

UNIVERSITÉ DE BORDEAUX
MASTER 2 INFORMATIQUE POUR L'IMAGE ET LE SON

Mémoire de stage

**Sous-représentation irrégulière duale pour le
traitement d'images**

Stagiaire:

Arthur LONGUEFOSSE

Encadrants:

Rémi GIRAUD

Michaël CLÉMENT

Enseignants référents:

Pierre HANNA

Lionel CLÉMENT

Résumé

La segmentation consiste à partitionner une image en ensembles de pixels homogènes partageant des caractéristiques communes. Dans ce domaine clé du traitement d’images, il existe un besoin important de méthodes rapides sans apprentissage et efficaces avec peu de données d’entraînement, notamment pour la segmentation d’images médicales. Ce stage de recherche, proposé par des enseignants-chercheurs de l’IMS et du LaBRI, s’intéresse à ces problématiques actuelles en considérant plusieurs méthodes autour des superpixels, une pré-segmentation réduisant le nombre d’éléments de l’image. Nous introduisons un logiciel interactif s’appuyant sur une hiérarchie de superpixel pour fournir une segmentation binaire d’un objet dans une image. Nous proposons également une étude sur des sous-représentations irrégulières duales, basées sur les centres et les contours des superpixels et permettant de capturer les informations structurelles à leurs frontières. Enfin, nous démontrons les performances de ces approches duales sur les applications de mise en correspondance, de génération de trimap semi-automatique et de matting.

Mots-clés — Superpixels, Segmentation d’images interactive, Descripteur dual, Correspondance de patchs, Trimap

Abstract

Segmentation is the process of partitioning an image into sets of homogeneous pixels that share common characteristics. In this key area of image processing, there is a great need for fast methods without learning steps and which are effective with limited training data, especially for the segmentation of medical images. This research internship, proposed by teacher-researchers from the IMS and LaBRI, addresses these current issues by considering several methods based on superpixels, a pre-segmentation reducing the number of image elements. We introduce an interactive software based on a superpixel hierarchy to provide a binary segmentation of an object in an image. We also propose a study of dual irregular under-representations, based on the center and contours of superpixels, to capture structural information at their boundaries. Finally, we demonstrate the performance of these dual approaches on patch-matching, semi-automatic trimap generation and matting applications.

Keywords — Superpixels, Interactive Image Segmentation, Dual descriptor, Patch-matching, Trimap

Table des matières

1	Introduction	4
1.1	Contexte	4
1.2	Planning prévisionnel	5
1.3	Environnement de travail	5
1.4	État de l'art et objectifs	6
2	Segmentation interactive avec hiérarchie de superpixels	12
2.1	Hiérarchie basée SLIC	12
2.2	Hiérarchie rapide basée graphe	15
2.3	Pose de labels et marqueurs binaires	17
2.4	Résultats et tests	19
3	Sous-représentations duales et DSPM	23
3.1	Décomposition duale basée barycentres	23
3.2	Décomposition duale basée dilatation des bords	30
3.3	Intégration au pipeline DSPM	31
3.4	Résultats et tests	33
4	Génération de trimap et matting	37
4.1	État de l'art	37
4.2	Génération de trimap à partir de la segmentation binaire	40
4.3	Intégration au logiciel Qt	41
5	Bilan	43
5.1	Contributions et conclusion sur les objectifs	43
5.2	Retours sur le déroulement du stage	43
	Bibliographie	45

Chapitre 1

Introduction

1.1 Contexte

La segmentation d'images est un sujet fondamental du traitement d'images et de la vision par ordinateur, avec des applications dans différents domaines tels que la détection d'objets, la vidéo-surveillance, l'analyse d'images médicales, et bien d'autres. La segmentation consiste à décomposer une image en différentes régions, souvent basées sur les caractéristiques des pixels. Dans ce contexte, l'utilisation de superpixels (cf Figure 1.1) peut être intéressante pour réduire la complexité des algorithmes de segmentation, en regroupant les pixels dans des régions homogènes qui respectent le contour des objets. Cette approche de pré-segmentation peut être par exemple utilisée dans des méthodes de correspondance rapide entre images, et permet de s'abstraire de réseaux de neurones s'appuyant sur un apprentissage supervisé coûteux, nécessitant souvent de grands ensembles de données annotées. En effet, dans de nombreuses applications de vision par ordinateur, telles que l'imagerie médicale, il existe un besoin de résultats rapides, automatiques et efficaces avec peu de données d'entraînement.

Récemment, l'article Dual SuperPatchMatch (DSPM) (Giraud et al., 2020, [1]), publié par mes encadrants de stage, met en avant un inconvénient majeur des algorithmes basés superpixels : l'irrégularité de forme de ces représentations, ce qui implique une difficulté à structurer l'information de voisinage et de contour. Ils proposent une solution basée sur des groupements de superpixels, appelés SuperPatch, et des interfaces de contour basées sur l'information inter-superpixels.

Ce stage de recherche s'intéresse à ces problématiques actuelles en cherchant à étudier l'intérêt d'une sous-représentation irrégulière duale basée sur les contours des superpixels, notamment dans les domaines de la correspondance de patch, la génération de trimap et du matting.

Tout d'abord, le planning prévisionnel et l'environnement du stage vont être exposés, avant de décrire l'état de l'art réalisé.

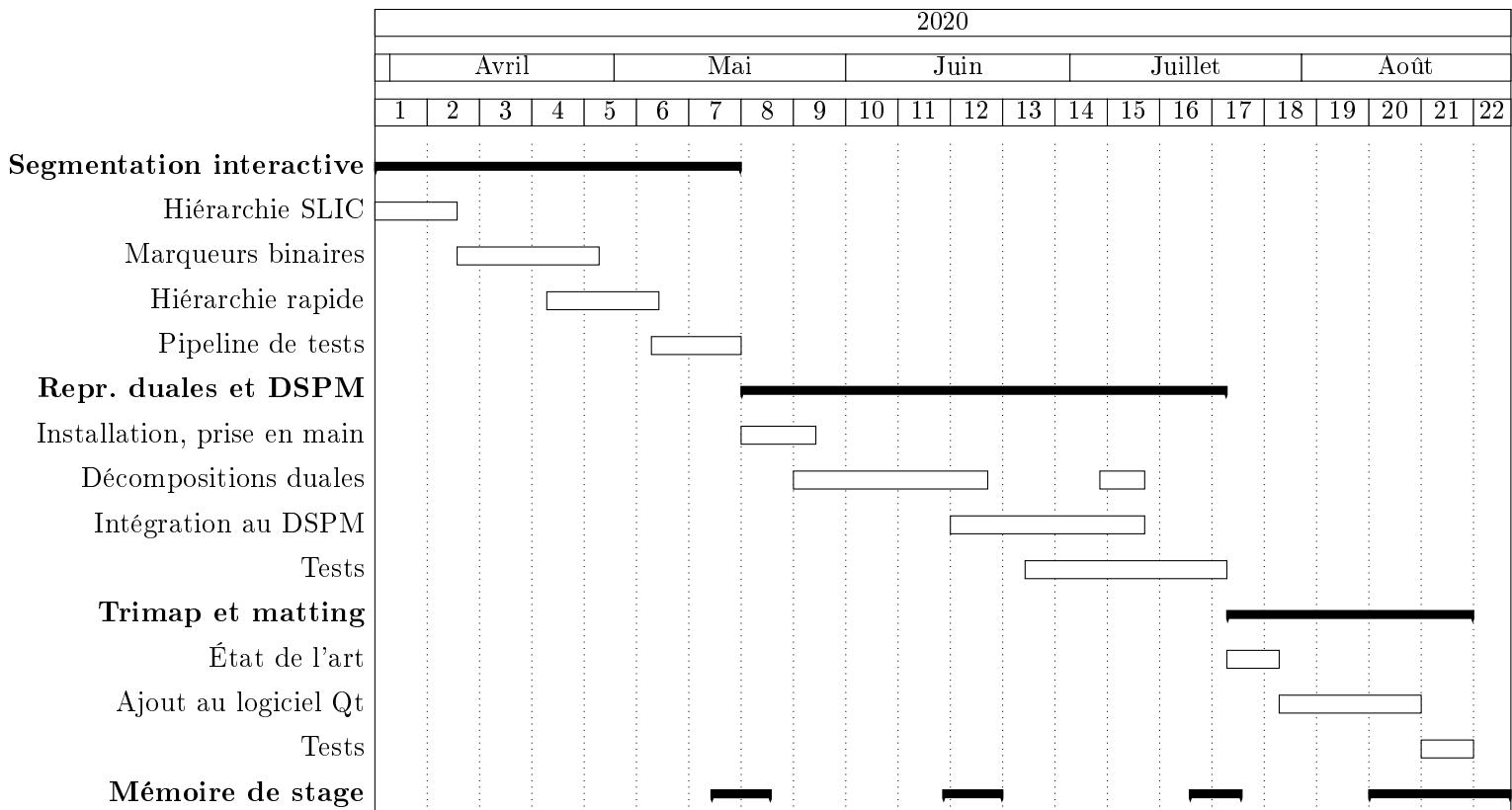
Nous allons ensuite présenter la première partie du stage, où nous avons conservé une continuité avec le Projet de Fin d'Etudes effectué auparavant avec mes encadrants (intitulé *PFE Superpixels*), en ajoutant au logiciel développé une version interactive avec pose de labels, s'appuyant sur une hiérarchie de superpixels.

