par la largeur de l'image et l'ordonnée par la hauteur de l'image. Nous obtenons un descripteur rapide à calculer lors de la phase d'initialisation et comprenant peu de caractéristiques, ce qui accélère son utilisation lors de l'étape de segmentation.

L'information de couleur permet de segmenter rapidement et avec un minimum de germes de larges zones homogènes telles que le ciel. Lorsque l'utilisateur souhaite séparer deux objets ayant des teintes similaires, l'information de localisation prend le relais. Cette dernière assure également que les germes conservent un comportement local c'est-à-dire que l'ajout de germes sur une partie de l'image ne perturbe pas la segmentation dans le reste de la photographie.

3.4.3 Problèmes liés au positionnement des germes

En contrepartie des deux qualités de ce descripteur – sa rapidité et l'influence intuitive des germes – la simplicité de ce dernier fait que, dans certains cas, les indications données par l'utilisateur peuvent ne pas être suffisantes pour trouver une segmentation pertinente de l'image.

Tout d'abord, les germes peuvent ne pas être suffisants pour créer un modèle des objets à extraire pertinent en termes de couleur. Un exemple est présenté sur la figure 3.3. Le pollen de cette dernière, en jaune, n'est pas correctement segmenté. Aucun pixel jaune n'étant sélectionné comme germe, cette teinte est rattachée à tort au fond, dont la couleur verte est plus proche du jaune que du rouge des pétales de la fleur. L'ajout de quelques germes sur l'une des deux zones centrales des fleurs suffit à améliorer netteent la qualité de la segmentation.

Inversement, les germes peuvent ne pas être répartis correctement dans l'image en termes de localisation, à l'instar de l'exemple donné sur la figure 3.4. Ici, même si théoriquement la couleur devrait permettre de séparer correctement l'éléphant, le ciel et la végétation, l'utilisation d'une information de localisation vient perturber la méthode de classification pour la partie gauche de l'image qui, contrairement à la moitié droite, ne contient pas de germes.

Afin d'éviter ce dernier problème, une consigne simple peut être donnée à l'utilisateur lui demandant de donner des germes « *près des contours* ».

3.4.4 Autres descripteurs testés

En remplaçant l'espace RGB par d'autres espaces colorimétriques (CIELab, HSV) nous n'avons pas constaté d'améliorations significatives.

Il en va de même avec l'utilisation d'un descripteur de texture, dont le coût en termes de temps de calcul n'est pas compensé par une amélioration notable des performances (réduction du nombre de germes nécessaires et/ou diminution des erreurs dans la segmentation). Ainsi, nos tests avec les LBP, descripteur de texture pourtant à l'origine d'avancées notables dans de nombreux travaux en vision par ordinateur, ne nous ont pas permis d'améliorer les scores de précision de notre méthode.

L'utilisation des histogrammes de LBP s'est en revanche traduite par une augmentation du temps d'exécution de 23% pour l'étape d'initialisation et de 76% pour l'étape de segmentation. Ce résultat n'est pas surprenant : les superpixels sont conçus sur la base de la couleur des pixels, de manière à former des ensembles homogènes. Ainsi, l'écart type moyen pour l'ensemble des données que nous avons utilisées ne dépasse pas les 4%, ce qui indique que les pixels au sein d'un même superpixel présentent peu de variations en termes de couleur.

FIGURE 3.3 – Illustration de l'impact de la répartition des germes en termes de couleur sur la qualité de la segmentation obtenue.

3.5 Choix d'une méthode de classification supervisée

L'approche de Wu *et al.* [56] est connue pour donner de bons résultats à la fois avec les méthodes de classification SVM et FA. Le choix de l'une de ces méthodes pour estimer la probabilité qu'un superpixel appartienne à une classe, nécessite de s'intéresser aux forces et aux faiblesses de chacune d'entre elles, dans le cadre spécifique du problème de la segmentation interactive tel que nous l'avons posé.

Ainsi, par rapport aux problèmes de classification par apprentissage canoniques, nous pouvons noter les particularités suivantes :

- nous disposons de peu de données d'apprentissage ;
- de nombreuses images présentent des classes *rares* ;
- le descripteur de chaque superpixel comprend peu de caractéristiques.

Les données d'apprentissage étant spécifiques à une image, elles correspondent à un sous-

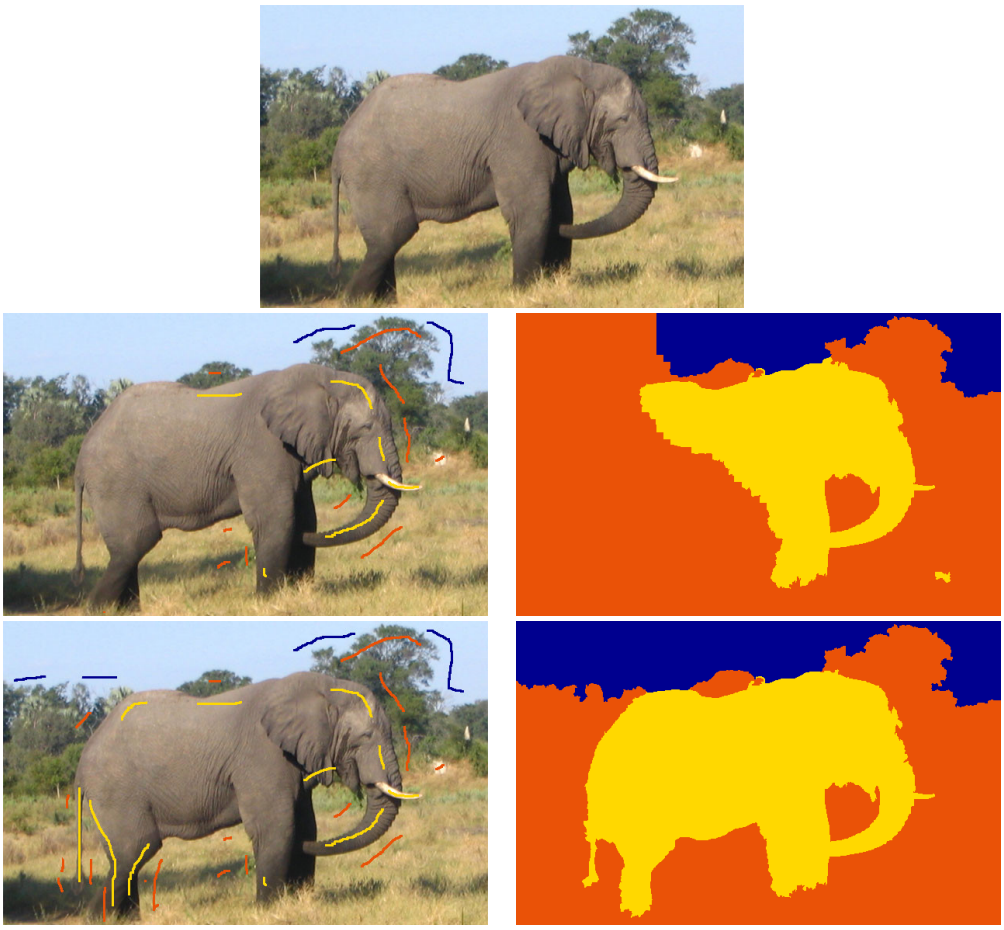


FIGURE 3.4 – Illustration de l'impact de la répartition des germes en termes de localisation sur la qualité de la segmentation obtenue.

ensemble des superpixels produits. Or, pour conserver des temps d'exécution raisonnables, le nombre de ces derniers ne doit pas dépasser quelques milliers. Si nous estimons que l'utilisateur annote environ 25% des superpixels, ce qui correspond déjà à un nombre élevé de germes et à un effort conséquent pour guider la méthode, nous disposons de seulement quelques centaines de données d'apprentissage, à répartir entre les différentes classes.

Parmi ces classes, certaines correspondent à de petits objets dont la surface en nombre de pixels est très inférieure à celles des autres composantes de l'image. Il est donc probable que les données d'apprentissage pour ces classes soit moins nombreuses (quelques dizaines de superpixels).

Le fait que les descripteurs utilisés ne contiennent que cinq caractéristiques constitue également une singularité, de nombreux problèmes de classification ayant plutôt à faire face à un descripteur de taille trop importante, qu'il convient de réduire, par exemple avec une analyse en composantes principales.

À ces spécificités inhérentes au contexte applicatif de la segmentation interactive, s'ajoutent celles liées à l'utilisation de la méthode de classification au sein de $S\alpha F$ et qui consiste à :

- estimer pour chaque superpixel une distribution de probabilité permettant de calculer le terme d'attache aux données ;
- produire une segmentation initiale dont dépend la pertinence du terme de régularisation.

Le dernier point est sans doute le plus simple à évaluer : sachant la classe réelle d'un superpixel, il suffit de vérifier qu'elle correspond à sa classe prédite. Vérifier la qualité de la distribution de probabilité est plus délicat. À partir de données de référence où chaque pixel est attribué à une classe, nous avons cherché à produire un nouvel ensemble de données de référence associant à chaque superpixel une distribution de probabilité de référence. Cela nous permet notamment de prendre en compte le fait qu'un superpixel en bordure d'un objet puisse avoir des caractéristiques proches de celles de plusieurs classes : dans ce cas de figure, nous préférons obtenir une distribution de probabilité témoignant de cette incertitude qu'une prédiction assurée pour l'une ou l'autre des classes.

Enfin, soulignons que nos tests ont été réalisés à l'échelle du superpixel et non du pixel. En effet, un pixel se voit attribuer la classe du superpixel auquel il appartient. Les sources d'erreurs dans la segmentation présentes à cette échelle peuvent donc provenir soit d'une distribution de probabilité erronnée, soit d'une erreur commise par la méthode de sur-segmentation employée. Ici, seul le premier cas de figure nous intéresse.

3.5.1 Fonctionnement des deux méthodes

Séparateur à vaste marge

Le principe d'un SVM est illustré par la figure 3.5. Il consiste à représenter une donnée comme un point dans un espace à N_D dimensions et à lui attribuer une classe en fonction de sa position par rapport à un hyperplan. Pour déterminer la position de l'hyperplan, le classificateur SVM réalise une étape d'apprentissage durant laquelle il dispose de données déjà classées et cherche l'équation d'un hyperplan séparant les deux ensembles de points, de manière à maximiser l'écart b entre les points périphériques (les vecteurs de support) et cet hyperplan de séparation. Si nous considérons nos données comme des vecteurs de N_d éléments, un cas idéal serait celui où $N_d = N_D$. Concrètement, pour trouver un hyperplan séparant toutes les données, N_D s'avère généralement très supérieur à N_d.

Il n'est pas toujours souhaitable que l'hyperplan sépare parfaitement l'ensemble des points : des points mal classés durant la phase d'apprentissage peuvent être tolérés afin de garantir qu'une marge entre les vecteurs de support soit plus importante. Par exemple, dans le cas de la segmentation interactive, il est possible que l'utilisateur ait légèrement débordé lors de son tracé du germe, et qu'ainsi un superpixel soit attribué à tort à une classe. Il est intéressant que ces données d'apprentissage erronées puissent ne pas être prise en considération. Un paramètre ω_C permet de pondérer l'importance des erreurs de classification par rapport à celle de la largeur de la marge.

L'évaluation de la distance entre deux points, dans un espace à N_D dimensions, est réalisé au moyen d'une fonction nommée noyau. Pour le problème de segmentation interactive que nous

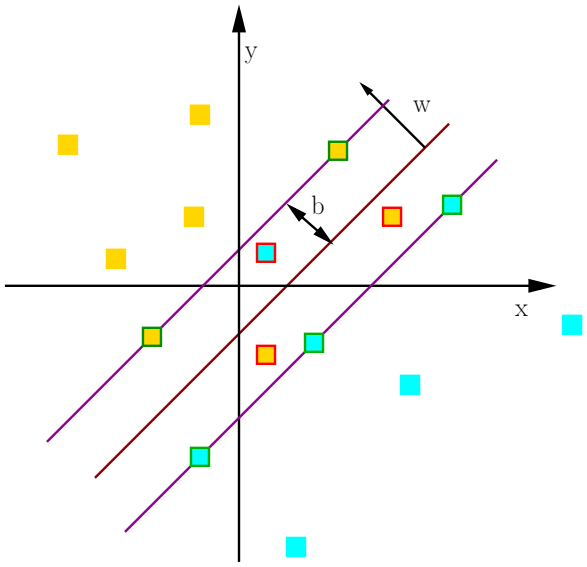


FIGURE 3.5 – L'hyperplan optimal (en rouge), caractérisé par le vecteur w, avec la marge maximale b. Les échantillons entourés en vert sont des vecteurs de support. Les échantillons entourés en rouge sont des erreurs tolérées lors de la phase d'apprentissage.

cherchons à résoudre, nous avons utilisé le noyau RBF, de l'anglais « *Radial Basis Function* » :

$$K_{RBF}(u, v) = \exp(-\omega_\Phi ||u - v||^2) \tag{3.8}$$

avec u et v les données dont nous voulons évaluer la distance et ω_Φ un paramètre du noyau. Les tests que nous avons menés avec d'autres noyaux n'ont pas permis une amélioration significative des résultats.

Forêt Aléatoire

Le principe des FA ayant déjà fait l'objet d'une présentation détaillée à la section 2.5.4, nous nous contenterons de rappeler ici les quelques points fondamentaux qui permettent de comprendre les résultats de notre évaluation :

- une FA correspond à un ensemble d'arbres de décision ;
- chaque arbre est entraîné séparément, avec un sous-ensemble des données d'apprentissage ;
- chaque arbre est spécialisé sur un petit nombre de caractéristiques, inférieur à la taille du descripteur ;
- lorsqu'une nouvelle donnée doit être classée, l'ensemble des arbres votent : la classe ayant obtenu le maximum de votes est attribuée à la donnée.

3.5.2 Protocole expérimental

Afin de comparer les deux méthodes de classification supervisée, nous avons utilisé, d'une part, l'implémentation d'une FA de la bibliothèque Alglib [1] et, d'autre part, l'implémentation C-SVM, contenue dans la bibliothèque libSVM [2]. Chacune des deux implémentations nécessite deux paramètres :

- le paramètre du noyau, ω_Φ, et le paramètre de prise en compte des erreurs dans les données d'apprentissage, ω_C dans le cas du SVM ;
- le nombre d'arbres de décision et le pourcentage de données d'apprentissage utilisées pour entraîner chaque arbre de décision dans le cas de la FA.

Nous nous sommes intéressés à l'évolution des performances de ces deux méthodes en fonction des couples de paramètres. Les trois critères quantitatifs que nous avons employés sont les suivants :

- l'adéquation entre les probabilités produites et la réalité, donnée par le taux d'erreur $\mathcal{F}_{Err}$;
- le temps d'exécution moyen pour l'entraînement de la méthode ;
- le temps d'exécution moyen nécessaire pour estimer les probabilités pour l'ensemble des classes et des superpixels, une fois que l'apprentissage est réalisé. Par la suite, nous parlerons de temps d'exécution pour la *phase d'exploitation*.

Soit un ensemble de données de référence composé de couples (I, R) associant à une image une segmentation de référence R où chaque pixel est attribué à une unique classe. L'image I est sur-segmentée en $N_\mathbb{S}$ superpixels, dont les caractéristiques sont extraites afin d'obtenir un ensemble de variables $X = \{x_1, \cdots, x_{N_\mathbb{S}}\}$ semblable à celui utilisé par $S\alpha F$. Pour chaque variable x_i, une estimation de la probabilité $P_R(\lambda_j|x_i)$ est obtenue en calculant la proportion de pixels rattachés à x_i et attribués à la classe de label λ_j dans R. En estimant $P_R(\lambda_j|x_i)$ pour l'ensemble des classes présentes dans la vérité terrain, nous obtenons $\mathcal{P}_R$ la distribution de probabilité de référence du superpixel. Soit $\mathcal{P}_{classif}$ la distribution de probabilité estimée à l'aide le l'approche de Wu *et al.* [56] :

$$\mathcal{F}_{Err}(P_R, P_{classif}) = \sum_{i=1}^{N_I} \sqrt{\sum_{j=1}^{N_\Lambda} (P_{classif}(\lambda_j|x_i) - P_R(\lambda_j|x_i))^2}. \tag{3.9}$$

Les performances des deux méthodes, SVM et FA, ont été analysées sur la base de deux ensembles de données de référence : celui de Santner *et al.* [46] et celui de McGuinness *et al.* [31]. Afin de ne pas sur-apprendre les paramètres de chacune des méthodes et de ne pas biaiser les résultats de l'évaluation de $S\alpha F$ qui utilise également ces données de référence, nous nous sommes contentés de prélever un tiers des segmentations de référence, soit 100 images pour Santner *et al.* [46] et 34 images pour McGuinness *et al.* [31]. Par la suite, nous désignerons ces deux ensembles respectivement par R^{MC} et R^{BI}. Le premier correspond à des problèmes de segmentation interactive multiclasse (MC), tandis que le second ne contient que des binarisations (BI).

1. www.alglib.net
2. www.csie.ntu.edu.tw/~cjlin/libsvm

La figure 3.6 donne la répartition du nombre de classes pour R^{MC}. Celui-ci varie de 2 à 7, avec une majorité de segmentations comprenant 2 et 4 classes.

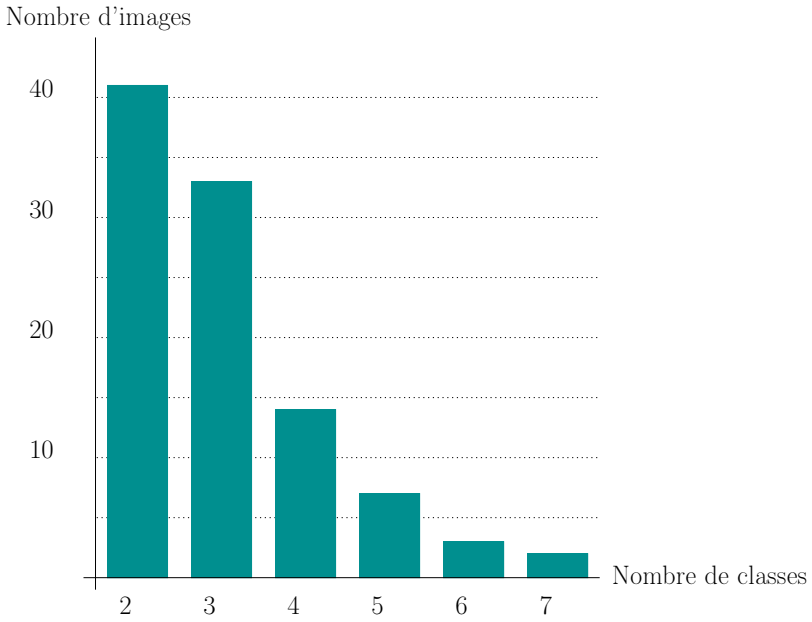


FIGURE 3.6 – Répartition du nombre de classes pour l'ensemble de données de référence R^{MC}.

Les temps d'exécution ont été obtenus sur un ordinateur de bureau, équipé d'un processeur *Intel Core i7* 2,6 GHz et d'une RAM de 16 Go. L'adéquation entre les probabilités estimées par une méthode et la réalité est calculée en faisant la moyenne du score $\mathcal{F}_{Err}$ obtenu pour tous les superpixels, de toutes les images d'un ensemble de données.

Les trois mesures sont calculées pour chaque couple de paramètres de chaque méthode, soit 256 tests pour le SVM et 100 tests pour la FA. Il n'est donc pas envisageable de mettre en place un protocole où un utilisateur donnerait les germes nécessaires à l'apprentissage de chacune des 356 variantes. Nous avons choisi d'automatiser la sélection des données d'apprentissage de la manière suivante :

1. pour chaque classe λ_i, nous créons un ensemble de données, constitué des superpixels x_j tels que $P_R(\lambda_i|x_j) > P_R(\lambda_k|x_j)\ \forall k \neq i$, c'est-à-dire ceux pour lesquels la classe la plus probable dans les données de référence est λ_i ;
2. nous classons les N_{λ_i} superpixels de cet ensemble par ordre d'apparition dans l'image, du haut vers le bas et de la gauche vers la droite ;
3. dans cet ensemble de superpixels classés, nous sélectionnons comme données d'apprentissage N_{app} superpixels, en laissant un écart constant de $N_{step} = \dfrac{N_{app}}{N_{\lambda_i}} - 1$ superpixels entre chaque paire de superpixels prélevés consécutivement.

Nous obtenons ainsi un ensemble de données d'apprentissage réparties uniformément (en termes de localisation) dans l'image. Nous avons fait des tests avec différentes valeurs pour N_{app}. Les

résultats que nous analyserons dans ce chapitre correspondent à $N_{app} = 100$. Les tendances demeurent les même en augmentant N_{app}.

3.5.3 Résultats avec le séparateur à vaste marge

Les tableaux 3.1 et 3.2 donnent l'évolution du taux d'erreur $\mathcal{F}_{err}$ pour la méthode de classification SVM, par rapport aux données de référence R^{MC} et R^{BI}. L'analyse de ces résultats montre que pour le couple de paramètre (ω_C, ω_Φ), un grand nombre de valeurs garantit un taux d'erreur inférieur à 5%. Ces valeurs forment un large palier et indiquent que le choix des paramètres ω_C et ω_ϕ n'est pas critique par rapport aux performances de la méthode SVM. Nous avons donc de bonnes raisons de penser qu'il n'y aura pas lieu de les modifier pour les adapter à d'autres jeux de données.

Pour les données de référence de R^{MC}, le taux d'erreur minimum est de 4%. Dans le cas des images de R^{BI} il descend autour de 1%. Les distributions de probabilité estimées semblent plus fidèles pour les images de R^{BI}, avec un taux d'erreur moins élevé. Ce résultat pourrait s'expliquer par le fait que la classification par SVM concerne, à l'origine, des problèmes à deux classes. L'extension à des problèmes multiclasses constitue, au moins dans le cadre de nos tests, une difficulté supplémentaire.

Les tableaux 3.3 et 3.4 contiennent les résultats concernant l'évolution du temps d'exécution lors de la phase d'entraînement. Ils montrent que lorsque le SVM est correctement paramétré, le temps nécessaire pour trouver les vecteurs de support diminue. Nous ne constatons pas de différence notable entre les deux ensembles de données.

Enfin, les tableaux 3.5 et 3.6 présentent les temps d'exécution lors de la phase d'exploitation. Nous pouvons en tirer les mêmes conclusions que pour la phase d'apprentissage, en constatant que, lorsqu'elle est correctement paramétrée, la méthode produit plus rapidement les distributions de probabilité des superpixels.

3.5.4 Résultats avec la forêt aléatoire

Les tableaux 3.7 et 3.8 donnent l'évolution du taux d'erreur $\mathcal{F}_{err}$ pour la méthode de classification FA, par rapport aux données de référence R^{MC} et R^{BI}. Les taux d'erreurs minimaux sont légèrement supérieurs à ceux de la méthode SVM, pour l'une et l'autre des deux ensembles de données de référence, avec de meilleures résultats pour les images de Santner *et al.*

Augmenter le nombre d'arbres de décision ainsi que les données d'apprentissage utilisées pour chacun d'entre eux permet d'améliorer la précision de la méthode. Ce résultat est d'autant plus cohérent que la dimension du descripteur est faible (seulement cinq caractéristiques) : il est donc peu probable que la configuration des arbres de décision varie fortement d'un arbre à l'autre. Dans ce contexte, seul le nombre d'arbres permet de fiabiliser le résultat en augmentant le nombre de suffrages. De même, le fait que le nombre de données d'apprentissage est faible explique qu'il faille en prélever un pourcentage élevé afin d'entraîner de manière pertinente les arbres de décision.

Malheureusement, donner des valeurs élevées à ces deux paramètres augmente les temps d'exécution, tant pour la phase d'apprentissage que pour celle d'exploitation. Les tableaux 3.9,

TABLEAU 3.1 – Taux d'erreur (exprimés en pourcentages) de la méthode SVM obtenus avec les données R^{MC}. Les valeurs sur la première ligne correspondent au paramètre ω_C, ceux sur la première colonne au paramètre ω_Φ.

	2^{-6}	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
2^{-6}	22	22	22	22	21	18	15	13	12	11	10	10	9	8	8	6
2^{-5}	23	22	22	21	18	15	13	12	11	10	10	9	8	7	6	5
2^{-4}	22	22	21	18	15	13	12	11	10	9	8	7	6	5	5	4
2^{-3}	22	21	18	15	12	11	11	10	9	7	6	5	5	4	4	4
2^{-2}	21	18	15	13	11	10	9	8	7	6	5	4	4	4	3	3
2^{-1}	19	15	12	11	9	8	7	6	5	4	4	4	3	3	3	3
1	16	12	10	9	7	6	5	4	4	3	3	3	3	3	3	3
2	13	10	8	7	6	5	4	4	3	3	3	3	3	3	3	3
2^2	13	8	6	5	4	4	3	3	3	3	3	3	3	3	3	3
2^3	14	7	5	4	4	3	3	3	2	2	2	2	3	3	3	3
2^4	16	10	5	4	3	3	3	2	2	2	2	3	3	3	3	3
2^5	17	15	7	3	3	3	2	2	2	2	2	3	3	3	3	3
2^6	19	18	14	6	3	3	3	3	3	3	3	3	3	3	3	3
2^7	18	18	17	13	5	3	3	3	3	3	3	3	3	3	3	3
2^8	20	20	19	18	13	6	4	4	4	4	4	4	4	4	4	4
2^9	20	20	21	20	20	14	7	7	7	7	7	7	7	7	7	7

TABLEAU 3.2 – Taux d'erreur (exprimés en pourcentages) de la méthode SVM obtenus avec les données R^{BI}. Les valeurs sur la première ligne correspondent au paramètre ω_C, ceux sur la première colonne au paramètre ω_Φ.

	2^{-6}	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
2^{-6}	10	11	11	10	10	10	9	8	8	8	7	7	6	6	5	4
2^{-5}	11	11	10	10	10	9	8	8	8	7	7	6	5	4	3	3
2^{-4}	10	11	10	9	8	8	8	8	7	6	5	4	4	3	3	2
2^{-3}	10	9	9	8	8	8	7	7	6	5	4	3	3	2	2	2
2^{-2}	9	9	8	8	7	7	6	5	4	3	3	2	2	2	2	2
2^{-1}	9	8	7	7	6	5	4	3	3	2	2	2	2	2	1	1
1	8	7	6	5	4	4	3	3	2	2	2	2	1	1	1	1
2	5	5	5	4	3	3	2	2	2	2	1	1	1	1	1	1
2^2	4	4	4	3	3	2	2	2	1	1	1	1	1	1	1	1
2^3	4	3	3	2	2	2	1	1	1	1	1	1	1	1	1	1
2^4	4	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
2^5	4	4	2	2	1	1	1	1	1	1	1	1	1	1	1	1
2^6	5	5	3	1	1	1	1	1	1	1	1	1	1	1	1	1
2^7	6	7	5	3	1	1	1	1	1	1	1	1	1	1	1	1
2^8	7	6	6	4	3	2	2	2	2	2	2	2	2	2	2	2
2^9	6	7	7	7	6	4	3	3	3	3	3	3	3	3	3	3

TABLEAU 3.3 – Temps d'exécution (en dixièmes de seconde) pour la phase d'entraînement de la méthode SVM obtenus avec les données R^{MC}. Les valeurs sur la première ligne correspondent au paramètre ω_C, ceux sur la première colonne au paramètre ω_Φ.

	2^{-6}	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
2^{-6}	4,4	4,2	4,4	4,2	4,2	4,2	3,9	3,5	2,9	2,5	2,1	1,9	1,8	1,8	2,1	2,5
2^{-5}	4,2	4,2	4,2	4,2	4,2	4	3,4	3	2,5	2,1	1,9	1,8	1,8	2	2,3	2,8
2^{-4}	4,2	4,2	4,2	4,2	3,9	3,5	2,9	2,4	2,1	1,8	1,7	1,7	1,8	2,1	2,6	3,4
2^{-3}	4,2	4,2	4,1	3,9	3,5	2,9	2,4	2,1	1,8	1,7	1,7	1,7	1,9	2,4	3	3,6
2^{-2}	4,2	4,2	4,3	3,5	3	2,5	2,1	1,8	1,6	1,5	1,6	1,7	2	2,6	3,1	4
2^{-1}	4,2	4,1	3,7	3	2,6	2,1	1,8	1,6	1,5	1,5	1,6	1,8	2	2,5	3,2	4,2
1	4,2	3,9	3,2	2,7	2,2	1,9	1,6	1,4	1,4	1,4	1,6	1,7	2,1	2,7	3,2	4
2	4,1	3,7	3	2,4	1,9	1,7	1,5	1,4	1,4	1,4	1,6	1,8	2,1	2,6	2,9	3,4
2^2	4,2	3,8	3	2,3	1,9	1,6	1,5	1,4	1,5	1,5	1,7	1,8	2	2,3	2,5	2,7
2^3	4,2	4	3,3	2,6	2,1	1,9	1,8	1,8	1,8	1,8	2	2	2,1	2,2	2,2	2,3
2^4	4,3	4,2	3,8	3,1	2,7	2,6	2,5	2,5	2,6	2,6	2,6	2,6	2,6	2,7	2,7	2,7
2^5	4,4	4,4	4,3	3,9	3,6	3,8	3,8	3,8	3,8	3,8	3,9	3,9	3,8	4	3,8	3,8
2^6	4,7	4,8	4,7	4,6	4,7	5,3	5,6	5,6	5,7	5,8	5,8	5,7	5,7	5,9	5,7	5,7
2^7	5,1	5,2	5,2	5,3	5,5	6,4	7,5	7,6	7,7	7,7	7,7	7,6	7,6	7,9	7,8	7,6
2^8	5,2	5,3	5,4	5,5	5,6	6,5	8,5	8,2	8,2	8,3	8,1	8,1	8,1	8,4	8,5	8,1
2^9	5,4	5,6	5,7	5,8	5,9	6,3	8,3	8,2	8,2	8,2	8,6	8,1	8,4	8,2	8,4	8,4

TABLEAU 3.4 – Temps d'exécution (en dixièmes de seconde) pour la phase d'entraînement de la méthode SVM obtenus avec les données R^{BI}. Les valeurs sur la première ligne correspondent au paramètre ω_C, ceux sur la première colonne au paramètre ω_Φ.

	2^{-6}	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
2^{-6}	1,6	1,6	1,6	1,7	1,6	1,7	1,5	1,4	1,3	1,2	1,2	1,1	1,1	1,3	1,4	1,7
2^{-5}	1,6	1,6	1,7	1,6	1,6	1,7	1,4	1,3	1,2	1,2	1,1	1,1	1,2	1,4	1,6	2,1
2^{-4}	1,7	1,7	1,7	1,6	1,6	1,4	1,3	1,2	1,2	1,1	1,1	1,1	1,2	1,5	1,9	2,5
2^{-3}	1,7	1,6	1,7	1,6	1,5	1,3	1,2	1,2	1,1	1,1	1	1,1	1,3	1,7	2,2	3,2
2^{-2}	1,7	1,6	1,7	1,4	1,3	1,2	1,1	1	1	1	1	1,1	1,3	1,9	2,6	3,4
2^{-1}	1,7	1,6	1,5	1,3	1,2	1,1	1	0,9	0,9	0,9	0,9	1,1	1,5	2	2,8	4
2^0	1,6	1,5	1,3	1,2	1,1	0,9	0,9	0,8	0,8	0,8	1	1,2	1,5	2	2,9	4,1
2^1	1,6	1,4	1,2	1,1	0,9	0,8	0,8	0,7	0,7	0,8	0,9	1,1	1,5	2	2,7	3,5
2^2	1,6	1,4	1,2	1	0,9	0,7	0,7	0,7	0,7	0,8	0,9	1,1	1,3	1,6	2,1	2,5
2^3	1,6	1,5	1,2	1	0,8	0,7	0,7	0,7	0,7	0,8	0,9	1	1,1	1,3	1,4	1,5
2^4	1,7	1,6	1,4	1,2	1	0,9	0,9	0,9	0,9	0,9	1	1	1,1	1,1	1,1	1,1
2^5	1,7	1,6	1,6	1,4	1,3	1,3	1,3	1,3	1,3	1,3	1,3	1,3	1,3	1,4	1,3	1,3
2^6	2	1,7	1,7	1,7	1,7	1,9	2,1	2,1	2,1	2,1	2	2	2,1	2,1	2,1	2,1
2^7	2	1,9	1,9	2	2,1	2,4	2,9	2,9	2,9	2,9	2,8	2,9	2,9	2,9	2,9	2,9
2^8	2	2	2	2,1	2,1	2,4	3,1	3,1	3,2	3,1	3,1	3,1	3,1	3,1	3,2	3,3
2^9	2	2,1	2,1	2,2	2,3	2,3	3	3,1	3	3,1	3,1	3	3,1	3,1	3,1	3,3

TABLEAU 3.5 – Temps d'exécution (en dixièmes de seconde) pour la phase d'exploitation de la méthode SVM obtenus avec les données R^{MC}. Les valeurs sur la première ligne correspondent au paramètre ω_C, ceux sur la première colonne au paramètre ω_Φ.

	2^{-6}	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
2^{-6}	5,2	5	5,2	5	5	5	4,7	4,5	4	3,6	3,2	2,9	2,5	2,3	2,1	1,9
2^{-5}	5	5	5	5	5	4,9	4,4	4,1	3,6	3,2	2,8	2,5	2,2	2,1	1,9	1,7
2^{-4}	5	5	4,9	5	4,8	4,4	4	3,6	3,2	2,8	2,5	2,3	2	1,9	1,7	1,5
2^{-3}	5	5	4,9	4,8	4,5	4	3,5	3,2	2,8	2,5	2,2	2	1,7	1,7	1,5	1,3
2^{-2}	5	5	5,1	4,5	4,1	3,6	3,2	2,8	2,5	2,2	2	1,7	1,5	1,5	1,3	1,2
2^{-1}	5	4,9	4,6	4,1	3,8	3,3	2,8	2,4	2,2	1,9	1,7	1,5	1,4	1,3	1,2	1,2
1	5	4,8	4,3	3,8	3,3	2,9	2,4	2,1	1,9	1,7	1,5	1,4	1,3	1,3	1,2	1,2
2	4,9	4,6	4	3,4	2,9	2,6	2,2	1,9	1,7	1,5	1,4	1,3	1,3	1,3	1,2	1,2
2^2	4,9	4,6	3,9	3,2	2,8	2,3	2	1,7	1,7	1,5	1,5	1,4	1,3	1,4	1,3	1,3
2^3	4,9	4,7	4	3,3	2,7	2,3	2	1,9	1,7	1,7	1,6	1,6	1,6	1,6	1,5	1,5
2^4	4,9	4,8	4,4	3,6	3	2,5	2,2	2,1	2,1	2	2	2	2	2	2	2
2^5	4,9	4,9	4,7	4,2	3,5	3,1	2,7	2,6	2,7	2,6	2,6	2,6	2,6	2,7	2,6	2,6
2^6	5	5,1	4,9	4,7	4,2	3,8	3,4	3,3	3,4	3,4	3,3	3,3	3,3	3,4	3,4	3,3
2^7	5	5,1	5	4,9	4,7	4,4	4,1	4	4,1	4,1	4,1	4	4	4,2	4,1	4
2^8	5,1	5,1	5	5	4,9	4,9	4,9	4,6	4,6	4,7	4,6	4,6	4,6	4,7	4,8	4,6
2^9	5,3	5,4	5,3	5,3	5,3	5,3	5,3	5,2	5,2	5,2	5,4	5,1	5,4	5,2	5,3	5,4

TABLEAU 3.6 – Temps d'exécution (en dixièmes de seconde) pour la phase d'exploitation de la méthode SVM obtenus avec les données R^{BI}. Les valeurs sur la première ligne correspondent au paramètre ω_C, ceux sur la première colonne au paramètre ω_Φ.

	2^{-6}	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
2^{-6}	1,7	1,7	1,7	1,9	1,7	1,9	1,7	1,6	1,4	1,3	1,2	1,1	1	1	0,9	0,8
2^{-5}	1,7	1,7	1,9	1,7	1,7	1,7	1,5	1,4	1,3	1,2	1,1	1	0,9	0,9	0,8	0,7
2^{-4}	1,9	1,9	1,9	1,7	1,7	1,5	1,4	1,3	1,2	1,1	1	0,9	0,8	0,8	0,7	0,6
2^{-3}	1,7	1,7	1,7	1,7	1,6	1,4	1,3	1,2	1,1	1	0,9	0,8	0,7	0,6	0,5	0,5
2^{-2}	1,9	1,7	1,7	1,6	1,4	1,3	1,1	1,1	1	0,8	0,7	0,6	0,6	0,5	0,4	0,4
2^{-1}	1,9	1,7	1,6	1,4	1,3	1,1	1	0,9	0,8	0,7	0,6	0,5	0,5	0,4	0,4	0,3
1	1,7	1,6	1,5	1,3	1,1	1	0,9	0,7	0,6	0,5	0,5	0,4	0,4	0,4	0,3	0,3
2	1,7	1,6	1,3	1,1	1	0,8	0,7	0,6	0,5	0,5	0,4	0,4	0,4	0,3	0,3	0,3
2^2	1,7	1,5	1,3	1	0,9	0,7	0,7	0,5	0,5	0,4	0,4	0,4	0,3	0,3	0,3	0,3
2^3	1,9	1,6	1,3	1	0,8	0,7	0,6	0,5	0,5	0,4	0,4	0,4	0,4	0,3	0,3	0,4
2^4	1,9	1,7	1,5	1,2	0,9	0,8	0,7	0,6	0,6	0,5	0,5	0,5	0,5	0,5	0,5	0,5
2^5	2	1,9	1,7	1,5	1,2	1	0,9	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
2^6	2,3	1,9	1,9	1,8	1,6	1,3	1,3	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2
2^7	2,1	2	2	2	1,9	1,7	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,6	1,7
2^8	2,1	2	2,1	2,1	2,1	2	1,9	1,9	2	1,9	1,9	1,9	1,9	1,9	1,9	2
2^9	2,1	2,1	2,2	2,1	2,2	2,1	2,1	2,2	2,1	2,1	2,2	2,1	2,1	2,1	2,2	2,3

TABLEAU 3.7 – Taux d'erreur (exprimés en pourcentages) de la méthode FA obtenus avec les données R^{MC}. Les valeurs sur la première ligne correspondent au pourcentage de données d'apprentissage utilisées pour chaque arbre de décision, ceux sur la première colonne au nombre d'arbres de décision.

	10	20	30	40	50	60	70	80	90	100
100	13	13	13	13	13	13	13	13	13	13
200	10	9	9	10	9	10	10	9	9	9
300	8	8	8	8	8	8	8	8	8	8
400	7	7	7	7	7	7	7	7	7	7
500	6	6	6	6	6	6	6	6	6	6
600	6	6	6	6	6	6	6	6	6	6
700	5	5	5	5	5	5	5	5	5	5
800	5	5	5	5	5	5	5	5	5	5
900	5	5	5	5	5	5	5	5	5	5
1000	5	5	5	5	5	5	5	5	5	5

TABLEAU 3.8 – Taux d'erreur (exprimés en pourcentages) de la méthode FA obtenus avec les données R^{BI}. Les valeurs sur la première ligne correspondent au pourcentage de données d'apprentissage utilisées pour chaque arbre de décision, ceux sur la première colonne au nombre d'arbres de décision.