Par la suite, nous avons développé plusieurs sous-représentations duales basées sur les barycentres des superpixels et sur la dilatation des contours, et testé ces différentes implémentations dans l'algorithme DSPM.

Enfin, la dernière partie du stage a été consacrée à l'élaboration d'un état de l'art exhaustif sur la génération de trimap et le matting, pour étudier l'intérêt des représentations duales dans ces domaines et implémenter dans le logiciel une méthode de génération de trimap semi-automatique.

1.2 Planning prévisionnel

Ce stage est défini selon trois axes de réflexion ; l'implémentation d'une segmentation binaire, l'étude de sous-représentations duales de superpixels, et le lien entre ces deux approches dans le domaine de la génération de trimap. Le planning de ce stage est présenté sous la forme d'un diagramme de Gantt :



1.3 Environnement de travail

Compte tenu de la situation sanitaire liée au COVID-19, le stage s'est déroulé entièrement en télétravail. Suite aux recommandations du responsable des stages, M. Lionel CLÉMENT, j'ai établi un lieu de travail isolé pour éviter d'être perturbé, et des horaires de travail dans la semaine pendant des heures ouvrables, ponctuées de quelques pauses. Ces choix m'ont permis d'avoir un rythme de travail régulier et efficace dès le début du stage.

Disposant d'un ordinateur personnel, j'ai pu installer toutes les bibliothèques nécessaires au bon déroulement du stage. Certaines parties nécessitaient néanmoins l'utilisation du langage et de l'environnement de développement MATLAB, qui est seulement disponible sous licence propriétaire payante. Après recherche avec mes encadrants, il s'est avéré qu'une licence "*COVID-19 Classroom Access*"¹ était disponible gratuitement du 15 mars au 31 août 2020, ce qui m'a permis de l'utiliser durant l'intégralité de mon stage.

Pour communiquer par écrit avec mes encadrants, nous avons utilisé la plateforme collaborative Slack², sur laquelle je faisais un compte-rendu quotidien de mon travail et des éventuelles questions. De plus, une fois par semaine, nous utilisions un serveur vocal de la plateforme Discord³ pour échanger oralement sur mon avancée, et éventuellement effectuer un partage d'écran pour montrer les fonctionnalités implémentées ou pour m'aider à corriger certains bugs.

¹https://fr.mathworks.com/licensecenter/classroom/COVID-19_Access, dernier accès le 27/08/2020

²<https://slack.com/> dernier accès le 27/08/2020

³<https://discord.com/> dernier accès le 27/08/2020

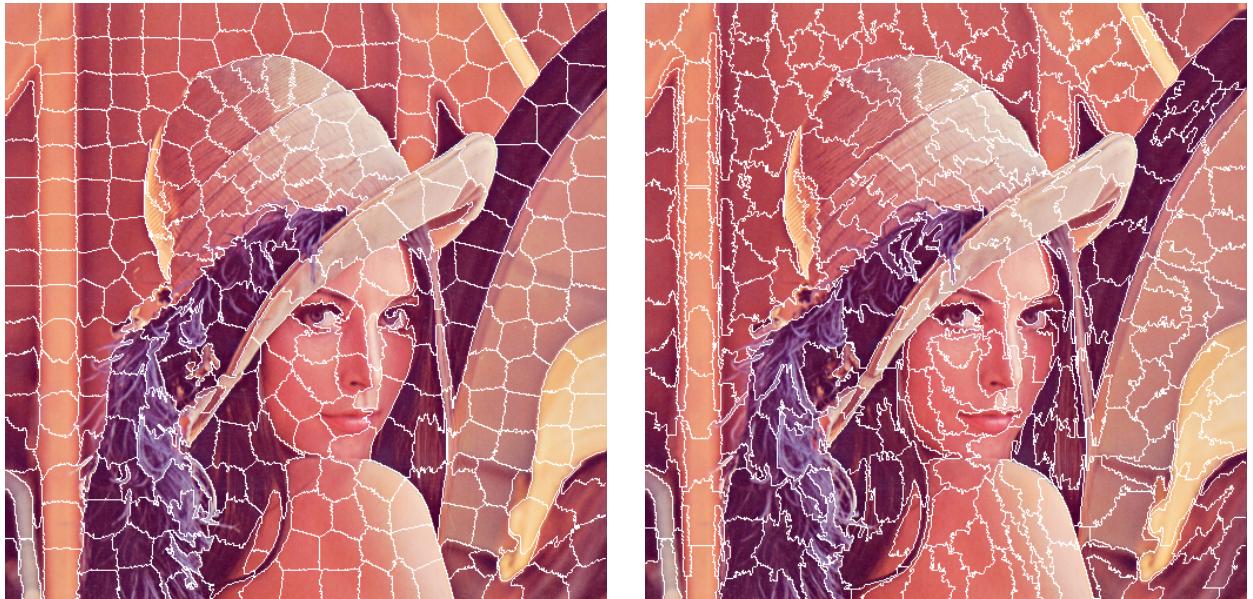
1.4 État de l'art et objectifs

Ce stage s'inscrivant dans le domaine de la recherche, une attention particulière a été portée sur l'étude de l'état de l'art tout au long du stage, c'est-à-dire les différents articles et méthodes relatifs à la segmentation d'images, et plus particulièrement aux domaines des superpixels, des outils interactifs, de la correspondance de patchs et de la génération de trimap.

1.4.1 Superpixels

Les superpixels, introduits en 2003 par Ren et Malik [2], correspondent à des groupes de pixels connexes partageant des caractéristiques communes. En décomposant une image avec des superpixels, on obtient alors des zones homogènes respectant les contours des objets. Cette représentation est souvent utilisée pour diminuer la complexité des algorithmes de traitement d'images, car elle réduit le nombre d'éléments à traiter et fournit des informations supplémentaires par rapport aux pixels.

L'utilisation des superpixels est devenu très populaire avec l'introduction de la méthode Simple Linear Iterative Clustering (SLIC) (Achanta et al., 2012 [3]). Cette méthode initialise les superpixels sur une grille régulière, puis effectue un groupement itératif des pixels selon la similarité des couleurs et la proximité spatiale dans l'image. Le compromis entre ces deux termes définit la *compacité* des superpixels (cf Figure 1.1). Enfin, une dernière étape est nécessaire pour assurer la connexité de chaque superpixel, en associant au superpixel le plus proche les pixels isolés.



(a) Superpixels à compacité élevée

(b) Superpixels à faible compacité

FIGURE 1.1 – Exemple de décompositions en superpixels (SLIC [3]), lena.png

1.4.2 Logiciel Qt Superpixels

En amont de ce stage, dans le cadre de l'UE *Projet de Fin d'Etudes* du dernier semestre du Master Informatique de l'Université de Bordeaux, nous avons réalisé un outil de segmentation basée superpixels, implémenté avec le framework **Qt** et le langage **C++**. Ce logiciel permet une segmentation rapide en superpixels (algorithme SLIC [3]) avec les paramètres définis par des sliders (nombre de superpixels et compacité). Nous avons également implémenté plusieurs fonctionnalités interactives telles que la sélection de superpixels (cf Figure 1.2), le zoom et la re-segmentation de la carte de superpixels, ainsi que la fusion de superpixels sélectionnés à différents niveaux de segmentation pour permettre un raffinement de la sélection (cf Figure 1.3).

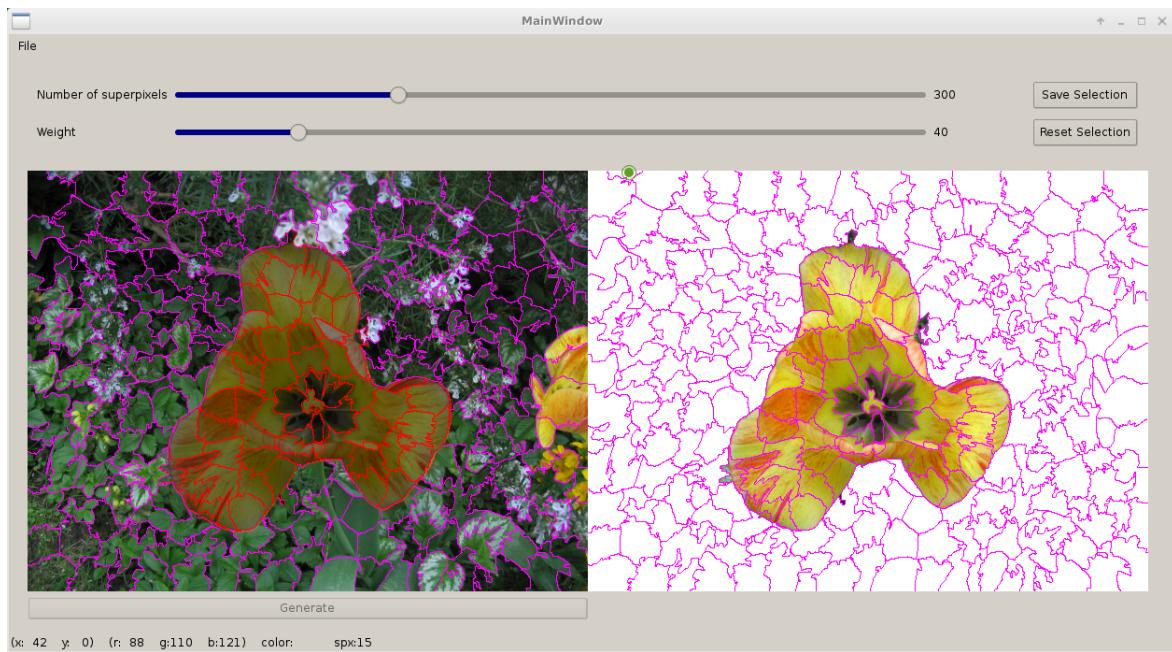


FIGURE 1.2 – Sélection des superpixels

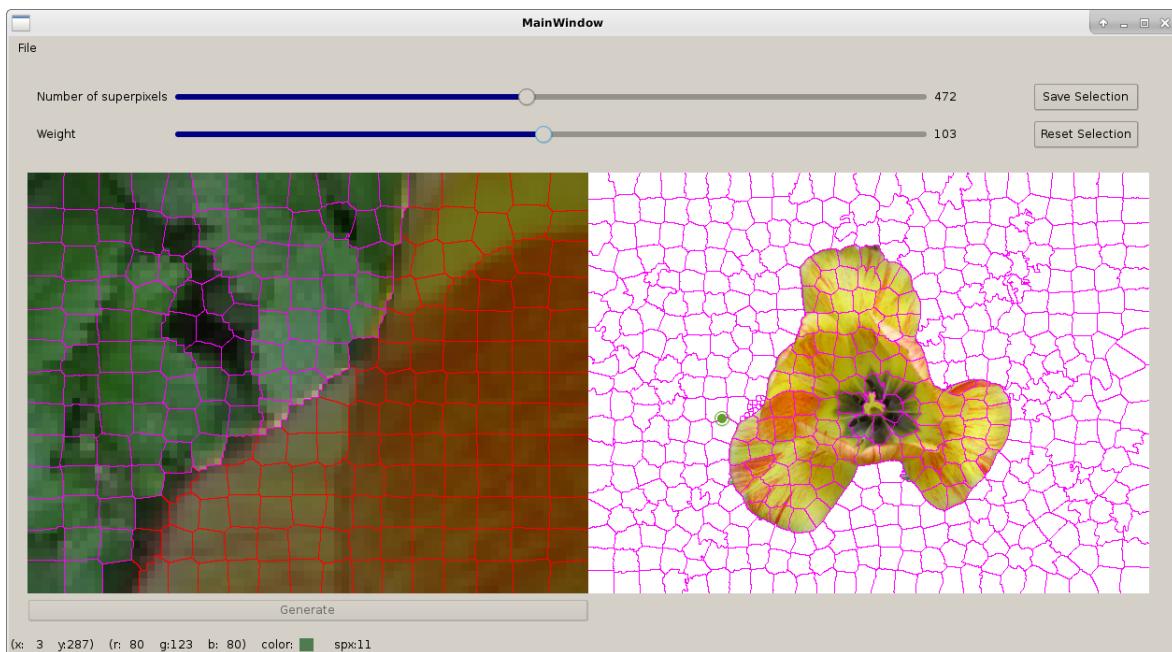


FIGURE 1.3 – Raffinement de la sélection en utilisant le zoom et la fusion de superpixels

A la fin de ce projet, nous avions également commencé à réfléchir à une sélection rapide des superpixels, avec par exemple la pose de labels (*scribble*). Il s'est avéré qu'une telle approche était trop difficile à implémenter avec une simple carte de superpixels, mais qu'une segmentation hiérarchique de superpixels pourrait être intéressante à exploiter. Celle-ci reposeraient sur la génération d'une segmentation initiale avec un grand nombre de superpixels (~ 2000 sp), et un calcul de segmentation hiérarchique par groupement ascendant (chaque superpixel est regroupé progressivement en se basant sur une distance couleur et spatiale). Par manque de temps, nous n'avions pas pu implémenter cette version.

Un des premiers objectifs de ce stage sera donc de terminer la version avec segmentation hiérarchique, puis d'implémenter la pose de label permettant une sélection rapide des superpixels.

1.4.3 Outils de segmentation interactive

La méthode du GrabCut (Rother et al., 2004 [4]) est une des premières méthodes semi-automatique pour la segmentation d’images. Elle nécessite seulement une interaction utilisateur à l’initialisation, sous forme d’une boîte englobante autour de l’objet à extraire. Le fond est alors défini à l’extérieur de la boîte et l’objet à l’intérieur, puis la méthode va converger vers une meilleure estimation de l’objet en cherchant à minimiser l’énergie de l’image pour trouver la segmentation binaire optimale, de façon itérative (cf Figure 1.4). GrabCut est basé sur la méthode de *graph cut*, permettant de représenter les pixels de l’image sous forme d’un graphe, et ainsi faciliter la segmentation rapide de l’image.



FIGURE 1.4 – Définition du GrabCut [4] © 2004 ACM : (a) l’utilisateur dessine une boîte englobante autour de l’objet ; (b) l’algorithme détecte la distribution des couleurs pour l’objet et le fond et applique une segmentation binaire ; (c) ce procédé est répété itérativement pour améliorer la segmentation.