	10	20	30	40	50	60	70	80	90	100
100	6	6	6	6	6	6	6	6	6	6
200	4	4	4	4	4	4	4	4	4	4
300	4	4	3	3	3	4	4	4	3	3
400	3	3	3	3	3	3	3	3	3	3
500	3	3	3	3	3	3	3	3	3	3
600	3	3	3	2	2	3	3	3	3	3
700	2	2	2	2	2	2	2	2	2	2
800	2	2	2	2	2	2	2	2	2	2
900	2	2	2	2	2	2	2	2	2	2
1000	2	2	2	2	2	2	2	2	2	2

3.10, 3.11 et 3.12 montrent que la durée nécessaire à la méthode FA pour analyser les données d'entraînement et estimer les distributions de probabilités est systématiquement supérieure à celle nécessaire à un SVM, sans pour autant permettre d'obtenir une meilleure précision.

3.5.5 Conclusion

L'ensemble des tests menés nous a conduit à privilégier l'utilisation d'une méthode de classification de type SVM. Comme valeurs des paramètres, nous avons utilisé $\omega_\Phi = 4$ et $\omega_C = 4$ durant l'ensemble de nos tests. Nous verrons que ces choix se sont révélés pertinents par la suite.

TABLEAU 3.9 – Temps d'exécution (en dixièmes de seconde) pour la phase d'entraînement de la méthode FA obtenus avec les données R^{MC}. Les valeurs sur la première ligne correspondent au pourcentage de données d'apprentissage utilisées pour chaque arbre de décision, ceux sur la première colonne au nombre d'arbres de décision.

	10	20	30	40	50	60	70	80	90	100
100	3,9	7,5	12,3	15,5	19,2	22,8	26,7	30,8	34,6	37,4
200	5,4	10,7	16,2	22,3	27,9	32,5	37,9	43,5	49,3	53,2
300	6,8	13,5	20,6	27,5	35	41,2	48,3	55,6	62,9	68,3
400	8,2	16,4	25,3	34,2	41,7	49,9	60,9	67	76	85,5
500	9,5	19,1	29,6	39,1	49	58,9	68,2	78,8	89,5	99,2
600	10,9	22	34,6	45,8	57,3	68	78,8	90,3	102,5	112,6
700	12,3	24,7	40,8	51,4	63,2	75,6	88,5	101,8	113,2	123,3
800	13,6	27,5	44,1	57,2	70,5	84	101,7	114,7	123,8	138
900	15,1	30,3	48,5	62,9	77,6	93,3	111,5	125,7	136,4	151,8
1000	16,4	34,7	52,1	67,7	86,5	101,6	121,6	137,7	149,3	166,6

TABLEAU 3.10 – Temps d'exécution (en dixièmes de seconde) pour la phase d'entraînement de la méthode FA obtenus avec les données R^{BI}. Les valeurs sur la première ligne correspondent au pourcentage de données d'apprentissage utilisées pour chaque arbre de décision, ceux sur la première colonne au nombre d'arbres de décision.

	10	20	30	40	50	60	70	80	90	100
100	1,9	3,7	5,6	7,6	9,4	11,3	12,9	14,8	16,7	19,6
200	2,6	5,2	8	10,6	13,7	15,9	17,9	20,8	23,8	26,7
300	3,4	6,5	9,9	13,3	17,6	20,2	22,4	26,8	30,3	34,2
400	3,9	7,7	11,8	15,8	19,6	25,3	26,8	31,1	36,1	42,9
500	4,5	9,2	13,8	18,5	23,3	27,6	31,3	36,6	41,1	51
600	5,1	10,6	15,8	21,4	27,1	31	36,7	41,6	47,1	55,6
700	5,7	11,7	17,8	24	29,3	36,7	40,7	47,1	53,4	60,6
800	6,4	13,5	19,7	26,6	33,9	40,2	46	54	62,8	68,2
900	7	14,1	21,6	29,3	38,8	45,9	49,9	57,6	69,1	73
1000	7,6	15,3	23,5	32,5	38,6	48,2	54,9	65,6	76,7	80

3.6 Optimisation dans un graphe de facteurs

Afin de sélectionner une méthode capable de minimiser $\mathcal{F}_{CAM}$, nous nous sommes appuyés sur les résultats de Kappes *et al.* [19]. Leur évaluation couvre 24 méthodes d'optimisation, reposant toutes sur une modélisation du problème par un graphe de facteurs. Le code des méthodes évaluées ayant été mis à disposition au travers de la bibliothèque OpenGm [3], nous étions assurés de pouvoir utiliser la même implémentation que Kappes *et al.*

Cette étude concerne une vingtaine d'applications de vision par ordinateur, dont un problème de segmentation sémantique très proche du problème de segmentation interactive qui nous oc-

3. https://github.com/opengm/opengm

TABLEAU 3.11 – Temps d'exécution (en dixièmes de seconde) pour la phase d'exploitation de la méthode FA obtenus avec les données R^{MC}. Les valeurs sur la première ligne correspondent au pourcentage de données d'apprentissage utilisées pour chaque arbre de décision, ceux sur la première colonne au nombre d'arbres de décision.

	10	20	30	40	50	60	70	80	90	100
100	4,5	8,6	13,8	17,7	21,9	26,1	30,4	35,1	39,4	42,6
200	5,2	10,3	15,8	22	27,3	31,8	37,1	42,6	48,4	52
300	5,7	11,3	17,5	23,5	29,8	35,3	41,4	47,8	54,6	59,2
400	6,1	12,2	19,1	26	31,6	38,1	46,9	51,5	59,2	66,6
500	6,3	12,8	20	26,7	33,6	40,7	47,5	55,1	64	69,3
600	6,6	13,3	21,8	29,1	35,9	43,2	50,2	57,9	67	72
700	6,9	13,7	24,2	29,7	36,5	43,9	52,1	60,4	67,1	72,3
800	6,9	14,1	23,9	30,4	37,8	45,3	56,9	64	67	75,4
900	7,1	14,4	24,7	31,4	38,9	47,3	58	65	68,9	77,4
1000	7,2	15,8	24,2	30,9	41,1	47,7	58,1	67,9	70,8	79,8

TABLEAU 3.12 – Temps d'exécution (en dixièmes de seconde) pour la phase d'exploitation de la méthode FA obtenus avec les données R^{BI}. Les valeurs sur la première ligne correspondent au pourcentage de données d'apprentissage utilisées pour chaque arbre de décision, ceux sur la première colonne au nombre d'arbres de décision.

	10	20	30	40	50	60	70	80	90	100
100	1,7	3,4	5,1	6,9	8,6	10,4	11,8	13,5	15,3	18,2
200	2	4	6,3	8,3	11	12,6	14,2	16,5	18,7	21,4
300	2,3	4,4	6,8	9,2	12,5	14,3	15,6	19	21,3	24,2
400	2,3	4,7	7,3	9,8	12,3	16,5	16,6	19,5	22,8	28,2
500	2,4	5	7,7	10,3	13,2	15,6	17,6	21,1	23,5	29,8
600	2,5	5,3	8	11,2	14,1	16,2	18,8	21,9	24,9	30,5
700	2,6	5,3	8,3	11,3	14,3	18,2	19,7	23	26	30,7
800	2,6	5,9	8,4	11,6	15,4	18,7	20,8	25	30	32,4
900	2,7	5,5	8,7	12	16,8	20,1	21	24,1	31,2	32
1000	2,8	5,6	8,8	13	15	19,1	22	27,2	33,1	33,6

cupe, avec notamment des superpixels comme primitives visuelles. Parmi les méthodes obtenant les meilleurs scores, la méthode α-fusion, proposée par Lempitsky *et al.* [22] garantit également les temps d'exécution les plus faibles.

3.6.1 Graphe de facteurs

Un graphe de facteurs est un graphe biparti représentant la factorisation d'une fonction et permettant la recherche efficace des optima de cette dernière. Soit $\mathcal{F}(X_1, \cdots, X_{N_\mathcal{F}})$, une fonction de $N_\mathcal{F}$ variables. Nous supposons qu'il existe un ensemble de N_f fonctions $\{f_1, \cdots, f_{N_f}\}$ telles

que $\mathcal{F}$ puisse être exprimée comme le produit de ces fonctions :

$$\mathcal{F}(X_1, \cdots, X_{N_\mathcal{F}}) = \prod f_i(X_j, \cdots, X_k). \tag{3.10}$$

La fonction $\mathcal{F}$ peut alors être représentée par un graphe $\mathcal{G}^\mathcal{F} = < X^\mathcal{F}, F^\mathcal{F}, E^\mathcal{F} >$ avec :

- $X^\mathcal{F} = \{X_1, \cdots, X_{N_\mathcal{F}}\}$, un ensemble de $N_\mathcal{F}$ sommets correspondant aux variables de $\mathcal{F}$;
- $F^\mathcal{F} = \{f_1, \cdots, f_{N_f}\}$, un ensemble de N_f sommets correspondant aux facteurs de $\mathcal{F}$;
- $E^\mathcal{F}$, un ensemble d'arêtes reliant un sommet de $X^\mathcal{F}$ à un sommet de $F^\mathcal{F}$.

Les ensembles $X^\mathcal{F}$ et $F^\mathcal{F}$ sont respectivement appelés *ensemble des sommets de variables* et *ensemble des sommets de facteurs*. Le graphe $\mathcal{G}^\mathcal{F}$ est biparti car il existe une partition de son ensemble de sommets en deux sous-ensembles telle que chaque arête de $E^\mathcal{F}$ ait une extrémité dans l'un des sous-ensembles et l'autre extrémité dans l'autre sous-ensemble. Un exemple de graphe de facteurs représentant la factorisation d'une fonction est donné sur la figure 3.7.

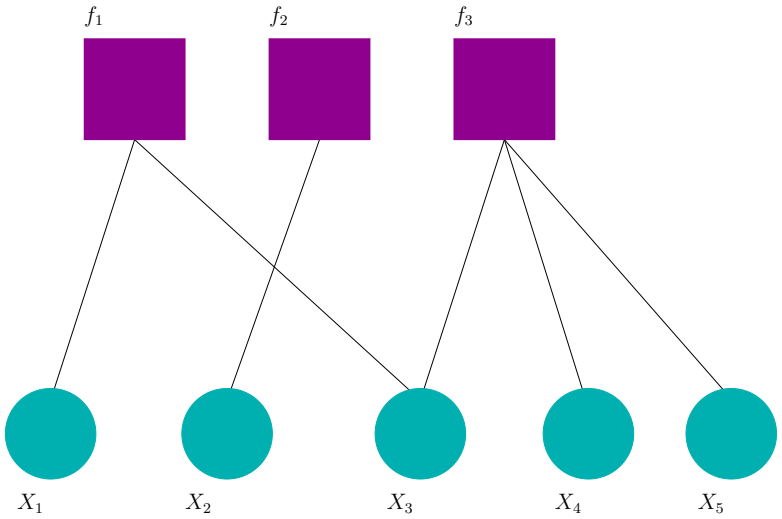


FIGURE 3.7 – Graphe de facteur $\mathcal{G}^\mathcal{F} = < X^\mathcal{F}, F^\mathcal{F}, E^\mathcal{F} >$ correspondant à la factorisation : $\mathcal{F}(X_1, X_2, X_3, X_4, X_5) = f_1(X_1, X_3) f_2(X_2) f_3(X_3, X_4, X_5)$.

Koppler *et al.* [21] montrent que les graphes de facteurs permettent une représentation plus fine et plus explicite de la structure sous-jacente d'un problème d'optimisation que les CAM.

3.6.2 Reformulation de la fonction à minimiser

Afin de factoriser la fonction de coût $\mathcal{F}_{CAM}$ (équation 3.1), nous nous sommes appuyés sur le fait que l'ensemble de labels C qui minimise $\mathcal{F}_{CAM}$ minimise également :

$$\mathcal{F}_{GF}(C, I) = \exp\Big(\sum_{i=1}^{N} \mathcal{F}_D(x_i, c_i) \; + \sum_{(x_i, x_j) \in E} \omega(x_i) \mathcal{F}_R(x_i, c_i, x_j, c_j)\Big). \tag{3.11}$$

L'introduction de la fonction exponentielle nous permet d'exprimer simplement notre fonction de coût sous la forme d'un produit de fonctions :

$$\mathcal{F}_{GF}(C,I) = \prod_{i=1}^{N} \exp(\mathcal{F}_D(x_i,c_i)) \prod_{(x_i,x_j)\in E} \exp(\omega(x_i)\mathcal{F}_R(x_i,c_i,x_j,c_j)). \tag{3.12}$$

3.6.3 Méthode α-fusion

Soit Λ un ensemble de labels possibles, X un ensemble de variables et C l'ensemble des labels attribués aux variables de X. Les relations entre les variables de X peuvent être représentées par un graphe $\mathcal{G} = < V, E >$, avec :

- l'ensemble de sommets $V = \{v_1 \cdots v_{N_x}\}$ associant à chaque variable x_i un sommet v_i ;
- l'ensemble d'arêtes E, où chaque arête $e_{i,j}$ relie une paire de sommets (v_i, v_j).

L'algorithme de Lempitsky *et al.* [22] recherche l'ensemble de labels C^* minimisant une fonction de coût de la forme :

$$\mathcal{F}_L(C,I) = \prod_{v_i\in V} \mathcal{F}_v(x_i,c_i) \prod_{(e_{i,j})\in E} \omega(x_i)\mathcal{F}_e(x_i,c_i,x_j,c_j). \tag{3.13}$$

Dans de nombreux cas, trouver une solution optimale est un problème NP complet et seule la recherche d'une approximation du minimum de $\mathcal{F}_L$ est envisageable.

Si nous nous plaçons dans le cas d'un problème à 2 classes avec $\Lambda = \{\lambda_0, \lambda_1\}$, l'algorithme de coupe minimale de Boykov *et al.* [4], décrit dans la section 2.4.3, permet dans la majorité des cas de trouver une solution approchée satisfaisante. En particulier, si $\mathcal{F}_L$ est une fonction sub-modulaire, c'est-à-dire que $\mathcal{F}_e(x_i,\lambda_0,x_j,\lambda_0) + \mathcal{F}_e(x_i,\lambda_1,x_j,\lambda_1) \leq \mathcal{F}_e(x_i,\lambda_0,x_j,\lambda_1) + \mathcal{F}_e(x_i,\lambda_1,x_j,\lambda_0)$, alors le minimum global de $\mathcal{F}_L$ peut être obtenu.

Dans le cas où $\mathcal{F}_L$ n'est pas une fonction sub-modulaire, l'ensemble de labels optimal C^* peut être trouvé en partie sous la forme d'un ensemble $C^{\frown}$ où à chaque sommet est associé soit l'un des deux labels possibles (λ_0 ou λ_1), soit le label $\lambda_?$, qui indique qu'aucun label pertinent n'a pu être trouvé.

Soit l'ensemble $C^\star$ obtenu en attribuant arbitrairement λ_0 ou λ_1 aux sommets ayant pour label $\lambda_?$. Selon Lempitsky *et al.* [22] :

$$\mathcal{F}_L(C^\star, I) \leq \mathcal{F}_L(C^{\frown}, I), \tag{3.14}$$

ce qui fait de $C^\star$ un ensemble de label pertinent pour un grand nombre de problèmes.

Dans le cas d'un problème multiclasse où $\Lambda = \{\lambda_0, \cdots, \lambda_{N_\Lambda}\}$, une approximation satisfaisante du minimum global de $\mathcal{F}_L$ peut être obtenue en trouvant deux ensembles sous-optimaux de labels attribués aux sommets de $\mathcal{G}$, C^0 et C^1, puis en les combinant. Par *ensembles sous-optimaux*, nous entendons que ni C^0, ni C^1, ne constituent une solution optimale au problème posé.

Lempitsky *et al.* [22] nomment *« procédure »* le traitement indiquant comment combiner C^0 et C^1, de manière à obtenir un nouvel ensemble de labels $C^{0\cup1}$ tel que :

$$\left(\mathcal{F}_L(C^{0\cup1}, I) \leq \mathcal{F}_L(c^0, I)\right) \ \wedge \ \left(\mathcal{F}_L(c^{0\cup1}, I) \leq \mathcal{F}_L(c^1, I)\right). \tag{3.15}$$

En particulier, ils proposent la procédure α-*fusion*. Cette procédure consiste à se ramener à un problème à 2 classes, avec comme ensemble de labels possibles $\Lambda^\dagger = \{\lambda_0^\dagger, \lambda_1^\dagger\}$, où le label $\lambda_0^\dagger$ indique que $c_i^{0\cup1} = c_i^1$ tandis que le label $\lambda_1^\dagger$ indique que $c_i^{0\cup1} = c_i^1$.

La fonction de coût minimisée à l'aide d'un algorithme de coupe minimale, $\mathcal{F}_\dagger$, correspond à la fonction de coût initiale, $\mathcal{F}_L$, où seul varie l'ensemble des labels possibles qui est réduit à $\Lambda^\dagger$:

$$\mathcal{F}_\dagger(c^{0\cup1}, I) = \prod_{v_i \in V} \mathcal{F}_v(x_i, c_i^{0\cup1}) \prod_{(e_{i,j}) \in E} \omega(x_i) \mathcal{F}_e(x_i, c_i^{0\cup1}, x_j, c_j^{0\cup1}). \tag{3.16}$$

Soit C^n, un ensemble de labels attribués aux sommets de $\mathcal{G}$. Soit C^{λ_i} un autre ensemble de labels, obtenu en attribuant à chaque sommet le même label $\lambda_i \in \Lambda$. La procédure α-fusion décrite précédemment permet de diminuer la valeur de $\mathcal{F}_L$ en combinant C^n et C^{λ_i} pour obtenir $C^{n+1} = C^{n\cup\lambda_i}$. Ce traitement est répété jusqu'à ce que la procédure α-fusion ne permette plus de diminuer la valeur de $\mathcal{F}_L$, c'est-à-dire que $\mathcal{F}_L(c^n, I) \leq \mathcal{F}_L(c^{n+1}, I)$.

3.7 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode de segmentation interactive reposant sur la minimisation d'une fonction de coût capable de produire une segmentation cohérente vis-à-vis des caractéristiques de l'image, des germes donnés par l'utilisateur et des relations de voisinage entre les superpixels. Le code source de cette méthode est disponible en ligne [4].

Nous avons jusqu'ici laissé de côté le choix d'un algorithme de sur-segmentation produisant les superpixels. Or, cette décision est cruciale pour le bon fonctionnement de $S\alpha F$. En particulier elle doit éviter deux écueils :

- le temps nécessaire pour produire les superpixels ne doit pas être supérieur au temps gagné en les utilisant à la place des pixels, c'est-à-dire que l'algorithme de sur-segmentation doit être en mesure de fournir un résultat en quelques secondes pour une image de plusieurs millions de pixel ;
- les superpixels doivent autant que possible éviter de chevaucher deux objets différents : en effet, l'étape de segmentation de $S\alpha F$ ne permettant pas d'attribuer les pixels d'un même superpixel à plusieurs classes, il ne sera pas possible de corriger ce type d'erreur, quel que soit le nombre de germes ajoutés par l'utilisateur.

Le chapitre suivant traite de manière détaillée cette problématique.

4. http://image.ensfea.fr/saf/

Chapitre 4

Évaluation des méthodes de sur-segmentation

4.1 Introduction

Nous devons l'introduction et la définition du concept de superpixel à Ren *et al.* [42]. Petites régions homogènes et régulières, les superpixels forment une sur-segmentation de l'image. Un exemple où les contours des superpixels sont tracés en blanc est donné sur la figure 4.1.

L'utilisation des superpixels comme primitives visuelles présente de nombreux avantages. Notamment :

- les superpixels constituent des primitives visuelles bien moins nombreuses que les pixels, ce qui permet de réduire considérablement les temps d'exécution de l'algorithme qui les manipule ;
- les superpixels, parce qu'ils correspondent à des ensembles de pixels, fournissent une information plus riche (il est notamment possible de décrire leur texture ou leur géométrie).

Durant les vingt dernières années, les superpixels ont été intégrés de manière fructueuse au sein de nombreux travaux, tels que la méthode de segmentation sémantique proposée par Gould *et al.* [17] ou, plus récemment, celle de Zang *et al.* [61] sur la catégorisation fine d'images. Ces succès ne doivent cependant pas masquer les difficultés inhérentes à la sur-segmentation et qui se cristallisent autour de la recherche d'un compromis capable de satisfaire au mieux trois objectifs :

- la production de superpixels ne chevauchant pas les contours de l'image ;
- l'obtention d'un faible nombre de superpixels afin de compresser au maximum la représentation de l'image ;

- l'obtention d'un faible temps de calcul, garantissant que le bénéfice à utiliser les superpixels à la place des pixels ne soit pas compensé par le temps nécessaire pour les produire.

Or, ces trois propriétés se révèlent contradictoires. En effet, la production de superpixels peu nombreux – donc regroupant un nombre plus important de pixels – augmente le risque de commettre des erreurs. Réduire cette erreur requiert une complexification des algorithmes, qui se répercute alors sur les temps de calcul. Ainsi, aujourd'hui encore, la sur-segmentation demeure un champ de recherche actif, où de nouvelles propositions sont faites régulièrement [1, 8, 29].

FIGURE 4.1 – Exemple de sur-segmentation. Les contours des superpixels sont tracés en blanc.

4.2 Quelques exemples pratiques d'utilisation des superpixels

L'analyse des méthodes intégrant une étape de sur-segmentation nous semble un préliminaire indispensable pour déduire les propriétés qui doivent être satisfaites par un algorithme de sur-segmentation, afin que celui-ci puisse être utilisé de manière fructueuse. La profusion des travaux reposant sur une étape sur-segmentation interdit toutefois de les décrire de manière exhaustive. Aussi avons-nous choisi trois applications représentatives.

4.2.1 Segmentation sémantique

Publiée en 2008, la méthode de segmentation sémantique de Gould *et al.* [17] constitue une étape notable, tant pour le domaine de la segmentation sémantique que pour celui de la sur-segmentation. Afin d'identifier et de localiser les différents objets composant une photographie, Gould *et al.* proposent une méthode capable, à partir d'un ensemble de données d'apprentissage, de déduire les caractéristiques visuelles d'une catégorie d'objet ainsi que ses relations spatiales avec d'autres types d'objets. Ainsi, la probabilité qu'un ensemble de pixels appartienne à la classe *vache* dépend à la fois de sa couleur, de sa texture et de la présence de pixels voisins attribués à des classes considérées comme probables, telle que *l'herbe*.

Les motivations évoquées pour justifier l'utilisation de superpixels à la place des pixels rejoignent celles avancées par Ren *et al.* [42] : la réduction du nombre de primitives manipulées, donc la diminution du temps de calcul global de l'application ainsi que l'utilisation de primitives visuelles contenant une information plus riche. Ainsi, chaque superpixel est décrit à la fois par sa couleur moyenne (espaces RGB et Lab), sa texture (textons) et sa géométrie.

Les superpixels étant organisés en graphe, la probabilité pour un superpixel d'appartenir à chacune des classes dépend de ses caractéristiques visuelles propres ainsi que de la probabilité de ses voisins d'appartenir à la même classe ou à une classe fréquemment voisine. Or, contrairement aux pixels qui sont organisés en grille régulière, les superpixels forment un graphe sur lequel peu d'hypothèses simplificatrices peuvent être faites. En particulier, la complexité du voisinage d'un superpixel, ou, en d'autres termes, son nombre de voisins, peut avoir un impact significatif sur le temps d'exécution.

4.2.2 Détection d'obstacle dans un environnement 3D

Dans leur article [39], Petrovai *et al.* décrivent un système complet d'assistance à la conduite, capable de générer une représentation 3D de l'environnement dans lequel évolue le véhicule et de s'en servir pour détecter les obstacles présents sur la route, afin d'alerter le conducteur. Pour le rendre indépendant du véhicule, ils utilisent une tablette équipée de deux caméras. Ainsi, l'algorithme de détection d'obstacles que Petrovai *et al.* [39] décrivent répond à la nécessité de produire des résultats en temps réel – permettant au conducteur de réagir promptement – avec des contraintes matérielles fortes, les tablettes actuelles disposant de capacités de mémoire et de calcul limitées par rapport à celles d'un ordinateur.

L'environnement 3D est déduit à partir des deux images prises par la tablette, grâce à une méthode de stéréovision : en identifiant pour chaque pixel de l'image de gauche son correspondant dans l'image de droite, la profondeur du point 3D correspondant à ces deux pixels est déduite. Pour des raisons d'efficacité, seules quelques correspondances peuvent être identifiées entre les pixels des deux images, par comparaison de leurs caractéristiques. Elles permettent une représentation 3D partielle, qui sera ensuite étendue sur la base de la sur-segmentation en superpixels. Petrovai *et al.* commencent par fusionner ces superpixels en fonction de leurs similarités en termes de couleur. Puis ils utilisent l'information 3D obtenue à l'étape précédente ainsi qu'une analyse de la géométrie et du niveau de gris moyen des superpixels pour identifier la route et les obstacles, puis pour calculer leur distance par rapport au véhicule.

Dans le cadre des travaux de Petrovai *et al.*, l'utilisation d'une méthode de sur-segmentation est clairement motivée par la volonté de produire une application finale aussi rapide que possible. Afin d'éviter que des erreurs ne soient commises au moment de la sur-segmentation et propagées, voire amplifiées lors des étapes suivantes, l'algorithme de sur-segmentation est paramétré pour produire des superpixels dont l'aire avoisine les 170 pixels.

4.2.3 Classification fine d'images

La classification fine d'images consiste à identifier, à partir d'une photographie, la présence d'objets appartenant à une même catégorie et présentant d'importantes similarités visuelles et sémantiques. Elle trouve de nombreuses applications en biologie, par exemple pour identifier plusieurs espèces d'oiseaux, ainsi qu'en agronomie, pour distinguer des plantes malades de celles en bonne santé. Une part conséquente des travaux de recherche dans ce domaine consiste à proposer des descripteurs capables de différencier chaque objet et ce, quel que soit le domaine de classification (champignons, marques de voitures, etc.). Comme les bases de données d'apprentissage peuvent, selon les domaines, contenir un nombre restreint de spécimens, des méthodes reposant sur une étape lourde d'apprentissage à partir d'un ensemble de données de référence composé de plusieurs centaines d'images ne sont pas toujours envisageables.

La méthode de Zhang *et al.* [61] repose sur l'utilisation de primitives visuelles correspondant à de petits graphes qu'ils nomment *graphlets*. Chaque sommet de ces graphlets correspond à un superpixel, donc un ensemble de pixels. Chaque paire de sommets ayant des pixels adjacents est associée par une arête. Les graphlets sont ensuite caractérisés par leurs histogrammes de couleur en neuf dimensions [48], leurs histogrammes de gradients orientés [9] et leurs structures [1]. Pour des raisons de complexité algorithmique, il est souhaitable que les graphlets ne soient composés que d'un petit nombre de sommets. Zhang *et al.* utilisent l'algorithme SLIC [60] proposé par Achanta *et al.* [1] pour produire les superpixels. Trois niveaux de sur-segmentation sont produits, le nombre de superpixels, et donc leur aptitude à capter les détails fins, augmentant à chaque niveau. Ainsi, les graphlets dont les sommets correspondent aux superpixels du premier niveau permettent de décrire les composantes les plus grandes de chaque objet (par exemple le corps d'un oiseau) tandis que ceux comprenant des superpixels du dernier niveau se focalisent sur des éléments plus petits (tels que le bec ou les pattes). Les graphlets comportent un nombre similaire de sommets, quel que soit le niveau de sur-segmentation.

Cette utilisation des superpixels est intéressante pour au moins trois raisons. Tout d'abord, l'un des intérêts des superpixels réside dans leur faculté à suivre les bordures des objets bien plus finement que des patchs rectangulaires. Avec des séries de photos où le fond peut varier de manière significative, le fait d'utiliser un descripteur qui permette de se focaliser sur l'objet à identifier n'est sans doute pas anecdotique dans les excellents résultats obtenus par la méthode de Zhang *et al.* Ensuite, chaque partie des objets est décrite en utilisant les relations de voisinage des superpixels. Enfin, le fait que trois niveaux de sur-segmentation soient utilisés découle de la difficulté actuelle pour un algorithme de sur-segmentation d'adapter l'aire des superpixels au niveau de détail.

1. Par structure, nous entendons la manière dont ces sommets sont connectés entre eux.

En effet, pour capter les détails les plus fins, de petits superpixels sont nécessaires. Comme la majorité des méthodes de sur-segmentation produisent des superpixels de taille quasi uniforme, l'ensemble de l'image sera découpé en de nombreux superpixels. Au niveau des composantes plus grandes, ce nombre important de superpixels se traduit par des graphlets comprenant de nombreux nœuds, ce qui augmente significativement la complexité des traitements ultérieurs.

4.3 Évaluation d'un algorithme de sur-segmentation

4.3.1 Propriétés à évaluer

De l'étude des trois applications précédentes, cinq propriétés permettant d'évaluer la qualité d'une sur-segmentation peuvent être énoncées.

Propriété 1 (validité) *une sur-segmentation est considérée comme valide si elle constitue une partition de l'image en sous-ensembles connexes.*

Propriété 2 (adhérence aux contours) *un superpixel ne doit pas recouvrir différents objets de l'image.*

Propriété 3 (concision) *un algorithme de sur-segmentation doit produire aussi peu de superpixels que possible.*

Propriété 4 (simplicité) *le nombre de voisins d'un superpixel doit être aussi petit que possible.*

Propriété 5 (rapidité) *un algorithme de sur-segmentation doit avoir un temps d'exécution aussi faible que possible.*

Pour que ces propriétés soient précises, il reste à définir ce que nous entendons par contour. À la différence d'autres travaux sur la sur-segmentation [29] nous ne pensons pas que ce terme doive se limiter aux contours en termes de couleur ou de niveau de gris. Au contraire, il nous semble essentiel de prendre en compte la texture, information dont le bénéfice est connu de longue date [63].

La propriété 1 indique que tout pixel doit être associé à un et un seul superpixel et qu'il existe, pour toute paire de pixels (p_i, p_j) appartenant à un même superpixel $\mathbf{s}_k$, un chemin de pixels voisins tous inclus dans $\mathbf{s}_k$, reliant p_i à p_j.

La propriété 2 correspond à la définition d'une sur-segmentation : les contours des superpixels incluent les contours des objets dans l'image, mais sont plus nombreux que ces derniers.

La propriété 3 est liée au fait que les superpixels permettent un traitement plus efficace de l'information visuelle contenue dans l'image : moins ils sont nombreux, plus les traitements qui leur sont appliqués sont rapides.

La propriété 4 s'explique par le fait que, contrairement aux pixels, les superpixels ne forment pas un graphe avec une structure régulière en grille, ce qui induit des calculs supplémentaires lorsque le voisinage est pris en compte. Il est donc souhaitable que le nombre moyen de voisins d'un superpixel ne soit pas trop élevé.

La propriété 5 se justifie par le fait que la sur-segmentation d'une image est, dans la majorité des cas, un traitement préalable réalisé dans le but d'abaisser les temps d'exécution d'une méthode. Or le temps nécessaire pour produire les superpixels ne doit pas être supérieur au gain de temps apporté par leur utilisation à la place des pixels.

4.3.2 Évaluations précédentes

La première évaluation des méthodes de sur-segmentation a été réalisée en 2012 par Achanta *et al.* [1]. Elle compare les méthodes de Ren *et al.* [42], Felzenszwalb *et al.* [13], Vedaldi *et al.* [52], Levinshtein *et al.* [23], Veksler *et al.* [53] et Achanta *et al.* [1]. Les six méthodes sont testées sur les 100 images de la base de donnée de Berkeley [30], BSD [2]. L'évaluation d'Achanta *et al.* se concentre sur les propriétés 2, 3 et 5.

Afin de quantifier l'adhérence aux contours d'une sur-segmentation (propriété 2), Achanta *et al.* utilisent deux mesures : le taux d'erreur de sous-segmentation ($\mathcal{F}_{ES}$) et le taux de rappel sur les contours ($\mathcal{F}_{RC}$). L'une comme l'autre sont calculées par rapport à une segmentation de référence R à laquelle est comparée la sur-segmentation obtenue. Soit $\mathbb{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_{N_\mathbb{S}}\}$ l'ensemble des superpixels. Le calcul du taux d'erreur de sous-segmentation repose sur la recherche, pour chaque région r_i présente dans R, de l'ensemble des superpixels nécessaires pour le recouvrir et sur le nombre de pixels de cet ensemble qui débordent de l'objet. Il est défini par :

$$\mathcal{F}_{ES}(\mathbb{S}, R) = \frac{1}{N_I} \sum_{r_i \in R} \sum_{\mathbf{s}_j \cap r_i \neq \emptyset} \min(|\mathbf{s}_j \cap r_i|, |\mathbf{s}_j - r_i|) \tag{4.1}$$

avec N_I le nombre de pixels de l'image. Le résultat obtenu est compris entre 0 et 1, un score de 0 correspondant à une sur-segmentation sans erreur.

Le taux de rappel sur les contours permet de vérifier que les contours des objets présents dans R se retrouvent dans $\mathbb{S}$. Soit C_R l'ensemble des points de contour de la segmentation de référence et $C_\mathbb{S}$ l'ensemble des points de contour de la sur-segmentation. Si nous faisons l'hypothèse que, pour chaque pixel, nous savons sans aucun doute s'il appartient ou non aux contours, nous pouvons évaluer la qualité d'une sur-segmentation en calculant la proportion de points de contour dans la segmentation de référence qui correspondent à des points de contour dans la sur-segmentation :

$$\mathcal{F}_{RC}(\mathbb{S}, R) = \frac{|C_\mathbb{S} \cap C_R|}{|C_R|}. \tag{4.2}$$

Concrètement, même pour un être humain, il n'est pas toujours aisé de déterminer au pixel près la position des contours dans une image. Achanta *et al.* se donnent une marge de deux pixels. Le score obtenu est compris dans l'intervalle $[0, 1]$, un score de 1 indiquant que tous les contours de R se retrouvent dans $\mathbb{S}$.

En outre, Achanta *et al.* réalisent une étude de la complexité de chaque méthode ainsi que de ses temps d'exécution (propriété 5). Les scores pour les mesures $\mathcal{F}_{ES}$ et $\mathcal{F}_{RC}$, ainsi que les temps d'exécution sont mis en relation avec le nombre de superpixels produits par la méthode,

2. https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

ce qui permet d'identifier les algorithmes capables de satisfaire aux mieux les propriétés 2 et 5 sous la contrainte de la propriété 3.

Trois ans plus tard, une seconde évaluation a été menée par Stutz *et al.* [49]. Leur première contribution consiste en l'évaluation de 7 méthodes supplémentaires : Liu *et al.* [26], Zhang *et al.* [62], Conrad *et al.* [8], Van *et al.* [51], Tang *et al.* [50], Weikersdorfer *et al.* [55] et Papon *et al.* [38]. Leur seconde contribution réside dans l'utilisation d'une deuxième base de données, celle de l'université de New York [47], NYU [3], qui comprend 400 images. Comme les tailles des photographies dans les deux bases de données ne sont pas identiques (481×321 pixels pour BSD, 640×480 pixels pour NYU), Stutz *et al.* proposent de modifier le seuil de tolérance de la mesure $\mathcal{F}_{RC}$ en autorisant tous les pixels à une distance de 0,007 $5 \times diag$, $diag$ étant la longueur de la diagonale de l'image.

Dans l'évaluation d'Achanta *et al.* [1], les méthodes de Felzenszwalb *et al.* [13] et d'Achanta *et al.* [1] obtiennent les meilleurs scores. L'étude de Stutz *et al.* [49] confirme ce résultat et révèle que les méthodes de Vedaldi *et al.* [52], Conrad *et al.* [8] et Liu *et al.* [26] obtiennent des scores similaires à ceux de Felzenszwalb *et al.* [13] et d'Achanta *et al.* [1]. Sur la base de données BSD, les scores optimaux sont atteints avec des sur-segmentations comprenant un millier de superpixels. Pour les cinq méthodes précédemment citées, le taux d'erreur de sous-segmentation est inférieur ou égal à 0,04 et le taux de rappel supérieur ou égal à 0,99. Pour la base de données NYU, il faut environ 1500 superpixels à ces mêmes méthodes pour obtenir des scores similaires. Le taux d'erreur de sous-segmentation est inférieur ou égal à 0,09 et le taux de rappel, là encore, est supérieur ou égal à 0,99. Pour les deux bases de données, ces performances sont obtenues pour un temps d'exécution d'environ une seconde par image, sur un ordinateur équipé de processeur *Intel Core i7-3770* 3,4 GHz et d'une RAM de 16 Go.

4.3.3 Nécessité d'une évaluation complémentaire

Les résultats obtenus par les deux évaluations précédemment décrites [1, 49] pourraient nous amener à la conclusion que le problème de la sur-segmentation d'une image est actuellement résolu, cinq méthodes obtenant des scores proches d'un résultat parfait, avec par exemple un taux de rappel de 0,99 alors que le score maximale atteignable est de 1. Ces deux évaluations s'avèrent néanmoins critiquables sur deux points :

- elles se cantonnent à l'utilisation de petites images de tailles similaires (de l'ordre de quelques milliers de pixels) qui ne permettent pas d'apprécier le passage à l'échelle des algorithmes ;
- elles se limitent à évaluer l'aptitude des méthodes à produire une sur-segmentation comprenant peu d'erreurs pour un nombre de superpixels donné, sans s'intéresser à leur capacité à s'adapter à la complexité de l'image.

Par *passage à l'échelle* nous entendons le fait qu'un algorithme conserve ses performances quelle que soit la taille de l'image fournie en entrée. En particulier, si les scores évoqués dans la section précédente sont excellents, il est justifié de se demander si les algorithmes évalués

3. http://cs.nyu.edu

conservent la même efficacité avec des images de plusieurs millions de pixels, bien plus proches de la taille des photographies obtenues aujourd'hui avec un appareil photo standard ou un smartphone.

La complexité de l'image désigne ici le nombre d'objets qu'elle contient, ainsi que leur niveau de détail. Nous donnons une illustration de ce concept sur la figure 4.2, au travers de deux images correspondant à deux situations extrêmes. Dans le premier cas (figure 4.2a), nous nous attendons à ce qu'une sur-segmentation en assez peu de superpixels fournisse une compression raisonnable de l'image, où aucune information visuelle importante ne soit perdue. Dans le second cas (figure 4.2b), la présence de nombreux petits objets, tels que les arbres, ainsi que le niveau de détail sur les bâtiments impose de produire des superpixels plus petits (donc plus nombreux), afin d'éviter que plusieurs superpixels ne chevauchent des objets différents. Cette notion de complexité globale d'une image peut aisément être étendue aux parties d'une même image, à l'instar de l'exemple donné par la figure 4.2c. Ainsi, si la zone correspondant au ciel peut être sur-segmentée en un nombre restreint de grands superpixels, afin de capter les détails des bâtiments, il est nécessaire de produire de nombreux petits superpixels.

Dans les trois cas de figure présentés, il est préférable de disposer d'un algorithme de sur-segmentation capable de faire varier automatiquement la taille des superpixels en fonction de la complexité de la zone dans laquelle ils se situent, sans qu'une modification *ad hoc* de ses paramètres ne soit nécessaire. Nous désignons cette capacité d'un algorithme par le terme d'*adaptabilité*. Par la suite, nous verrons que l'analyse de l'adaptabilité d'une méthode de sur-segmentation peut se faire par l'étude de l'évolution conjointe des propriétés 2 et 3.

4.4 Proposition d'un nouveau protocole expérimental

4.4.1 Une nouvelle base de données : HSID

L'ensemble de données de référence HSID, que nous avons construit, a été créé à partir de 100 images issues de la base de données de Wikimedia Commons [4]. Nous avons sélectionné les photographies pour qu'elles présentent une large variété de difficultés : flou, bruit, ombre, faible contraste, reflet. Pour chacune de ces images, des contours ont été extraits par un être humain.