L’article du GrabCut indique également la possibilité de corriger la segmentation semi-automatique avec une interaction utilisateur basée sur un pinceau-fond et pinceau-objet qui correspondent à des marqueurs binaires (également appelé *scribbles* ou *brush stroke*).

De nombreux articles utilisent des méthodes similaires basées sur ces interactions avec pinceaux, comme par exemple *Geodesic Graph Cut* (Price et al., 2010 [5], cf Figure 1.5) ou encore *Geodesic Star Convexity for Interactive Image Segmentation* (Gulshan et al., 2010 [6]), qui introduit des contraintes sur la forme des objets à segmenter en se basant sur des chemins géodésiques (i.e. plus court chemin), et en minimisant la fonction d’énergie globale dans l’image. Contrairement au GrabCut, ces méthodes reposent uniquement sur cette pose de labels manuels, et permettent donc d’obtenir des segmentations précises avec peu d’interaction utilisateur.

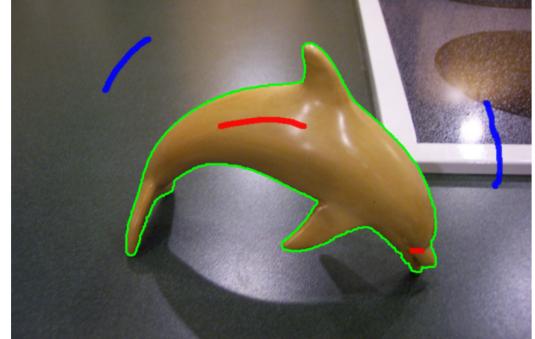


FIGURE 1.5 – Segmentation depuis des labels manuels (Price et al, 2010 [5]) © 2010 IEEE

Plus récemment, l’outil *Interactive segmentation with morphological hierarchies* (Perret, 2014 [7], disponible en ligne⁴) permet d’allier une structure hiérarchique de superpixels avec une pose de labels reposant sur des pinceaux objets-fonds. Cet outil se base sur la bibliothèque *Hierarchical Graph Analysis* (Higra [8]) pour fournir une segmentation binaire rapide et précise de l’image, mais est dépendant d’une méthode de bassin couvrant (*watershed*), et peut présenter des résultats très irréguliers et peu adaptés à la sélection manuelle. De plus, cet outil manque de fonctionnalités essentielles, telles qu’un zoom sur l’image pour raffiner la sélection, ou le fait de pouvoir changer les paramètres des superpixels (nombre, compacité) pour s’adapter à différentes formes d’objets.

Un des objectifs du stage est donc d’implémenter la version hiérarchique avec pose de labels dans notre logiciel, pour permettre d’allier les fonctionnalités déjà présentes dans le logiciel (zoom, raffinement, slider des paramètres) avec cette méthode interactive de segmentation.

⁴<https://perso.esiee.fr/~perretb/ISeg/>

1.4.4 Correspondance de patchs et superpixels

Les algorithmes de recherche de correspondance sont généralement basés sur l'utilisation de patchs, et permettent d'extraire et de transférer de l'information depuis des images d'exemples. Ces algorithmes sont notamment utiles dans les domaines de la reconstruction ou suppression d'objets dans une image (*inpainting*), de l'estimation de flux optique ou de la correspondance stéréo.

La méthode PatchMatch (Barnes et al, 2009 [9]) est une approche rapide et efficace permettant de calculer des correspondances de patchs entre deux images. Son principe repose sur le fait qu'une bonne correspondance entre deux patchs peut être propagée aux patchs adjacents. La méthode est divisée en 3 parties : initialisation, propagation et recherche aléatoire (cf Figure 1.6).

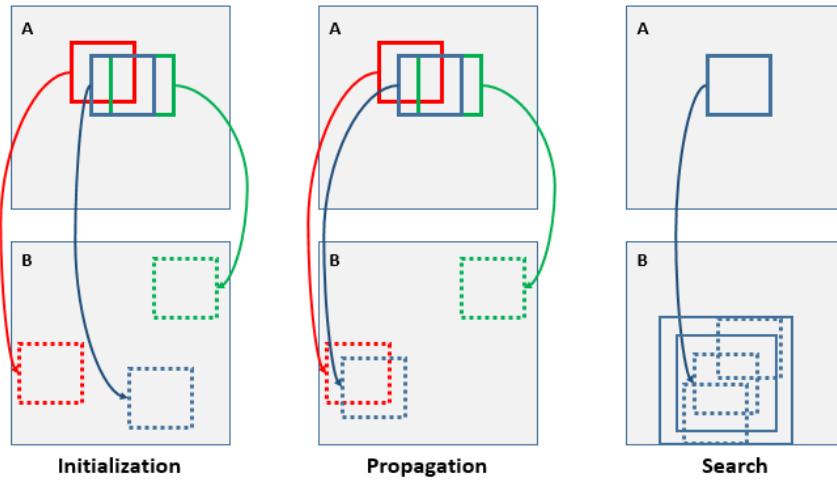


FIGURE 1.6 – Étapes du PatchMatch entre deux images A et B © 2017 Medium

La première étape consiste à initialiser aléatoirement des correspondances entre les patchs des deux images A et B. L'étape de propagation tente d'améliorer itérativement les correspondances de patchs en assumant que si le patch $f(x, y)$ de l'image A a une bonne correspondance avec le patch $f(x', y')$ de l'image B, alors les patchs voisins de $f(x, y)$ ont une bonne correspondance avec les patchs voisins de $f(x', y')$. Enfin, l'étape de recherche consiste à échantillonner aléatoirement autour de la meilleure correspondance afin d'améliorer les résultats et éventuellement sortir d'une zone de minimum local.

Lorsque la méthode de PatchMatch est appliquée pour de grandes images ou dans une bibliothèque d'images, la recherche de correspondance peut devenir coûteuse et nécessiter de nombreuses itérations avant de converger. Pour pallier à cela, un article récent associe la méthode de PatchMatch avec la décomposition en superpixels (Giraud et al, 2017 [10]). Le principe est d'appliquer les étapes d'initialisation, de propagation et de recherche aléatoire pour établir une correspondance entre des patchs de superpixels voisins, appelés *SuperPatchs*. Un SuperPatch est défini par tous les superpixels dont le barycentre est inclus dans un rayon de recherche autour d'un superpixel central (cf Figure 1.7).

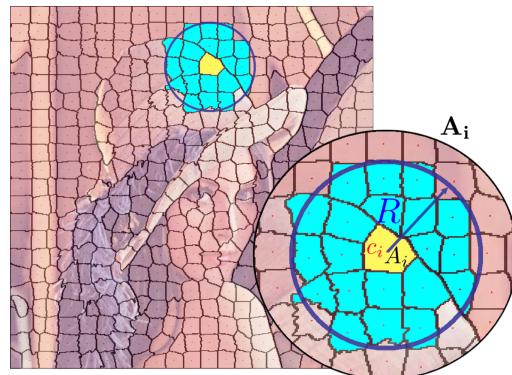


FIGURE 1.7 – Définition du SuperPatch à partir d'une décomposition en superpixels © 2017 IEEE

Une nouvelle méthode de comparaison de patchs à également été présentée, appelée SuperPatch-Match (SPM), pour permettre d'appliquer les étapes du PatchMatch à cette nouvelle structure en définissant une notion d'adjacence entre les SuperPixels.

En se basant sur le voisinage de superpixels, cette méthode permet de réaliser des comparaisons robustes de motifs visuels sur des domaines irréguliers de l'image. Cependant, SPM ne calcule des descripteurs qu'à l'intérieur de chaque région, et ne prend donc pas en compte l'information de contour disponible aux frontières des superpixels.

L'article SuperPatch Dual (DSPM, Giraud et al, 2019 [1]) vise à améliorer cette structure de SuperPatch en y ajoutant une information de contour ; chaque SuperPatch possède donc désormais deux types d'information (cf Figure 1.8). Les descripteurs intra-superpixels, basés sur l'histogramme RGB de chaque superpixel avec un décalage de β pixels à ses bordures, permettent d'extraire uniquement l'information pertinente à l'intérieur des superpixels (notamment ceux ayant des formes irrégulières). Les descripteurs inter-superpixels, quant à eux, sont basés sur des interfaces entre superpixels (régions de taille 3×3 pixels où au moins trois superpixels sont présents). Sur ces régions, des descripteurs de contour peuvent être calculés, comme par exemple l'histogramme de gradient orienté (HoG [11]).

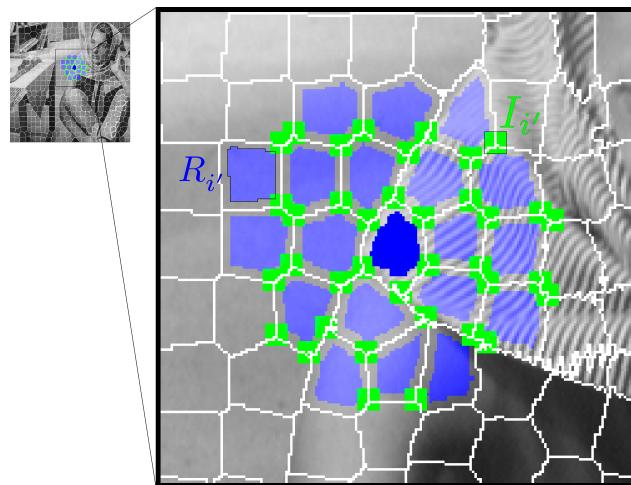


FIGURE 1.8 – Descripteur dual de SuperPatch © 2020 PRL : en bleu l'information intra-région (R'_i) avec un décalage de $\beta = 3$ pixels, en vert l'information inter-région (I'_i) basé sur les interfaces entre superpixels.

Cette nouvelle structure de SuperPatch Dual permet de répondre aux principales limites des méthodes de correspondance de superpixels existantes, en apportant une robustesse à l'imprécision des frontières des superpixels. Cet article a donc illustré l'intérêt de l'utilisation d'une information de contour de superpixels.

Dans ce contexte, l'objectif de ce stage est d'étudier l'intérêt d'une sous-représentation irrégulière duale, qui serait basée, non pas sur des interfaces entre superpixels, mais directement sur une carte de superpixels duaux, obtenue par exemple en reliant les barycentres des superpixels adjacents. Cette nouvelle structure devrait en théorie capturer efficacement les contours des superpixels, et pourra être implémentée dans l'algorithme du DSPM à la place des interfaces duales, pour permettre une comparaison des résultats avec l'existant.

1.4.5 Génération de trimap et matting

À partir d'une image donnée, le matting consiste à extraire un objet (élément du premier plan) du fond de l'image (arrière-plan). Pour faciliter l'extraction, la plupart des méthodes de matting sont initialisées avec une trimap (cf Figure 1.9), c'est-à-dire une partition de l'image en trois régions : l'objet, le fond, et une région inconnue où chaque pixel P peut être un mélange des couleurs de l'objet F ("Foreground") et du fond B ("Background"). Ce mélange de couleurs peut être défini selon l'équation suivante :

$$P = \alpha \times F + (1 - \alpha) \times B, \quad \alpha \in [0, 1]$$

Cette équation est composée de 7 inconnues (canaux RGB de l'objet F, canaux RGB du fond B, coefficient α), l'étape la plus importante étant la définition du coefficient α ; on parle alors d'alpha-matting.

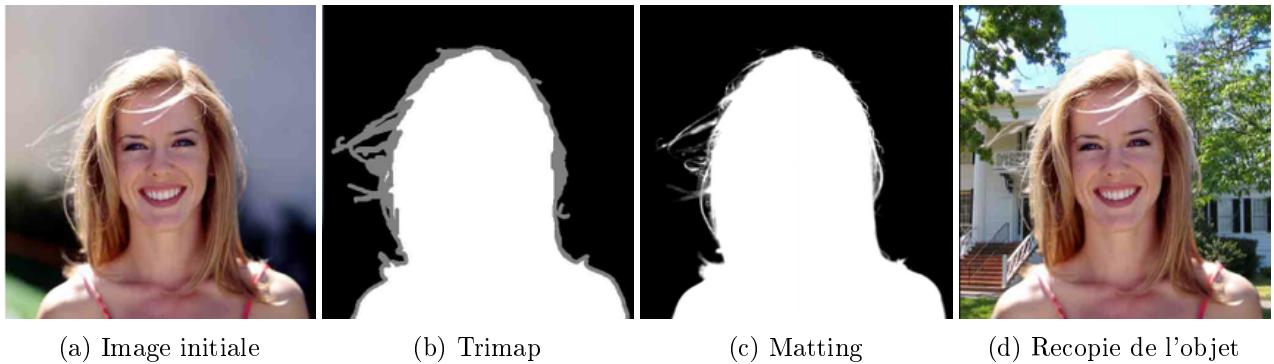


FIGURE 1.9 – Exemple de matting à partir d'une image et d'une trimap, et application à la recopie d'objet © 2010 IEEE

De nos jours, il existe un grand nombre d'algorithme de matting, et la majorité nécessitent en entrée une trimap en plus de l'image initiale. Or, peu d'algorithme permettent une génération automatique ou semi-automatique de ces trimaps de façon précise ; dans la plupart des domaines d'application (tels que l'industrie du cinéma, où le matting est très largement utilisée), ces trimaps sont alors annotées manuellement par l'utilisateur, ce qui est relativement long et contraignant.

En dernière partie du stage, nous allons donc procéder à un état de l'art exhaustif sur les méthodes de matting et de génération de trimap semi-automatique, pour déterminer les éventuels intérêts de la représentation dual basée superpixels dans ces domaines.

Enfin, un des objectifs finaux du stage sera d'implémenter la génération de trimap et/ou le matting dans le logiciel fourni, qui pourront alors être utilisés en parallèle de la segmentation interactive avec pose de labels, pour permettre d'extraire plus précisément les objets sélectionnés en raffinant la segmentation en superpixels initiale.

Chapitre 2

Segmentation interactive avec hiérarchie de superpixels

Comme indiqué précédemment, la première partie du stage consiste à implémenter une segmentation hiérarchique de superpixels, avec comme support le logiciel Qt réalisé lors du *PFE Superpixels*. Ce logiciel est basé sur une architecture Modèle-Vue-Contrôleur :

- Le modèle repose sur la structure de données `SLIC()`, qui contient les fonctions et données relatives aux superpixels et à la sélection ;
- La vue est gérée par la classe `MainWindow`, qui permet d'afficher les données du modèle et de recevoir les actions de l'utilisateur (sliders des paramètres, boutons de l'interface, sélection, zoom) ;
- Le contrôleur correspond à la classe `ClickableLabel` et va permettre la synchronisation du modèle et de la vue, en appliquant les modifications sur les données et en demandant la mise à jour de la vue.

2.1 Hiérarchie basée SLIC

Le principe de la hiérarchie est, à partir d'une segmentation initiale en un grand nombre de superpixels, de fusionner chaque superpixel les plus proches un à un, jusqu'à ce que tous les superpixels soient regroupés en une seule région.

2.1.1 Crédit de la hiérarchie

Pour cela, nous avons implémenté la fonction `createHierarchy`, qui va se baser sur la carte de superpixels (de taille $h \times w$, les dimensions de l'image en entrée) fournie par la fonction `generateSuperpixels` pour un nombre de superpixels N donné. Cette fonction `createHierarchy` calcule pour chaque superpixel de l'image sa distance avec ses superpixels adjacents, en se basant sur une distance couleur (dans l'espace *Lab*) et une distance spatiale (ici, la distance euclidienne entre les barycentres). Les deux superpixels les plus proches sont ensuite fusionnés en un seul superpixel, puis les distances avec le voisinage sont actualisées et une nouvelle fusion entre deux superpixels est effectuée, jusqu'à ce que tous les superpixels soient fusionnés. On dispose alors d'une matrice de taille $(N \times N)$ rassemblant chaque niveau de segmentation à partir de ce groupement hiérarchique.