Tout d'abord, pour chaque image, les objets qui seront séparés dans la segmentation de référence sont identifiés. Ces derniers sont choisis de manière à correspondre à des objets cohérents. Les objets sélectionnés correspondent aux objets principaux de la photographie ainsi qu'à des objets au second plan, mais présentant des difficultés intéressantes.

Ensuite, une segmentation de référence est réalisée, sous forme d'aplats de couleurs, de manière à ce que deux objets voisins aient des couleurs différentes (figure 4.3b). Nous avons utilisé une tablette Intuos Pen ainsi que le logiciel de manipulation d'image Gimp. Enfin, les contours des segmentations sont extraits, un contour correspondant à un pixel ayant au moins un voisin (au sens du 8-voisinage) de couleur différente (figure 4.3c). Cette dernière étape permet une vérification visuelle de la segmentation de référence créée. Si elle met en évidence des erreurs commises, celles-ci sont corrigées et une nouvelle vérification est effectuée.

4. https://commons.wikimedia.org/wiki/Accueil

(a) Image pouvant être considérée comme simple : un seul objet est visible. Ce dernier contient peu de détails. Sa couleur comme celle du fond sont relativement uniforme.

(b) Image pouvant être considérée comme complexe : de nombreux objets sont visibles (les différents bâtiments, les arbres, etc.). En outre, les bâtiments comprennent de nombreux détails. Même pour un être humain, la tâche consistant à tracer le contour de chaque objet visible se révèle fastidieuse.

(c) Image contenant des objets de complexité variable : tandis que le ciel correspond à une zone uniforme, les bâtiments contiennent de nombreux détails.

FIGURE 4.2 – Illustration de la notion de complexité à partir de trois photographies issues des données de référence de Berkeley [30].

4.4.2 Mesure de l'adhérence aux contours

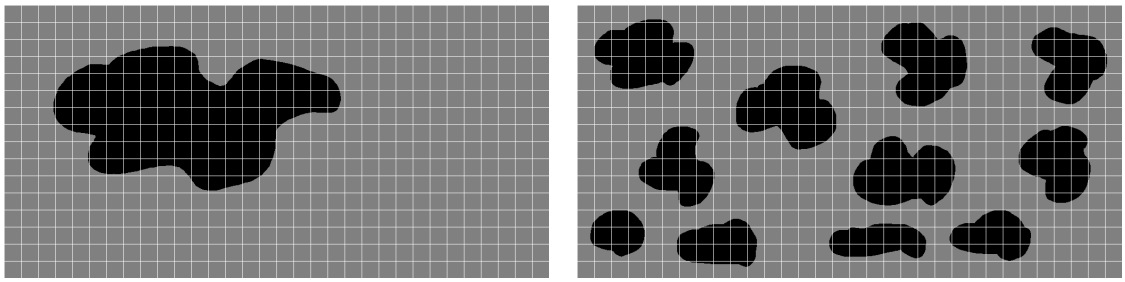
Les mesures d'adhérence aux contours tentent d'évaluer la capacité d'un algorithme de sur-segmentation à produire des superpixels ne chevauchant pas deux objets distincts. Traditionnellement, le taux d'erreur de sous-segmentation $\mathcal{F}_{ES}$ (équation 4.1) et la mesure de rappel sur les contours $\mathcal{F}_{RC}$ (équation 4.2) sont utilisés. Les sections suivantes montrent cependant que HSID soulève des difficultés spécifiques, demandant de repenser ces deux mesures.

(a) Image d'origine (b) Segmentation (c) Extraction des contours

FIGURE 4.3 – Procédure suivie pour la création de HSID.

HSID : un cas limite pour l'utilisation du taux d'erreur de sous-segmentation

La figure 4.4 montre deux images de synthèse (512 × 1024 pixels), correspondant à deux types d'images opposés : la première (figure 4.4a) contient un seul objet, de grande taille ; la seconde (figure 4.4b) contient de multiples petits objets. Le premier cas correspond à ce que nous obtenons avec une photographie de type portrait, se focalisant sur un seul objet d'intérêt. La seconde situation peut être assimilée à ce que nous obtenons avec une vue panoramique.



(a) Image de type portrait et sur-segmentation sous la forme d'une grille régulière. $\mathcal{F}_{ES}(\mathbb{S}, R) = 0{,}05$.

(b) Image de type panoramique et sur-segmentation sous la forme d'une grille régulière. $\mathcal{F}_{ES}(\mathbb{S}, R) = 0{,}1$.

FIGURE 4.4 – Images de synthèse mettant en évidence les limites du taux d'erreur de sous-segmentation. Les zones grises correspondent au fond, les zones noires aux objets. Les contours des superpixels sont tracés en blanc.

Pour ces deux images, nous avons produit une sur-segmentation en grille régulière, avec des cases carrées de 32×32 pixels. Comme le montre l'affichage des contours de cette sur-segmentation (figure 4.4), dans les deux cas, la sur-segmentation obtenue comporte de nombreuses erreurs, avec des superpixels dont les contours ne coïncident pas avec ceux des objets visibles. Nous sommes capables de ce diagnostic, car nous focalisons notre attention sur les superpixels au niveau des contours des objets. En effet, les superpixels à l'intérieur des différents objets ne contiennent pas d'erreurs et il en serait de même, quelle que soit leur forme. Or, ces derniers sont malgré tout pris en compte lors du calcul du taux d'erreur de sous-segmentation $\mathcal{F}_{ES}$, qu'ils contribuent à diminuer. Ainsi dans le cas de l'image de la figure 4.4a où ces superpixels sont plus nombreux, le score $\mathcal{F}_{ES}$ bien meilleur : 0,05 contre 0,1 pour l'image de la figure 4.4b.

Les images des données de référence BSD et NYU correspondant essentiellement à des prises de vue panoramiques, le taux moyen d'erreur de sous-segmentation est un indicateur acceptable. Il n'en va pas de même pour HSID, qui comprend les deux situations extrêmes décrites ci-dessus, avec des segmentations de référence allant de deux à une dizaine d'objets. Dans ces conditions, le score moyen, médian, maximum ou minimum obtenu pour la mesure $\mathcal{F}_{ES}$ ne permet de quantifier correctement l'adhérence aux contours. Nous avons donc choisi de ne pas utiliser la mesure $\mathcal{F}_{ES}$.

Amélioration de la mesure d'adhérence aux contours

Lors de la réalisation des segmentations de référence, nous avons constaté que, pour un certain nombre des images de HSID, il n'était pas possible de déterminer au pixel près où se situent les contours. C'est par exemple le cas lorsqu'un objet est flou ou en présence de fourrure ou de cheveux qui comprennent une part de transparence. La grande variété des tailles des images de HSID rend inutilisable la mesure d'adhérence proposée par Achanta *et al.* [1], $\mathcal{F}_{RC}$ (équation 4.2), qui utilise un seuil de tolérance fixe, une erreur de deux pixels n'ayant pas du tout la même importance pour une image de 300×400 pixels que pour une image de 3000×4000 pixels. Quant à celle proposée par Stutz *et al.* [49], elle s'avère inadaptée pour des images de grande taille, le seuil de tolérance s'élevant alors à une trentaine de pixels, ce qui dépasse largement l'erreur acceptable, y compris dans les pires cas. Ces raisons nous ont conduit à concevoir une nouvelle mesure d'adhérence aux contours, dont la définition s'appuie sur la théorie des sous-ensembles flous.

Soit R la segmentation de référence pour une image donnée et $\mathbb{S}$ la sur-segmentation obtenue pour cette image. L'ensemble $R = \{r_1, \cdots, r_{N_R}\}$ est une partition en N_R composantes connexes et l'ensemble $\mathbb{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_{N_\mathbb{S}}\}$ est une partition en $N_\mathbb{S}$ composantes connexes. Nous supposons que $N_\mathbb{S} > N_R$.

Un pixel p_i est un point de contour dans $\mathbb{S}$, si

$$\exists p_j \in \mathcal{F}_{\text{voi}}(p_i),\ (p_i \in \mathbf{s}_n \wedge p_j \notin \mathbf{s}_n).$$

De même, un pixel p_i est un point de contour dans R, si

$$\exists p_j \in \mathcal{F}_{\text{voi}}(p_i),\ (p_i \in r_n \wedge p_j \notin r_n).$$

Soit C_R l'ensemble des points de contour de R et $C_\mathbb{S}$ l'ensemble des points de contour de $\mathbb{S}$. La proportion de points de contour de R qui correspondent à des points de contour de $\mathbb{S}$ est donnée par le taux de rappel des contours $\mathcal{F}_{RC}$ (équation 4.2).

Comme les points de contour des superpixels qui se situent à l'intérieur d'un objet ne sont pas pris en compte, $\mathcal{F}_{RC}$ est similaire à une mesure d'omission qui serait calculée dans le cadre d'une classification des éléments de C_R en deux classes :

- *l'élément appartient au contour d'un superpixel* ;

- *l'élément n'appartient pas au contour d'un des superpixels.*

Soit le sous-ensemble flou $C^*_{R\cap\mathbb{S}}$ défini à partir de C_R, avec la fonction d'appartenance $f_{R\cap\mathbb{S}}$ qui donne pour un élément de C_R son degré d'appartenance à un point de contour dans $C_\mathbb{S}$. La fonction $f_{R\cap\mathbb{S}}$ est définie de la manière suivante :

$$f_{R\cap\mathbb{S}}(p_i) = \exp\left(-\frac{d(p_i - p'_i)^2}{2\sigma^2}\right) \tag{4.3}$$

avec :

- $d(p_i - p_j)$ la distance entre p_i et p_j (par exemple la distance euclidienne) ;
- σ un paramètre pondérant l'influence de la distance entre p_i et p_j, que nous avons fixé à 4 après quelques tests empiriques ;
- $p'_i = \underset{p_j\in C_\mathbb{S}}{\arg\min}(d(p_i - p_j))$.

La fonction $f_{R\cap S}$ donne une valeur dans l'intervalle $[0, 1]$, la valeur 1 correspondant au cas où un pixel dans C_R coïncide parfaitement avec un pixel dans $C_\mathbb{S}$. Il est ainsi possible de définir une mesure floue d'adhérence au contour :

$$\mathcal{F}_{FRC}(\mathbb{S}, R) = \frac{1}{|C_R|}\sum_{p\in C_R} f_{R\cap\mathbb{S}}(p) \tag{4.4}$$

C'est cette mesure que nous proposons d'utiliser pour évaluer l'adhérence aux contours d'une sur-segmentation.

4.5 Algorithmes évalués

À ce jour, il existe une vingtaine de méthodes de sur-segmentation. Un quart d'entre elles sont initialement des algorithmes de segmentation [7, 13, 42, 52, 54] reconvertis grâce à une modification de leurs paramètres. Parmi les méthodes conçues uniquement pour réaliser une tâche de sur-segmentation, une dizaine [1, 8, 23, 26, 29, 38, 50, 51, 53, 55, 62] bénéficient d'une bonne notoriété, notamment parce qu'une implémentation ou un exécutable mis à disposition par leurs auteurs facilite leur réutilisation au sein d'autres travaux.

Nous nous sommes concentrés sur les cinq méthodes ayant obtenues les meilleurs scores lors de l'évaluation la plus récente, celle de Stutz *et al.* [49] :

- la méthode Quick Shift [52] (QS), qui consiste en une démarche similaire à celle de l'algorithme mean shift de Comaniciu *et al.* [7] ;
- la méthode de Felzenszwalb *et al.* [13] (FZ), qui repose sur un algorithme de coupure de graphe ;
- la méthode de Liu *et al.* [26] (ERS, de l'anglais : « *Entropy Rate Superpixels* »), qui groupe les pixels en ensembles homogènes et de même taille, par maximisation d'une fonction objectif ;

- la méthode d'Achanta *et al.* [1] (SLIC), dont le traitement principal correspond à l'algorithme des k-moyennes et qui est sans doute l'algorithme de sur-segmentation le plus prisé ;
- la méthode de Conrad *et al.* [8] (CRS, de l'anglais : « *Contour Relaxed Superpixels* »), dont l'originalité consiste à rechercher une sur-segmentation optimale à partir d'une sur-segmentation initiale, en ne modifiant que les pixels en bordure des superpixels.

À ces cinq algorithmes, nous avons ajouté la méthode de Machairas *et al.* [29] (WP, de l'anglais : « *WaterPixels.* ») et celle de Rubio *et al.* [44] (BASS, de l'anglais : « *Boundary-Aware Superpixel Segmentation* »), dont les publications sont concomitantes de celle de l'évaluation de Stutz *et al.* [49] et pour lesquelles aucune comparaison complète avec l'état de l'art n'existe.

4.5.1 Méthode de Vedaldi *et al.* (QS)

La méthode QS, proposé par Vedaldi *et al.* [52] appartient à la catégorie des algorithmes de segmentation par recherche des modes. Soit I une image. Vedaldi *et al.* [52] décrivent les N_I pixels qui la composent par leurs localisations et leurs couleurs. Ils obtiennent alors $X = \{x_1, \cdots, x_{N_I}\}$, un ensemble de N_I vecteurs aléatoires à 5 dimensions. Les modes de la fonction de densité de probabilité associés à ces vecteurs aléatoires correspondent aux instances les plus fréquentes de ces vecteurs (donc aux maxima locaux de cette fonction).

La méthode QS permet de calculer efficacement ces modes et d'associer chaque pixel au mode dont les caractéristiques visuelles sont les plus proches des siennes. Les groupes de pixels ainsi obtenus forment une sur-segmentation de l'image.

Soit ϕ, une fonction fenêtre d'observation gaussienne, dont les pondérations sont données par :

$$w_\phi(n) = \exp\left(\frac{-n^2}{2\sigma^2}\right) \tag{4.5}$$

avec :

- $\frac{-(N_\phi - 1)}{2} < n < \frac{(N_\phi - 1)}{2}$, l'indice de la pondération ;
- N_ϕ la taille de la fenêtre ;
- σ l'écart type de la variable aléatoire considérée.

La méthode QS [52] repose sur une estimation de la densité de Parzen :

$$\mathcal{F}_P(x) = \frac{1}{N_I} \sum_{i=1}^{N_I} \phi(d(x - x_i)),\ x_i \in X. \tag{4.6}$$

À chaque itération de la méthode, chaque point correspondant à un vecteur x_{i1} est déplacé vers x_{i2}, son plus proche voisin correspondant à une valeur plus élevée de la fonction de densité de Parzen :

$$i_2 = \underset{i_3 \in [1, N_I]}{\arg\min}\, d(x_{i1}, x_{i3}),\ \mathcal{F}_P(x_{i1}) < \mathcal{F}_P(x_{i3}). \tag{4.7}$$

4.5.2 Méthode de Felzenszwalb *et al.* (FZ)

L'algorithme de segmentation proposé par Felzenszwalb *et al.* [13] groupe les pixels en régions, de manière à ce que les différences entres les pixels voisins et appartenant à des régions distinctes soient supérieures aux différences entre les pixels appartenant à une même région.

L'image I est modélisée sous la forme d'un graphe $\mathcal{G} = < V, E >$ où $V = \{v_1, \cdots v_{N_I}\}$ est un ensemble de N_I sommets correspondant aux pixels de I et E un ensemble d'arêtes pondérées reliant les sommets v_i et v_j qui correspondent à des pixels voisins. Les régions sont formées en retirant un certain nombre d'arêtes satisfaisant un prédicat, de manière à obtenir une partition de $\mathcal{G}$ en composantes connexes correspondant à des ensembles de pixels homogènes.

Soit $e_{i,j}$, l'arête reliant les sommets v_i et v_j. La pondération $w_{i,j}$ associée à $e_{i,j}$ est une mesure de la dissemblance entre les pixels i et j. Felzenzswalb *et al.* proposent d'utiliser la valeur absolue de la différence entre les niveaux de gris des deux pixels.

Soit $\mathbf{s}_i$ une composante connexe du graphe $\mathcal{G}$ et $E_i \subset E$ l'ensemble des arêtes reliant les sommets de $\mathbf{s}_i$. La fonction

$$\mathcal{F}_{Int}(\mathbf{s}_i) = \max_{e_{j,k} \in E_i} (e_{j,k}) \tag{4.8}$$

correspond à une mesure du degré de dissemblance entre les pixels composant $\mathbf{s}_i$.

Soit $E_{i,j}$ l'ensemble des arêtes reliant un sommet de la composante connexe $\mathbf{s}_i$ à un sommet de la composante connexe $\mathbf{s}_j$. La fonction

$$\mathcal{F}_{Dif}(\mathbf{s}_i, \mathbf{s}_j) = \min_{e_{k,l} \in E_{i,j}} (e_{k,l}) \tag{4.9}$$

correspond à une mesure du degré de dissemblance entre les deux composantes connexes $\mathbf{s}_i$ et $\mathbf{s}_j$. Si $E_{i,j} = \varnothing$, $\mathcal{F}_{Dif}(\mathbf{s}_i, \mathbf{s}_j) = \infty$. Le prédicat $\mathcal{F}_P$ défini par Felzenszwalb *et al.* [13] est une fonction binaire, qui renvoie vrai si les deux composantes connexes doivent rester séparées, faux si elles doivent être fusionnées :

$$\mathcal{F}_P(\mathbf{s}_i, \mathbf{s}_j) = \begin{cases} \text{vrai si } \mathcal{F}_{Dif}(\mathbf{s}_i, \mathbf{s}_j) > \min\Big(\mathcal{F}_{Int}(\mathbf{s}_i) + \tau(\mathbf{s}_i), \mathcal{F}_{Int}(\mathbf{s}_j) + \tau(\mathbf{s}_j)\Big), \\ \text{faux sinon,} \end{cases} \tag{4.10}$$

avec $\tau(\mathbf{s}_i) = \dfrac{\omega_k}{|\mathbf{s}_i|}$, où ω_k est un paramètre à déterminer lors de l'utilisation de l'algorithme et $|\mathbf{s}_i|$ correspond au nombre de sommets composant $\mathbf{s}_i$.

4.5.3 Méthode de Liu *et al.* (ERS)

L'algorithme ERS [26], reprend la représentation d'une image I sous forme d'un graphe $\mathcal{G}$, semblable à celui défini pour la méthode FZ. De manière similaire, Liu *et al.* [26] cherchent à retirer des arêtes de E, afin d'obtenir un ensemble de N_S composantes connexes dont chacune correspond à un superpixel. Toutefois, contrairement aux travaux de Felzenszwalb *et al.* [13] :

- le nombre de composantes connexes est fixé par l'utilisateur et est respecté scrupuleusement par l'algorithme ;

- les arêtes $e_{i,j} \in E$ sont pondérées par une mesure de similarité au lieu d'une mesure de dissemblance ;
- pour chaque sommet v_i, est ajoutée une arête e_i, qui boucle sur ce même sommet : lorsqu'une arête $e_{i,j}$ est supprimée, sa pondération est ajoutée sur les arêtes e_i et e_j ($e_i \leftarrow e_i + e_{i,j}$ et $e_j \leftarrow e_j + e_{i,j}$;
- la partition de $\mathcal{G}$ en composantes connexes est obtenue par maximisation d'une fonction objectif $\mathcal{F}_{ERS}$: alors que l'algorithme FZ [13] repose sur une décision locale (l'arête $e_{i,j}$ doit-elle être conservée ou non ?), ERS [26] évalue l'impact global de la suppression de cette même arête.

Soit $\mathcal{G}^* = < V, E^* >$ le graphe résultat correspondant à la sur-segmentation. L'ensemble des arêtes E^* correspond au sous-ensemble de E pour lequel la fonction $\mathcal{F}_{ERS}$ atteint son maximum. Liu *et al.* [26] se donnent pour objectif d'obtenir des superpixels :

- dont les aires, en termes de nombre de pixels, sont similaires ;
- qui ont une forte homogénéité interne en termes de couleur.

La fonction de coût est :

$$\mathcal{F}_{ERS}(\mathcal{G}') = \mathcal{F}_R(\mathcal{G}') + \mathcal{F}_E(\mathcal{G}') \tag{4.11}$$

où les deux termes $\mathcal{F}_R$ et $\mathcal{F}_E$ s'assurent du respect des deux propriétés citées précédemment et $\mathcal{G}' = < V, E' >$ est un graphe intermédiaire, avec $E' \in E$.

Soit $\mathbb{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_{N_\mathbb{S}}\}$ une partition de $\mathcal{G}$ en composantes connexes, après avoir supprimé une partie des arêtes de E. Nous notons $|\mathbf{s}_i|$ la cardinalité d'un ensemble $\mathbf{s}_i$. Liu *et al.* [26] définissent

$$\mathcal{F}_R(\mathcal{G}') = -N_\mathbb{S} - \sum_{i=1}^{N_\mathbb{S}} \frac{|\mathbf{s}_i|}{|V|} \log\Big(\frac{|\mathbf{s}_i|}{|V|}\Big). \tag{4.12}$$

Cette fonction favorise non seulement des superpixels ayant la même aire, mais également les partitions des sommets de V en exactement $N_\mathbb{S}$ composantes connexes.

Afin de mesurer l'homogénéité interne de chaque superpixel, Liu *et al.* [26] utilisent le taux d'entropie, une mesure qui permet de quantifier le taux d'incertitude d'un processus aléatoire. Soit $\overline{E'}$ l'ensemble des arêtes supprimées. Les ensembles E' et $\overline{E'}$ forment une partition de E.

Soient v_i et v_j deux sommets. Liu *et al.* [26] s'intéressent à la probabilité $P_{i,j}(\mathcal{G}', E')$ qu'un marcheur partant de v_i arrive au sommet v_j. Cette probabilité est obtenue à partir de la mesure de similarité entre v_i et v_j :

$$P_{i,j}(\mathcal{G}', E') = \begin{cases} \dfrac{w_{i,j}}{w_i} & \text{si } i \neq j \text{ et } e_{i,j} \in E', \\ 0 & \text{si } i \neq j \text{ et } e_{i,j} \in \overline{E'}, \\ 1 - \dfrac{1}{w_i} \displaystyle\sum_{e_{i,k} \in E'} w_{i,k} & \text{si } i = j. \end{cases} \tag{4.13}$$

Le graphe $\mathcal{G}' = < V, E' >$ peut être modélisé comme une marche aléatoire, sur laquelle il

devient possible de calculer le taux d'entropie, mesurant l'homogénéité interne des superpixels :

$$\mathcal{F}_E(\mathcal{G}', E') = -\sum_{v_i \in V} \left(\frac{w_i}{w_V} \sum_{v_j \in V} P_{i,j}(\mathcal{G}', E') \log(P_{i,j}(\mathcal{G}', E')) \right) \tag{4.14}$$

avec $w_V = \sum_{v_i \in V} w_i$.

La recherche du graphe $\mathcal{G}^*$ permettant de maximiser $\mathcal{F}_{ERS}$ est réalisée par un algorithme glouton. En partant d'un ensemble $E' = \varnothing$, les arêtes de l'ensemble E sont progressivement ajoutées à E', en ajoutant à chaque itération l'arête permettant d'aboutir à l'augmentation la plus importante de $\mathcal{F}_{ERS}$. L'algorithme s'arrête lorsqu'une partition de V en exactement $N_{\mathbb{S}}$ composantes connexes est obtenue. Ces composantes connexes correspondent alors aux superpixels.

4.5.4 Méthode d'Achanta *et al.* (SLIC)

La méthode SLIC proposée par Achanta *et al.* [1], correspond à une version modifiée de l'algorithme des k-moyennes. Elle est constituée de trois étapes :

- l'initialisation, où une première sur-segmentation est donnée ;
- le regroupement de pixels en ensembles, de manière à ce que chaque pixel soit rattaché à l'ensemble dont les caractéristiques visuelles sont les plus proches des siennes ;
- le post-traitement s'assurant que les ensembles obtenus à l'étape précédente forment une partition en composantes connexes.

Lors de l'étape d'initialisation, les pixels sont regroupés en superpixels correspondant à un découpage régulier de l'image sous la forme d'une grille. D'autres configurations peuvent également être envisagées, telles que des superpixels hexagonaux.

La deuxième étape repose sur un processus itératif, répétant une dizaine de fois les actions suivantes :

1. la couleur et la position moyennes de ces $N_{\mathbb{S}}$ ensembles de pixels correspondant aux superpixels initiaux sont calculées ;
2. chaque pixel, décrit par sa couleur et sa localisation dans l'image, est assigné au superpixel dont il est le plus proche au sens d'une mesure de similarité ;
3. les caractéristiques des superpixels sont mises à jour.

L'une des clés du succès de SLIC réside dans le fait que chaque pixel est comparé uniquement aux ensembles les plus proches, permettant à la méthode de produire une sur-segmentation avec une complexité quasi linéaire vis-à-vis du nombre de pixels dans l'image.

Soit un superpixel $\mathbf{s}_i$ et p_j un pixel. Nous notons l_i, a_i et b_i la couleur moyenne exprimée dans l'espace CIELab des pixels composant $\mathbf{s}_i$, x_i et y_i les coordonnées du barycentre de $\mathbf{s}_i$, l_j, a_j et b_j la couleur de p_j, x_j et y_j ses coordonnées. La similarité entre $\mathbf{s}_i$ et p_j est évaluée par la

fonction suivante :

$$\mathcal{F}_{SLIC}(\mathbf{s}_i, p_j) = \sqrt{\mathcal{F}_c(\mathbf{s}_i, p_j)^2 + \mathcal{F}_p(\mathbf{s}_i, p_j)^2 \left(\frac{\omega_C N_I}{\sqrt{\omega_{\mathbb{S}}}}\right)^2} \tag{4.15}$$

avec :

- $\mathcal{F}_c(\mathbf{s}_i, p_j) = \sqrt{(l_i - l_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2}$, la distance euclidienne entre la couleur moyenne du superpixel et celle du pixel ;
- $\mathcal{F}_p(\mathbf{s}_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, la distance euclidienne entre le barycentre du superpixel et la position du pixel ;
- N_I le nombre de pixels de l'image ;
- ω_C un paramètre de compacité, pondérant l'influence de la position du pixel par rapport à sa couleur ;
- $\omega_{N_{\mathbb{S}}}$ un paramètre indiquant le nombre de superpixels souhaités.

Soit $\mathbb{S}'$ l'ensemble des superpixels ainsi obtenus. Les pixels étant regroupés aussi en fonction de leur couleur, il n'est pas garanti que $\mathbb{S}'$ forme une partition de l'image en composantes connexes. Afin d'assurer le respect de la propriété 1 (validité), les composantes connexes de $\mathbb{S}'$ sont extraites. Celles dont le nombre de pixels est en dessous d'une taille minimale sont fusionnées avec une composante connexe voisine, donnant $\mathbb{S}$, la sur-segmentation finale.

4.5.5 Méthode de Rubio *et al.* (BASS)

L'algorithme BASS de Rubio *et al.* [44] est une version modifiée de SLIC, capable d'adapter le nombre de superpixels à la complexité locale de l'image : aux endroits contenant des objets de petite taille, de nombreux superpixels sont produits, tandis que les zones uniformes sont sur-segmentées en un nombre restreint de superpixels. Rubio *et al.* introduisent deux changements :

- le premier, au niveau de l'étape d'initialisation, permet d'adapter le nombre de superpixels de la sur-segmentation initiale ;
- le second, au moment de l'évaluation de la similarité est l'ajout d'un terme capable de prendre en compte la présence d'un contour entre un pixel p_j et le barycentre de l'ensemble $\mathbf{s}_i$.

Lors de l'étape d'initialisation, au moment de produire les superpixels initiaux suivant une grille, la méthode de Rubio *et al.* commence par appliquer sur l'image le détecteur de contour de Dollár *et al.* [11] (évoqué dans la section 2.5.5). À partir du résultat de ce détecteur, une image binaire est produite où sont conservés, comme appartenant aux contours, tous les pixels pour lesquels la valeur du détecteur de contours se situe dans les 70% des scores les plus élevés.

Soit $\mathcal{F}_C$ la fonction qui, pour un ensemble de pixels, donne la portion de pixels appartenant à un contour. En particulier, $\mathcal{F}_C(I)$ correspond à la proportion de pixels appartenant à un contour dans l'image, et $\mathcal{F}_C(\mathbf{s}_i)$ à celle des pixels appartenant à un contour au sein du superpixel $\mathbf{s}_i$. Les deux prédicats suivants permettent d'adapter le nombre de superpixels initiaux à la complexité de l'image :

- Si $\mathcal{F}_C(\mathbf{s}_i) > 3\mathcal{F}_C(I)$, alors le superpixel $\mathbf{s}_i$ appartient à une zone de l'image comprenant de nombreux détails : il est découpé en quatre nouveaux superpixels ;
- Si $\mathcal{F}_C(\mathbf{s}_i) < \mathcal{F}_C(I)$, alors le superpixel $\mathbf{s}_i$ appartient à une zone de l'image relativement homogène et il est supprimé.

Afin de terminer l'étape d'initialisation, les composantes connexes correspondant aux superpixels supprimés sont extraites : elles viennent s'ajouter à l'ensemble des superpixels initiaux, en plus des superpixels conservés et des superpixels nouvellement créés.

La seconde contribution de Rubio *et al.* consiste à proposer une nouvelle fonction permettant de calculer la similarité entre un superpixel $\mathbf{s}_i$ et un pixel p_j :

$$\mathcal{F}_{BASS}(\mathbf{s}_i, p_j) = \sqrt{\mathcal{F}_c(\mathbf{s}_i, p_j) + \omega_p \mathcal{F}_p(\mathbf{s}_i, p_j) + \omega_g \mathcal{F}_g(\mathbf{s}_i, p_j)} \tag{4.16}$$

avec :

- $\mathcal{F}_c$ et $\mathcal{F}_p$ les deux fonctions décrites lors de la présentation de l'algorithme SLIC, dans la section précédente ;
- $\mathcal{F}_g$ une fonction donnant la distance géodésique entre p_j et $\mathbf{s}_i$;
- ω_p et ω_g, deux paramètres permettant de pondérer l'influence de chacun des termes.

Soit $p_i \in \mathbf{s}_i$ et $\zeta_{i,j} = (p_i, \cdots, p_j)$ un chemin discret le reliant à p_j. Soit $\Upsilon_{i,j}$ l'ensemble des chemins reliant p_j à un pixel appartenant à $\mathbf{s}_i$. Rubio *et al.* définissent :

$$\mathcal{F}_g(\mathbf{s}_i, p_j) = \min_{\zeta_{i,j} \in \Upsilon_{i,j}} \sum_{p_k \in \zeta_{i,j}} I(p_k) \tag{4.17}$$

avec $I(p_k)$, le niveau de gris du pixel p_k.

Le déroulement de la deuxième étape de l'algorithme, ainsi que le post-traitement permettant de s'assurer que les superpixels forment une partition en composantes connexes de l'image, restent les mêmes que pour l'algorithme SLIC.

4.5.6 Méthode de Conrad *et al.* (CRS)

À l'instar de la méthode Quick Shift, l'algorithme de Conrad *et al.* [8] repose sur la représentation d'une image I par un ensemble de vecteurs $X = \{x_1, \cdots, x_{N_I}\}$, contenant les caractéristiques visuelles de chaque pixel. Soit $\mathbb{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_{N_\mathbb{S}}\}$ une partition de cette image en $N_\mathbb{S}$ superpixels. Les vecteurs qui correspondent aux pixels appartenant à un même superpixel $\mathbf{s}_i$ correspondent à un processus stochastique spécifique à $\mathbf{s}_i$. Ce processus peut être modélisé sous la forme d'une loi de probabilité de paramètres Θ_i. Conrad *et al.* [8] se donnent une loi de probabilité par composante du vecteur x_i. L'ensemble des paramètres pour la totalité des superpixels est noté Θ

L'algorithme de Conrad *et al.* [8] repose sur la recherche de la partition $\mathbb{S}^*$ et des paramètres Θ ayant la plus forte probabilité d'avoir généré X :

$$\mathbb{S}^* = \arg\max_{\mathbb{S}} P(\mathbb{S}, \Theta | X). \tag{4.18}$$

Le théorème de Bayes ainsi que le fait que $P(X)$ soit une constante, permettent à Conrad *et al.* d'obtenir :

$$P(\mathbb{S}, \Theta|X) = P(X|\mathbb{S}, \Theta)P(\Theta|\mathbb{S})P(\mathbb{S}) \tag{4.19}$$

avec :

- $P(X|\mathbb{S}, \Theta)$, la probabilité d'observer l'image I, sachant que la partition $\mathbb{S}$ de paramètres Θ est donnée ;
- $P(\Theta|\mathbb{S})$, la probabilité que les paramètres Θ soit corrects, sachant que la partition $\mathbb{S}$ est donnée ;
- $P(\mathbb{S})$, la probabilité que la partition $\mathbb{S}$ soit correcte.

Les probabilités $P(X|\mathbb{S}, \Theta)$ et $P(\Theta|\mathbb{S})$ sont obtenues en modélisant le processus stochastique à l'intérieur de chaque superpixel par une loi de probabilité telle que la loi gaussienne ou la loi laplacienne puis en déduisant les paramètres de cette loi à partir de la moyenne et de la variance sur chacune des composantes des vecteurs de l'ensemble X.

Une approximation de la probabilité $P(\mathbb{S})$ est obtenue en pénalisant les pixels adjacents n'appartenant pas au même superpixel :

$$P(\mathbb{S}) \propto w_1 \exp(-n_{4V} w_{4V} - n_D w_D) \tag{4.20}$$

avec :

- n_{4V} le nombre de paires de pixels voisins, au sens du 4-voisinages, étant attribué à des superpixels différents ;
- n_D le nombre de paires de pixels voisins diagonaux, étant attribué à des superpixels différents ;
- w_1, w_{4V}, w_D des paramètres fixés de manière empirique par Conrad *et al.*

Afin que l'utilisateur puisse choisir d'obtenir des superpixels plus ou moins réguliers, Conrad *et al.* proposent une version modifiée de la fonction à maximiser :

$$\mathcal{F}_{CRS}(\mathbb{S}, X) = P(\mathbb{S}, \Theta|X) + w_R \sum_{i=1}^{N_{\mathbb{S}}} \sum_{x_j \in \mathbf{s}_i} (x_j^p - x_i^*)^\top (x_j^p - x_i^*) \tag{4.21}$$

avec x_i^* le barycentre du superpixel $\mathbf{s}_i$, x_j^p la position du j^{e} pixel et w_R un paramètre fixé de manière empirique.

Comme la recherche de $\mathbb{S}^*$ maximisant la fonction $\mathcal{F}_{CRS}$ n'est pas possible en un temps raisonnable, l'algorithme proposé par Conrad *et al.* [8] recherche une approximation de cette solution optimale en sélectionnant un sous-ensemble des pixels sur les contours des superpixels d'une partition et en les attribuant à un superpixel voisin si cela augmente la valeur de la fonction $\mathcal{F}_{CRS}$.

Les tests menés par Conrad *et al.* [8] montrent que leur algorithme obtient de bons résultats en décrivant chaque pixel par sa couleur exprimée dans l'espace colorimétrique CIELuv.

4.5.7 Méthode de Machairas *et al.* (WP)

Les algorithmes par ligne de partage des eaux (ALPE) appartiennent à une famille de méthodes de segmentation issues de la morphologie mathématique. Ils reposent sur la modélisation d'une image en niveaux de gris comme un relief topographique puis sur la simulation de l'inondation de ce relief. Une analogie fréquemment évoquée pour expliquer le principe des ALPE est celui de la ligne continentale de partage des eaux d'Amérique du Nord, qui correspond à une succession de crêtes situées dans les chaînes côtières du Pacifique et les montagnes Rocheuses. Si nous imaginons deux gouttes d'eau tomber de part et d'autre de cette ligne, la première descendra jusqu'à atteindre l'océan Atlantique, tandis que la seconde se dirigera vers l'océan Pacifique : ainsi, cette ligne sépare deux bassins versants, chacun étant associé à un océan.

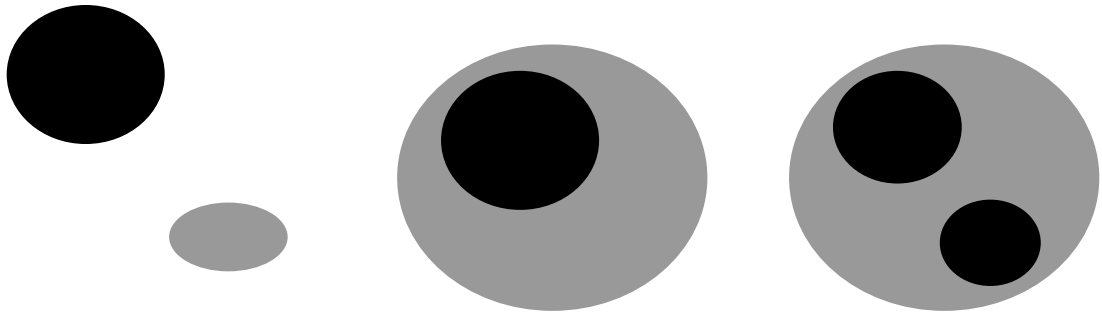
De manière similaire, nous pouvons considérer une image en niveau de gris comme un relief, les niveaux de gris les plus élevés correspondant aux altitudes les plus hautes. Dans le cas de l'algorithme WP [29], les différents bassins obtenus donnent une partition de l'image en superpixels, tandis que les lignes de partage des eaux qui les séparent forment les bordures de ces superpixels. La méthode de Machairas *et al.* [29] repose sur une modélisation de l'inondation par immersion : les niveaux de gris les plus faibles forment des sources depuis lesquelles l'eau s'écoule pour remplir petit à petit les bassins versants. Les lignes de partage des eaux correspondent aux endroits où les eaux provenant de deux bassins différents se rencontrent.

Afin de simuler ce processus d'immersion, les pixels de l'image sont triés par ordre décroissant de leurs niveaux de gris. Soit $\min(I)$, le plus petit niveau de gris de l'image I. Soit $CC^{\min(I)}$, l'ensemble des composantes connexes formées par les pixels ayant le niveau de gris $\min(I)$ et $CC^{\min(I)+1}$, l'ensemble des composantes connexes formées par les pixels ayant un niveau de gris inférieur ou égal à $\min(I)+1$. Soit $cc_i^{\min(I)+1} \in CC^{\min(I)+1}$ une composante connexe à traiter :

- si $\forall\ cc_j^{\min(I)} \in CC^{\min(I)} : cc_j^{\min(I)} \cap cc_i^{\min(I)+1} = \varnothing$, alors $cc_i^{\min(I)+1}$ correspond à un nouveau minimum de I et forme un nouveau bassin versant (un exemple de cette configuration est donné sur la figure 4.5a) ;
- si $\exists cc_i^{\min(I)}, \forall\ cc_k^{\min(I)} \in CC^{\min(I)}$ avec $k \neq i : cc_j^{\min(I)} \cap cc_i^{\min(I)+1} \neq \varnothing \ \wedge cc_j^{\min(I)} \cap cc_k^{\min(I)+1} = \varnothing$, alors $cc_j^{\min(I)}$ et $cc_i^{\min(I)+1}$ appartiennent au même bassin versant (un exemple de cette configuration est donné sur la figure 4.5b) ;
- si $\exists\ cc_j^{\min(I)} \neq cc_k^{\min(I)}$, $cc_j^{\min(I)} \cap cc_i^{\min(I)+1} \neq \varnothing \wedge cc_k^{\min(I)} \cap cc_i^{\min(I)+1} \neq \varnothing$, alors $CC^{\min(I)+1}$ contient plusieurs bassins versants, un par composante connexe de $CC^{\min(I)}$ incluse dans $CC^{min(I)+1}$ (un exemple de cette configuration est donné sur la figure 4.5c).

Lorsque le troisième cas de figure se produit, les éléments de $CC^{\min(I)+1}$ doivent être répartis entre les différents bassins versants. Machairas *et al.* utilisent la notion de zone d'influence géodésique pour réaliser cette répartition.