Par exemple, avec une carte de 10 superpixels (labélisés de 1 à 10) :

$$\begin{array}{ll} \text{ligne 1} & \left[\begin{matrix} 1 & 2 & 3 & \dots & 9 & 10 \end{matrix} \right] \\ \text{ligne 2} & \left[\begin{matrix} 1 & 3 & 3 & \dots & 9 & 10 \end{matrix} \right] \text{ Fusion des superpixels 2 et 3} \\ \text{ligne 3} & \left[\begin{matrix} 1 & 3 & 3 & \dots & 1 & 10 \end{matrix} \right] \text{ Fusion des superpixels 1 et 9} \\ \dots & \dots \\ \text{ligne 9} & \left[\begin{matrix} 4 & 4 & 4 & \dots & 4 & 10 \end{matrix} \right] \\ \text{ligne 10} & \left[\begin{matrix} 4 & 4 & 4 & \dots & 4 & 4 \end{matrix} \right] \text{ Un seul superpixel restant} \end{array}$$

À partir de cette matrice, il est possible de récupérer une segmentation à un nombre de superpixels donné, en accédant à la ligne correspondante, sachant que la ligne 1 de la matrice correspond à N superpixels affichés, et la ligne N correspond à un seul superpixel. Dans le logiciel, nous avons directement relié ces niveaux de segmentation avec les niveaux de zoom ; un zoom $\times 2$ sur l'image va afficher le niveau de segmentation représentant deux fois plus de superpixels.

Par exemple, avec une segmentation initiale de 1000 superpixels, si nous avions un affichage contenant 300 superpixels (ligne 700 de la matrice), un zoom $\times 2$ va désormais afficher 600 superpixels (ligne 400 de la matrice), de manière à afficher une représentation plus précise de la partie zoomée.

L'intérêt d'une telle représentation est donc de pouvoir rassembler tous les calculs au lancement du logiciel (segmentation SLIC puis groupement ascendant), ce qui permet de naviguer entre les différents niveaux de segmentation sans aucun temps de calcul (seulement un accès à une ligne de la matrice, contrairement à l'implémentation précédente qui nécessitait une re-segmentation de l'image avec l'algorithme SLIC entre chaque zoom), mais également de simplifier la sélection de superpixels à différents niveaux, car chaque grande région est composée de superpixels initiaux.

De plus, cette hiérarchie de superpixels permet de connaître les niveaux de fusion entre chaque superpixel, et semble donc compatible avec une sélection par pose de marqueurs binaires (appelés *scribble*).

2.1.2 Calcul de la carte de saillance

Pour permettre un affichage adapté à la segmentation hiérarchique, nous avons implémenté la fonction `createSaliency`, qui calcule une carte de saillance (appelée *saliency map*) dépendant des niveaux de fusion des superpixels. Pour cela, à partir de la dernière ligne de la matrice hiérarchique ($N \times N$), nous remontons les niveaux de segmentation en dessinant à chaque fois les contours des superpixels plus ou moins foncé en fonction de leur taille. Les contours des superpixels contenant beaucoup de superpixels initiaux seront donc d'abord dessinés en noir, puis les contours des superpixels de taille moyenne seront dessinés par-dessus en gris foncé, et enfin les contours des superpixels initiaux situés en haut de la matrice, en gris clair.

Le logiciel affiche donc désormais à gauche les superpixels correspondants au niveau de segmentation actuel, et à droite la carte de saillance (cf Figure 2.1), qui est indépendante du niveau de segmentation, car calculée pour toute la hiérarchie au démarrage du logiciel ou au changement du paramètre de compacité de la segmentation initiale (Slider *Weight*).

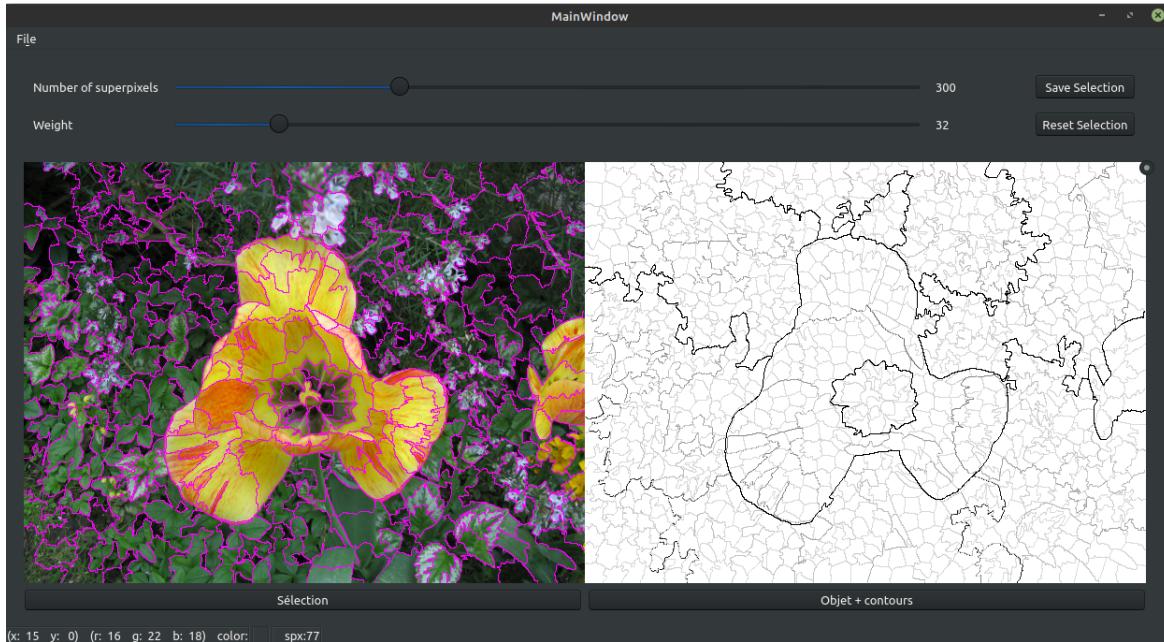


FIGURE 2.1 – Carte de saillance des superpixels (à droite), 1000 superpixels initiaux, `flower.jpg`

Cette représentation permet d'avoir un aperçu global de la hiérarchie de superpixels, et donc de savoir si les paramètres actuels (nombre de superpixels initiaux et compacité) fournissent une segmen-

tation précise de l'objet souhaité.

2.1.3 Limites

Bien qu'elle apporte des fonctionnalités intéressantes par rapport à l'implémentation précédente, cette approche hiérarchie de superpixels basée SLIC présente des inconvénients majeurs, notamment au niveau de la dépendance des paramètres. En effet, il est nécessaire de fixer un nombre de superpixels initial pour la segmentation SLIC sur laquelle va se baser la hiérarchie. Ces superpixels initiaux vont définir la précision maximale de la segmentation de l'image ; idéalement, il faudrait donc que ce nombre initial soit très élevé pour que les superpixels puissent capturer tous les détails de l'image, ce qui implique un temps de calcul très important au lancement du logiciel, et à chaque variation du paramètre de compacité.

Par exemple, avec une implémentation C++ et interface Qt, une hiérarchie de superpixels basée sur 1000 superpixels initiaux implique un temps de calcul d'environ 3 secondes sur une image 400×600 , tandis qu'une hiérarchie basée sur 2000 superpixels initiaux apportent de bien meilleurs résultats mais un temps de calcul d'environ 17 secondes sur la même image.

Ce compromis entre temps de calcul et précision de la segmentation n'est pas souhaitable ; pour y remédier, nous avons travaillé sur une autre version basée sur l'état de l'art actuel, qui ne dépend plus d'une segmentation initiale SLIC.

2.2 Hiérarchie rapide basée graphe

La recherche de l'état de l'art a notamment fait ressortir qu'un article récent (Wei, 2016 [12]) propose un calcul rapide de hiérarchie de superpixels qui s'abstient de l'utilisation de SLIC.

La méthode repose sur la construction d'un graphe, les sommets étant les pixels de l'image, et les arêtes le lien entre les pixels voisins. En se basant sur la méthode d'arbre couvrant minimum (*Minimum spanning tree*), les plus proches voisins du graphe sont fusionnés un à un jusqu'à ce qu'il ne reste qu'un seul sommet.

Pour permettre une fusion rapide des voisins du graphe, les auteurs ont utilisé un système de contraction des arêtes (*edge contraction*), illustré dans l'exemple de la Figure 2.2.

Dans cet exemple, les sommets représentent les labels des superpixels, et les arêtes les poids entre les superpixels voisins, basés sur la distance des histogrammes RGB et la confiance entre les arêtes (*edge confidence*, une distance entre les noeuds du graphes, qui permet de s'abstraire de la distance euclidienne dépendante des barycentres). Une contraction d'arêtes est effectuée entre les sommets 2 et 4 (a), ce qui va créer une boucle (ligne verte) sur le superpixel fusionné, ainsi que deux arêtes parallèles (lignes rouge et bleue) (b). Une opération d'aplatissement est alors effectuée, pour supprimer la boucle et remplacer les arêtes parallèles par la plus faible des deux (c). Enfin, la dernière étape consiste à mettre à jour les poids des arêtes du graphe (lignes rouges), en se basant sur les distances couleur et la confiance entre les arêtes (d).

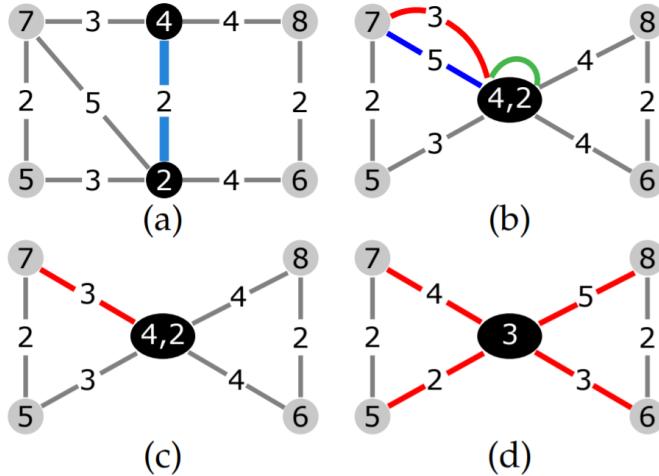


FIGURE 2.2 – Exemple de contraction d'arêtes dans un graphe [12] © 2018 IEEE

2.2.1 SuperpixelHierarchy

Le code de la hiérarchie rapide basée graphe étant disponible en ligne¹, nous avons pu implémenter cette méthode dans le logiciel, en remplaçant la structure **SLIC** par la nouvelle structure **SuperpixelHierarchy**. La hiérarchie s'articule principalement autour de deux fonctions. La fonction **buildHierarchy** initialise le graphe à partir de l'image et des paramètres et construit la hiérarchie avec des contractions d'arêtes. Les données de cette hiérarchie étant trop volumineuses pour être stockées dans une simple matrice, nous utilisons la fonction **getSuperpixels** qui permet de récupérer un niveau de superpixels précis à partir du graphe.

2.2.2 Correction des paramètres

Après avoir implémenté cette version, il s'est avéré que le paramètre de compacité, présent dans la structure **SuperpixelHierarchy** et relié aux sliders du logiciel, n'avait pas le comportement souhaité ; seules deux valeurs de compacité étaient possibles, contrairement à la version SLIC précédente où nous pouvions faire varier la compacité entre 0 et 100. Après avoir contacté l'auteur de l'article et du code

¹<https://github.com/xingxjtu/SuperpixelHierarchy>

disponible, celui-ci nous a confirmé que la version présentée dans l'article et mise en ligne ne permettait pas de faire varier la compacité des superpixels. Une deuxième version nous a été transmise, qui cette fois permet de choisir en entrée les paramètres de poids des arêtes (donc le compromis entre distance couleur et confiance des arêtes), et de compacité (qui va définir la forme plus ou moins carrée des superpixels).

2.2.3 Ajout de la détection de contours

Une particularité intéressante de cette version hiérarchique est qu'il est possible de fournir en entrée, en plus de l'image à segmenter, les contours de cette image, fournis par la vérité terrain ou par un algorithme de détection de contours (cf Figure 2.3), pour améliorer la segmentation de la hiérarchie.



(a) Image en entrée, `banana.bmp`



(b) Image de contours correspondante

FIGURE 2.3 – Détection des contours avec la méthode *Holistically-Nested Edge Detection*, 2015 [13]

La structure `SuperpixelHierarchy` va alors se baser sur ces contours de l'image pour initialiser les régions de la hiérarchie, ce qui va permettre d'améliorer la précision et la fusion des superpixels. Dans notre logiciel, à l'ouverture de l'image, nous avons ajouté un calcul de contours basé sur une implémentation PyTorch de l'article Holistically-Nested Edge Detection [13] disponible sur GitHub [14].

Pour cela, un appel à la fonction `QProcess` du framework Qt est nécessaire, pour faire le lien entre la méthode `pytorch-hed` (en Python), et le logiciel (en C++). Cette fonction récupère en entrée l'image à segmenter, et retourne en sortie la matrice de contours, qui est utilisée par la suite dans la méthode `buildGraph` de la structure `SuperpixelHierarchy` pour améliorer la segmentation en superpixels hiérarchiques.

En conclusion, la version `SuperpixelHierarchy` fournit une segmentation plus précise que la version précédente basée SLIC, notamment car elle ne dépend pas d'un nombre de superpixels initial et se base sur une information de contour supplémentaire. De plus, le temps de calcul de cette version hiérarchique basée graphe est inférieur à la version SLIC grâce aux méthodes de contraction d'arêtes qui permettent une fusion rapide des superpixels. En moyenne, sur une image de taille 400×600 , la détection de contours est réalisée en environ 2 secondes, et la construction de la hiérarchie en environ 0.5 secondes, ce qui est largement inférieur à la version SLIC initialisée à 2000 superpixels (environ 17 secondes).

2.3 Pose de labels et marqueurs binaires

Maintenant que nous avons une segmentation hiérarchique précise et rapide quelle que soit l'image en entrée, l'objectif est d'implémenter une méthode de pose de labels avec des marqueurs binaires (objet ou fond) qui permet de détourer l'objet souhaité de manière interactive et rapide.

2.3.1 Bibliothèque Higra

Dans un premier temps, nous avons observé le fonctionnement de l'outil de Benjamin Perret [7], dépendant d'une méthode basique de bassin couvrant (*watershed*) qui peut présenter des résultats irréguliers, mais également basé sur la bibliothèque **Higra** [8], qui utilise des marqueurs binaires (respectivement en vert et en rouge) pour segmenter rapidement l'image en deux parties (respectivement objet et fond). À partir d'une segmentation hiérarchique basée forêt couvrante minimum (*minimum spanning forest*), l'algorithme de labélisation binaire utilisant les marqueurs est le suivant :

Algorithm 1 Labélisation binaire à partir de marqueurs

Require: Tree, object markers, background markers

```
for i in leaves(tree) do                                ▷ Parcours des feuilles de l'arbre pour les labéliser
    if (background_marker(i)) attr(i)=1;                  ▷ Initialisation des structures
    else if (object_marker(i)) attr(i)=2;
end for

for i in leaves_to_root(tree) do                         ▷ Parcours ascendant pour faire remonter les marqueurs
    for c in children(i, tree) do
        attr(i) |= attr(c);                            ▷ Si les deux fils ont des labels différents, le noeud parent sera
                                                       considéré comme étant un fond
    end for
end for

if (attr(root(tree) == 0) attr(root(tree)) = 1;          ▷ Si les marqueurs sont vides, racine=fond

for i in root_to_leaves(tree) do                         ▷ Parcours descendant pour labéliser les noeuds indéterminés
    if (attr(i) == 0)
        if (attr(parent(i,tree)) == 2) attr(i)=2;
        else attr(i)=1;
    end for
```

En résumé, les labels des marqueurs sont remontés à partir des feuilles jusqu'à la racine, puis les labels sont descendus à partir de la racine jusqu'aux feuilles pour labéliser les feuilles dépourvues de marqueurs initiaux.