Soit A un ensemble et x et y deux éléments de A. La distance géodésique entre x et y correspond à la longueur du plus petit chemin reliant x à y et n'étant constitué que d'éléments inclus dans A. Soit B et C, deux composantes connexes dont les tous les éléments sont inclus dans A (c'est-à-dire que $B \subset A$ et $C \subset A$). Supposons que les éléments de B et de C sont parfaitement distincts. Soit D le complémentaire de $B \cup C$, contenant tous les éléments de A



(a) Découverte d'un nouveau bassin versant.

(b) Agrandissement d'un bassin versant existant.

(c) Deux bassins versants au sein d'une même composante connexe.

FIGURE 4.5 – Les trois cas de figure possibles lors de la recherche de bassins versants par simulation d'inondation.

n'appartenant ni à B ni à C. Soit $d_i \in D$. Soit b_i (respectivement c_i), l'élément de B le plus proche de d_i au sens de la distance géodésique (respectivement l'élément de C le plus proche de d_i au sens de la distance géodésique). Si b_i est plus proche de d_i que c_i, alors d_i appartient à la zone d'influence de B. Si b_i et c_i sont à égale distance de d_i alors cet élément appartient au squelette des zones d'influences de B et de C.

Ce procédé d'immersion est répété du niveau de gris le plus faible jusqu'au niveau de gris le plus élevé, afin d'obtenir une partition de l'image en superpixels.

4.6 Résultats de l'évaluation

Les sections suivantes présentent les résultats obtenus par chacun des algorithmes évalués. Pour chaque méthode, nous avons réalisé 8 tests, en faisant varier les paramètres de manière à ce que le nombre moyen de superpixels de l'image soit respectivement proche de 500 (test 1), 700 (test 2), 900 (test 3), 1100 (test 4), 1300 (test 5), 1500 (test 6), 1700 (test 7) et 1900 (test 8) superpixels. Chaque test est effectué avec l'ensemble des images de HSID.

Dans les paragraphes suivants nous noterons :

- $N_{\mathbb{S}}$ le nombre de superpixels ;
- N_V : le nombre de voisins d'un superpixel ;
- T : le temps d'exécution de la méthode ;
- $\mathcal{F}_{FRC}$: le score obtenu avec la mesure floue d'adhérence aux contours (équation 4.4).

L'ensemble des algorithmes évalués respectant par construction la propriété 1 (validité), nous ne nous attarderons pas sur cette dernière. La mesure $\mathcal{F}_{FRC}$ permet d'apprécier la capacité d'un algorithme à satisfaire la propriété 2 (adhérence aux contours). Un score moyen proche de 1 et un écart type faible, indiquent que la méthode évaluée sur-segmente correctement l'ensemble des images de HSID.

La mise en relation de cette mesure avec le nombre de superpixels $N_{\mathbb{S}}$ conduit à une analyse du respect de la propriété 3 (concision), en identifiant les méthodes capables d'obtenir une bonne adhérence aux contours tout en conservant un nombre peu élevé de superpixels.

La mesure N_V permet d'évaluer le respect de la propriété 4 (simplicité) : il est souhaitable que le score moyen et l'écart type obtenus pour cette mesure soient aussi bas que possible.

La propriété 5 (rapidité) est, quant à elle, évaluée à l'aide de la mesure T du temps d'exécution.

Tous les algorithmes évalués ont obtenu d'excellents scores avec les mesures et les données de référence des deux évaluations précédentes [1, 49]. Le constat d'une performance moindre sur HSID, avec les mesures précédemment présentées, peut donc venir :

- d'une difficulté à traiter des images de grande taille, avec notamment un écart type important pour les mesures T et $\mathcal{F}_{FRC}$;
- d'une difficulté à traiter avec les mêmes paramètres des images de tailles différentes, ce qu'indiqueraient des écarts types importants pour les mesures N_V, T et $\mathcal{F}_{FRC}$.

Dans les deux cas, nous considérons que la méthode ne passe pas à l'échelle, puisque l'introduction d'images de taille importante se révèle problématique.

Reste la question de l'adaptabilité d'une méthode, c'est-à-dire de sa capacité à sur-segmenter correctement des images de différentes complexités ou des images dont certaines zones sont plus complexes que d'autres. Une manière d'identifier les algorithmes possédant cette propriété consiste à s'intéresser à l'écart type de la mesure $N_{\mathbb{S}}$. Nous nous attendons à ce que ce dernier soit élevé, un faible écart type indiquant que la méthode est restée très proche du nombre moyen de superpixels recherché, sans s'adapter aux données.

L'ensemble des sur-segmentations produites par chacun des algorithmes peut être consulté en ligne [5].

4.6.1 Quelques résultats généraux

Sur l'ensemble des tests, tous les algorithmes produisent des sur-segmentations avec en moyenne 6 voisins par superpixel, avec un écart type inférieur à 0,2. En ce qui concerne la propriété de simplicité, les méthodes évaluées réalisent donc une performance équivalente.

Les figures 4.6 et 4.7 montrent, pour chaque algorithme, l'évolution de l'adhérence aux contours en fonction du nombre de superpixels. Le score $\mathcal{F}_{FRC}$ donné correspond à la moyenne obtenue pour l'ensemble de HSID. Les méthodes ERS, FZ et SLIC réalisent une performance significativement meilleure que celles des algorithmes QS, CRS, BASS et WP. Dans les sections suivantes, nous aurons l'occasion d'analyser plus en détail ce que ces résultats impliquent.

La figure 4.7 donne les temps d'exécution moyens de chaque algorithme en fonction du nombre de superpixels. Les deux extrêmes sont, d'une part, l'algorithme QS dont le temps d'exécution moyen dépasse la minute et, d'autre part, SLIC dont la durée moyenne est de seulement 3 secondes. Par rapport aux évaluations précédemment menées [1, 49], nos tests font apparaître de manière nette les différences entre les algorithmes.

5. http://image.ensfea.fr/hsid/

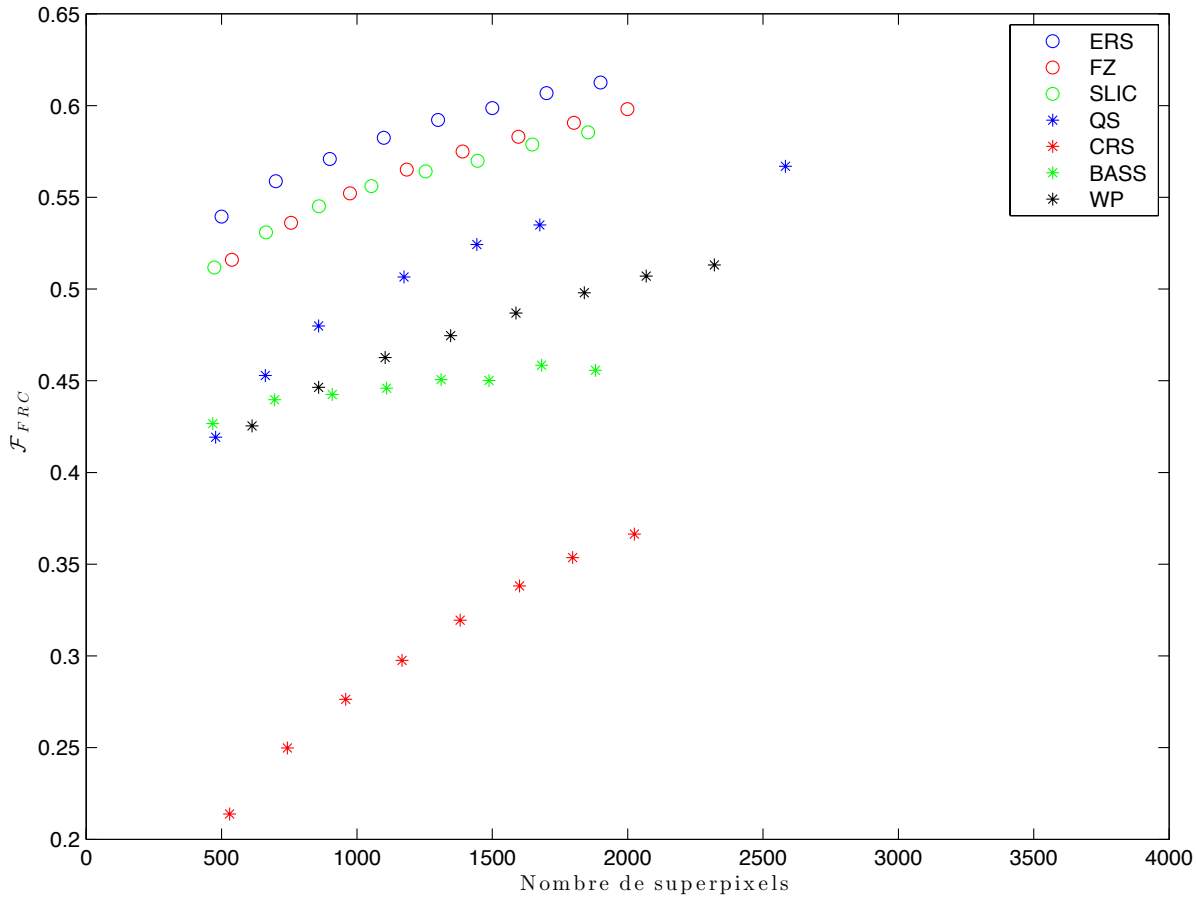


FIGURE 4.6 – Évolution de l'adhérence aux contours ($\mathcal{F}_{FRC}$) en fonction du nombre de superpixels.

À part QS pour lequel une variation importante est visible, le nombre de superpixels produits n'influe que peu le temps d'exécution moyen. En effet, la complexité algorithmique des méthodes étudiées dépend essentiellement du nombre de pixels.

4.6.2 Méthode de Vedaldi *et al.*

Pour évaluer la méthode QS [52], nous avons utilisé l'implémentation de la bibliothèque VLFeat [6]. Trois paramètres doivent être spécifiés :

- ω_c, qui permet de pondérer l'importance de la couleur du pixel par rapport à sa position dans l'image ;
- ω_ϕ, qui correspond à la taille de la fenêtre d'observation gaussienne ;
- ω_d, qui est la distance maximale entre deux pixels p_i et p_j pour que p_j soit considéré comme appartenant à l'ensemble des plus proches voisins de p_i.

6. http://www.vlfeat.org/overview/quickshift.html

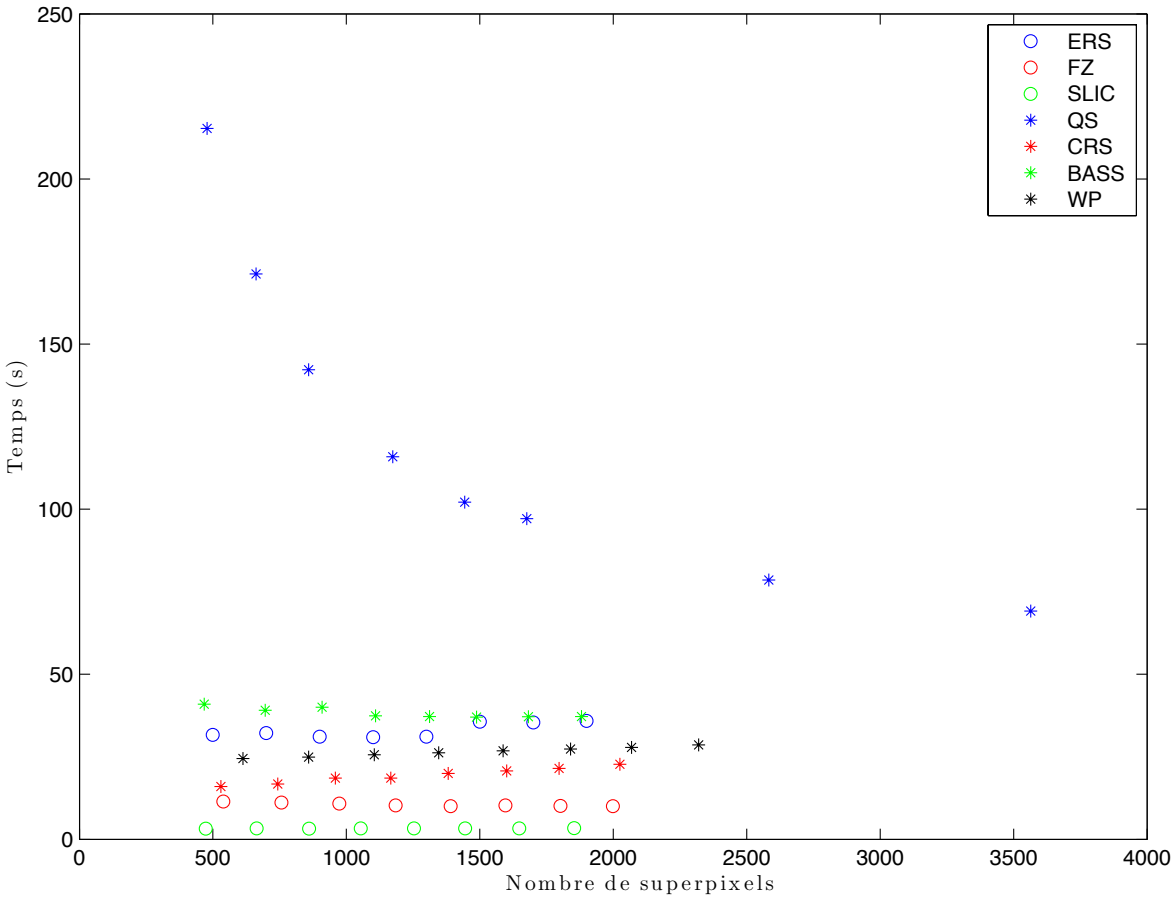


FIGURE 4.7 – Évolution du temps d'exécution en fonction du nombre de superpixels.

Durant les 8 tests, nous avons utilisé $\omega_c = 0{,}75$ et $\omega_\phi = 3$, qui correspondent aux valeurs recommandées par Stutz *et al.* [49] lors de leur évaluation. Le tableau 4.1 récapitule les valeurs utilisées pour le paramètre ω_d.

TABLEAU 4.1 – Valeurs du paramètre ω_d utilisées pour évaluer la méthode QS.

Test	1	2	3	4	5	6	7	8
ω_d	47	50	35	30	27	25	20	10

L'analyse de la figure 4.6 montre que l'algorithme QS obtient la quatrième position en termes de respect de l'adhérence aux contours par rapport au nombre de superpixels. Le score $\mathcal{F}_{FRC}$ moyen obtenu pour chacun des tests reste significativement inférieur à celui des algorithmes ERS, FZ et SLIC. Le tableau 4.2 montre que l'écart entre le nombre minimum de superpixels et le nombre maximum de superpixels pour chaque test est conséquent.

La figure 4.8 permet de voir qu'il existe une corrélation entre le nombre de pixels et le nombre de superpixels produits. Si nous étudions l'ensemble des sur-segmentations produites par Quick Shift, nous constatons que les images de grande taille produisent des sur-segmentations avec

TABLEAU 4.2 – Bornes de l'intervalle de variation du nombre de superpixels produits par l'algorithme QS.

Test	1	2	3	4	5	6	7	8
min	12	16	24	29	38	42	65	80
max	2045	2768	3596	4750	5803	6763	10222	13969

davantage de superpixels que les images de petite taille. Nous n'avons par ailleurs pas su trouver de relation simple entre ω_d et N_I, le nombre de pixels de l'image, qui permette d'obtenir un nombre de superpixels pertinent. Cette dépendance entre l'un des paramètres de Quick Shift et la taille de la photographie traitée se révèle problématique lorsque les tailles des images à sur-segmenter n'est pas connue à l'avance, notamment lorsque ces dernières proviennent de sources diverses. C'est en particulier le cas pour le domaine d'application qui nous intéresse : celui de la segmentation interactive.

Le faible score d'adhérence aux contours s'explique par le problème évoqué précédemment : pour les petites images, le nombre de superpixels n'est pas assez important pour permettre une sur-segmentation précise, ce qui se répercute sur la moyenne du score $\mathcal{F}_{FRC}$. Par ailleurs, les images contenant des objets dont les contours ne sont pas nets, notamment parce qu'ils représentent des animaux à fourrure ou les cheveux d'une personne, posent également de grosses difficultés à Quick Shift.

La figure 4.9 illustre ce phénomène. Elle montre les superpixels obtenus pour deux images. Dans les deux cas, le paramètre ω_d vaut 47. Pour la figure 4.9a, qui correspond à une photographie de 732×509 pixels, nous obtenons 57 superpixels. Au contraire, la photographie de la figure 4.9b, qui contient 3648×2736 pixels, est sur-segmentée en 1204 superpixels. Enfin, le temps d'exécution moyen, même s'il décroît à mesure que le nombre de superpixels augmente, se classe parmi les plus élevés, certaines images nécessitant plusieurs minutes pour être sur-segmentées.

Ces résultats nous conduisent à conclure que, malgré ses bonnes performances dans les études précédentes, la méthode Quick Shift ne passe pas à l'échelle et échoue à sur-segmenter convenablement HSID.

4.6.3 Algorithme de Felzenszwalb *et al.*

Afin d'évaluer l'algorithme de Felzenszwalb *et al.* [13], nous avons utilisé l'implémentation mise à disposition par les auteurs. Elle nécessite de fixer trois paramètres :

- ω_k qui pondère l'influence des différences internes à une composante connexe, par rapport aux différences entre deux composantes connexes ;
- ω_σ, le paramètre du filtre gaussien qui, lors d'un prétraitement, permet de réduire le bruit au sein des images ;
- ω_{min}, la taille minimale des superpixels.

Lors de l'ensemble de nos tests, nous avons utilisé $\omega_k = 24$ et $\omega_\sigma = 0{,}5$. Ces deux paramètres découlent de tests effectués sur deux autres ensembles d'images : celles mises à disposition par McGuinness *et al.* [31] et celles de Santner *et al.* [46], les paramètres trouvés par Stutz *et al.*

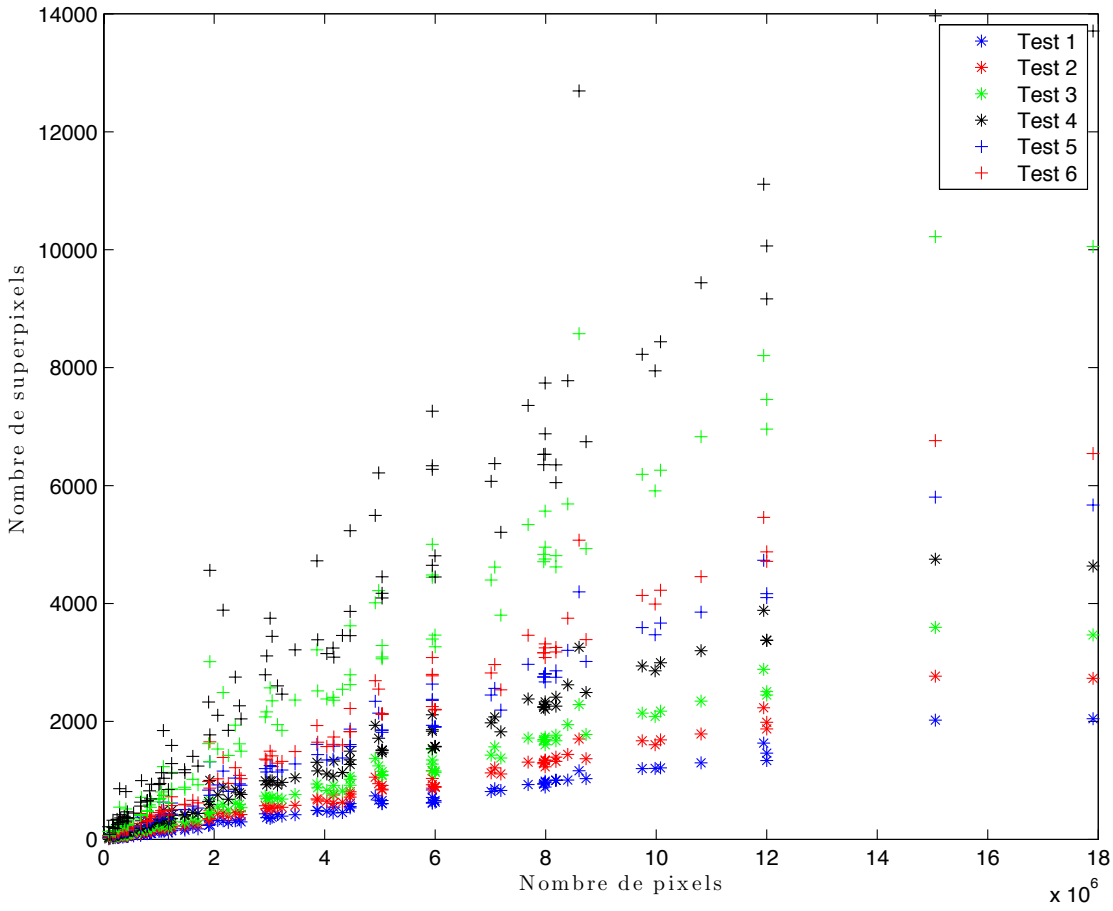


FIGURE 4.8 – Algorithme QS : évolution du nombre de superpixels par rapport au nombre total de pixels dans l'image.

ne se révélant pas pertinents pour HSID. Pour le paramètre ω_{min}, nous avons utilisé les valeurs données dans le tableau 4.3.

TABLEAU 4.3 – Valeurs du paramètre ω_{min} utilisées pour l'algorithme de Felzenszwalb *et al.* La variable N_I correspond au nombre de pixels de l'image.

Test	1	2	3	4	5	6	7	8
ω_{min}	$N_I/2500$	$N_I/3500$	$N_I/4500$	$N_I/5500$	$N_I/6500$	$N_I/7500$	$N_I/7500$	$N_I/8500$

Les résultats obtenus par l'algorithme FZ montrent qu'il permet de garantir une bonne adhérence aux contours. Pour l'ensemble des tests, nous constatons un écart type élevé pour la mesure $N_{\mathbb{S}}$, égal à environ un cinquième de la valeur de la moyenne. Mis en relation avec les bons scores d'adhérence aux contours, ce résultat révèle que l'algorithme de Felzenszwalb *et al.* bénéficie d'une bonne adaptabilité, ce que confirme l'étude des sur-segmentations produites et ce qu'illustrent les deux exemples de la figure 4.10.

L'un des reproches fréquemment adressés à la méthode de Felzenszwalb *et al.* concerne l'aspect

(a) Superpixels obtenus pour l'image img-100 de HSID. Cette image comporte 732×509 pixels.

(b) Superpixels obtenus pour l'image img-041 de HSID. Cette image comporte 3648×2736 pixels.

FIGURE 4.9 – Algorithme Quick Shift : exemples de sur-segmentations obtenues lors du test 1, pour des images de tailles différentes.

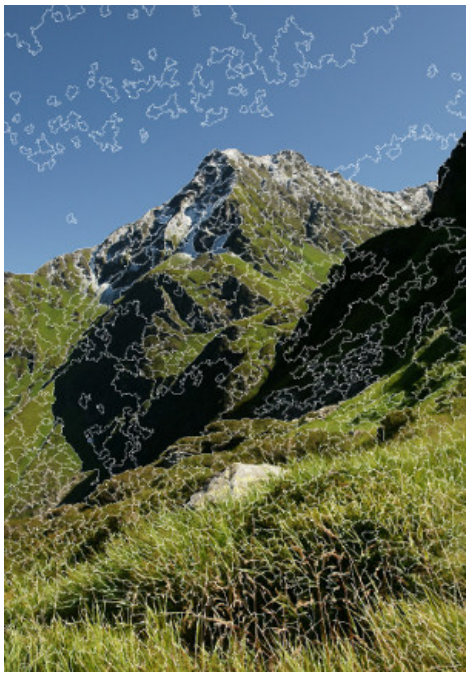
des superpixels qu'elle produit, avec des bordures souvent très sinueuses, donnant un aspect moins visuellement plaisant que des algorithmes tels que SLIC ou celui de Liu *et al.*. Les superpixels de la figure 4.10 sont effectivement assez irréguliers. De plus, la topologie des deux sur-segmentations présente des aspects surprenants tels qu'un superpixel inclus à l'intérieur d'un autre superpixel de grande taille. La moyenne et l'écart type de la mesure N_V révèlent toutefois qu'en termes de nombre de superpixels voisins, l'algorithme de Felzenszwalb produit des résultats similaires à ceux des autres algorithmes évalués, avec un score de 6, ce qui reste inférieur au nombre de

voisins d'un pixel si nous considérons le 8-voisinage.

Le temps d'exécution moyen de l'algorithme gravite autour d'une dizaine de secondes et tend à diminuer légèrement lorsque le nombre de superpixels augmente.

L'ensemble de ces résultats nous conduit à conclure que l'algorithme de Felzenszwalb *et al.* passe correctement à l'échelle, en réussissant à sur-segmenter les images de HSID de manière à conserver une bonne adhérence aux contours, sans que le nombre de superpixels n'explose.

(a) Superpixels obtenus pour l'image img-086 de HSID.



(b) Superpixels obtenus pour l'image img-028 de HSID.

FIGURE 4.10 – Algorithme de Felzenszwalb *et al.* : exemples de sur-segmentations obtenues lors du test 8.

4.6.4 Algorithme de Liu *et al.*

Pour évaluer l'algorithme de Liu *et al.* [26], nous avons utilisé l'implémentation fournie par les auteurs ainsi que les paramètres recommandés par Stutz *et al.* [49]. Les paramètres à spécifier sont les suivants :

- ω_σ, qui correspond à l'écart type d'une fonction gaussienne utilisée pour calculer la mesure de similarité entre deux pixels à partir de la valeur absolue de la différence entre les niveaux de gris des pixels ;
- ω_λ, qui permet de pondérer l'influence entre les termes $\mathcal{F}_R$ et $\mathcal{F}_E$ de la fonction de coût ;
- $\omega_{N_\mathbb{S}}$, le nombre de superpixels souhaité.

Durant tous les tests, nous avons considéré des pixels voisins au sens du 8-voisinage, $\omega_\sigma = 0{,}21$ et $\omega_\lambda = 0{,}5$. Pour le paramètre $\omega_{N_\mathbb{S}}$, nous avons utilisé les valeurs présentées dans le tableau 4.4.

TABLEAU 4.4 – Valeurs du paramètre ω_{N_S} utilisées pour évaluer l'algorithme de Liu *et al.*

Test	1	2	3	4	5	6	7	8
ω_{N_S}	500	700	900	1100	1300	1500	1700	1900

De toutes les méthodes évaluées, celle de Liu *et al.* obtient la meilleure adhérence aux contours avec un minimum de superpixels. Cette très bonne performance doit cependant être nuancée par des temps d'exécution importants (plus d'une minute pour certaines images) et l'absence complète d'adaptabilité, le nombre de superpixels étant un paramètre fixé par l'utilisateur de la méthode. Le premier de ces inconvénient limite l'utilisation de la méthode à des applications où les temps de calcul ne sont pas trop critiques. Ce n'est malheureusement pas le cas du domaine applicatif qui nous intéresse, celui de la segmentation interactive. Quant à l'absence d'adaptabilité, elle se traduit par deux désavantages : d'une part, certaines images relativement simples sont sur-segmentées en davantage de superpixels que nécessaire. D'autre part, pour les images les plus complexes, même avec 1900 superpixels, certains détails sont perdus.

(a) Superpixels obtenus pour l'image img-042 de HSID.

(b) Superpixels obtenus pour l'image img-058 de HSID.

FIGURE 4.11 – Algorithme de Liu *et al.* : exemples de sur-segmentations obtenues lors du test 6.

Les deux exemples présentés sur la figure 4.11 soulignent ce paradoxe : si les sur-segmentations produites suivent parfaitement les contours des objets, malgré le fait que les photographies présentent de nombreuses difficultés (objets voisins de couleurs similaires, bruit, ombre, etc.), elles comprennent de nombreux petits superpixels qui pourraient être fusionnés.

4.6.5 Méthode d'Achanta *et al.*

Afin d'évaluer l'algorithme SLIC [1], nous avons utilisé l'implémentation mise à disposition par Achanta *et al.* Cette dernière requiert de spécifier :

- ω_C, le paramètre de compacité pondérant l'influence de la localisation d'un pixel par rapport à sa couleur ; , le nombre approximatif de superpixels souhaité, qui permet de produire la sur-segment

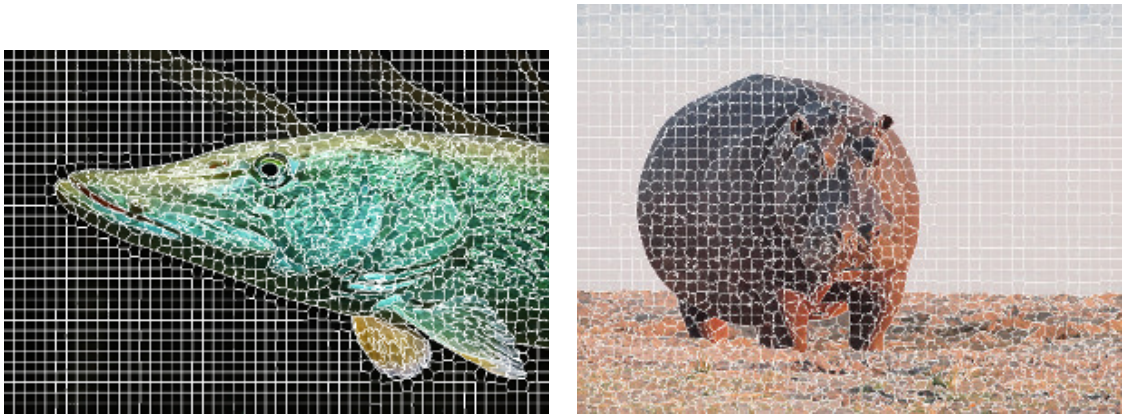
- $\omega_{N_\mathbb{S}}$, le nombre approximatif de superpixels souhaité, qui permet de produire la sur-segmentation de départ, sous forme de grille.

Suivant les recommandations d'Achanta *et al.* [1] et celles de Stutz *et al.* [49], nous avons utilisé $\omega_C = 10$ pour l'ensemble de nos tests. Les valeurs du paramètre $\omega_{N_\mathbb{S}}$ sont données dans le tableau 4.5.

TABLEAU 4.5 – Valeurs du paramètre $\omega_{N_\mathbb{S}}$ utilisées pour évaluer l'algorithme SLIC.

Test	1	2	3	4	5	6	7	8
$\omega_{N_\mathbb{S}}$	500	700	900	1100	1300	1500	1700	1900

SLIC nécessite davantage de superpixels que les algorithmes de Felzenszwalb *et al.* [13] et de Liu *et al.* [26] pour obtenir une adhérence aux contours semblable, avec un score $\mathcal{F}_{FRC}$ égal à 0,59 atteint avec quasiment 1900 superpixels, tandis que les deux méthodes précédemment citées réalisent la même performance avec respectivement 1600 et 1300 superpixels. En termes d'adaptabilité, l'algorithme de Felzenszwalb se révèle également légèrement meilleur que SLIC : adapter le nombre de superpixels au niveau de détail de l'image reste à la charge de l'utilisateur. Le principal atout de SLIC reste cependant ses temps d'exécution, largement inférieurs à ceux des autres méthodes, avec un temps moyen de 3 secondes.



(a) Superpixels obtenus pour l'image img-046 de HSID.

(b) Superpixels obtenus pour l'image img-052 de HSID.

FIGURE 4.12 – Algorithme SLIC : exemples de sur-segmentations obtenues lors du test 8.

4.6.6 Algorithme de Rubio *et al.*

Afin de tester l'algorithme de Rubio *et al.* [44], nous avons utilisé l'implémentation mise à disposition par ses auteurs. Le seul paramètre de cette méthode est $\omega_{N_\mathbb{S}}$, le nombre initial de superpixels, pour lequel nous avons utilisé les valeurs du tableau 4.6.

L'analyse des résultats de l'algorithme de Rubio *et al.*, permet de mettre en exergue l'importance de la mise à disposition de HSID. Dans l'article [44] où ils décrivent leur algorithme, Rubio *et al.* le comparent à quelques méthodes de l'état de l'art, dont SLIC, sur sept ensembles

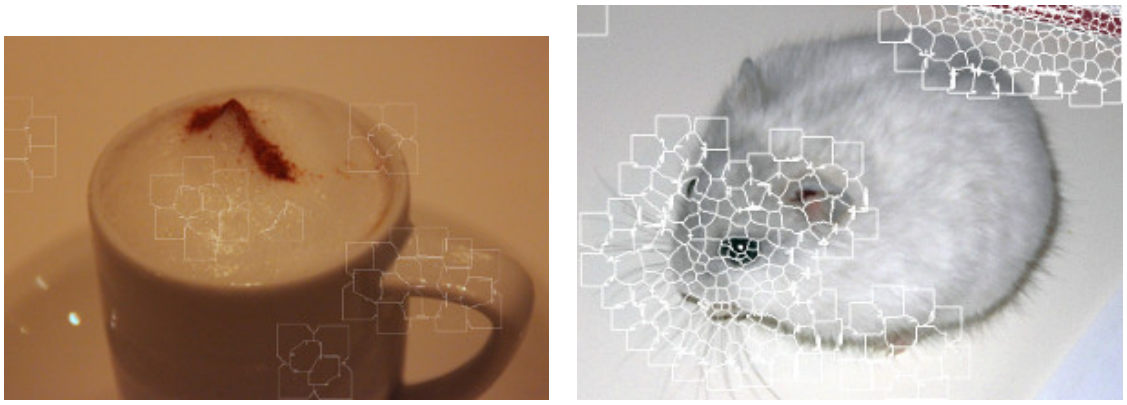
TABLEAU 4.6 – Valeurs du paramètre ω_{N_S} utilisées lors des tests effectués avec la méthode de Rubio *et al.*

Test	1	2	3	4	5	6	7	8
ω_{N_S}	700	1100	1500	1900	2300	2700	3100	3500

de données de référence différents [20, 27, 30, 58, 59], totalisant plusieurs milliers d'images comprenant des difficultés très variées mais dont les tailles sont toujours proches : quelques centaines de pixels en largeur comme en hauteur, pour un nombre total de pixels $N_I \backsimeq 15000$. Une autre propriété commune à l'ensemble des photographies est qu'elles contiennent en majorité des objets bénéficiant d'un bon contraste avec le fond. Les résultats obtenus par Rubio *et al.* montrent que leur méthode obtient de meilleurs résultats que SLIC, avec un nombre inférieur de superpixels.

Nos tests avec HSID ne permettent pas d'aboutir à la même conclusion : le score d'adhérence aux contours de la méthode de Rubio *et al.* est largement en dessous de celui de SLIC. Les deux exemples donnés sur la figure 4.13 montrent que les mécanismes qui garantissent à l'algorithme de Rubio *et al.* une bonne adaptabilité sont responsables des erreurs importantes commises, là où la présence des contours des objets est difficilement perceptible.

Par ailleurs, les modifications introduites par Rubio *et al.* afin d'obtenir une méthode avec une meilleure adaptabilité se paient par une augmentation considérable des temps d'exécution, avec une moyenne d'environ 40 secondes, soit plus de dix fois le temps d'exécution moyen de SLIC. Notons que les algorithmes de Felzenszwalb *et al.* [13] et de Liu *et al.* [26] (avec lesquels Rubio *et al.* ne comparent pas leur méthode), obtiennent eux aussi une bien meilleure adhérence aux contours, avec des temps d'exécution inférieurs.



(a) Superpixels obtenus pour l'image img-012 de HSID.

(b) Superpixels obtenus pour l'image img-040 de HSID.

FIGURE 4.13 – Algorithme de Rubio *et al.* : exemples de sur-segmentations obtenues lors du test 1.

4.6.7 Algorithme de Conrad *et al.*

Afin de tester l'algorithme de Conrad *et al.* [8], nous avons utilisé l'implémentation que les auteurs ont mise à disposition. Les paramètres de cette implémentation sont les suivants :

- ω_R, le paramètre permettant de pondérer l'influence du terme de régularisation de la fonction à maximiser ;
- ω_{4V}, le paramètre permettant de pondérer la pénalité prise en compte lorsque l'un des quatre voisins d'un pixel n'appartient pas au même superpixel ;
- ω_D, le paramètre permettant de pondérer la pénalité prise en compte lorsque l'un des voisins diagonaux d'un pixel n'appartient pas au même superpixel ;
- ω_{Iter}, le nombre d'itérations de l'algorithme ;
- $\omega_{N_\mathbb{S}}$, le nombre initial de superpixels.

Pour l'ensemble de nos tests, nous avons utilisé $\omega_R = 0{,}045$, $\omega_{4V} = 0{,}3$, $\omega_D = 0{,}21$ et $\omega_{Iter} = 3$, valeurs recommandées par Stutz *et al.* [49]. Pour le paramètre $\omega_{N_\mathbb{S}}$ nous avons utilisé les valeurs données dans le tableau 4.7.

TABLEAU 4.7 – Valeurs du paramètre $\omega_{N_\mathbb{S}}$ utilisées lors des tests effectués avec la méthode de Conrad *et al.*

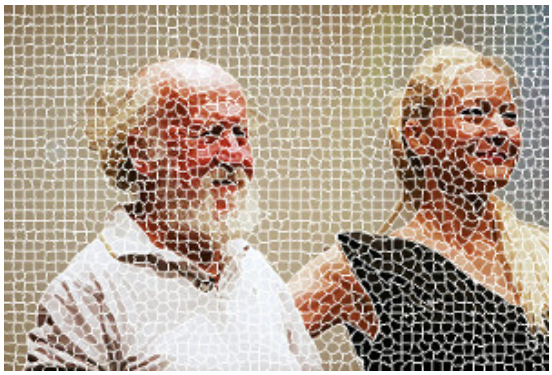
Test	1	2	3	4	5	6	7	8
$\omega_{N_\mathbb{S}}$	500	700	900	1100	1300	1500	1700	1900

L'analyse des résultats obtenus par CRS [8] amène au constat que la méthode ne parvient pas à sur-segmenter correctement HSID, avec des scores pour la mesure $\mathcal{F}_{FRC}$ faibles, y compris pour un nombre élevé de superpixels. Une étude visuelle des sur-segmentations produites montre que si les images de taille modeste (quelques milliers de pixels) sont sur-segmentées correctement, la méthode de Conrad *et al.* échoue à produire des superpixels corrects pour les images de grande taille (plusieurs millions de pixels). Une illustration est donnée sur la figure 4.14.

Le fait que l'algorithme de Conrad *et al.* ne passe pas à l'échelle s'explique par sa trop grande dépendance vis-à-vis de la sur-segmentation initiale, une grille régulière de $\omega_{N_\mathbb{S}}$ cases. Pour les images de grande taille contenues dans HSID, cette sur-segmentation initiale correspond à une erreur importante par rapport à la sur-segmentation souhaitée. En ne modifiant, à chaque itération, que les pixels au niveau des contours, la convergence vers un résultat acceptable devient très lente. Il n'est par ailleurs pas assuré qu'en augmentant le nombre d'itérations, un meilleur score moyen pour la mesure $\mathcal{F}_{FRC}$ soit atteint : les caractéristiques (moyenne et variance de la couleur des pixels) initiales calculées étant elles aussi entachées d'erreurs conséquentes. Lors de tests complémentaires, nous avons multiplié par 100 le nombre d'itérations, sans constater d'amélioration notable de l'adhérence aux contours. Les temps d'exécution, quant à eux, dépassaient les trentes minutes par image.

(a) Superpixels obtenus pour l'image img-003 de HSID, avec 2121×2816 pixels.

(b) Superpixels obtenus pour l'image img-003 de HSID (zoom).



(c) Superpixels obtenus pour l'image img-095 de HSID, avec 800×535 pixels.

(d) Superpixels obtenus pour l'image img-0095 de HSID (zoom).

FIGURE 4.14 – Algorithme de Conrad *et al.* : exemples de sur-segmentations obtenues lors du test 8.

4.6.8 Algorithme de Machairas *et al.*

Afin d'évaluer la méthode Machairas *et al.* [29], nous avons utilisé l'exécutable[7] que ses auteurs ont mis à disposition. Ce dernier nécessite deux paramètres :

- ω_R, un paramètre de régularisation, permettant d'obtenir des superpixels dont les contours

7. Nous n'avons malheureusement pas pu avoir accès au code source.

sont plus ou moins réguliers ;

- $\omega_{N_\mathbb{S}}$, le nombre approximatif de superpixels souhaité.

Durant l'ensemble de nos tests, nous avons utilisé $\omega_R = 8$, qui correspond à la valeur recommandée par Machairas *et al.* et qui donne les meilleurs résultats sur HSID. Les valeurs du paramètre $\omega_{N_\mathbb{S}}$ sont récapitulées dans le tableau 4.8.

TABLEAU 4.8 – Valeurs du paramètre $\omega_{N_\mathbb{S}}$ utilisées lors des tests effectués avec la méthode de Machairas *et al.*

Test	1	2	3	4	5	6	7	8
$\omega_{N_\mathbb{S}}$	500	700	900	1100	1300	1500	1700	1900

Contrairement aux autres algorithmes, celui de Machairas *et al.* échoue à produire un résultat pour les images *img-012*, *img-066* et *img-072* de HSID. Ne disposant pas du code source, nous n'avons pas pu déterminer la cause de ce comportement. Nos tests ont donc été réalisés sur une version tronquée de HSID, ne contenant pas ces trois images.

Les résultats obtenus par l'algorithme de Machairas *et al.* sont un nouvel exemple d'une méthode obtenant de bons résultats sur les données de référence BSD, mais échouant à sur-segmenter correctement les images de HSID. Ainsi, malgré un nombre important de superpixels (supérieur à 2000), l'adhérence aux contours reste très inférieure à celle obtenue avec des algorithmes tels que celui de Liu [26], de Felzenszwalb [13] ou la méthode SLIC [1]. Les temps d'exécution sont quant à eux assez élevés, une vingtaine de secondes en moyenne.

L'analyse visuelle des résultats obtenus montre d'importantes disparités entre les sur-segmentations des images de grande taille et celles de petite taille. Ainsi, comme l'illustre la figure 4.15, alors que les superpixels obtenus pour des images de grande taille (figures 4.15a et 4.15b) sont souvent erronés, ceux des photographies de quelques milliers de pixels (voir les figures 4.15c et 4.15d) garantissent une adhérence aux contours bien meilleure.

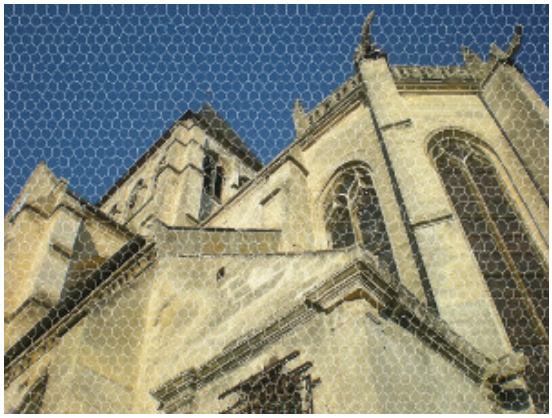
4.6.9 Choix d'une méthode de sur-segmentation

L'évaluation conduite à partir des images de HSID montre que seulement trois méthodes – SLIC [1], l'algorithme de Felzenszwalb *et al.* [13] et celui de Liu *et al.* [26] – sur-segmentent correctement un ensemble d'images de taille variable, dont certaines contiennent plusieurs millions de pixels. Dans le domaine applicatif qui nous concerne, la méthode de sur-segmentation choisie doit être intégrée au sein d'une méthode de segmentation interactive, capable de produire un résultat en quelques secondes. À ce titre, malgré son excellente adhérence aux contours, l'algorithme de Liu *et al.* [26] ne peut être utilisé en raison de ses temps d'exécution élevés.

Si l'algorithme de Felzenszwalb *et al.* [13] bénéficie d'une meilleure adaptabilité que SLIC, il reste néanmoins significativement plus lent. Par ailleurs, les superpixels produits par cet algorithme peuvent avoir une aire importante, entraînant des erreurs conséquentes lorsque la bordure entre deux objets est difficile à détecter. L'une des contraintes fortes de l'algorithme de segmentation interactive que nous proposons réside dans le fait qu'une erreur commise lors de l'étape de

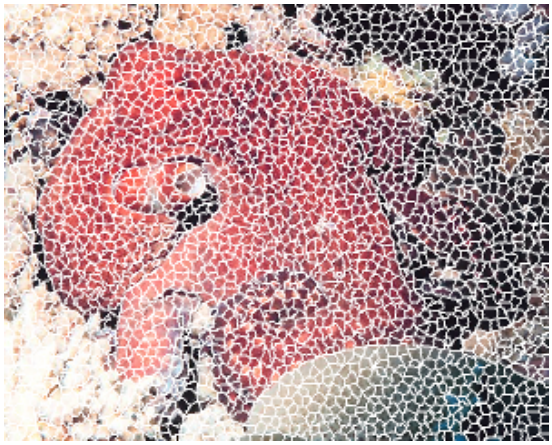
sur-segmentation ne peut être corrigée par la suite, quels que soient les germes données par l'utilisateur. Même si ce type d'erreur demeure marginal, lorsqu'il survient, il place l'utilisateur dans l'impossibilité d'aboutir à un résultat acceptable. *A contrario*, les faibles temps d'exécution de SLIC permettent de compenser le fait qu'un nombre plus important de superpixels doivent être produits pour sur-segmenter correctement l'image. Par ailleurs, ces superpixels étant de petite taille, lorsqu'une erreur est commise, son impact sur le résultat final reste faible.

Toutes ces raisons nous ont conduit à privilégier l'utilisation de SLIC au sein de la méthode de segmentation interactive que nous proposons.

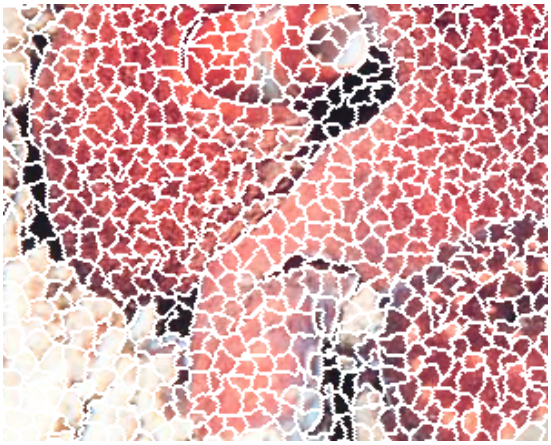


(a) Superpixels obtenus pour l'image img-020 de HSID.

(b) Superpixels obtenus pour l'image img-020 de HSID (zoom).



(c) Superpixels obtenus pour l'image img-099 de HSID.



(d) Superpixels obtenus pour l'image img-099 de HSID (zoom).

FIGURE 4.15 – Algorithme de Machairas *et al.* : exemples de sur-segmentations obtenues lors du test 8.

4.7 Conclusion

Si l'objectif du protocole d'évaluation présenté dans ce chapitre concernait le choix d'une méthode de sur-segmentation pertinente dans le cadre d'un domaine d'application particulier, celui de la segmentation interactive, l'analyse des résultats obtenus met également en évidence l'apport de HSID.

En particulier, nous notons que le fait que HSID contienne des images dont la taille va de quelques milliers à plusieurs millions de pixels permet :

- d'identifier des méthodes dont les paramètres dépendent de la taille de la photographie à sur-segmenter, à l'instar des conclusions obtenues avec Quick Shift [52] ;
- de constater que certains algorithmes, comme celui de Conrad *et al.* [8], ont été optimisés pour de petites images et n'arrivent pas à sur-segmenter correctement des images de taille bien supérieure ;
- de repérer des difficultés de passage à l'échelle, notamment vis-à-vis des temps d'exécution, par exemple avec l'algorithme de Liu *et al.* [26] dont l'écart avec des méthodes telles que SLIC [1] ou celle de Felzenszwalb *et al.* [13] devient nettement plus perceptible.

Par ailleurs, les résultats obtenus par la méthode de Rubio *et al.* [44] ou par celle de Machairas *et al.* [29] montrent, qu'au delà des aspects liés à la taille des photographies, les images de HSID présentent des difficultés plus marquées que dans d'autres ensembles de données de référence.

Chapitre 5

Évaluation de l'algorithme $S\alpha F$

5.1 Introduction

5.1.1 Comparaison avec l'état de l'art

Ce chapitre présente l'évaluation de l'algorithme $S\alpha F$. Il commence par une analyse des ses performances par rapport à celles des méthodes de l'état de l'art :

- en binarisation interactive par recherche des contours ;
- en binarisation interactive par recherche des régions ;
- en segmentation interactive multiclasse.

Se comparer à ces méthodes est une tâche complexe. Pour la majorité des algorithmes de l'état de l'art, aucune implémentation n'a été mise à disposition par les auteurs, nous contraignant à reproduire leurs conditions de test afin d'obtenir pour $S\alpha F$ des résultats qui pourront être confrontés à ceux obtenus par les autres méthodes. Afin de produire les conditions d'une comparaison équitable, nous avons été amené à répondre aux questions suivantes :

- **Quelles sont les données employées ?** Trois ensembles de données de référence – celui Rother *et al.* [43], celui de McGuinness *et al.* [31] et celui Santner *et al.* [46] – sont utilisés pour l'évaluation des méthodes de binarisation interactive. Cependant leur utilisation est loin d'être uniforme : à notre connaissance, $S\alpha F$ est le seul algorithme de segmentation interactive évalué sur l'intégralité des trois ensembles. Toutes les autres méthodes sont évaluées au mieux sur deux de ces ensembles, souvent sur un seul et parfois même uniquement sur quelques images de chacun des ensembles. Le tableau 5.1 récapitule pour chaque méthode les données utilisées.

- **Comment la qualité de la segmentation est-elle évaluée ?** Nous verrons que trois mesures sont utilisées : l'indice de Jaccard, une mesure floue d'adéquation aux contours et la mesure DICE. Là encore, selon les évaluations conduites, une ou plusieurs de ces mesures sont utilisées.

- **Comment la rapidité de la méthode est-elle évaluée ?** Le tableau 5.2 récapitule les informations dont nous disposons pour chaque méthode. La rapidité d'une méthode peut être évaluée :

  - soit en mesurant le temps nécessaire pour produire une segmentation, une fois les germes donnés – nous nommons cette durée *temps d'exécution* ;
  - soit, dans le cas d'évaluation permettant à l'utilisateur de venir modifier les germes, en mesurant le temps nécessaire pour aboutir à la segmentation recherchée. Cette durée, que nous nommons *durée de convergence*, inclut le temps requis pour donner les germes, le temps utilisé pour les modifier et les temps d'exécution de la méthode pour chaque segmentation produite, chaque fois que les germes sont mis à jour. Dans certains cas, une durée limite est imposée à l'utilisateur : au delà de ce temps imparti, la modification des germes n'est plus possible. Nous nommons cette durée *temps de convergence maximum*.

TABLEAU 5.1 – Données de référence utilisées pour l'évaluation des différents algorithmes de l'état de l'art.

Algorithme	Données de références
Milles *et al.* [32]	10 images de Rother *et al.* [43]
Mortensen *et al.* [33]	Rother *et al.* [43]
Boykov *et al.* [4]	McGuinness *et al.* [31]
Salembier *et al.* [45]	McGuinness *et al.* [31]
Friedland *et al.* [15]	McGuinness *et al.* [31]
Adams *et al.* [2]	McGuinness *et al.* [31]
Jian *et al.* [18]	McGuinness *et al.* [31] (en partie)
Santner *et al.* [46]	Santner *et al.* [46]
Müller *et al.* [35]	Santner *et al.* [46]
Changjae *et al.* [6]	Rother *et al.* [43] et 50 images de Santner *et al.* [46]

5.1.2 Analyse des propriétés de $S\alpha F$

Une fois la comparaison avec les méthodes de l'état de l'art réalisée, nous décrirons un ensemble de tests visant à mettre en exergue des propriétés spécifiques de $S\alpha F$. Tout d'abord, nous évaluerons sa capacité de passage à l'échelle. Ensuite, nous étudierons son ergonomie. Enfin, nous conclurons avec deux applications de notre algorithme.

TABLEAU 5.2 – Évaluation de la rapidité des méthodes.

Algorithme	Temps d'exécution	Modification des germes	Temps de convergence maximum
Milles *et al.* [32]	OUI	NON	–
Mortensen *et al.* [33]	OUI	OUI	2 minutes
Boykov *et al.* [4]	OUI	OUI	2 minutes
Salembier *et al.* [45]	OUI	OUI	2 minutes
Friedland *et al.* [15]	OUI	OUI	2 minutes
Adams *et al.* [2]	OUI	OUI	2 minutes
Jian *et al.* [18]	OUI	OUI	–
Santner *et al.* [46]	OUI	NON	–
Müller *et al.* [35]	OUI	NON	–
Changjae *et al.* [6]	OUI	NON	–

5.1.3 Conditions générales des tests réalisés

Les tests décrits ont été réalisés sur un ordinateur équipé d'un processeur 2,6 GHz Intel Core i7 et de 16 Go de mémoire vive. Les images sont sur-segmentées en 3000 superpixels et le paramètre de compacité de l'algorithme de sur-segmentation SLIC a été fixé à 10. Nous avons utilisé comme méthode de classification un SVM, au travers de l'implémentation LibSVM avec le noyau suivant :

$$\mathcal{F}_{RBF}(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \tag{5.1}$$

où σ vaut 4. Le paramètre pondérant l'importance des erreurs de classification par rapport à celle de la largeur de la marge de tolérance au bruit dans les données est lui aussi fixé à 4. Le choix de ces deux paramètres est détaillé dans la section 3.5.

5.2 Comparaison avec les méthodes de binarisation interactive par recherche des contours

Nous avons comparé les performances de $S\alpha F$ à celles de deux méthodes de binarisation interactive par recherche des contours :

- la méthode de Milles *et al.* [32] ;
- l'algorithme de Mortensen *et al.* [33].

Aucune implémentation n'étant disponible pour l'algorithme de Milles *et al.* [32], nous avons cherché à reproduire au mieux les expérimentations menées par les auteurs. Pour l'algorithme de Mortensen *et al.* [33], nous avons utilisé l'implémentation incluse dans le logiciel de manipulation d'images Gimp [1].

1. https://www.gimp.org/

5.2.1 Méthode de Milles *et al.*

Conditions de test de Milles *et al.*

Afin d'évaluer leur méthode, Milles *et al.* utilisent un sous-ensemble de dix images, extraites des données de référence mises à disposition par Rother *et al.* [43]. À chaque image, une segmentation de référence est associée, correspondant à un problème de binarisation où l'objet à extraire correspond à une unique composante connexe, sans trou. L'adéquation entre l'objet extrait et la segmentation de référence est quantifiée en utilisant l'indice de Jaccard, converti en pourcentage. Soit R une segmentation de référence et S la segmentation produite par une méthode.

Soit R_O les pixels appartenant à l'objet dans la segmentation de référence et S_O ceux attribués à ce même objet dans la segmentation résultat. L'indice de Jaccard mesure la précision pour la classe *objet à extraire* et correspond à la proportion de pixels appartenant à l'objet (à la fois dans la segmentation de référence et dans la segmentation produite) par rapport au nombre total de pixels attribués à l'objet dans l'une ou l'autre des segmentations. En le multipliant par 100 pour obtenir un pourcentage, nous avons :

$$\mathcal{F}_{Jaccard}(R_O, S_O) = 100\frac{|R_O \cap S_O|}{|R_O \cup S_O|}. \tag{5.2}$$

Pour chaque image, Milles *et al.* donnent le minimum, le maximum, la valeur moyenne et l'écart type obtenus avec les différents ensembles de germes.

Interprétation des résultats

Les tests de Milles *et al.* présentent deux particularités qui compliquent leur analyse. Premièrement, ils ne concernent que dix images. Elles correspondent à des problèmes de binarisation relativement simples où l'objet se détache nettement du fond.

Deuxièmement, Milles *et al.* ont fait le choix d'utiliser des germes générés automatiquement. À partir de la segmentation de référence, le contour de l'objet à extraire est découpé en N segments de tailles identiques. Pour chaque segment, un germe est prélevé en sélectionnant un pixel de manière aléatoire. Pour chaque image, 20 ensembles de germes différents sont générés, en faisant varier leur nombre et leur position. Les résultats obtenus par Milles *et al.* dans ces conditions ont été recopiés dans le tableau 5.3.

Le tableau 5.4 montre les résultats obtenus par $S\alpha F$ avec les mêmes images. Les germes ont été donnés manuellement. L'utilisateur disposait d'une durée de deux minutes (maximum) par image pour leur apporter des modifications et améliorer la qualité de la segmentation produite. Le pourcentage moyen de pixels annotés par l'utilisateur est égal à 0,56% des pixels de l'image. Les figures 5.1, 5.2 et 5.3 présentent les germes données à $S\alpha F$ et les segmentations réalisées.

Au vu des conditions de test décrites précédemment, il n'est pas possible de comparer strictement les deux méthodes. D'une part la méthode de Milles *et al.* utilise des germes bien plus précis (ce sont des points de contour) que ceux de $S\alpha F$. D'autre part, les germes de $S\alpha F$ ont été modifiés par l'utilisateur durant le processus de segmentation.

TABLEAU 5.3 – Scores $\mathcal{F}_{Jaccard}$ minimum, moyen (avec l'écart type) et maximum obtenus par la méthode de Milles *et al.* (extraits de l'article [32]).

Nom de l'image	Minimum	Moyenne	Maximum
BANANA1	20,0	$60{,}4 \pm 0{,}20$	89,1
BANANA2	0,7	$47{,}3 \pm 0{,}25$	88,3
BANANA3	26,1	$62{,}5 \pm 0{,}15$	86,6
CERAMIC	74,8	$85{,}6 \pm 0{,}03$	89,8
DOLL	72,5	$80{,}8 \pm 0{,}04$	87,7
FLOWER	1,4	$88{,}1 \pm 0{,}29$	98,2
MUSHROOM	33,0	$61{,}3 \pm 0{,}17$	91,1
MUSIC	97,3	$97{,}8 \pm 0{,}01$	98,6
SHEEP	4,5	$77{,}0 \pm 0{,}18$	90,2
TEDDY	17,6	$74{,}9 \pm 0{,}17$	96,7
Toutes	0,7	$73{,}5 \pm 0{,}23$	91,4

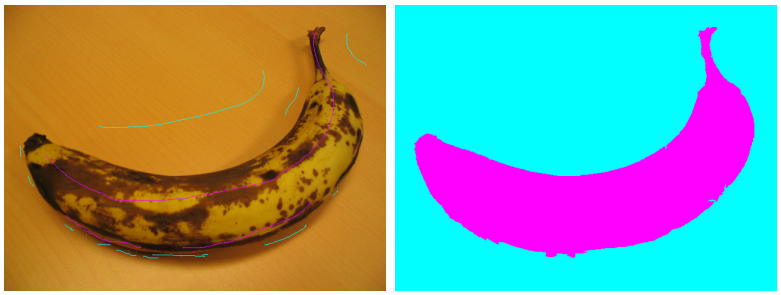
TABLEAU 5.4 – Scores $\mathcal{F}_{Jaccard}$ obtenus par la méthode $S\alpha F$.

Nom de l'image	$S\alpha F$
BANANA1	97,1
BANANA2	96,4
BANANA3	97,9
CERAMIC	97,5
DOLL	98,9
FLOWER	98,6
MUSHROOM	97
MUSIC	98,9
SHEEP	96,2
TEDDY	95,8
Tous	97,2

Avec moins de trois itérations (donc de trois modifications des germes), $S\alpha F$ obtient des résultats très satisfaisants. Par ailleurs, alors que le temps d'exécution de la méthode de Milles *et al.* dépend à la fois du nombre de pixels et du nombre de germes, celui de $S\alpha F$ est relié uniquement au nombre de pixels durant l'étape d'initialisation et au nombre de superpixels durant l'étape de segmentation. Ainsi, tandis que l'algorithme de Milles *et al.* affiche des temps de calcul allant de 3 à 17 secondes pour produire une segmentation une fois que les germes ont été donnés, le temps d'exécution de $S\alpha F$ pour réaliser la même tâche est de seulement 0,6 seconde, dans des conditions matérielles équivalentes.

5.2.2 Méthode de Mortensen *et al.*

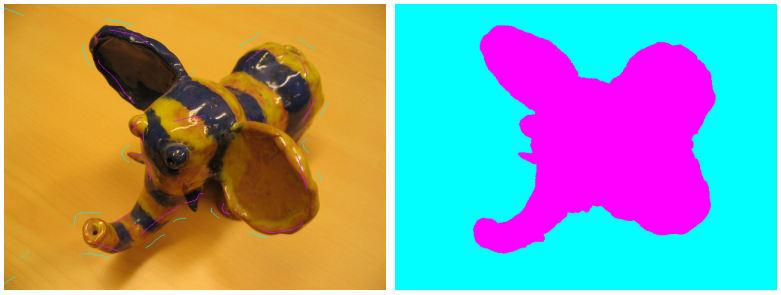
Comme nous disposions d'une implémentation de l'algorithme de Mortensen *et al.* [33], nous avons réalisé des tests sur les données de référence de Rother *et al.* [43]. Les autres ensembles de



(a) Image BANANA1.

(b) Image BANANA2.

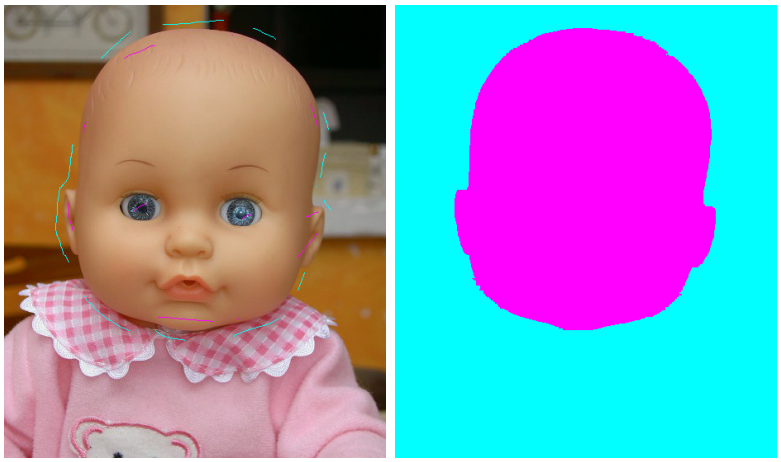
(c) Image BANANA3.



(d) Image CERAMIC.

FIGURE 5.1 – Germes donnés et segmentations produites par $S\alpha F$, pour les images utilisées lors des tests de Milles *et al.*

données de référence ne sont pas adaptés à l'évaluation de méthodes de binarisation interactive par recherche des contours, qui visent à extraire un unique objet correspondant à une surface sans trou. En effet, soient ils contiennent plus de deux classes, soit les objets sont composés de plusieurs composantes connexes ou d'une composante connexe avec des trous. Pour la méthode de



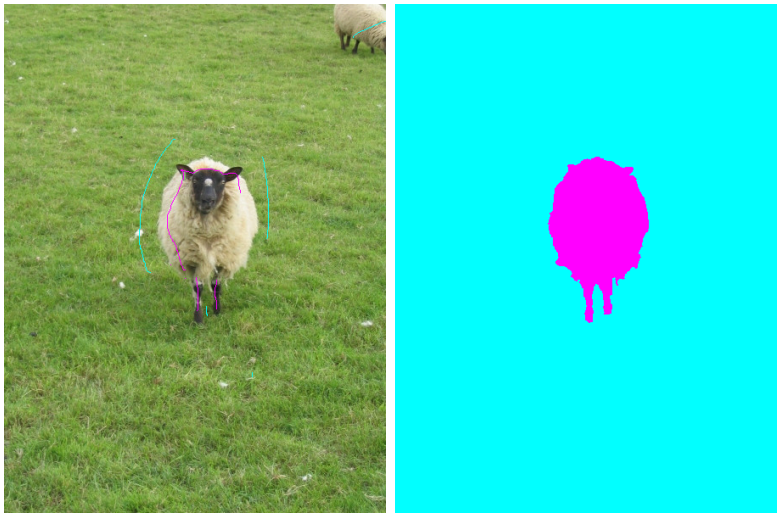
(a) Image DOLL.

(b) Image FLOWER.

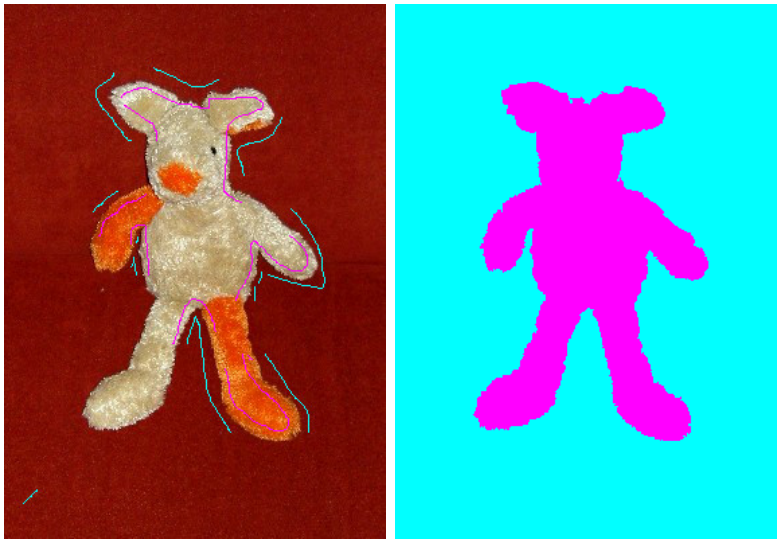
(c) Image MUSHROOM.

FIGURE 5.2 – Germes donnés et segmentations produites par $S\alpha F$, pour les images utilisées lors des tests de Milles *et al.* (suite).

(a) Image MUSIC.



(b) Image SHEEP.



(c) Image TEDDY.

FIGURE 5.3 – Germes donnés et segmentations produites par $S\alpha F$, pour les images utilisées lors des tests de Milles *et al.* (fin).

Mortensen *et al.*, comme pour $S\alpha F$, les germes sont donnés manuellement. Le temps maximal de convergence est de 2 minutes : au delà de cette durée l'utilisateur n'est plus autorisé à modifier les germes. En pratique, pour la totalité des images et les deux algorithmes, le temps de convergence est de quelques secondes.

La méthode de Mortensen *et al.* obtient un score $\mathcal{F}_{Jaccard}$ de 94,5%. L'algorithme $S\alpha F$ permet une amélioration de pratiquement 2%, avec un score de 96,3%.

En termes de temps d'exécution, sur les images traitées, les deux méthodes fonctionnent en temps interactif, avec un temps d'exécution inférieur à 1 seconde.

5.3 Comparaison avec les méthodes de binarisation interactive par recherche des régions

5.3.1 Protocole expérimental de McGuinness *et al.*

L'étude menée en 2010 par McGuinness *et al.* [31] est, à notre connaissance, l'évaluation la plus complète concernant le problème de la binarisation interactive. Elle analyse les performances de quatre algorithmes, proposés respectivement par Boykov *et al.* [4], Salembier *et al.* [45], Friedland *et al.* [15] et Adams *et al.* [2] [2].

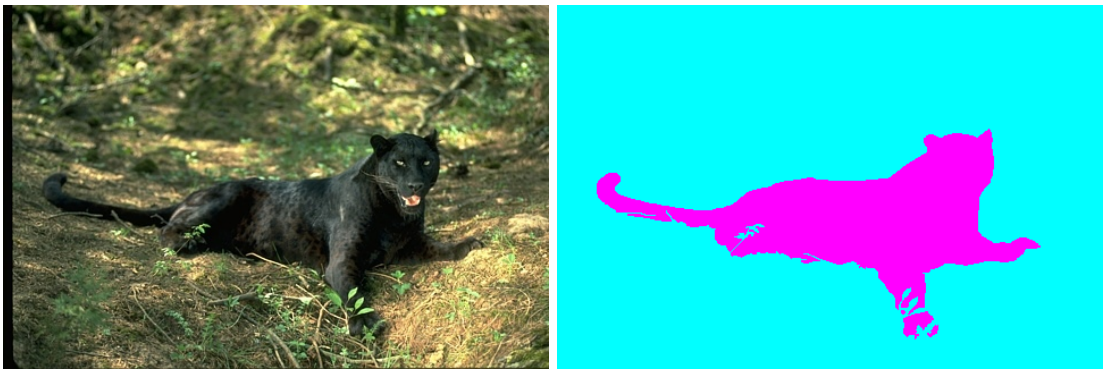
À partir des images de la base de données de Berkeley [30], McGuinness *et al.* ont créé un ensemble de données de référence totalisant 96 images et 100 segmentations de référence. Chaque segmentation de référence correspond à un problème de binarisation où un objet doit être extrait du fond. Ainsi, lorsqu'une image contient plusieurs objets intéressants, plusieurs segmentations de référence lui sont associées. Les objets peuvent contenir des trous ou être constitués de plusieurs régions.

De petite taille (481×321 pixels), les photographies utilisées comprennent malgré tout de nombreux détails, restitués avec précision dans les segmentations de référence. La figure 5.4a en fournit un exemple. En outre, les objets à extraire n'occupent pas nécessairement la majorité de l'image, comme dans le cas du dromadaire de la figure 5.4b. Tous ces aspects contribuent à faire des données de référence de McGuinness *et al.* un ensemble de problèmes de binarisation difficiles.

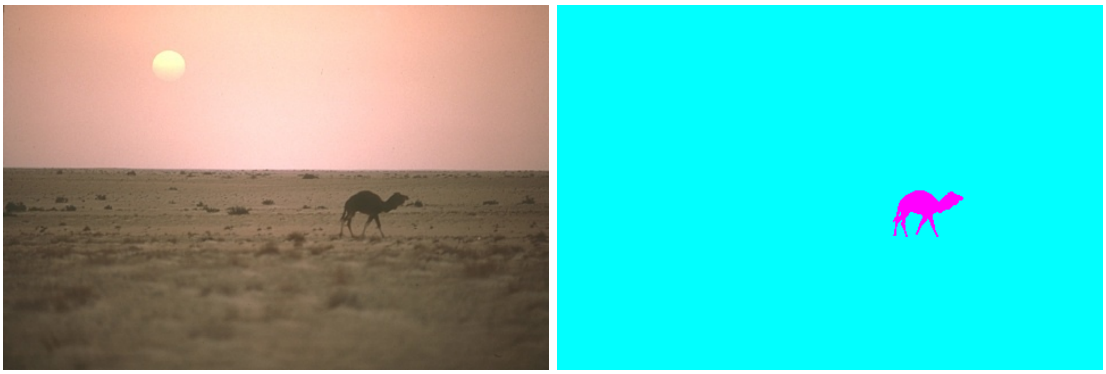
Les tests mis en place par McGuinness *et al.* font intervenir un panel d'utilisateurs, testant chacune des méthodes en lui donnant manuellement des germes. Afin d'aboutir à une segmentation la plus proche possible de chaque segmentation de référence, McGuinness *et al.* autorisent l'utilisateur à ajouter ou supprimer autant de germes que nécessaire, durant au plus deux minutes (il s'agit de ce que nous nommons le temps maximal de convergence). Au delà de cette durée, l'utilisateur n'est plus autorisé à modifier les germes. Ce temps ne doit pas être confondu avec le temps d'exécution nécessaire à un algorithme pour produire une segmentation une fois les germes donnés.

À la fin des tests, la qualité des segmentations est évaluée grâce à deux mesures : l'indice de Jaccard $\mathcal{F}_{Jaccard}$, présenté dans la section précédente et une mesure floue d'adéquation aux

2. L'algorithme d'Adams *et al.* [2] permet aussi de résoudre des problèmes multiclasses.



(a) Problème de segmentation où le fond comprend de petits éléments, complexes à extraire.



(b) Problème de segmentation où l'objet à extraire est très petit par rapport au fond.

FIGURE 5.4 – Exemples d'images (à gauche) et de segmentations de référence (à droite) mises à disposition par McGuinness *et al.* [31].

contours, $\mathcal{F}_{Cnt}$.

Soit R une segmentation de référence et S une segmentation produite par une méthode de segmentation interactive. Comme nous nous intéressons à des problèmes de binarisation, les segmentations R et S donnent chacune une partition de l'image en deux ensembles :

- R_O et R_F, qui séparent les pixels de l'objet des pixels du fond dans la segmentation de référence ;
- S_O et S_F, qui séparent les pixels de l'objet des pixels du fond dans la segmentation résultat.

Un pixel appartenant à l'un des ensembles et ayant un de ses voisins dans un autre ensemble, constitue un point de contour. McGuinness *et al.* souhaitent vérifier que les contours dans S suivent les contours dans R.

Soit C_R l'ensemble des points de contour de R et C_S l'ensemble des points de contours de S. La mesure suivante, quantifie l'adéquation entre les contours dans R et ceux dans S :

$$\mathcal{F}_{Cnt} = \frac{|C_R \cap C_S|}{|C_R \cup C_S|}. \tag{5.3}$$

L'un des inconvénients de cette mesure est qu'elle ne prend en compte aucune marge d'erreur :

les pixels des contours doivent coïncider parfaitement. L'idée de McGuinness *et al.* consiste à étendre les ensembles C_R et C_S à l'aide de la théorie des sous-ensembles flous, afin de permettre que deux pixels de contour proches l'un de l'autre contribuent également à améliorer le score.

Ainsi, pour un pixel p_i de l'image, son degré d'appartenance à un contour d'une segmentation S est donné par la fonction gaussienne :

$$\mathcal{F}_{App}(p_i, S) = \exp\left(-\frac{||p_i - p_j||}{\sigma^2}\right) \tag{5.4}$$

où

- $||p_i - p_j||$ est la distance euclidienne entre les pixels p_i et p_j ;
- $p_j = \underset{p_k \in C_S}{\arg\min} ||p_i - p_k||$ est le pixel, appartenant à un contour, le plus proche de p_i ;
- σ est un paramètre permettant de régler le seuil de tolérance ; McGuinness *et al.* proposent de le fixer à 4.

La mesure d'adéquation aux contours, exprimée en pourcentage, devient alors :

$$\mathcal{F}_{FCnt} = 100 \frac{\sum_{i=1}^{N_I} \min(\mathcal{F}_{App}(p_i, S), \mathcal{F}_{App}(p_i, R))}{\sum_{i=1}^{N_I} \max(\mathcal{F}_{App}(p_i, S), \mathcal{F}_{App}(p_i, R))}. \tag{5.5}$$

Pour chacune de ces deux mesures, $\mathcal{F}_{Jaccard}$ et $\mathcal{F}_{FCnt}$ les scores moyens obtenus sur l'ensemble des images sont donnés.

5.3.2 Méthodes testées par McGuinness *et al.*

Le tableau 5.5 résume les scores obtenus par les algorithmes de Boykov *et al.* [4], de Salembier *et al.* [45], de Friedland *et al.* [15] et d'Adams *et al.* [2] lors de l'évaluation réalisée par McGuinness *et al.* [31]. Il donne également les scores atteints par $S\alpha F$ dans des conditions expérimentales similaires. Le temps maximum de convergence, de deux minutes par image, a été respecté. Dans la majorité des cas, le temps réel de convergence pour $S\alpha F$ est bien inférieur : de quelques secondes pour les images les plus simples et jusqu'à une minute, pour la majorité des images.

Tableau 5.5 – Comparaison de $S\alpha F$ avec les méthodes évaluées par McGuinness *et al.*

Algorithme	$\mathcal{F}_{Jaccard}$	$\mathcal{F}_{FCnt}$
Boykov *et al.*	92	77
Salembier *et al.*	92	78
Friedland *et al.*	85	64
Adams *et al.*	88	70
$S\alpha F$	95	83

Les résultats du tableau 5.5 montrent que les segmentations produites par $S\alpha F$ sont plus proches des segmentations de référence que celles des autres algorithmes. En particulier, le score de $S\alpha F$ est supérieur de 2% à celui obtenu par les méthodes de Boykov *et al.* et de Salembier *et al.* pour la mesure $\mathcal{F}_{Jaccard}$ et de plus de 5% pour la mesure $\mathcal{F}_{FCnt}$. Ils indiquent que, malgré l'utilisation des superpixels, $S\alpha F$ réussit à extraire précisément les détails contenus dans les images. Ces résultats ont été obtenus avec des temps d'exécution autour de 0,7 seconde, ce qui est similaire aux temps d'exécution des méthodes comparées. La figure 5.5 illustre par quelques exemples le comportement de $S\alpha F$.

Sur l'image de la figure 5.5a, l'objet à extraire est nettement distinguable du fond. L'utilisation d'une information de couleur permet d'obtenir une segmentation correcte avec très peu de germes. Au contraire, l'objet à extraire de la figure 5.5b est l'un des trois porcelets blancs. L'information de couleur n'est donc pas suffisante pour l'isoler. Ici, c'est l'information de localisation qui permet d'obtenir la segmentation désirée. Enfin, la figure 5.5c correspond à l'un des exemples d'images comprenant de petits détails et qui a été présentée précédemment.

5.3.3 Algorithme de Jian *et al.*

Nous n'avons malheureusement pas pu réaliser une comparaison rigoureuse entre $S\alpha F$ et la méthode de binarisation interactive par recherche des régions la plus récente, proposé par Jian *et al.* [18]. En effet, les auteurs affirment avoir utilisé comme données d'évaluation une version étendue de l'ensemble de données de référence de McGuinness *et al.*, sans la mettre à disposition. Nous soulignerons toutefois que les performances atteintes par $S\alpha F$ sur les images de McGuinness *et al.* correspondent à une amélioration de 2% [3] par rapport à l'algorithme de Jian *et al.* Les temps d'exécution de $S\alpha F$ sont supérieurs de quelques dixièmes de secondes.

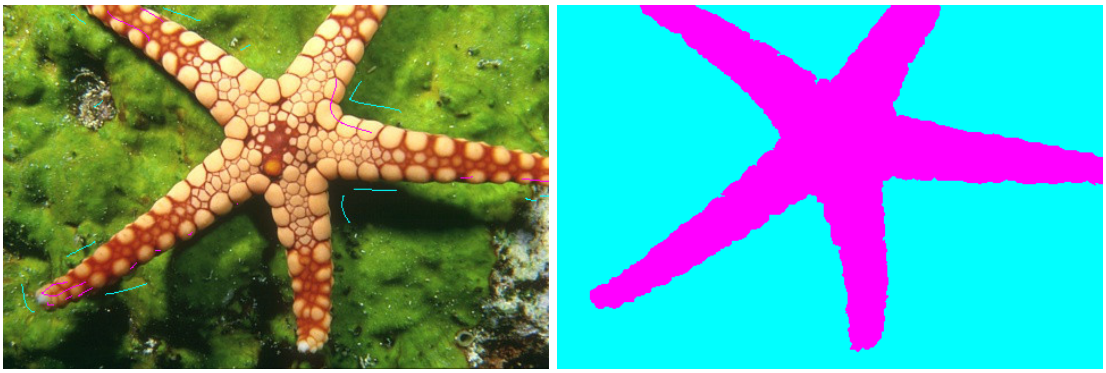
5.4 Comparaison avec les méthodes de segmentation interactive multiclasse

Nous avons comparé $S\alpha F$ à trois méthodes de segmentation multiclasse : celle de Santner *et al.* [46], celle de Müller *et al.* [35] et celle de Changjae *et al.* [6].

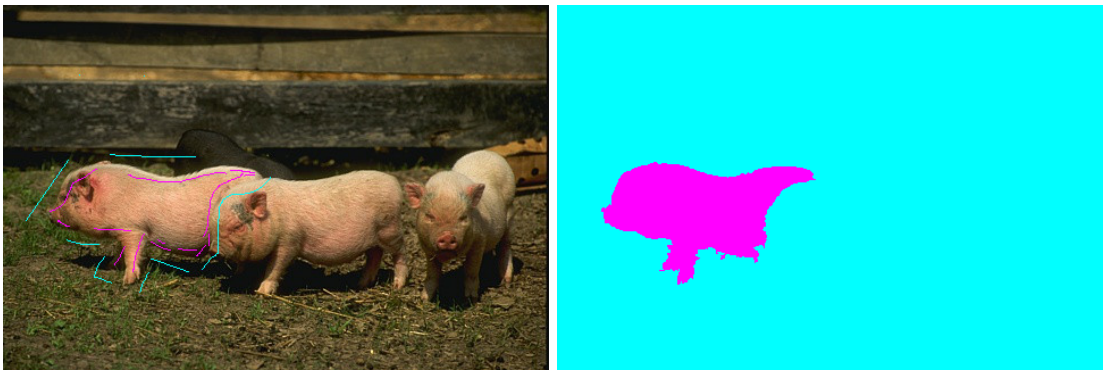
5.4.1 Données utilisées

Ces trois méthodes ont été évaluées avec les données de référence de Santner *et al.*, qui comprennent 158 photographies de 625×391 pixels et 262 segmentations de référence, pour des problèmes multiclasses. La figure 5.6 donne la répartition du nombre de classes pour ces segmentations. Elle montre qu'une majorité des problèmes posés comprennent entre 2 et 4 classes.

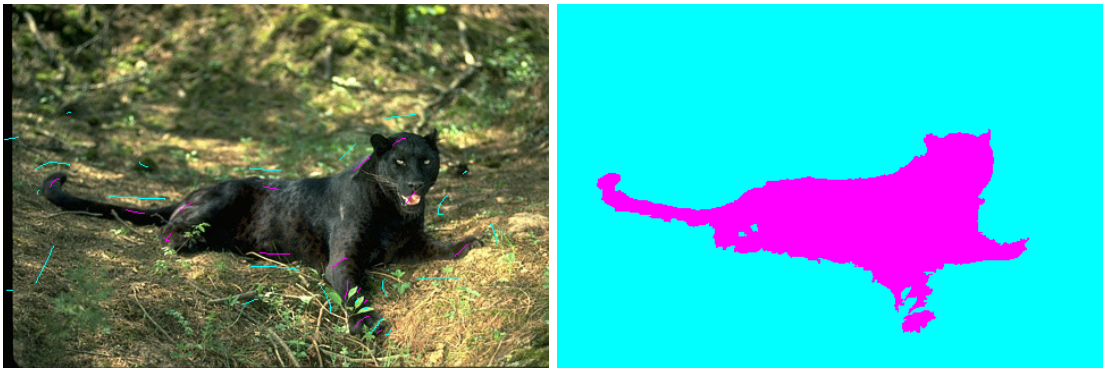
3. Ce pourcentage correspond à la différence obtenue pour la mesure $\mathcal{F}_{Jaccard}$, Jian *et al.* n'ayant pas communiqué de chiffres pour la mesure $\mathcal{F}_{FCnt}$.



(a) Objet fortement contrasté avec le fond : l'utilisation d'une information de couleur permet d'obtenir une segmentation précise à l'aide de seulement quelques germes.



(b) Image où l'objet à extraire est très similaire à une partie du fond : l'information de localisation permet malgré tout d'obtenir une segmentation précise.



(c) Image où le fond comprend de petits éléments difficiles à extraire.

FIGURE 5.5 – Exemples de germes donnés et de résultats obtenus par $S\alpha F$ sur les données de référence de McGuinness *et al.*

5.4.2 Mesure de la précision d'une segmentation

Afin d'évaluer l'adéquation entre une segmentation de référence R et une segmentation produite S, Santner *et al.*, Müller *et al.* et Changjae *et al.* utilisent la mesure $\mathcal{F}_{DICE}$ [46], dont la

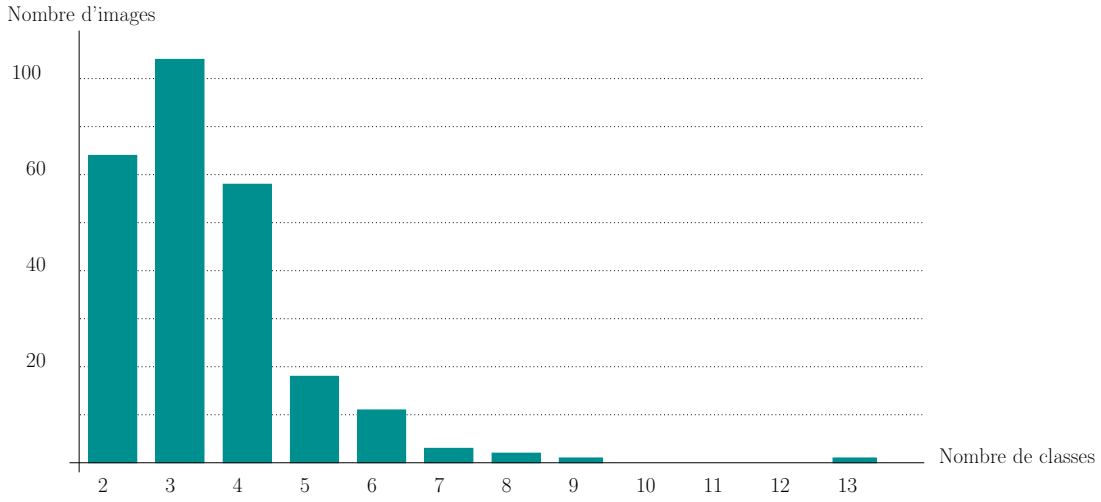


FIGURE 5.6 – Répartition du nombre de classes pour les données de référence mises à disposition par Santner *et al.* [46].

valeur convertie en pourcentage vaut :

$$\mathcal{F}_{DICE} = 100 \sum_{i=1}^{N_\Lambda} 2 \frac{|R_i \cap S_i|}{|R_i \cup S_i|}. \tag{5.6}$$

Les ensembles R_i et S_i correspondent à l'ensemble des pixels de label λ_i, respectivement dans la segmentation de référence et dans la segmentation produite. Santner *et al.*, Müller *et al.* et Changjae *et al.* donnent les scores $\mathcal{F}_{DICE}$ moyens obtenus par leurs méthodes pour l'ensemble des images testées.

5.4.3 Méthodes de Santner *et al.* et de Müller *et al.*

En plus des données de référence, Santner *et al.* [46] ont mis à disposition les germes utilisés par leur méthode. Ces germes ont été donnés à l'aveugle, c'est-à-dire sans voir le résultat produit par la méthode de segmentation interactive. En utilisant ces germes, leur méthode obtient un score $DICE$ moyen de 93. Celle de Müller *et al.* [35] atteint 94.

Résultats avec les germes de Santner *et al.*

Avec les germes de Santner *et al.*, $S\alpha F$ obtient de mauvais résultats, avec un score $\mathcal{F}_{DICE}$ moyen de 83%. Pour cette mesure, l'écart type est élevé (14%). L'analyse de l'histogramme des résultats obtenus pour l'ensemble des images montre que :

- pour 32% des couples (germes, image) le score $\mathcal{F}_{DICE}$ se situe entre 94 et 99 ;
- pour 35% des couples (germes, image) le score $\mathcal{F}_{DICE}$ se situe entre 80 et 93 ;
- pour 33% des couples (germes, image) le score $\mathcal{F}_{DICE}$ se situe entre 40 et 79.

Pour 67% des données, nous obtenons donc des résultats « *normaux* » (compris entre 80 et 99). L'étude des 33% de données aboutissant à des segmentations fortement erronées permet de mieux comprendre les limites de $S\alpha F$.

Les figures 5.7 et 5.8 donnent deux exemples où les germes de Santner *et al.* ne permettent pas à $S\alpha F$ de produire une segmentation adéquate. Elles comprennent également un exemple, où, avec des germes différents, une segmentation bien meilleure est obtenue. L'étude de la différence entre les deux types de germes montre que pour que $S\alpha F$ produise une segmentation pertinente les germes doivent positionnés près des contours des différents objets.

Dans certains cas, comme avec l'exemple de la figure 5.7, cela implique de donner des germes plus nombreux que ceux de Santner *et al.* Dans de nombreux autres cas, à l'instar de l'exemple présenté sur la figure 5.8, les germes ont simplement besoin d'être positionnés différemment.

Comme $S\alpha F$ repose sur l'utilisation d'un SVM et d'un descripteur contenant une information de localisation, les germes de Santner *et al.* sur les figures 5.7 et 5.8 ne sont pas répartis de manière à permettre au SVM de trouver tous les vecteurs de support adéquats pour produire une segmentation correcte. Cette spécificité a été décrite dans la section 3.4.3. Dans la section 5.7 nous montrerons que cette limitation est compensée par le fait que l'impact des germes donnés est aisément prédictible, ce qui facilite l'utilisation de $S\alpha F$.

Tests avec de nouveaux germes

Nous avons effectué des tests supplémentaires avec deux ensembles de germes, différents de ceux de Santner *et al.* [46] :

- $S\alpha F_0$, l'ensemble des germes initiaux, donnés avant que l'utilisateur ne voit le résultat de l'étape de segmentation ;
- $S\alpha F_N$, l'ensemble des germes corrigés par l'utilisateur, correspondant donc à la dernière itération), afin que la segmentation produite soit la plus proche possible de la segmentation de référence.

À l'instar des germes de Santner *et al.* [46], les germes de $S\alpha F_0$ ont été obtenus sans permettre à l'utilisateur de voir le résultat de la segmentation et de s'en servir pour guider la méthode. Cependant, une consigne supplémentaire a été donnée à ce dernier : donner des germes initiaux proches des contours des objets.

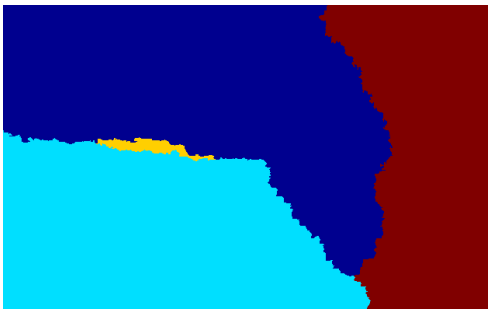
Avec les germes $S\alpha F_0$ et les images de Santner *et al.*, $S\alpha F$ obtient une score $\mathcal{F}_{DICE}$ moyen de 97. En permettant à l'utilisateur d'enlever ou d'ajouter des germes afin de corriger les erreurs commises par $S\alpha F$ (ce qui correspond aux germes de l'ensemble $S\alpha F_N$) ce score passe à 98. Les erreurs restantes proviennent de l'étape de sur-segmentation et ne peuvent généralement pas être corrigées.

Le temps d'exécution moyen par image est de 1,2 seconde. Celui des méthodes de Santner *et al.* et de Müller *et al.* avoisine les 2 secondes. Cependant, alors que ces deux algorithmes reposent sur une implémentation tirant parti de la puissance de calcul d'une carte graphique, $S\alpha F$ n'a fait l'objet d'aucune optimisation particulière et ne nécessite pas une configuration matérielle spécifique. Nous verrons dans la section 5.8.2 que cette faible complexité permet d'utiliser $S\alpha F$ au sein d'une application mobile.

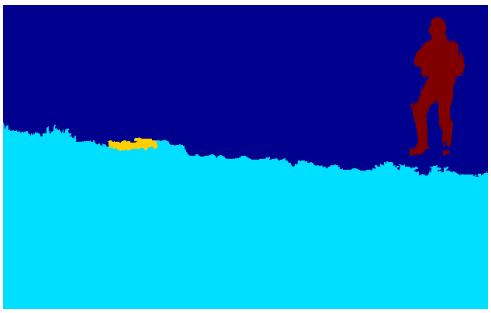
(a) Image originale.

(b) Germes de Santner *et al.*

(c) Germes donnés en demandant à l'utilisateur de donner des germes près des contours.



(d) Résultat de $S\alpha F$ avec les germes de Santner *et al.*



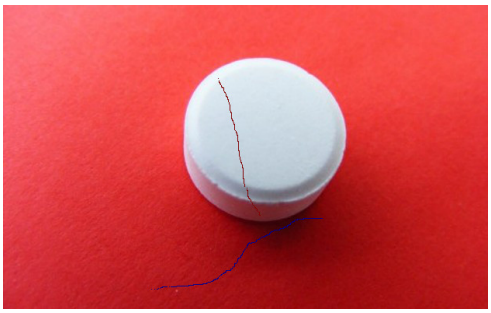
(e) Résultat de $S\alpha F$ avec les germes près des contours.

FIGURE 5.7 – Illustration de l'un des cas où les germes de Santner *et al.* ne conviennent pas à $S\alpha F$. Ici des germes plus nombreux doivent être donnés pour marquer les délimitations entre la rivière et l'herbe ainsi qu'entre l'herbe et le pêcheur.

Ces résultats nous permettent de conclure que $S\alpha F$, malgré la contrainte qu'il impose au niveau des germes, est une méthode compétitive par rapport à celles de Santner *et al.* et de Müller *et al.*

Quelques autres exemples de segmentations produites par $S\alpha F$ pour les problèmes de segmentation multiclasse de Santner *et al.* sont donnés sur la figure 5.9.

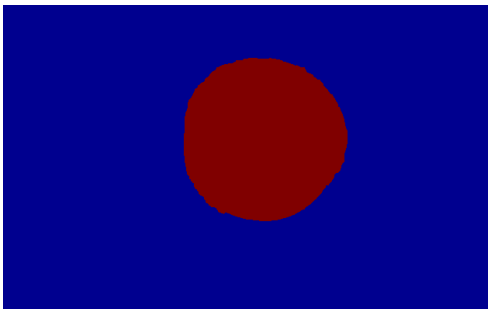
(a) Image originale.



(b) Germes de Santner *et al.*

(c) Germes donnés en demandant à l'utilisateur de donner des germes près des contours.

(d) Résultat de $S\alpha F$ avec les germes de Santner *et al.*



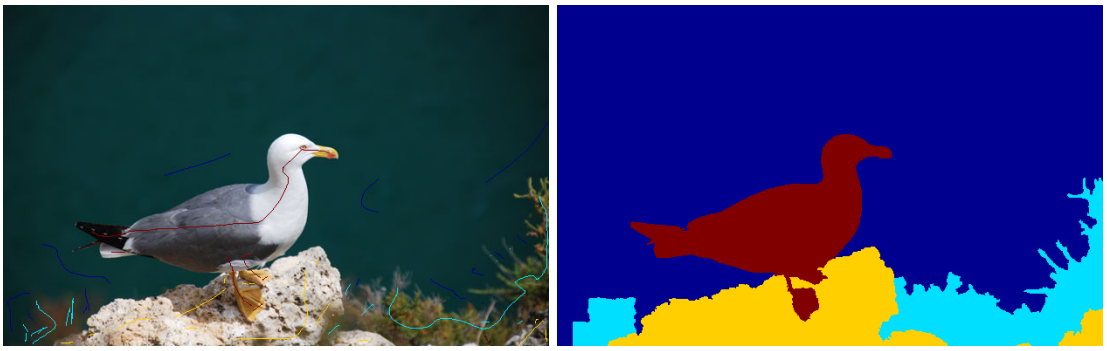
(e) Résultat de $S\alpha F$ avec les germes près des contours.

FIGURE 5.8 – Illustration de l'un des cas où les germes de Santner *et al.* ne conviennent pas à $S\alpha F$. Ici, il n'est pas nécessaire d'ajouter davantage de germes : ils doivent seulement être positionnés différemment.

5.4.4 Algorithme de Changjae *et al.*

Changjae *et al.* [6] ont évalué leur méthode sur les données de référence de Rother *et al.* [43] ainsi que sur 50 images parmi les données de référence de Santner *et al.* [46]. Ils utilisent eux aussi des germes donnés manuellement, mais n'étant pas modifié par l'utilisateur. Comme ces germes n'ont pas été mis à disposition, une comparaison avec $S\alpha F$ est délicate.

Les scores $\mathcal{F}_{DICE}$ obtenus par l'algorithme de Changjae *et al.* [6] sont, dans le meilleur des cas, de 93 pour les données de Rother *et al.* [43] et de 90 sur les 50 images issues des données



(a) Ici, peu de germes sont nécessaires pour segmenter correctement le fond de couleur unie. Les branchages, pourtant moins nets, sont également correctement extraits. Tandis que les contours du goéland sont réguliers et arrondis, ceux du rocher restituent la granularité de ce dernier.

(b) Bien qu'ils constituent des détails de faible dimension, les ergots du perroquet sont correctement extraits.

(c) Bien qu'il contienne des petits détails architecturaux, le bâtiment est correctement segmenté.

FIGURE 5.9 – Exemples de germes donnés et de résultats obtenus sur les données de référence de Santner *et al.* [46].

de référence de Santner *et al.* [46]. Sans permettre à l'utilisateur de modifier les germes, $S\alpha F$ améliore le score sur les données de Rother *et al.* [43] de 3 points. Comme nous ne connaissons pas les 50 images retenues parmi les données de Santner *et al.*, nous ne pouvons pas tirer de conclusion, même si le score obtenu par $S\alpha F$ pour l'ensemble de ces données est de 97, sans modification des germes.

Le bénéfice le plus clair de $S\alpha F$ reste au niveau du temps d'exécution, significativement

plus court : à peine plus d'une seconde pour $S\alpha F$, tandis que 40 secondes sont nécessaires à la méthode Changjae *et al.*

5.5 Bénéfice apporté par le terme de régularisation

Afin d'analyser l'impact du terme de régularisation, nous avons conçu une version simplifiée de $S\alpha F$ qui, au lieu de la fonction décrite par l'équation 3.12, minimise :

$$\mathcal{F}_{SCIS}(C, I) = \prod_{i=1}^{N_I} \exp(\mathcal{F}_D(x_i, c_i)). \tag{5.7}$$

Comme seul le terme d'attache aux données apparaît dans la fonction $\mathcal{F}_{SCIS}$, la segmentation correspondant à son minimum est obtenue en attribuant à chaque superpixel la classe la plus probable, c'est-à-dire en utilisant directement la classe ayant la plus forte probabilité, selon l'approche de Wu *et al.* [56]. Nous avons nommé cette variante $SCIS$, de l'anglais « *Superpixel Classification-based Interactive Segmentation* ».

Nous avons comparé $SCIS$ et $S\alpha F$ à partir des données de référence de Rother *et al.* [43], de McGuinness *et al.* [31] et de Santner *et al.* [46]. En donnant de nombreux germes, nous avons réussi à obtenir avec $SCIS$:

- un score $\mathcal{F}_{Jaccard}$ égal à celui de $S\alpha F$ pour les données de Rother *et al.* ;
- des scores $\mathcal{F}_{Jaccard}$ et $\mathcal{F}_{FCnt}$ identiques à ceux de $S\alpha F$ pour les données de McGuiness *et al.* ;
- un score $\mathcal{F}_{DICE}$ similaire pour les données de Santner *et al.*

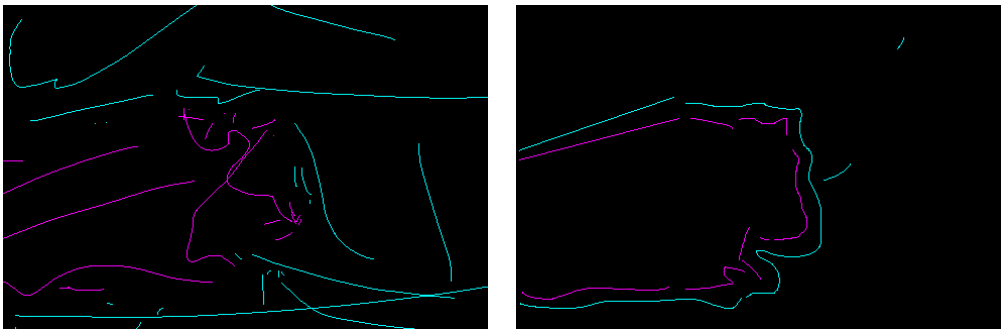
La complexité algorithmique de $SCIS$ est moins élevée que celle de $S\alpha F$. Ainsi, lors de l'étape d'initialisation, il n'est pas nécessaire d'estimer $P_i(j)$, la probabilité pour un superpixel de classe de label λ_i d'avoir comme voisin un superpixel attribué à la classe de label λ_j. Quant à l'étape de segmentation, elle ne nécessite pas d'utiliser l'algorithme α-fusion. Néanmoins, cette simplicité se paie par un nombre de germes plus élevé.

Données	Rother *et al.*		McGuinness *et al.*		Santner *et al.*	
	$SCIS$	$S\alpha F$	$SCIS$	$S\alpha F$	$SCIS$	$S\alpha F$
Pr (%)	$2{,}08 \pm 0{,}36$	$0{,}74 \pm 0{,}36$	$2{,}08 \pm 0{,}49$	$1{,}26 \pm 0{,}69$	$1{,}42 \pm 0{,}42$	$0{,}89 \pm 0{,}43$
T_{init} (s)	$0{,}6 \pm 0{,}18$	$0{,}61 \pm 0{,}19$	$0{,}4 \pm 0{,}01$	$0{,}41 \pm 0{,}01$	$0{,}64 \pm 0{,}02$	$0{,}66 \pm 0{,}02$
T_{seg} (s)	$0{,}04 \pm 0{,}02$	$0{,}27 \pm 0{,}02$	$0{,}06 \pm 0{,}03$	$0{,}68 \pm 0{,}04$	$0{,}06 \pm 0{,}04$	$0{,}51 \pm 0{,}28$

Tableau 5.6 – Comparaison entre $SCIS$ et $S\alpha F$. Moyennes et écarts types pour le pourcentage de pixels annotés par l'utilisateur (Pr) ainsi que les temps d'exécution lors de l'étape d'initialisation (T_{init}) et de l'étape de segmentation (T_{seg}). Ces valeurs sont obtenues en produisant des segmentations dont la précision est similaire.

Le tableau 5.6 donne, pour chaque ensemble de données de référence et pour chaque méthode, le temps d'exécution pour l'étape d'initialisation, le temps d'exécution pour l'étape de segmen-

tation et le pourcentage de pixels annotés par l'utilisateur, à l'issue de la dernière itération. Ces résultats montrent que $SCIS$ nécessite environ 1% de germes supplémentaires (ce pourcentage étant donné par rapport à l'ensemble des pixels de l'image). Concrètement, il s'agit de plus 200 000 pixels supplémentaires assignés à une classe par l'utilisateur. Ces germes doivent par ailleurs être répartis de manière beaucoup plus uniforme dans l'image en termes de localisation, et cela dans l'ensemble de l'image et pas uniquement près des contours. La figure 5.10 montre un exemple des différences entre les germes donnés pour $SCIS$ et ceux pour $S\alpha F$.



(a) Germes nécessaires à $SCIS$. (b) Germes nécessaires à $S\alpha F$.

FIGURE 5.10 – Réduction du nombre de germes par l'utilisation du terme de régularisation.

Si la différence entre les germes à donner pour $S\alpha F$ et ceux nécessaires à $SCIS$ correspond à un effort nettement perceptible par l'utilisateur, les variations en termes de temps d'exécution sont plus ténues. Elles concernent très peu l'étape d'initialisation : dans le pire des cas l'écart est de seulement 0,02 seconde. Avec des images de grande taille, il est cependant probable que la différence entre les deux méthodes se creuse, en faveur de $SCIS$.

Pour l'étape de segmentation, la différence entre les durées est plus marquée. Néanmoins, la complexité de cette étape étant liée au nombre de superpixels et pas à la taille de l'image, le fait d'augmenter le nombre de pixels ne la fait pas croître. Comme nous le verrons dans la section suivante, le temps nécessaire à $S\alpha F$ pour segmenter l'image après avoir donné les germes reste autour de 0,5 seconde, même en augmentant sensiblement la taille de l'image.

Afin de conclure cette analyse, il convient de décrire un peu le fonctionnement de $S\alpha F$ tel qu'il est perçu par l'utilisateur. L'étape d'initialisation ne nécessite pas la présence des germes et peut parfaitement être réalisé à l'insu de l'utilisateur, pendant que ce dernier est en train de sélectionner les germes. Lorsque l'utilisateur a terminé, la méthode termine si nécessaire l'initialisation et effectue, en moins d'une seconde, l'étape de segmentation. À partir de ce moment, seule l'étape de segmentation est répétée, tant que l'utilisateur modifie les germes.

En définitive, le temps passé par l'utilisateur à ajouter les germes nécessaires à $SCIS$ est nettement plus perceptible que la différence de temps d'exécution entre $SCIS$ et $S\alpha F$. L'utilisation d'un terme de régularisation améliore donc l'ergonomie de $S\alpha F$.

5.6 Passage à l'échelle

5.6.1 Protocole

Afin d'analyser le passage à l'échelle de l'algorithme $S\alpha F$, nous avons créé, à partir des données de référence de HSID, trois nouveaux ensembles de référence $HSID'_1$, $HSID'_2$ et $HSID'_3$. Ils sont composés des mêmes images mais à des tailles différentes. Chaque ensemble de référence comprend 50 images. Les photographies de $HSID'_1$ contiennent 1800×1201 pixels. Celles de $HSID'_2$ ont été ramenées à 1350×901 de pixels et celles de $HSID'_3$ à 1013×676 pixels.

Une unique segmentation de référence est associée à chaque image de chaque ensemble. Les segmentations de référence de $HSID'_{i-1}$ correspondent aux segmentations de référence de $HSID'_i$ dont les tailles ont été réduites. Lorsque cette réduction de la taille a engendré des erreurs, elles ont été corrigées manuellement. Les germes pour chaque ensemble ont été obtenu selon le même procédé, en partant des germes donnés pour $HSID'_1$.

Comme le montre la figure 5.11, le nombre de classes dans les segmentations se situent principalement entre 2 et 4 classes. Le nombre de classes est le même pour chacun des trois ensembles de données de référence.

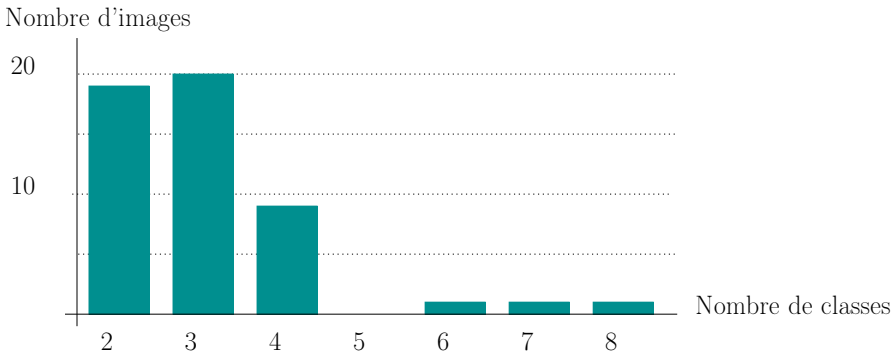


FIGURE 5.11 – Répartition du nombre de classes pour chaque ensemble de référence $HSID'_i$.

Pour évaluer l'adéquation entre les segmentations produites par $S\alpha F$ et les segmentations de références nous avons utilisé la mesure $\mathcal{F}_{DICE}$. Nous nous sommes également intéressés au temps d'exécution de l'étape d'initialisation et de l'étape de segmentation. Nous avons calculé la valeur moyenne, l'écart type, le minimum et le maximum de ces trois mesures, pour chacun des trois ensembles de référence.

Enfin, durant l'ensemble de nos tests, nous n'avons pas modifié les paramètres. En particulier, toutes les images sont sur-segmentées en 3000 superpixels environ.

5.6.2 Analyse des résultats

Les résultats obtenus par $S\alpha F$ sont présentés dans :

- le tableau 5.7, pour le score $\mathcal{F}_{DICE}$;

- le tableau 5.8, pour le temps d'exécution lors de l'étape d'initialisation ;
- le tableau 5.9, pour le temps d'exécution lors de l'étape de segmentation.

Données	Moyenne ± écart type	Minimum	Maximum
$HSID'_3$	$99\% \pm 1{,}03$	95	100
$HSID'_2$	$99\% \pm 0{,}99$	95	100
$HSID'_1$	$99\% \pm 1{,}12$	95	100

TABLEAU 5.7 – Valeurs de la mesure $\mathcal{F}_{DICE}$ obtenues par $S\alpha F$.

L'analyse de l'adéquation entre les segmentations produites et les segmentations de référence montre que la qualité des résultats de $S\alpha F$ n'est pas influencée par l'augmentation ou la diminution des dimensions de l'image. L'écart type relativement faible, ainsi que la différence ténue (seulement 5%) entre la valeur minimale et la valeur maximale, indiquent par ailleurs que les performances de $S\alpha F$ demeurent stables, quelle que soit la segmentation de référence à atteindre.

Données	Moyenne ± écart type	Minimum	Maximum
$HSID'_3$	$1{,}79 \pm 0{,}06$	1,71	1,98
$HSID'_2$	$3{,}25 \pm 0{,}12$	3,06	3,64
$HSID'_1$	$5{,}73 \pm 0{,}25$	5,34	6,33

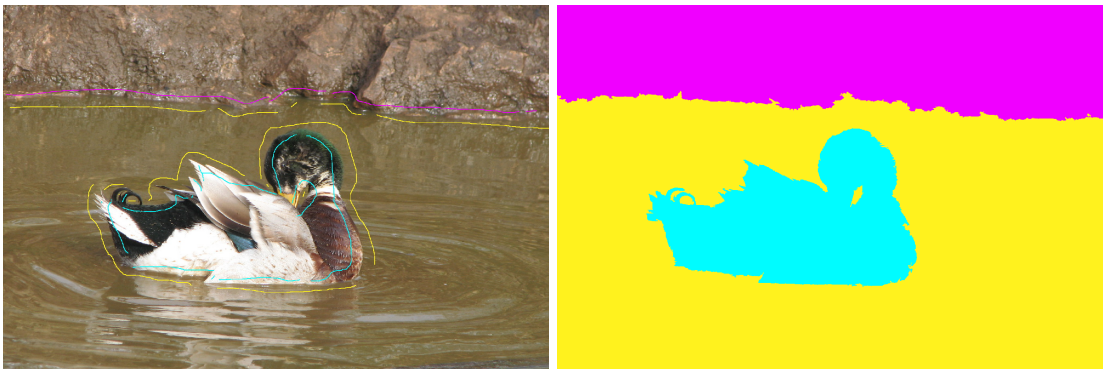
TABLEAU 5.8 – Temps d'exécution de $S\alpha F$ (en secondes) lors de l'étape d'initialisation.

Données	Moyenne ± écart type	Minimum	Maximum
$HSID'_3$	$0{,}46 \pm 0{,}28$	0,21	1,64
$HSID'_2$	$0{,}46 \pm 0{,}28$	0,21	1,68
$HSID'_1$	$0{,}41 \pm 0{,}22$	0,2	1,2

TABLEAU 5.9 – Temps d'exécution de $S\alpha F$ (en secondes) lors de l'étape de segmentation.

Le résultat le plus intéressant de cette évaluation concerne les temps d'exécution pour chacune des deux étapes. Tandis que le temps nécessaire pour l'initialisation augmente avec le nombre de pixels, la durée de la phase de segmentation demeure stable et en dessous de la seconde. Or, l'étape d'initialisation ne nécessite pas la présence des germes. Elle peut donc être réalisée avant ou pendant que l'utilisateur les donne. Le temps perçu est alors réduit à la seule durée de la segmentation, permettant à l'utilisateur de percevoir une très bonne réactivité de $S\alpha F$.

Ce résultat est tout à fait cohérent, puisque l'étape d'initialisation travaille à l'échelle du pixel tandis que l'étape de segmentation manipule les superpixels. Or, si le nombre de pixels augmente avec la taille de l'image, le nombre de superpixels, lui, ne varie pas. La figure 5.12 donne quelques exemples des germes utilisés et des segmentations obtenues sur $HSID'_2$.



(a) Le faible contraste entre l'eau et la pierre n'empêche pas de séparer ces deux éléments.

(b) Utiliser des superpixels n'empêche pas de segmenter avec précision la fourrure du glouton. Quelques erreurs sont cependant visibles au niveau de la patte avant gauche dont la couleur se confond avec l'ombre. Ces erreurs sont commises au niveau des superpixels et ne peuvent pas être corrigées même en ajoutant des germes.

FIGURE 5.12 – Exemples de germes donnés et de résultats obtenus sur l'ensemble de référence $HSID'_2$.

5.7 Ergonomie

5.7.1 Problématique

Les questionnaires pour les utilisateurs réalisés et analysés par McGuinness *et al.* [31] montrent que, pour qu'une méthode de segmentation interactive soit reçue favorablement, il ne lui suffit pas de permettre la production rapide d'une segmentation précise : il est également essentiel que l'utilisateur, même néophyte, comprennent rapidement comment positionner les germes pour arriver au résultat souhaité. Ainsi, même si l'algorithme de Salembier *et al.* [45] obtient un meilleur score d'adhérence aux contours que celui de Boykov *et al.* [4], c'est ce dernier que les utilisateurs plébiscitent.

Par sa structure, la méthode de Boykov *et al.* favorise un impact local des germes : en particulier, ajouter des germes à un endroit de la photographie n'influe pas, ou très peu, le reste de la segmentation. Nous nous sommes appuyés sur ce résultat lors de la conception de $S\alpha F$, notamment en choisissant d'introduire une information de localisation dans le descripteur afin de

tempérer l'influence de l'information de couleur. Cette caractéristique, en plus d'être nécessaire à la séparation de deux objets similaires (comme les porcelets de la figure 5.4b), permet de retrouver l'influence locale des indications données qui fait la force de l'algorithme de Boykov *et al.*

5.7.2 Quelques études qualitatives

Les figures 5.13, 5.14 et 5.15 montrent l'impact de l'ajout de germes sur les segmentations produites par $S\alpha F$ pour trois images.

L'analyse de ces trois figures montre qu'une première segmentation relativement proche de la segmentation de référence peut presque toujours être obtenue en positionnant les germes près de la bordure interne des objets à extraire, ce qui constitue une indication relativement simple à transmette à l'utilisateur. Cette caractéristique de $S\alpha F$ est liée à l'utilisation d'un SVM, un point près de la bordure d'un objet ayant davantage de chance de constituer un vecteur de support au sens de l'information de localisation.

Cette première segmentation comprend toutefois quelques erreurs. Afin de les corriger, il suffit d'ajouter de nouveaux germes sur les zones erronées. Les exemples présentés couvrent deux itérations de $S\alpha F$. Le fait que l'utilisateur puisse identifier d'un simple coup d'œil les zones erronées et qu'en positionnant des germes sur ces dernières, ces erreurs soient corrigées sans dégrader la segmentation des autres parties de l'image, constitue un point de fort de $S\alpha F$, à la fois parce que son comportement est facilement prévisible et parce qu'il reste intuitif pour l'utilisateur.

À l'issue de la troisième étape, la segmentation obtenue est quasiment similaire à la segmentation souhaitée. Les erreurs résiduelles sont dues à l'étape de sur-segmentation et ne peuvent pas être corrigées par l'ajout de germes.

5.8 Applications

5.8.1 Greffon pour le logiciel de manipulation d'images Gimp

Le logiciel de manipulation d'images Gimp [4] est une alternative gratuite et libre à Photoshop [5] qui offre aux utilisateurs un panel d'outils pour modifier leurs photographies. Si ces opérations peuvent rester légères (redresser l'image, réduire le bruit, augmenter le contraste), Gimp permet également de réaliser des manipulations complexes, en appliquant des traitements spécifiques à certaines parties d'une image ou en prélevant des extraits de différentes images et en les organisant à la manière d'un collage. Les outils de sélection sont donc une composante essentielle du logiciel Gimp. Actuellement ils prennent cinq formes :

- des outils réalisant des sélections à l'aide de formes géométriques simples (rectangle, ellipse, polygone) ;
- un outil sélectionnant tous les pixels d'une même couleur ;

4. http://www.gimp.org/
5. http://www.photoshop.com/

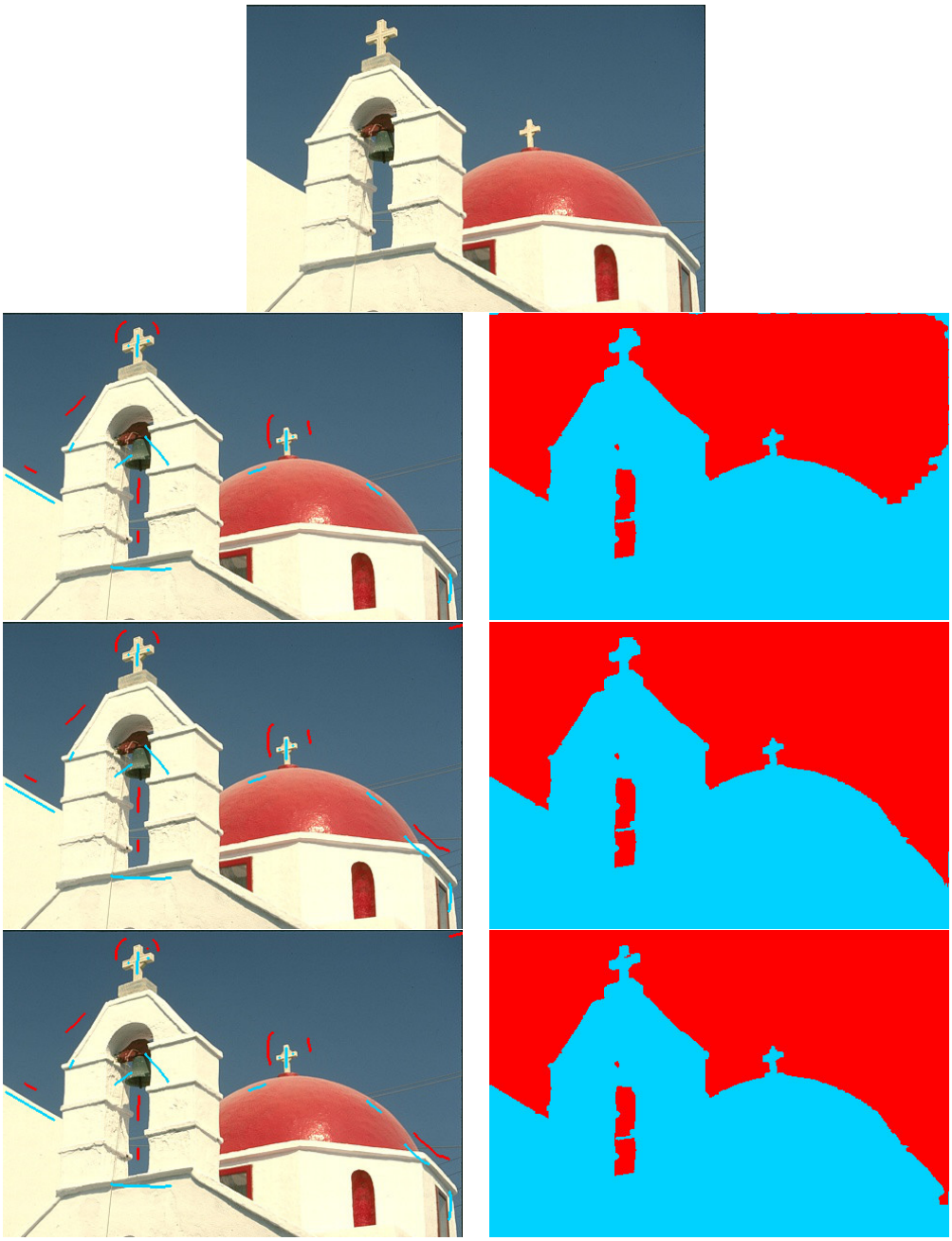


FIGURE 5.13 – Évolutions concomitantes des germes donnés et des segmentations produites par $S\alpha F$ sur un problème de binarisation issu des données de référence de McGuinness *et al.* [31].

- un outil de binarisation interactive par recherche des contours, qui correspondant à l'implémentation de l'algorithme de Mortensen *et al.* [33] ;
- un outil de binarisation interactive par recherche des régions, qui est une implémentation de l'algorithme de Friedland *et al.* [15].

L'algorithme $S\alpha F$, implémenté comme greffon pour le logiciel Gimp, vient compléter ces

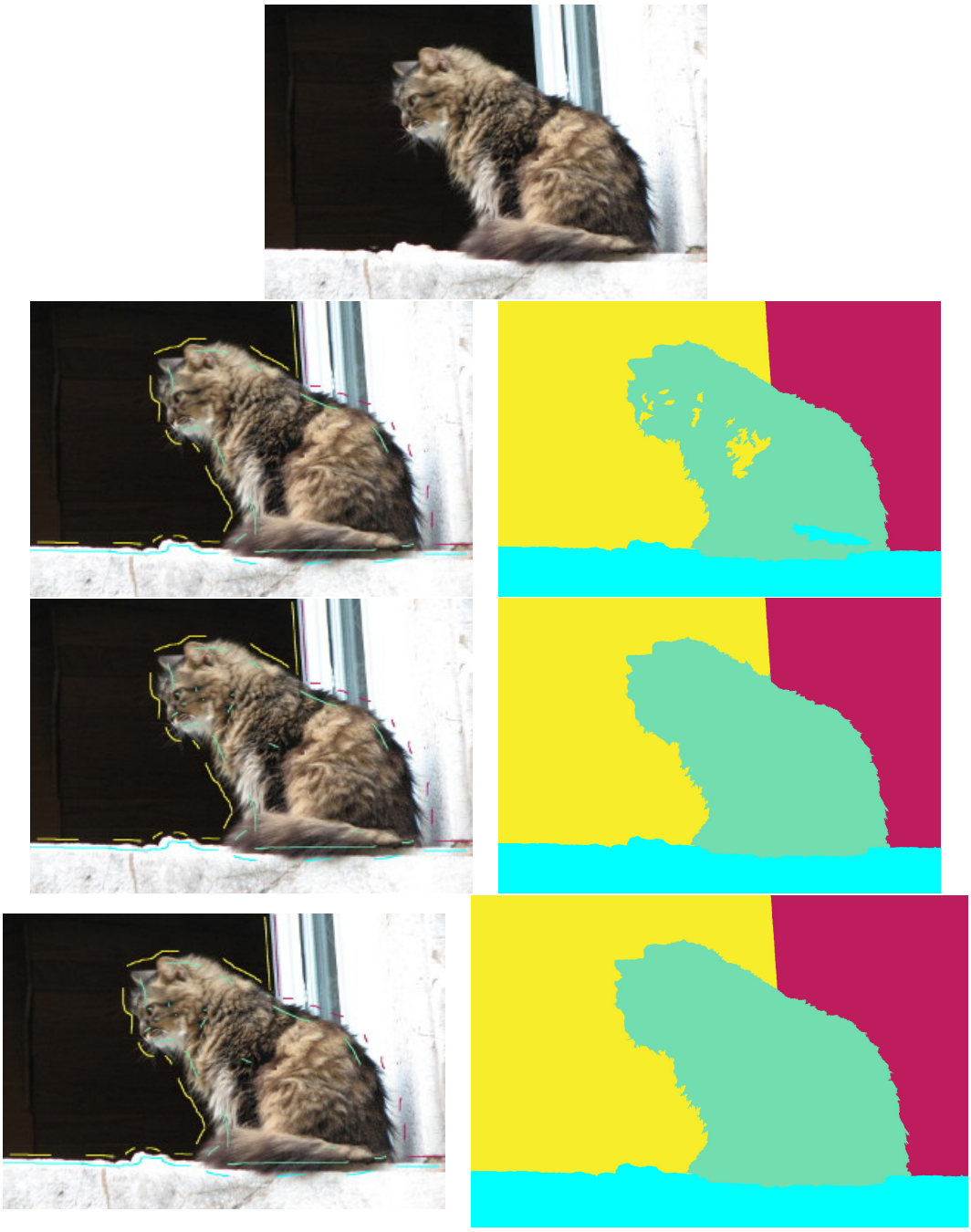


FIGURE 5.14 – Évolutions concomitantes des germes donnés et des segmentations produites par $S\alpha F$ sur un problème à 4 classes issu des données de référence $HSID'_1$.

outils. En effet, nous avons montré qu'il assure une sélection plus précise que les méthodes de Mortensen *et al.* [33] et de Friedland *et al.* [15]. Par ailleurs, alors que ces deux algorithmes se contentent de produire des binarisations, $S\alpha F$ offre la possibilité de sélectionner en une seule passe plusieurs objets.

FIGURE 5.15 – Évolutions concomitantes des germes donnés et des segmentations produites par $S\alpha F$ sur un problème à 4 classes issu des données de référence $HSID'_1$.

La figure 5.16 illustre le fonctionnement de $S\alpha F$ intégré au logiciel Gimp. Cet exemple part d'une image (figure 5.16a) souffrant d'un faible contraste et d'un peu de flou. Nous souhaitons faire intervenir trois retouches locales pour l'améliorer :

- nous voudrions colorer le ciel en rose et orangé, afin de donner un effet de coucher de soleil ;
- au niveau des bâtiments, nous aimerions renforcer le contraste ;

- nous souhaiterions augmenter le contraste du canal et le colorer avec des teintes ambrées afin de faire apparaître les reflets de l'eau.

(a) Image originale.

(b) Germes.

(c) Segmentation obtenue.

(d) Image modifiée.

FIGURE 5.16 – Utilisation de $S\alpha F$ comme greffon pour le logiciel Gimp.

La figure 5.16b montre les germes que nous avons fournis à $S\alpha F$ et la figure 5.16c la segmentation obtenue. Les germes et le résultat sont donnés sur des *calques* qui viennent se superposer à l'image originale sans en modifier les caractéristiques. En sélectionnant, par exemple, toute la zone bleue sur le calque de la segmentation et en basculant sur le calque contenant l'image originale, il est possible d'appliquer un traitement local sur tous les pixels du ciel. La figure 5.16c correspond à l'image modifiée.

5.8.2 Observatoires photographiques du paysage

Traité du Conseil de l'Europe adopté en octobre 2000, la Convention Européenne du Paysage vise à mieux prendre en compte et préserver les paysages. Elle est à l'origine de la mise en place de nombreux observatoires photographiques du paysage [6]. Ces derniers consistent en une sélection de plusieurs lieux représentatifs d'une thématique (évolution du paysage après un chantier d'autoroute, zones périurbaines, etc.) qui seront photographiés à intervalle de temps régulier (par exemple tous les mois) selon un protocole rigoureux garantissant que les images de la même série correspondront au même point de vue.

L'algorithme $S\alpha F$ a été intégré au sein d'une application distribuée, afin de faciliter la mise à jour et l'enrichissement d'un observatoire photographique. Cette application se compose d'une partie serveur (en PHP), permettant la consultation et la comparaison de photographies, et d'un logiciel mobile, développé pour le système d'exploitation Android. L'application mobile gère la re-photographie des sites choisis. En particulier l'utilisateur peut :

- localiser le site à re-photographier le plus proche ;
- disposer d'une image guide qui facilite la rephotographie ;
- créer une segmentation sémantique associée à la photographie prise en indiquant, pour chaque pixel, à quel type d'élément paysager il appartient (ciel, végétation, bâtiment, etc.).

Cette dernière étape est réalisée grâce à $S\alpha F$. La puissance de calcul et les ressources mémoire d'un smartphone étant limitées, $S\alpha F$ a fait l'objet d'une implémentation optimisée où l'étape d'initialisation est réalisée dès que la photographie est prise, pendant que l'utilisateur donne les premiers germes. Ainsi, seule la durée nécessaire à l'étape de segmentation est perçue par l'utilisateur, ce qui réduit considérablement le temps d'attente de ce dernier.

L'intérêt du dispositif, tant sur le plan géographique que sur le plan éducatif, a fait l'objet d'une publication [**?**] et d'une présentation lors des *Journées Géomatique, Enseignement et Apprentissage* qui sont déroulées à Toulouse en janvier 2017.

5.9 Conclusions et perspectives

Analyser l'apport de l'algorithme $S\alpha F$ par rapport à l'état de l'art est une tâche délicate, les conditions d'évaluation de certaines méthodes n'étant pas reproductibles. Nous avons néanmoins montré que notre méthode est compétitive par rapport à l'état de l'art.

6. Deux exemples : http://observatoiredespaysages.fr, http://www.paysages-citoyens.be

Nous nous sommes ensuite intéressés à certaines propriétés de $S\alpha F$. Nous avons montré l'importance de l'utilisation d'un terme de régularisation ainsi que les bénéfices de l'utilisation d'une méthode de sur-segmentation, lorsque des images de taille importante sont traitées. Enfin, nous avons abordé la question de l'ergonomie de $S\alpha F$ et notamment les particularités des germes à lui fournir.

La piste qui nous paraît actuellement la plus prometteuse pour améliorer $S\alpha F$ concerne l'étape de sur-segmentation. Tout d'abord, les erreurs commises par rapport aux segmentations de référence proviennent en majorité de cette étape. Les réduire augmenterait la précision des segmentations produites. Ensuite, le fait que les superpixels ne contiennent, par construction, pas d'information de texture annihile les bénéfices que nous pourrions retirer à utiliser un descripteur un peu plus complexe, qui faciliterait sans doute la phase d'apprentissage des caractéristiques visuelles de chaque classe et se traduirait par une réduction du nombre de germes nécessaires. Enfin, réduire le nombre de superpixels permettrait de gagner encore un peu plus de temps lors de l'étape de segmentation de $S\alpha F$. Le chapitre suivant constitue un premier pas dans la direction d'une nouvelle méthode de sur-segmentation prenant en compte ces pistes d'amélioration.

Chapitre 6

Sur-segmentation adaptative par fusion de superpixels

6.1 Introduction

Ce chapitre présente un nouvel algorithme de sur-segmentation. Il s'agit du fruit d'une première réflexion, amorcée dans le but de concevoir une méthode de génération des superpixels capable d'améliorer les performances de $S\alpha F$. Au sein de $S\alpha F$, l'impact de la méthode de prétraitement générant les superpixels est multiple. En particulier :

- la sur-segmentation est la seule des opérations travaillant au niveau du pixel : il s'agit donc de l'étape de $S\alpha F$ ayant la complexité algorithmique la plus élevée ;
- les erreurs commises au niveau des superpixels ne peuvent pas être corrigées par la suite : l'algorithme de sur-segmentation a donc une influence forte sur la qualité du résultat produit par $S\alpha F$;
- les superpixels étant utilisés comme primitives visuelles, ils déterminent en grande partie les caractéristiques des descripteurs employés pour différencier chacune des classes constituant la segmentation résultat.

Concrètement, qu'il s'agisse de l'adéquation de la segmentation produite avec la segmentation désirée, des temps d'exécution ou du nombre de germes demandés à l'utilisateur, tous les points sensibles de $S\alpha F$ dépendent d'une manière ou d'une autre des propriétés des superpixels générés.

6.1.1 Analyse des forces et faiblesses de SLIC

Actuellement, $S\alpha F$ utilise l'algorithme SLIC [1] proposé par Achanta *et al.* L'un des principaux attraits de SLIC tient dans sa complexité linéaire qui lui assure des temps d'exécution significativement en dessous de ceux des méthodes concurrentes.

L'algorithme SLIC souffre cependant de deux faiblesses :

- il s'adapte peu à la complexité interne de l'image ;
- il crée des superpixels homogènes uniquement au sens de la couleur.

Le premier désavantage est directement lié à la propriété d'adaptabilité décrite au chapitre 3 : le choix d'un nombre de superpixels pertinent par rapport au niveau de détail de la photographie est à la charge de l'utilisateur. En particulier, nous regrettons que SLIC ne soit pas capable de réduire le nombre de superpixels dans des zones homogènes telles que le ciel ou de l'augmenter pour s'assurer de sur-segmenter correctement même les détails les plus fins de l'image. Nous y perdons, tant en termes de temps d'exécution qu'en termes de qualité des segmentations générées par $S\alpha F$.

Le second désavantage nous limite essentiellement dans le choix du descripteur : les superpixels n'étant pas conçus pour contenir des variations de couleurs, il y a une très faible probabilité que ces derniers contiennent une information de texture suffisamment discriminante pour améliorer l'identification des classes.

En outre, le fait que SLIC ne groupe les pixels qu'en fonction de leur localisation et de leur couleur engendre parfois des erreurs de sur-segmentation. Si nous prenons par exemple l'image de la figure 6.1 et que nous cherchons à la sur-segmenter en superpixels dont l'aire moyenne est d'environ 300 pixels, nous constatons que l'algorithme se trompe de manière surprenante, en intégrant dans un même superpixel des pixels de l'herbe et des pixels du zèbre. Ces aberrations s'expliquent par le fait que certaines rayures du zèbre sont plus petites que l'aire moyenne d'un superpixel : comme les pixels blancs n'ont, au sens de la couleur, aucune raison d'être regroupés avec des pixels noirs, ils sont rattachés à un ensemble regroupant d'autres pixels blancs, proche en termes de localisation, alors même que les deux ensembles ne sont pas voisins et que leur union ne forme pas une composante connexe.

Lors de la dernière étape de l'algorithme SLIC, ces ensembles se voient fragmentés en minuscules composantes connexes. Pour former des superpixels d'une taille moyenne proche de celle attendue, l'algorithme les fusionne avec des composantes connexes voisines, sans faire de vérification sur leur similarité au sens de la couleur. Ainsi, les rayures blanches peuvent se retrouver rattachées au rayures noires (créant l'illusion de superpixels texturés) ou bien à l'herbe, formant les superpixels erronés que nous observons.

6.1.2 Démarche

Les constats précédents nous ont conduit à travailler sur un nouvel algorithme, ASARI, de l'anglais « *Adaptive Superpixel Algorithm with Rich Information* » (algorithme de sur-segmentation adaptatif avec une information riche) [**?**]. Comme l'indique son nom, la première caractéristique

FIGURE 6.1 – Exemple de superpixels (environ 500) produits par l'algorithme SLIC et comprenant des erreurs introduites lors du post-traitement. Les contours des superpixels sont tracés en bleu. Les contours de certains superpixels erronés sont tracés en rouge.

d'ASARI consiste à s'adapter à la complexité de l'image, en réduisant le nombre de superpixels dans les zones uniformes de l'image.

La seconde spécificité d'ASARI réside dans le fait que, contrairement à SLIC, il produit des superpixels homogènes à la fois au sens de la couleur et de la texture. À notre connaissance, la seule méthode de sur-segmentation à utiliser une information de texture est celle de Ren *et al.* [42] dont les temps d'exécution importants en freinent malheureusement l'utilisation. Avec ASARI, nous montrons qu'il est tout à fait possible d'avoir une méthode rapide bien que manipulant des caractéristiques de texture.

Nous avons testé ASARI, à la fois selon les protocoles de l'état de l'art [1, 49] et selon le protocole que nous proposons avec l'ensemble de données de référence HSID. L'ensemble de ces tests montre qu'ASARI constitue un apport intéressant par rapport aux autres méthodes de sur-segmentation. Néanmoins, il doit encore faire l'objet d'adaptations avant de pouvoir être intégré fructueusement à $S\alpha F$.

6.2 ASARI : un algorithme de sur-segmentation par fusion

L'algorithme ASARI que nous proposons appartient à la catégorie des méthodes de segmentation par fusion, formalisée pour la première fois en 2000, par Salembier *et al.* [45]. Ce type d'algorithme sert de base, aujourd'hui encore, à un certain nombre de travaux fructueux en analyse d'image [24, 57].

Soit une image I et un graphe $\mathcal{G}_0 = < V_0, E_0 >$ la représentant, avec :

- V_0 un ensemble de sommets ;
- E_0 les arêtes qui les relient.

Les sommets de V_0 correspondent aux pixels de l'image ou bien à de petites régions. Les arêtes relient les sommets voisins.

Un algorithme de segmentation par fusion appliqué sur le graphe initial $\mathcal{G}_0$ produit un graphe résultat $\mathcal{G}_K = < V_K, E_K >$ tel que V_K soit une partition de V_0 en composantes connexes et E_K soit un sous-ensemble de E_0.

Le passage de $\mathcal{G}_0$ à $\mathcal{G}_K$ s'effectue par la fusion des sommets de V_0 et la suppression des arêtes correspondantes. Salembier *et al.* montrent qu'il existe une famille de méthodes dites de *segmentation par fusion* assurant la production de $\mathcal{G}_K$. Ces méthodes respectent la procédure suivante :

- l'une des arêtes est sélectionnée ;
- les deux sommets reliés par cette arête sont examinés ;
- s'ils satisfont un critère dit *de fusion* :
  - l'arête est supprimée ;
  - les deux sommets sont fusionnés en un seul sommet ;
  - les caractéristiques du nouveau sommet sont calculées.

Cette procédure est répétée jusqu'à qu'il n'existe plus d'arête reliant deux sommets qui peuvent être fusionnés.

Ainsi, n'importe quel algorithme de segmentation par fusion se définit en donnant :

- un ordre de parcours, guidant la sélection des arêtes ;
- un critère de fusion, indiquant si deux sommets doivent ou pas rester distincts ;
- un modèle de région, spécifiant comment les caractéristiques d'un nouveau sommet sont produites à partir de deux sommets fusionnés.

6.2.1 Ordre de parcours

L'ordre de parcours permet de déterminer quelles arêtes du graphe seront sélectionnées afin d'être éventuellement supprimées [1]. Dans le cas d'un algorithme standard de segmentation, l'ordre de parcours est généralement induit par une simple mesure de similarité, le but étant de créer des régions homogènes. Les régions produites correspondant aux objets présents dans l'image, il n'est pas choquant qu'elles soient de tailles différentes.

L'une des particularités des algorithmes de sur-segmentation réside dans la contrainte supplémentaire de produire des régions de tailles relativement proches. L'ordre de parcours doit ainsi permettre une croissance homogène des régions en plus de garantir la sélection d'une arête ayant une probabilité raisonnable d'être supprimée.

Pour ce faire, nous associons à chaque sommet v_i un compteur Δ_i correspondant au nombre de fois où le sommet v_i a été sélectionné. À chaque itération, l'algorithme choisit la région ayant la valeur Δ_i la plus faible. En cas d'égalité entre plusieurs régions, l'une d'entre elles est sélectionnée de manière arbitraire. Parmi l'ensemble des arêtes reliant v_i à ses voisins, l'algorithme sélectionne celle qui maximise une mesure de similarité. Que v_i soit fusionné ou pas, la valeur de Δ_i est incrémentée.

1. Si l'algorithme autorise la fusion des deux sommets qu'elles relient.

6.2.2 Mesure de similarité

Afin de produire des superpixels qui ne soient pas uniquement homogènes au sens de la couleur, il est crucial qu'ASARI intègre une mesure de similarité au sens de la couleur et de la texture. Toutefois l'intégration d'une information visuelle plus riche ne doit pas faire perdre de vue la contrainte de rapidité qui pèse lourdement sur tous les algorithmes de sur-segmentation et se révèle particulièrement critique lorsque ces derniers sont intégrés à une méthode telle que $S\alpha F$. C'est donc en grande partie pour leur rapidité que nous avons choisi d'utiliser les *motifs ternaires locaux* (LTP, de l'anglais « *Local Ternary Pattern* ») [50].

Descripteur de texture

Soient p_i un pixel et $I(p_i)$ son niveau de gris. La fonction seuil $\mathcal{F}_{LTP}$ est définie par :

$$\mathcal{F}_{LTP}(p_i, p_j) = \begin{cases} 1, \text{ si } I(p_i) - I(p_j) > \omega_{LTP} \\ 0 \text{ sinon.} \end{cases} \tag{6.1}$$

Le paramètre ω_{LTP} ajuste le comportement de la fonction.

Trouver le motif ternaire local de p_i consiste à lui attribuer deux identifiants, LTP_N et LTP_P, représentant les variations du niveau de gris dans son voisinage. L'identifiant LTP_N se construit à partir des voisins de p_i ayant un niveau de gris significativement inférieur au sien, tandis que LTP_P se calcule à partir de ceux ayant un niveau de gris significativement supérieur.

Soit $\{I(p_i^0), \cdots, I(p_i^7)\}$ les niveaux de gris 8 voisins de p_i. L'identifiant LTP_N est obtenu grâce à la fonction suivante :

$$LTP_N(p_i) = \sum_{n=0}^{7} 2^n \mathcal{F}_{LTP}(I(p_i), I(p_i^n)). \tag{6.2}$$

Ainsi, pour que la fonction $\mathcal{F}_{LTP}$ soit égale à 1, il faut que $I(p_i^n)$ soit strictement inférieur à $I(p_i)$ et que $|I(p_i) - I(p_i^n)| > \omega_{LTP}$.

De même, l'identifiant LTP_P est obtenu grâce à la fonction suivante :

$$LTP_P(p_i) = \sum_{n=0}^{7} 2^n \mathcal{F}_{LTP}(I(p_i^n), I(p_n)). \tag{6.3}$$

Ici, pour que la fonction $\mathcal{F}_{LTP}$ soit égale à 1, il faut que $I(p_i^n)$ soit strictement supérieur à $I(p_i)$ et que $|I(p_i) - I(p_i^n)| > \omega_{LTP}$.

Détection des zones homogènes

Les deux fonctions LTP_N et LTP_P sont des entiers compris dans l'intervalle $[0, 255]$. Ces entiers peuvent être représentés sous la forme de nombres binaires à huit bits. Le premier bit correspond au résultat de la comparaison entre le pixel p_i et son premier voisin, le deuxième au résultat de la comparaison avec le deuxième voisin, etc.

Trois cas de figure peuvent se présenter :

1. le niveau de gris de p_i est nettement inférieure à celui de son k^{e} voisin : dans ce cas, le k^{e} bit de l'identifiant LTP_N est égal à 0 et celui de l'identifiant LTP_P est égal à 1 ;
2. la valeur du niveau de gris de p_i est nettement supérieure à celle de son k^{e} voisin : dans ce cas, le k^{e} bit de l'identifiant LTP_N est égal à 1 et celui de l'identifiant LTP_P est égal à 0 ;
3. les deux niveaux de gris sont similaires, c'est-à-dire que $|I(p_i) - I(p_i^n)| \leq \omega_{LTP}$: dans ce cas, le k^{e} bit de l'identifiant LTP_N et celui de l'identifiant LTP_P sont égaux à 0.

En particulier, si tous les voisins d'un pixel ont des niveaux de gris similaires au sien, les identifiants LTP_P et LTP_N valent tous les deux 0. Soit

$$v_i^0 = \{p_j \in v_i\},\ LTP_N(p_j) = LTP_P(p_j) = 0, \tag{6.4}$$

l'ensemble des pixels de la région v_i ayant tout ses voisins avec des niveaux de gris similaires.

Il est aisé d'obtenir une estimation de la probabilité que la région v_i soit non texturée en calculant :

$$P_{\neg T}(v_i) = \frac{|v_i^0|}{|v_i|}, \tag{6.5}$$

$|A|$ étant la cardinalité de l'ensemble A.

Définition de la mesure de similarité

Nous postulons que la mesure de similarité doit être faible lorsqu'elle compare une région non texturée avec une région texturée. Nous nous donnons les fonctions suivantes :

- $\mathcal{F}_c(i,j)$, une mesure de similarité au sens de la couleur entre les régions v_i et v_j ;
- $\mathcal{F}_t(i,j)$, une mesure de similarité au sens de la texture entre les régions v_i et v_j ;
- $\mathcal{F}_h(i)$, une fonction binaire, valant 1 si la région v_i est non texturée, 0 sinon.

À partir de la probabilité $P_{\neg T}(v_i)$ dont l'estimation est décrite précédemment, nous calculons :

$$\mathcal{F}_h(i) = \begin{cases} 1 & \text{si } P_{\neg T}(v_i) > \omega_T \\ 0 & \text{sinon.} \end{cases} \tag{6.6}$$

Le paramètre ω_T correspond au seuil à partir duquel une région est considérée comme texturée. Nous pouvons alors définir la fonction de similarité $\mathcal{F}_S$ comme :

$$\mathcal{F}_S(i,j) = \begin{cases} \mathcal{F}_c(i,j), & \text{si } \mathcal{F}_h(i) + \mathcal{F}_h(j) = 2 \\ \dfrac{\mathcal{F}_t(i,j) + \mathcal{F}_c(i,j)}{2}, & \text{si } \mathcal{F}_h(i) + \mathcal{F}_h(j) = 0 \\ 0 & \text{sinon.} \end{cases} \tag{6.7}$$

Similarité au sens de la couleur

Soit v_i une région et c_i le vecteur de dimension 3 contenant sa couleur. Le vecteur c_i correspondant aux valeurs normalisées obtenues pour chacun des canaux d'un espace colorimétrique fixé. Une mesure de la similarité au sens de la couleur entre v_i et v_j est donnée par :

$$\mathcal{F}_c(i,j) = \exp(-\frac{1}{N_c}||c_i - c_j||) \tag{6.8}$$

avec N_c le nombre de composantes pour la couleur (généralement 3) et $||.||$ la norme euclidienne.

Nous avons testé les espaces RGB et Lab. Ces deux espaces donnent des résultats similaires. L'espace Lab nécessitant des calculs intermédiaires pour convertir la couleur des pixels, nous avons privilégié l'utilisation de l'espace RGB durant l'évaluation d'ASARI.

Similarité au sens de la texture

Soit t_i la concaténation des histogrammes normalisés des deux identifiants LTP_N et LTP_P, pour les pixels rattachés au sommet v_i. La mesure de similarité au sens de la texture utilisée par ASARI fait intervenir la distance du χ^2 entre les deux histogrammes :

$$\begin{aligned}\mathcal{F}_t(i,j) &= \exp(-\chi^2(t_i,t_j)) \\ &= \exp\left(-\frac{1}{N_t}\sum_{k=0}^{N_t}\frac{\left(t_i^k - t_j^k\right)^2}{t_i^k + t_j^k}\right).\end{aligned} \tag{6.9}$$

La valeur N_t correspond au nombre de classes ayant un nombre d'occurrences strictement positif, dans l'histogramme t_i ou dans l'histogramme t_j.

Temps d'exécution

Distinguer les régions non texturées des régions texturées comporte deux avantages. D'une part, nous évitons de décrire uniquement par sa couleur moyenne une région dont la texture peut comporter des teintes très différentes. D'autre part, nous réduisons considérablement le temps d'exécution d'ASARI, la complexité de $\mathcal{F}_c$ étant très inférieure à celle de $\mathcal{F}_t$. En effet, dans le premier cas, seules les trois composantes de couleur sont comparées, tandis que, dans le second cas, il faut prendre en compte l'intégralité des éléments des histogrammes de LTP, soit 2×256 comparaisons.

6.2.3 Critère de fusion

Dans le cas d'une méthode de segmentation, le critère de fusion se contente de garantir la cohérence de la région créée lorsque deux sommets sont fusionnés. Une méthode de sur-segmentation nécessite également de vérifier que la surface de la région obtenue ne soit pas disproportionnée par rapport à celles des autres régions. Ce contrôle s'ajoute à celui de l'ordre de parcours.

Le critère de fusion d'ASARI est une fonction booléenne qui se décompose de la manière suivante :

$$\mathcal{F}_F(i,j) = \mathcal{F}_{F,S}(i,j) \wedge \mathcal{F}_{F,R}(i,j), \tag{6.10}$$

avec $\mathcal{F}_{F,S}$ un critère prenant en compte la similarité et $\mathcal{F}_{F,R}$ un critère dépendant de l'aire de la nouvelle région.

Critère de similarité

La fonction booléenne $\mathcal{F}_{F,S}$ vérifie que la mesure de similarité entre les deux régions se situe au dessus d'un seuil :

$$\mathcal{F}_{F,S}(i,j) = \mathcal{F}_S(i,j) > \omega_S. \tag{6.11}$$

Critère de régularité

La fonction booléenne $\mathcal{F}_{F,R}$ s'assure que la taille de la nouvelle région ne sera pas disproportionnée par rapport à celles des autres régions.

Soient $|v_i|$ le nombre de pixels appartenant à la région v_i et N_V le nombre de régions. La vérification effectuée est la suivante :

$$\mathcal{F}_{F,R}(i,j) = \left((|v_i| + |v_j|) < \frac{\omega_R}{N_V} \sum_{k=0}^{N_V - 1} |v_k| \right). \tag{6.12}$$

Le paramètre ω_R correspond à un *a priori* sur la rapidité de croissance des régions : plus ce paramètre est élevé, plus la différence entre la taille de la nouvelle région et la taille des régions existantes est importante. Ainsi, si ω_R vaut 2, l'aire de la nouvelle région ne devra pas dépasser le double de l'aire moyenne des régions existantes.

6.3 Liens avec un problème d'optimisation

6.3.1 Reformulation sous la forme d'un problème d'optimisation

Si nous convertissons le résultat booléen de la fonctions $\mathcal{F}_F$ en un résultat binaire égal à 0 lorsque le prédicat n'est pas respecté et à 1 lorsque l'arête doit être supprimée, nous constatons que l'algorithme ASARI revient à rechercher le graphe $\mathcal{G}_K^*$ permettant d'atteindre le minimum de la fonction :

$$\mathcal{F}_{ASARI}(\mathcal{G}_K) = \sum_{e_{i,j} \in E_K} \mathcal{F}_F(i,j). \tag{6.13}$$

La fonction $\mathcal{F}_F$ ne pouvant valoir que 0 ou 1, ce minimum vaut 0, lorsque toutes les fusions ont été réalisées. Comme les arêtes sont sélectionnées en fonction de la valeur du compteur Δ_i, chaque sommet du graphe initial $\mathcal{G}_0$ est parcouru au moins une fois et l'une des arêtes le reliant à l'un de ses voisins est sélectionnée. Soit N_{V_0} le nombre initial de sommets : dans le pire des

cas, une seule fusion est réalisée et un parcours est relancé sur les $N_{V_0} - 1$ sommets restants. Toujours dans le pire des cas, le processus se poursuit jusqu'à l'obtention d'un unique sommet. Dans ce cas de figure, la complexité d'ASARI vaut :

$$O_{ASARI} = N_{V_0} + (N_{V_0} - 1) + \cdots + 1 \tag{6.14}$$

$$= \frac{N_{V_0}(N_{V_0} + 1)}{2} \tag{6.15}$$

$$= \frac{{N_{V_0}}^2 + N_{V_0}}{2}. \tag{6.16}$$

Même si ce cas de figure ne se produit en pratique jamais (il correspond à une fusion de l'ensemble des pixels en un seule région), il a le mérite de montrer qu'une implémentation naïve d'ASARI dépasse largement la complexité algorithmique tolérable pour un algorithme de sur-segmentation.

Pour obtenir des temps d'exécution raisonnables et permettre à ASARI de passer à l'échelle, nous nous sommes contentés de rechercher une approximation de $\mathcal{G}_K^*$. En nous donnant un critère d'arrêt approprié et en réalisant une première étape d'initialisation permettant d'obtenir un graphe $\mathcal{G}_1$ tel que :

$$\mathcal{F}_{ASARI}(\mathcal{G}_1) > \mathcal{F}_{ASARI}(\mathcal{G}_0), \tag{6.17}$$

nous permettons à l'algorithme ASARI de produire rapidement une segmentation correcte.

6.3.2 Critère d'arrêt

De manière similaire à un algorithme de segmentation standard, ASARI s'arrête lorsque plus aucune arête ne peut être supprimée, c'est-à-dire lorsque plus aucune paire de sommets ne satisfait le critère de fusion $\mathcal{F}_F$. Au sein de ce critère, la fonction $\mathcal{F}_{F,S}$ permet à l'algorithme de s'adapter à complexité de l'image : lorsque celle-ci contient de nombreux petits détails, elle arrête la fusion des régions, tandis que dans les zones homogènes, elle l'autorise, conduisant à de plus grands superpixels. En parallèle, la fonction $\mathcal{F}_{F,R}$ arrête la fusion des régions avant que des disparités de taille trop importantes ne soient atteintes.

Pour les raisons d'efficacité évoquées précédemment, il est également souhaitable d'arrêter l'algorithme lorsque tous les sommets ont été sélectionnés au moins ω_{Itr} fois et d'obtenir, non pas un résultat optimal mais une approximation. L'évaluation d'ASARI conduite dans la section 6.5 montre que $\omega_{Itr} = 10$ permet d'obtenir des résultats satisfaisants.

Enfin, si le contexte applicatif nécessite qu'un nombre minimal de superpixels soit imposé, cette contrainte peut également être intégrée dans le critère de terminaison.

6.3.3 Initialisation

Afin de réduire les calculs nécessaires à ASARI, nous avons choisi de regrouper les pixels de l'image sur-segmentée en petits ensembles connexes.

Le fait que les sommets de V_0 soient des petites régions facilite également l'utilisation du descripteur de texture. En effet, ce dernier correspond à l'histogramme des deux identifiants LTP_P et LTP_N au sein de chaque région. Lorsque celle-ci ne contient qu'un très petit nombre de pixels (ou pire, un unique pixel), cet histogramme n'est pas exploitable. En nous assurant que les sommets de V_0 contiennent au moins une cinquantaine de pixels, nous évitons de manipuler des descripteurs de texture qui n'auraient pas de sens.

Les régions donnant les sommets de V_0 sont obtenues avec l'algorithme SLIC [1], paramétré pour produire de tout petits superpixels. L'algorithme SLIC garantit une complexité linéaire par rapport au nombre de pixels, ce qui assure que l'obtention de V_0 n'augmente pas de manière drastique la complexité générale d'ASARI. De plus, le fait que SLIC soit paramétré pour produire de minuscules régions réduit considérablement le nombre d'itérations internes à SLIC qui sont nécessaires à l'obtention d'une sur-segmentation, tout en garantissant un faible taux d'erreur de sous-segmentation. Durant l'ensemble de nos tests, trois itérations (au lieu d'une dizaine, lors d'une utilisation normale de SLIC) se sont révélées suffisantes.

L'inconvénient de cette approche est qu'elle introduit deux paramètres, ω_{V_0} et ω_C qui correspondent respectivement à la taille moyenne souhaitée pour les régions de V_0 et au paramètre de compacité de l'algorithme SLIC.

6.4 Détermination des paramètres

L'algorithme ASARI comporte 6 paramètres :

- ω_{LTP} : le seuil permettant de décider si deux niveaux de gris sont similaires ;
- ω_T : un seuil au dessus duquel la probabilité d'une région d'être homogène est jugée suffisamment haute pour que cette dernière soit considérée comme non texturée ;
- ω_{V_0} : l'aire moyenne souhaitée pour les régions de départ ;
- ω_C : le paramètre de compacité de l'algorithme SLIC ;
- ω_S : le paramètre du critère de similarité ;
- ω_R : le paramètre du critère de régularité.

Comme ces paramètres ont une influence sur des aspects différents (nombre de superpixels, régularité, adhérence aux contours, etc.), il n'est pas aisé de les optimiser en une seule fois, de manière à maximiser un score de réussite. Nous avons choisi de procéder en trois étapes successives : optimisation de la détection des zones non texturées, paramétrage de SLIC et détermination des paramètres du critère de fusion.

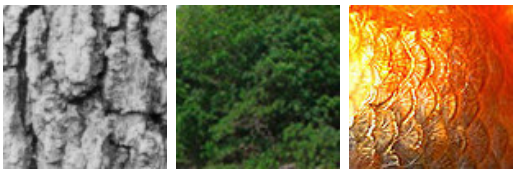
L'évaluation menée dans la section 6.5 montre qu'une fois ces paramètres fixés, ils permettent d'obtenir de bons résultats sur des photographies couleur ou noir et blanc, prises à l'extérieur ou à l'intérieur, de tailles et de complexités variées. Les étapes d'apprentissage des paramètres détaillées dans cette section n'ont donc pas besoin d'être répétées.

6.4.1 Détection des zones non texturées

Comme expliqué dans la section 6.2, la détection des zones non texturées est effectuée via les LTP, tout d'abord en détectant les pixels ayant un voisinage non texturé, ensuite en repérant les régions composées en majorité de tels pixels. Deux paramètres doivent donc être fixés : $\omega_{LTP} \in [0, 255]$, le seuil à partir duquel les niveaux de gris de deux pixels sont considérés comme similaires et $\omega_T \in [0, 1]$ la proportion de pixels avec un voisinage non texturé, à partir de laquelle une région est considérée comme non texturée.

Nous avons constitué un ensemble de 100 images de 100×100 pixels, la moitié correspondant à des zones non texturées, le reste correspondant à des zones texturées. Ces images sont des patchs extraits d'images de la base de données de Wikimedia Commons [2] ainsi que de la base de données de Brodatz [3]. La figure 6.2 montre quelques exemples d'images utilisées durant les tests.

(a) Exemples d'images correspondant à des zones non texturées.



(b) Exemples d'images correspondant à des zones texturées

FIGURE 6.2 – Exemples d'images utilisées pour le paramétrage de la détection des zones non texturées.

À partir de ce jeu de données, nous nous sommes ensuite ramenés à un problème de classification à deux classes :

- *image texturée* ;
- *image non texturée.*

Soient :

- R_t l'ensemble des images texturées ;
- R_h l'ensemble des images non texturées ;
- C_t l'ensemble des images classées comme texturées ;
- C_h l'ensemble des images classées comme non texturées.

2. https://commons.wikimedia.org/wiki/Accueil
3. http://sipi.usc.edu/database/database.php?volume=textures

Nous avons cherché le couple (ω_{LTP}, ω_T) permettant de maximiser le score de précision :

$$\mathcal{F}_P = \frac{|R_t \cap C_t|}{|C_t|} + \frac{|R_h \cap C_h|}{|C_h|}. \tag{6.18}$$

Le paramètre ω_{LTP} étant un nombre entier dans un intervalle borné, nous avons testé toutes les valeurs possibles. Pour ω_T, nous avons choisi un pas de 0,01. Les valeurs retenues sont de 12 pour ω_{LTP} et de 0,85 pour ω_T.

6.4.2 Paramètres de SLIC

Les résultats obtenus par la méthode SLIC [1, 49] montrent qu'une valeur de 10 pour le paramètre de compacité assure une forme régulière des régions, proche du rectangle, tout en conservant une bonne adhérence aux contours. Nous avons donc conservé cette valeur pour ω_C.

Le paramètre ω_{V_0} fixe la taille des régions de départ. Il est donc nécessaire qu'il soit inférieur à l'aire en pixels du plus petit objet d'intérêt. Il doit également permettre de produire des régions suffisamment grandes pour que le calcul de la probabilité qu'elles soient texturées ait du sens. Nous recommandons de ne pas fixer la valeur de ce paramètre en dessous de 50 pixels. Si aucun *a priori* n'est émis sur la taille du plus petit objet d'intérêt, une valeur entre 50 et 150 pixels permet d'obtenir des résultats satisfaisants pour une grande variété d'images.

6.4.3 Paramètres du critère de fusion

Le paramètre ω_S correspond à la similarité minimale requise pour fusionner deux régions. Nous avons essayé de déterminer automatiquement ce seuil à partir des distances entre les superpixels. Il est par exemple possible de supposer que les sommets de V_0 peuvent, pour la majorité, être fusionnés avec le sommet voisin le plus proche (au sens d'un critère de similarité), d'analyser la distribution de la mesure de similarité entre chaque sommet et son voisin le plus proche, en cherchant les modes de l'histogramme.

Malheureusement, cette analyse statistique se révèle coûteuse, allant jusqu'à multiplier par trois les temps de calcul nécessaires à l'étape d'initialisation. Bien que n'ayant pas abouti, cette piste s'est néanmoins révélée fructueuse, révélant que les seuils obtenus de manière automatique varient peu. L'analyse de ces derniers nous a conduit à fixer $\omega_S = 0{,}95$.

Le paramètre ω_R permet d'introduire un *a priori* sur la croissance des régions. Dans le cas idéal où les superpixels pourraient conserver la structure en grille des pixels, chaque région serait fusionnée avec 3 de ses voisines, afin de doubler sa largeur et sa hauteur. Dans ce cas de figure l'aire est multipliée par 4, ce qui nous a conduits à fixer ω_R à 4.

6.5 Protocole d'évaluation

Lors de l'évaluation de l'algorithme ASARI, nous avons cherché à mesurer sa pertinence vis-à-vis des propriétés désirées pour une sur-segmentation, telles qu'elles ont été définies au chapitre 3 :

- la propriété 1 (validité) imposant qu'une sur-segmentation soit considérée comme valide si elle constitue une partition de l'image en sous-ensembles connexes ;
- la propriété 2 (adhérence aux contours) indiquant qu'un superpixel ne doit pas recouvrir différents objets de l'image ;
- la propriété 3 (concision) énonçant qu'un algorithme de sur-segmentation doit produire aussi peu de superpixels que possible ;
- la propriété 4 (simplicité) stipulant que le nombre de voisins d'un superpixel doit être aussi petit que possible ;
- la propriété 5 (rapidité) recommandant qu'un algorithme de sur-segmentation doit avoir un temps d'exécution aussi faible que possible.

Afin de réaliser une évaluation aussi complète que possible, nous avons utilisé d'une part les conditions expérimentales fixées par Stutz *et al.* [49] et, d'autre part, notre propre protocole de test, incluant les données de référence HSID [**?**].

6.5.1 Rappel du protocole expérimental de Stutz *et al.*

Les données utilisées par Stutz *et al.* [49] rassemblent :

- les 100 images de l'ensemble de données de référence BSD [4], mise à disposition par Martin *et al.* [30] ;
- les 400 images de l'ensemble de données de référence de l'université de New York, NYU [5], mis à disposition par Silberman *et al.* [47].

Pour chaque méthode, Stutz *et al.* calculent :

- le temps d'exécution moyen ;
- le taux d'erreur de sous-segmentation, $\mathcal{F}_{ES}$;
- le score d'adhérence aux contours, $\mathcal{F}_{RC}$.

Soit R une segmentation de référence et $\mathbb{S}$ une sur-segmentation. Nous rappelons que le calcul du taux d'erreur de sous-segmentation repose sur la recherche, pour chaque région r_i présente dans R, de l'ensemble des superpixels nécessaires pour la recouvrir et sur le nombre de pixels de cet ensemble qui débordent de la région. Il est défini par :

$$\mathcal{F}_{ES}(\mathbb{S}, R) = \frac{1}{N_I} \sum_{r_i \in R} \sum_{\mathbf{s}_j \cap r_i \neq \emptyset} \min(|\mathbf{s}_j \cap r_i|, |\mathbf{s}_j - r_i|) \tag{6.19}$$

avec N_I le nombre de pixels de l'image.

Le taux de rappel sur les contours permet de vérifier que les contours des régions présentes dans R se retrouvent dans $\mathbb{S}$. Soit C_R l'ensemble des points de contour dans une segmentation de référence et $C_{\mathbb{S}}$ l'ensemble des points de contours dans la sur-segmentation :

$$\mathcal{F}_{RC}(\mathbb{S}, R) = \frac{|C_{\mathbb{S}} \cap C_R|}{|C_R|}. \tag{6.20}$$

4. https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
5. http://cs.nyu.edu

Stutz *et al.* considèrent comme valides tous les pixels à une distance de 0,007 5 $\times$ *diag*, *diag* correspondant à la longueur de la diagonale de l'image. Le score obtenu est dans l'intervale $[0, 1]$, un score de 1 indiquant que tous les contours de R se retrouvent dans $\mathbb{S}$.

6.5.2 Rappel du protocole expérimental associé à HSID

HSID a été créé à partir de 100 images issues de Wikimedia Commons [6]. À chaque image, une segmentation de référence est associée.

Soit R la segmentation de référence pour une image donnée et $\mathbb{S}$ la sur-segmentation obtenue pour cette image. R est une partition en N_R composantes connexes $(r_1, \cdots, r_{N_R})$ et $\mathbb{S}$ est une partition en $N_{\mathbb{S}}$ composantes connexes $(\mathbf{s}_1, \cdots, \mathbf{s}_{N_{\mathbb{S}}})$. Soit C_R l'ensemble des points de contour pour R et $C_{\mathbb{S}}$ l'ensemble des points de contour pour $\mathbb{S}$. L'ensemble flou $C^*_{R\cap\mathbb{S}}$ est défini à partir de C_R, avec la fonction d'appartenance :

$$f_{R\cap\mathbb{S}}(p_i) = \exp(-\frac{d(p_i - p'_i)^2}{2\sigma^2}) \tag{6.21}$$

où :

- $d(p_i - p_j)$ correspond à la distance entre p_i et p_j (par exemple la distance euclidienne) ;
- σ est un paramètre pondérant l'influence de la distance entre p_i et p_j, que nous avons fixé à 4 ;
- $p'_i = \underset{p_j \in C_{\mathbb{S}}}{\arg\min}(d(p_i - p_j))$.

L'adhérence aux contours des superpixels est évaluée à l'aide de la mesure floue d'adhérence aux contours :

$$\mathcal{F}_{FRC}(\mathbb{S}, R) = \frac{1}{|C_R|} \sum_{p_i \in C_R} f_{R\cap\mathbb{S}}(p_i). \tag{6.22}$$

6.5.3 Méthodes compétitrices

Sur HSID, nous avons testé ASARI ainsi que les méthodes :

- de Vedaldi *et al.* [52] ;
- de Felzenszwalb *et al.* [13] ;
- de Liu *et al.* [26] ;
- d'Achanta *et al.* (SLIC) [1] ;
- de Rubio *et al.* [44] ;
- de Conrad *et al.* [8] ;
- de Machairas *et al.* [29].

6. https://commons.wikimedia.org/wiki/Accueil

Sur les données de Stutz *et al.* [49] (qui rassemblent à la fois des images de la base de données de Berkeley [30] et de celle de NYU [47]) nous avons uniquement testé l'algorithme ASARI et nous avons interprété les scores obtenus à l'aune de ceux présentés par Stutz *et al.* [49] pour les méthodes :

- de Vedaldi *et al.* [52] ;
- de Felzenszwalb *et al.* [13] ;
- de Liu *et al.* [26] ;
- d'Achanta *et al.* (SLIC) [1] ;
- de Conrad *et al.* [8].

Les méthodes de Machairas *et al.* [29] et de Rubio *et al.* [44] n'ont pas été évaluées par Stutz *et al.* [49], leurs publications étant concomitantes.

6.6 Analyse des résultats

6.6.1 Respect de la propriété de validité

Avant toute évaluation qualitative ou quantitative, il est fondamental de vérifier qu'ASARI respecte la propriété 1, qui implique la validité ou non de la sur-segmentation produite.

Comme l'algorithme SLIC [1] respecte cette propriété, les sommets de $\mathcal{G}_1$ correspondent à des régions formant une partition en composantes connexes de l'image. Par ailleurs, ASARI ne fusionne deux sommets qu'à la condition qu'ils soient voisins. Le résultat de cette fusion constitue donc bien une composante connexe. Enfin, un pixel est attribué à un et un seul sommet, garantissant que le résultat demeure une partition de l'image. La propriété 1 est donc respectée.

6.6.2 Respect de la propriété de simplicité

Le nombre moyen de voisins pour un sommet, sur l'ensemble des images des trois ensembles de données (BSD, NYU et HSID), est de 6, ce qui correspond exactement au score des autres méthodes.

6.6.3 Adaptabilité

Protocole de Stutz *et al.*

Les tableaux 6.1 et 6.2 comparent les résultats obtenus par ASARI avec les scores atteints par les méthodes évaluées par Stutz *et al.* À part pour ASARI, les scores rapportés sont ceux de l'étude de Stutz *et al.* issus de l'article [49].

L'adhérence aux contours est donnés par les mesures $\mathcal{F}_{ES}$ (équation 6.19) et $\mathcal{F}_{RC}$ (équation 6.20). Les performances atteintes sont mises en regard du nombre de superpixels $N_{\mathbb{S}}$. Dans le cas d'ASARI, ce nombre n'est pas connu à l'avance : il est le résultat du nombre de fusions pratiquées par l'algorithme. *A contrario*, les méthodes évaluées par Stutz *et al.* ont été paramétrées pour produire environ 1500 superpixels pour les images de NYU et 1000 pour celles de BSD.

TABLEAU 6.1 – Scores obtenus lors de l'évaluation de Stutz *et al.* [49] avec l'ensemble de données de référence NYU. Pour chacune des méthodes, le taux d'erreur de sous-segmentation moyen ($\mathcal{F}_{ES}$), le taux de rappel sur les contours moyen ($\mathcal{F}_{RC}$) et le nombre moyen de superpixels ($N_{\mathbb{S}}$) sont donnés.

Méthodes	$\mathcal{F}_{ES}$	$\mathcal{F}_{RC}$	$N_{\mathbb{S}}$
Vedaldi *et al.*	0,07	0,99	$\simeq 1500$
Felzenszwalb *et al.*	0,07	0,99	$\simeq 1500$
Liu *et al.*	0,08	0,99	$\simeq 1500$
Achanta *et al.*	0,09	0,99	$\simeq 1500$
Conrad *et al.*	0,09	1	$\simeq 1500$
ASARI	0,1	0,99	1897

Sur la base de données NYU, le taux d'erreur de sous-segmentation égal à 0,1 d'ASARI est moins bon que celui des méthodes de référence. Son taux de rappel sur les contours, atteignant 0,99, est équivalent aux autres. Le nombre moyen de superpixels par image est par contre sensiblement supérieur, avec près de 400 superpixels supplémentaires.

L'analyse visuelle des résultats obtenus montre qu'ASARI rencontre des difficultés dans les zones avec un faible contraste local au niveau des bordures des différents objets. Pour ces parties de l'image, la fusion entre régions appartenant à des objets différents conduit à des erreurs de sous-segmentation plus nombreuses que pour les méthodes de référence. Il s'agit là d'une limite liée aux caractéristiques d'ASARI qui, en contrepartie, lui confèrent une bonne adaptabilité.

TABLEAU 6.2 – Scores obtenus lors de l'évaluation de Stutz *et al.* [49] avec l'ensemble de données de référence BSD. Pour chacune des méthodes, le taux d'erreur de sous-segmentation moyen ($\mathcal{F}_{ES}$), le taux de rappel sur les contours moyen ($\mathcal{F}_{RC}$) et le nombre moyen de superpixels ($N_{\mathbb{S}}$) sont donnés.

Méthodes	$\mathcal{F}_{ES}$	$\mathcal{F}_{RC}$	$N_{\mathbb{S}}$
Vedaldi *et al.*	0,03	1	$\simeq 1000$
Felzenszwalb *et al.*	0,03	0,99	$\simeq 1000$
Liu *et al.*	0,04	0,99	$\simeq 1000$
Achanta *et al.*	0,04	0,99	$\simeq 1000$
Conrad *et al.*	0,04	1	$\simeq 1000$
ASARI	0,04	0,99	899

Sur la base de données BSD, ASARI obtient un taux d'erreur de sous-segmentation de 0,04 et un score d'adhérence aux contours de 0,99. Ces performances sont donc équivalentes à celles des méthodes de l'état de l'art. De plus, le nombre moyen de superpixels par image est inférieur à celui des autres méthodes, avec en moyenne 100 superpixels de moins. Enfin, soulignons que certaines images sont sur-segmentées en seulement 500 superpixels tout en donnant des scores semblables.

Les segmentations présentées sur la figure 6.3 illustrent la capacité d'ASARI à s'adapter à la complexité interne d'une image, au travers de deux exemples tirés des deux ensembles de données

de référence. Dans les régions homogènes (le ciel pour la sur-segmentation sur la figure 6.3a, la porte pour celle de la figure 6.3b, de grands superpixels sont produits. Dans les zones de l'image comprenant de nombreux détails (la bibliothèque sur la figure 6.3b) ou de petits éléments (les pattes de l'oiseau sur la figure 6.3a) les superpixels deviennent beaucoup plus petits et nombreux.

(a) Image 42049 de BSD. Nombre de superpixels : 472. Taux d'erreur de sur-segmentation : 0,03. Taux de rappel sur les contours : 0,99.

(b) Image 00001315 de NYU. Nombre de superpixels : 905. Taux d'erreur de sur-segmentation : 0,07. Taux de rappel sur les contours : 1.

FIGURE 6.3 – Exemples de résultats obtenus par ASARI sur les bases de données BSD et NYU.

Résultats obtenus avec HSID

Les résultats obtenus par ASARI sur la base de données HSID sont donnés dans le tableau 6.3. Avec 1528 superpixels en moyenne, ASARI obtient le meilleur score d'adhérence aux contours. L'écart est significatif par rapport aux méthodes :

- de Vedaldi *et al.*, +4% avec 1100 superpixels en moins ;
- de Rubio *et al.*, +15% avec 300 superpixels en moins ;
- de Conrad *et al.*, +24% avec 500 superpixels en moins ;
- de Machairas *et al.*, +10% avec 500 superpixels en moins.

La différence sur le score d'adhérence aux contours est moins important avec les méthodes de Felzenszwalb *et al.* (+1%) et d'Achanta *et al.* (+2%), mais le nombre de superpixels reste significativement inférieur avec, respectivement, 500 et 300 superpixels de moins. Enfin, ASARI obtient un score $\mathcal{F}_{FRC}$ identique à celui de l'algorithme de Liu *et al.* en produisant 150 superpixels de moins.

TABLEAU 6.3 – Valeurs moyennes pour la mesure floue d'adhérence aux contours ($\mathcal{F}_{FRC}$) et nombre moyen de superpixels par image ($N_{\mathbb{S}}$), avec les données de référence HSID.

Méthodes	$\mathcal{F}_{FRC}$	$N_{\mathbb{S}}$
Vedaldi *et al.*	0,57	2583
Felzenszwalb *et al.*	0,60	1998
Liu *et al.*	0,61	1700
Achanta *et al.*	0,59	1854
Rubio *et al.*	0,46	1881
Conrad *et al.*	0,37	2025
Machairas *et al.*	0,51	2069
ASARI	0,61	1528

La figure 6.4 montre deux exemples de segmentations produites par ASARI sur les images de la base de données HSID. La figure 6.4a est un autre exemple du caractère adaptatif d'ASARI, avec des superpixels capables de s'adapter à la taille des différents objets. Sur la figure 6.4b, l'information de texture permet de séparer l'objet principal du fond, pourtant de couleurs similaires.

(a) Image img-010. Nombre de superpixels : 1445. Score d'adhérence aux contours : 0,66.

(b) Image img-036. Nombre de superpixels : 928. Score d'adhérence aux contours : 0,61.

FIGURE 6.4 – Deux résultats obtenus par ASARI sur la base de données HSID.

6.6.4 Temps de calcul

Les temps d'exécution moyens nécessaires à l'obtention des résultats précédents, pour chacun des ensembles de données de référence sont donnés dans le tableau 6.4.

Pour de petites images (BSD et NYU), ASARI est plus lent que les méthodes de l'état de l'art. En revanche, lorsque des images de grande taille comme celle d'HSID doivent être sursegmentées, ASARI obtient des temps d'exécution significativement inférieurs à la quasi-totalité des méthodes avec :

- 73 secondes de moins que l'algorithme de Vedaldi *et al.* ;
- 4 secondes de moins que l'algorithme de Felzenszwalb *et al.* ;

TABLEAU 6.4 – Temps d'exécution moyens en secondes pour chacune des bases de données.

Méthodes	BSD	NYU	HSID
Vedaldi *et al.*	1, 24	1, 24	79
Felzenszwalb *et al.*	0, 10	0, 1	10
Liu *et al.*	1, 11	2, 21	36
Achanta *et al.*	0, 09	0, 17	3
Rubio *et al.*	-	-	37
Conrad *et al.*	0, 9	1, 19	23
Machairas *et al.*	-	-	29
ASARI	1, 38	2, 89	6

- 30 secondes de moins que l'algorithme de Liu *et al.* ;
- 31 secondes de moins que l'algorithme de Rubio *et al.* ;
- 17 secondes de moins que l'algorithme de Conrad *et al.* ;
- 23 secondes de moins que l'algorithme de Machairas *et al.*.

La seule méthode plus rapide qu'ASARI est l'algorithme SLIC d'Achanta *et al.* qui produit une sur-segmentation en seulement 3 secondes.

6.7 Conclusion

6.7.1 Bilan par rapport à l'état de l'art

Les évaluations que nous avons menées montrent que sur de petites images, semblables à celles qui composent les données de référence des ensembles BSD et NYU, le bénéfice d'ASARI par rapport à l'état de l'art est ténu. Ces résultats s'expliquent par deux phénomènes :

- les mécanismes visant à réduire la complexité algorithmique d'ASARI ont un impact moindre, ce qui contribue à augmenter l'écart entre les temps d'exécution de cet algorithme et ceux des méthodes concurrentes ;
- la petite taille des images implique que les pixels à partager entre les régions de V_0 sont peu nombreux. En particulier, il est difficile de fixer la taille de ces dernières de manière à ce que le nombre de pixels les composant soit suffisamment élevé pour le descripteur de texture mais suffisamment faible pour éviter d'introduire des erreurs de sur-segmentation dès cette étape.

Ajoutons que l'une des contreparties du fait qu'ASARI fusionne des régions réside dans le fait que, lorsqu'une arête est supprimée à tort, l'erreur commise est plus importante que pour des algorithmes travaillant au niveau du pixel.

Avec des images de grande taille, les avantages d'ASARI deviennent plus marqués. Nous constatons notamment que :

- avec la méthode de Liu *et al.*, ASARI est le seul algorithme à atteindre un score d'adhérence aux contours de 0,61 ;

- les superpixels produits par ASARI sont significativement moins nombreux ;
- en termes de rapidité d'exécution, ASARI obtient la deuxième place, avec seulement une durée moyenne de 6 secondes, là où tous les autres algorithmes, à l'exception de SLIC, sont au delà des 10 secondes.

Le code source d'ASARI a été mis à disposition [7].

6.7.2 Perspectives

Malgré ces résultats encourageants, nous ne pensons pas qu'ASARI puisse être, dans l'état actuel, intégré de manière fructueuse à $S\alpha F$. Les deux principaux problèmes sur lesquels nous souhaitons poursuivre nos travaux concernent :

- la diminution des erreurs commises pour les petites images ;
- la réduction du temps d'exécution.

Afin d'améliorer les performances d'ASARI sur les ensembles de données BSD et NYU, nous réfléchissons à l'introduction de mécanismes permettant de remettre en cause l'intégrité d'un sommet et de pouvoir le séparer en plusieurs régions. Nous envisageons la prise en compte d'une information sur les contours et recherchons un procédé capable de répartir de manière efficace les pixels d'une région en plusieurs sous-ensembles connexes.

Dans l'optique de réduire les temps d'exécution, nous travaillons actuellement à identifier les étapes d'ASARI les plus coûteuses, afin d'en réduire la durée, soit en parallélisant ces parties de l'algorithme, soit en recherchant une approximation des informations calculées.

Si ces deux pistes s'avèrent fructueuse, ASARI permettra sans aucun doute d'améliorer la méthode de segmentation interactive $S\alpha F$.

7. http://image.ensfea.fr/asari/

Chapitre 7

Conclusion

Dans ce mémoire, nous avons présenté une synthèse de nos travaux menés autour deux thématiques différentes : la segmentation interactive et la sur-segmentation. Le fil conducteur de ces recherches est la mise au point d'un outil de sélection et d'annotation permettant l'enrichissement d'un observatoire photographique du paysage. Ce dernier permet d'associer à une photographie une segmentation sémantique indiquant pour chaque pixel la catégorie d'objet à laquelle il appartient.

7.1 Segmentation interactive

Ces deux dernières décennies, de très nombreuses méthodes de segmentation interactive ont été proposées. Traditionnellement, ces méthodes sont classés selon leur mode d'interaction, avec des algorithmes axés sur la recherche des contours [32, 33] tandis que d'autres se basent sur celle des régions [18, 31, 46].

Dans ce mémoire, nous avons montré que le choix du mode d'interaction influence le type de problème que la méthode pourra résoudre. Tandis que les méthodes par recherche des contours se limitent à la sélection d'objets correspondant à une seule composante connexe, sans trou, le mode d'interaction propre aux méthodes par recherche des régions n'impose aucune limitation sur la topologie des objets ou leur nombre.

Les trois catégories que nous avons proposé dans le chapitre 2 – méthodes de binarisation interactive par recherche des contours, méthodes de binarisation interactive par recherche des régions, méthodes de segmentation interactive multiclasse – permettent d'identifier le type de problème résolu par l'algorithme.

7.1.1 Proposition d'une nouvelle méthode

L'une des principales contributions de nos travaux est la conception d'une nouvelle méthode de segmentation interactive multiclasse, $S\alpha F$. Décrite en détail dans le chapitre 3, cette dernière repose sur l'extraction de primitives visuelles nommées superpixels, qui correspondent à de petites régions décrites par leur couleur moyenne et la position de leur barycentre.

L'utilisation d'un SVM, entraîné à partir des germes donnés par l'utilisateur, permet de tirer le meilleur parti de ce descripteur minimaliste. Nous montrons comment estimer à la fois la probabilité qu'un superpixel $\mathbf{s}_i$ appartienne à une classe λ_m et celle que deux superpixels voisins, $\mathbf{s}_{i1}$ et $\mathbf{s}_{i2}$ appartiennent respectivement aux classes λ_{m1} et λ_{m2}. Ces deux types de probabilités nous permettent de calculer les facteurs d'une fonction que nous minimisons dans un graphe de facteurs, afin de trouver une segmentation adéquate de l'image.

7.1.2 Mise à disposition d'un nouvel ensemble de données de référence

L'évaluation et la comparaison des différentes méthodes de segmentation interactive est une tâche difficile. Les codes sources sont rarement disponibles, les données de référence sont utilisées selon des modalités très différentes et l'ergonomie des méthodes est prise en compte de manière sporadique. Un travail reste à faire qui permettrait une analyse rigoureuse des avantages et inconvénients de chaque algorithme.

Notre contribution concerne la création et la mise à disposition de trois ensembles de données comprenant des images de tailles différentes afin de tester le passage à l'échelle d'une méthode de segmentation interactive multiclasse. Les images du dernier ensemble, contenant 1800×1201 pixels, ont en outre l'avantage de se rapprocher davantage des dimensions des photographies prises par les appareils pour le grand public que les données de référence utilisées jusqu'à maintenant [31, 43, 46].

7.1.3 Évaluation de la méthode proposée

La comparaison d'une nouvelle méthode de segmentation interactive avec les méthodes existantes est une tâche délicate. En mettant en regard les résultats obtenus par $S\alpha F$ avec ceux de l'état de l'art, nous avons montré l'intérêt de l'algorithme que nous proposons.

Nous nous sommes également intéressés à certaines de ses propriétés :

- intérêt du terme de régularisation ;
- passage à l'échelle ;
- ergonomie ;
- caractéristiques des germes à donner.

Ces tests ont permis de mettre en évidence les capacités de $S\alpha F$, ses points forts et ses faiblesses.

7.2 Sur-segmentation

7.2.1 Conception d'un nouveau protocole d'évaluation

L'une des principales motivations des algorithmes de sur-segmentation est de créer des groupes de pixels homogènes, les superpixels, moins nombreux que les pixels, donc plus rapides à traiter que ces derniers. Cependant, les données de référence utilisées pour évaluer les méthodes de sur-segmentation sont des images de taille modeste : à peine quelques milliers de pixels.

Afin d'y remédier, nous avons conçu et mis à disposition un nouvel ensemble de données, $HSID$, composé d'images de tailles diverses, avec toutefois une majorité de grandes images, contenant plusieurs millions de pixels. Les particularités de ces données nous ont conduit à repenser les mesures traditionnellement utilisées pour évaluer la qualité d'une sur-segmentation.

Enfin, une analyse des méthodes utilisant les superpixels nous a permis de formaliser les attentes propres à un algorithme de sur-segmentation, au travers de cinq propriétés (validité, adhérence aux contours, concision, simplicité, rapidité) et du concept d'adaptabilité. L'ensemble de ce travail a été détaillé au chapitre 4.

7.2.2 Analyse des méthodes existantes

Présentée dans le chapitre 4, l'analyse des résultats obtenus par les méthodes de l'état de l'art appliquées aux images de $HSID$ révèlent que seulement trois d'entre elles conservent leurs bonnes performances lorsqu'elles sont amenées à traiter des images de tailles différentes ou des images de grande taille.

Le choix d'une méthode parmi les trois obtenant de bons résultats dépend du domaine d'application. En particulier nos test, montrent que :

- l'algorithme ERS de Liu *et al.* [26] permet d'obtenir une excellente adhérence aux contours avec un nombre peu élevé de superpixels. Cependant, cette méthode est lente et n'a aucune adaptabilité ;
- l'algorithme FZ de Felzenszwalb *et al.* [13] fournit un compromis intéressant : bénéficiant d'une bonne adaptabilité, il est capable d'atteindre une adhérence aux contours correcte avec un nombre de superpixels raisonnable. Ses temps d'exécution sont trois fois plus rapides que ceux de l'algorithme ERS.
- l'algorithme SLIC d'Achanta *et al.* [1] est de loin la méthode la plus rapide, avec un temps d'exécution dix fois inférieur à celui d'ERS. Il requiert toutefois quelques superpixels supplémentaires pour obtenir la même adhérence aux contours que ses deux concurrents.

7.2.3 Proposition d'une nouvelle méthode

L'étude des algorithmes de sur-segmentation existants nous a conduit à proposer, dans le chapitre 6, une nouvelle méthode, $ASARI$. Cette dernière s'appuie sur un premier résultat obtenu grâce à l'algorithme SLIC, qui correspond à une sur-segmentation en de très nombreux superpixels. Certains de ces superpixels sont alors fusionnés afin d'aboutir à un résultat meilleur

que celui de SLIC en ce qui concerne l'adhérence aux contours et le nombre de superpixels, tout en conservant des temps d'exécution aussi faibles que possible.

Les deux principales particularités d'ASARI sont :

- le fait que les superpixels produits soient non seulement homogènes au sens de la couleur mais également au sens de la texture ;
- une bonne adaptabilité, qui permet d'obtenir de grands superpixels dans les zones uniformes et de nombreux petits superpixels dans celles riches en détails.

Les résultats d'ASARI sont encourageants, en particulier pour les images de grandes tailles comme celle de $HSID$ pour lesquelles ses performances sont nettement meilleures que celle du reste de l'état de l'art.

7.3 Perspectives

7.3.1 Évaluation des méthodes de segmentation interactive

Si l'évaluation que nous avons conduite permet de montrer l'intérêt de $S\alpha F$, une comparaison plus complète des divers algorithmes de l'état de l'art reste à réaliser. Elle nécessite un travail conséquent pour implémenter les méthodes existantes (leur code n'étant pas disponible) et la mise en place d'un protocole permettant de mieux prendre en compte la place de l'utilisateur.

De manière similaire à ce qui a été réalisé par McGuinness *et al.* [31], il serait souhaitable que chaque méthode soit évaluée par différents utilisateurs. Les questions posées à ces derniers devront permettre d'obtenir des données quantitatives plus fiables concernant les différents aspects que nous avons évoqués dans le chapitre 5, avec notamment :

- l'influence du temps d'exécution sur la pénibilité d'utilisation d'un outil de segmentation interactive ;
- les caractéristiques des germes à donner à chaque méthode et la manière dont ils permettent d'obtenir plus ou moins rapidement un résultat satisfaisant.

Par ailleurs, une telle évaluation nécessitera de poursuivre notre effort pour la mise à disposition d'ensembles de données de référence contenant des images de tailles similaires à celles produites actuellement par les appareils photographiques grand public. Les ensembles que nous avons proposés doivent être étendus. Il serait également souhaitable de travailler à la réalisation d'un ensemble contenant des données de très grandes tailles (plusieurs milliers de pixels en largeur et en hauteur).

7.3.2 Intégration d'ASARI au sein de $S\alpha F$

Comme évoqué à la fin du chapitre 5, l'étape de $S\alpha F$ ouvrant le plus de perspectives pour l'amélioration de ses performances est celle de la sur-segmentation. Dans le chapitre 6, nous avons présenté un nouvel algorithme de sur-segmentation : ASARI.

Actuellement, ASARI est trop lent pour être intégré avec succès au sein d'une méthode comme $S\alpha F$. En particulier, ses temps d'exécution compromettent une implémentation utilisable sur

smartphone. Un travail important doit donc être réalisé, afin d'accélérer ASARI. La conclusion du chapitre 6 nous a donné l'occasion d'évoquer quelques pistes, telles que la parallélisation de certaines portions de l'algorithme.

Une fois ce travail effectué, l'intégration d'ASARI à la méthode $S\alpha F$ reposera la question du descripteur, les superpixels produits par cet algorithme ayant la propriété de ne pas être seulement homogènes au sens de la couleur, mais également en termes de texture. La mise en place de tests supplémentaires devra permettre de décider si l'utilisation d'un descripteur plus complexe, intégrant la texture en plus de la couleur et de la localisation, influence la qualité des segmentations produites ou les germes à donner.

7.3.3 Observatoires photographiques du paysage

La version de $S\alpha F$ que nous avons décrite au chapitre 3 a été implémentée avec succès dans une application distribuée facilitant l'enrichissement d'un observatoire photographique du paysage [**?**]. L'algorithme $S\alpha F$ permet d'associer à chaque photographie une segmentation indiquant la classe sémantique des pixels et permettant d'identifier et de localiser avec précision les objets présents. La faible complexité de $S\alpha F$ permet une utilisation fluide de cet outil.

Nous souhaitons tester l'utilisation de cette application distribuée dans un contexte de jeu-sérieux où la mise en place d'un observatoire et la contribution à son amélioration serviraient de supports dans des cours sur l'aménagement du territoire. Nous envisageons d'ajouter à l'application mobile une fonctionnalité d'envoi sur le réseau social Twitter, afin qu'un message accompagne chaque nouvelle image ajoutée à l'observatoire. De la sorte, les utilisateurs qui suivent le mot-clé correspondant seront informés des changements et les participants à un jeu sérieux seront avisés de l'avancée de leurs partenaires ou de leurs concurrents. Nous allons mener des tests avec deux classes et leurs enseignants, l'une en classe de terminale technologique « Sciences et technologies de l'agronomie et du vivant », l'autre en en classe de terminale professionnelle « Aménagements paysagers ».

Annexe

Codes et jeu de données mis à disposition

1 Greffon Gimp $S\alpha F$

L'implémentation de $S\alpha F$ comme greffon pour le logiciel Gimp est disponible à l'adresse suivante : http://image.ensfea.fr/saf/.

2 Application Android pour l'enrichissement d'un observatoire photographique du paysage

Une version de démonstration de cette application Android (avec les principales fonctionnalités) est disponible à l'adresse suivante : https://github.com/BerengereMathieu/SafDemoAndroidStudio

3 Base de données HSID

L'ensemble des images et des segmentations de référence constituant HSID sont disponibles à l'adresse suivante : http://image.ensfea.fr/hsid/.

4 Implémentation d'ASARI

Le code source d'ASARI est disponible à l'adresse suivante : http://image.ensfea.fr/asari/.

Bibliographie

[1] A. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua et S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] R. Adams et L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[3] S. Boyd et L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[4] Y. Boykov et M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. *In IEEE International Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] O. Changjae, H. Bumsub et S. Kwanghoon. Robust interactive image segmentation using structure-aware labeling. *Expert Systems with Applications*, 79:90 – 100, 2017.

[7] D. Comaniciu et P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[8] C. Conrad, M. Mertz et R. Mester. Contour-relaxed superpixels. *In International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 280–293, 2013.

[9] N. Dalal et B. Triggs. Histograms of oriented gradients for human detection. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[10] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[11] P. Dollár et C. L. Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570, 2015.

[12] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[13] P. F. Felzenszwalb et D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[14] D. Fourure, R. Emonet, E. Fromont, D. Muselet, N. Neverova, A. Trémeau et C. Wolf. Multi-task, multi-domain learning: application to semantic segmentation and pose regression. *Neurocomputing*, 251:68–80, 2017.

[15] G. Friedland, K. Jantz et R. Rojas. Siox: Simple interactive object extraction in still images. *In IEEE International Symposium on Multimedia*, pages 253 –260, 2005.

[16] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez et J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[17] S. Gould, J. Rodgers, D. Cohen, G. Elidan et D. Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008.

[18] M. Jian et C. Jung. Interactive image segmentation using adaptive constraint propagation. *IEEE Transactions on Image Processing*, 25(3):1301–1311, 2016.

[19] J. Kappes, B. Andres, F. Hamprecht, C. Schnorr, S. Nowozin, D. Batra, S. Kim, B. Kausler, J. Lellmann, N. Komodakis *et al.* A comparative study of modern inference techniques for discrete energy minimization problems. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1328–1335, 2013.

[20] A. Kolesnikov, M. Guillaumin, V. Ferrari et C. H. Lampert. Closed-form approximate CRF training for scalable image segmentation. *In European Conference on Computer Vision*, pages 550–565, 2014.

[21] D. Koller et N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[22] V. Lempitsky, C. Rother, S. Roth et A. Blake. Fusion moves for Markov random field optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1392–1405, 2010.

[23] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson et K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009.

[24] M. Li, A. Stein, W. Bijker et Q. Zhan. Region-based urban road extraction from VHR satellite images using binary partition tree. *International Journal of Applied Earth Observation and Geoinformation*, 44:217–225, 2016.

[25] D. Lin, J. Dai, J. Jia, K. He et J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 3159–3167, 2016.

[26] M. Y. Liu, O. Tuzel, S. Ramalingam et R. Chellappa. Entropy rate superpixel segmentation. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2104, 2011.

[27] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang et H.-Y. Shum. Learning to detect a salient object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):353–367, 2011.

[28] J. Long, E. Shelhamer et T. Darrell. Fully convolutional networks for semantic segmentation. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[29] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter et E. Decencière. Waterpixels. *IEEE Transactions on Image Processing*, 24(11):3707–3716, 2015.

[30] D. Martin, C. Fowlkes, D. Tal et J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *International Conference on Computer Vision*, 2:416–423, 2001.

[31] K. McGuinness et N. E. O'connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010.

[32] J. Mille, S. Bougleux et L. D. Cohen. Combination of piecewise-geodesic paths for interactive segmentation. *International Journal of Computer Vision*, 112(1):1–22, 2015.

[33] E. N. Mortensen et W. A. Barrett. Intelligent scissors for image composition. *In Annual Conference on Computer Graphics and Interactive Techniques*, pages 191–198, 1995.

[34] E. N. Mortensen et W. A Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

[35] S. Müller, P. Ochs, J. Weickert et N. Graf. Robust interactive multi-label segmentation with an advanced edge detector. *In German Conference on Pattern Recognition*, pages 117–128, 2016.

[36] C. Nieuwenhuis et D. Cremers. Spatially varying color distributions for interactive multilabel segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1234–1247, 2013.

[37] T. Ojala, M. Pietikainen et T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[38] J. Papon, A. Abramov, M. Schoeler et F. Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[39] A. Petrovai, A. Costea, F. Oniga et S. Nedevschi. Obstacle detection using stereovision for Android-based mobile devices. *In IEEE International Conference on Intelligent Computer Communication and Processing*, pages 141–147, 2014.

[40] M. Pietikäinen, A. Hadid, G. Zhao et T. Ahonen. Local binary patterns for still images. *In Computer vision using local binary patterns*, pages 13–47. Springer, 2011.

[41] T. Pock, A. Chambolle, D. Cremers et H. Bischof. A convex relaxation approach for computing minimal partitions. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 810–817, 2009.

[42] X. Ren et J. Malik. Learning a classification model for segmentation. *In IEEE International Conference on Computer Vision, Proceedings*, volume 1, pages 10–17, 2003.

[43] C. Rother, V. Kolmogorov et A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. 23(3):309–314, 2004.

[44] A. Rubio, L. Yu, E. Simo-Serra et F. Moreno-Noguer. Bass: Boundary-aware superpixel segmentation. *In International Conference on Pattern Recognition*, pages 2824–2829, 2016.

[45] P. Salembier et L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576, 2000.

[46] J Santner, T. Pock et H. Bischof. Interactive multi-label segmentation. *In Asian Conference on Computer Vision*, pages 397–410, 2010.

[47] N. Silberman, D. Hoiem, K. Pushmeet et R. Fergus. Indoor segmentation and support inference from RGBD images. *In European Conference on Computer Vision*, pages 746–760, 2012.

[48] M. A. Stricker et M. Orengo. Similarity of color images. *In IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 381–392, 1995.

[49] D. Stutz. Superpixel segmentation: An evaluation. *In German Conference on Pattern Recognition*, pages 555–562, 2015.

[50] D. Tang, H. Fu et X. Cao. Topology preserved regular superpixel. *In IEEE International Conference on Multimedia and Expo*, pages 765–768, 2012.

[51] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani et L. Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *In European Conference on Computer Vision*, pages 13–26, 2012.

[52] A. Vedaldi et S. Soatto. Quick shift and kernel methods for mode seeking. *In European Conference on Computer Vision*, pages 705–718, 2008.

[53] O. Veksler, Y. Boykov et P. Mehrani. Superpixels and supervoxels in an energy optimization framework. *In European Conference on Computer Vision*, pages 211–224, 2010.

[54] L. Vincent et P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[55] D. Weikersdorfer, D. Gossow et M. Beetz. Depth-adaptive superpixels. *In International Conference on Pattern Recognition*, pages 2087–2090, 2012.

[56] T.-F. Wu, C.-J. Lin et R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.

[57] Y. Xu, T. Géraud et L. Najman. Connected filtering on tree-based shape-spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1126–1140, 2016.

[58] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz et T. L. Berg. Parsing clothing in fashion photographs. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, 2012.

[59] Q. Yan, J. Shi, L. Xu et J. Jia. Hierarchical saliency detection on extended CSSD. *arXiv preprint arXiv:1408.5418*, 2014.

[60] L. Zhang, Y. Yang, Y. Gao, Y. Yu, C. Wang et X. Li. A probabilistic associative model for segmenting weakly supervised images. *IEEE Transactions on Image Processing*, 23(9):4150–4159, 2014.

[61] L. Zhang, Y. Yang, M. Wang, R. Hong, L. Nie et X. Li. Detecting densely distributed graph patterns for fine-grained image categorization. *IEEE Transactions on Image Processing*, 25(2):553–565, 2016.

[62] Y. Zhang, R. Hartley, J. Mashford et S. Burn. Superpixels via pseudo-boolean optimization. *In IEEE International Conference on Computer Vision*, pages 1387–1394, 2011.

[63] M. Özden et E. Polat. Image segmentation using color and texture features. *European Signal Processing Conference*, pages 1–4, 2005.