À la suite de cet algorithme, toutes les feuilles de l'arbre sont labélisées comme faisant partie de l'objet (`attr(i) = 2`) ou du fond (`attr(i) = 1`). Il suffit donc de sélectionner toutes les feuilles objet pour obtenir une segmentation binaire de l'image.

2.3.2 Lien graphe-matrice

Dans notre cas, **SuperpixelHierarchy** est basée sur un graphe pour la construction de la hiérarchie de superpixels, mais une fois que cette hiérarchie est terminée, la seule information à laquelle nous pouvons accéder simplement est un niveau précis de superpixels, en utilisant la fonction `getSuperpixels`. Pour pallier cela, nous avons créé une matrice d'adjacence `_hierarchyClusters` qui, à la création de la hiérarchie, stocke différents niveaux de superpixels pour représenter la hiérarchie de façon globale. Après différents tests sur le temps de calcul et la précision de la segmentation, nous avons récupéré une dizaine de niveaux allant de 4000 superpixels à 1 superpixel de la manière suivante :

```

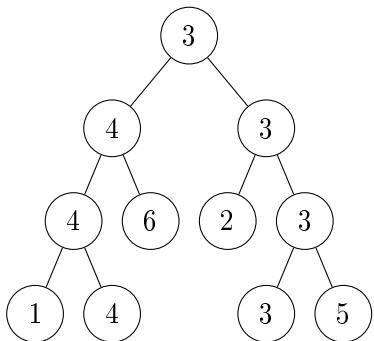
n = 4000 ;
while (n > 0) do
    _hierarchyClusters.add(getSuperpixels(tree,n)) ;
    n = n/2 ;
end while

```

Ce qui permet de récupérer les niveaux [4000, 2000, 1000, ..., 4, 2, 1].

Le problème est que l'algorithme de labélisation binaire de la bibliothèque `Higra` ne fonctionne que pour une structure en graphe, et nous n'avons accès qu'à la matrice `_hierarchyClusters`, de taille finale (`_nbClusters`, 10), `_nbClusters` étant ici le nombre de pixels dans l'image.

Le fait est que les structures de graphe et de matrice possèdent plusieurs caractéristiques communes. Par exemple, dans la figure 2.4, une hiérarchie de superpixels est représentée sous forme de graphe et également sous forme de matrice d'adjacence.



(a) Représentation sous forme de graphe

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 2 & 3 & 4 & 5 & 6 \\ 4 & 2 & 3 & 4 & 3 & 6 \\ 4 & 2 & 3 & 4 & 3 & 4 \\ 4 & 3 & 3 & 4 & 3 & 4 \\ 3 & 3 & 3 & 3 & 3 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 2 & 3 & 4 & 5 & 6 \\ 4 & 2 & 3 & 4 & 3 & 6 \\ 4 & 2 & 3 & 4 & 3 & 4 \\ 4 & 3 & 3 & 4 & 3 & 4 \\ 3 & 3 & 3 & 3 & 3 & 3 \end{bmatrix}$$

(b) Représentation sous forme de matrice d'adjacence

FIGURE 2.4 – Exemple de plusieurs représentations possibles d'une hiérarchie avec 6 labels

À partir de ces similarités graphe-matrice, nous avons adapté l'algorithme de binarisation binaire pour qu'il puisse fonctionner d'une manière identique mais avec la matrice `_hierarchyClusters`.

Dans un premier temps, les matrices `<object>` et `<background>` stockent les labels des superpixels touchés par des marqueurs binaires, placés par l'utilisateur ; un cliquer-glisser du bouton gauche de la souris dessine un trait bleu correspondant à l'objet, et un cliquer-glisser du bouton droit de la souris dessine un trait rouge correspondant au fond.

Ensuite, la fonction `binaryLabelisation` détermine, pour chaque superpixel SP_i dans `<object>`, le niveau de fusion minimum (noté $minFusion$) avec les superpixels SP_j de `<background>`. Ce niveau de fusion, calculé avec la fonction `levelOfFusion(SP_a, SP_b)`, correspond au niveau de la hiérarchie où les superpixels SP_a et SP_b ont fusionné. Il suffit d'ajouter à la sélection tous les superpixels initiaux contenus dans le superpixel actuel SP_i de `<object>` au niveau $minFusion - 1$, de sorte à récupérer les superpixels de l'objet avant la fusion avec les superpixels du fond. Une fois tous les superpixels de `<object>` parcourus, les superpixels sélectionnés correspondent à l'objet à détourer, et les superpixels non-sélectionnés correspondent au fond de l'image.

D'un point de vue de graphe, ce niveau de fusion minimum est l'équivalent, pour chaque feuille-objet, du plus haut noeud-objet dans l'arbre, car c'est précisément ce noeud qui va permettre de labéliser en objet les feuilles indéterminées, lors de l'étape de parcours descendant.

2.3.3 Fonctionnalités ajoutées au logiciel

Pour permettre la pose de label, nous avons séparé le logiciel en 2 onglets *Sélection* et *Scribble*, le premier permettant une sélection manuelle des superpixels, et le deuxième permettant de dessiner des marqueurs de couleurs, représentant l'objet ou le fond, et appliquer directement une segmentation binaire sur l'image.

Des modifications sur les sliders ont également été apportées, pour permettre de séparer les paramètres de poids couleurs-distance et le paramètre de compacité. De plus, nous avons ajouté la possibilité de définir le seuil maximal du slider de nombre de superpixels directement depuis l'interface, ce qui permet

à l'utilisateur de définir la précision de la segmentation (entre 1 et 10 000 superpixels).

Enfin, la carte de saillance, présentée dans la section 2.1, apportait un visuel intéressant sur la segmentation hiérarchique globale. Pour l'ajouter avec la nouvelle hiérarchie basée graphe, nous avons utilisé la matrice `_hierarchyClusters`, en dessinant chaque contour des superpixels de la matrice avec un niveau de gris différent. Le premier niveau de la matrice contenant 4000 superpixels sera donc dessiné en gris clair, puis le deuxième niveau contenant 2000 superpixels en gris, etc jusqu'à l'avant-dernier niveau de la matrice contenant 2 superpixels qui sera dessiné en gris foncé (le dernier niveau contenant un seul supepixel n'est pas dessiné, car il englobe toute l'image et n'apporte donc aucune information supplémentaire).

En conclusion, après avoir étendu SLIC à une hiérarchie de superpixels, nous avons adapté une nouvelle méthode hiérarchique plus rapide au logiciel Qt, en corrigeant les problèmes d'implémentation et en ajoutant un lien avec la gestion de paramètres déjà présente dans le logiciel. Enfin, nous avons adapté les algorithmes de segmentation binaire de l'outil de B. Perret, qui étaient jusque-là basés sur une méthode de bassin couvrant (*watershed*) peu régulière. Un exemple de la nouvelle interface est disponible en Figure 2.5.

2.4 Résultats et tests

Après avoir implémenté la pose de labels permettant une segmentation binaire de l'image, nous avons réalisé différents tests sur l'interface avec plusieurs images. Dans certains cas, des "points" de couleur posés par l'utilisateur sont suffisants pour obtenir une segmentation précise de l'objet (cf Figure 2.5).

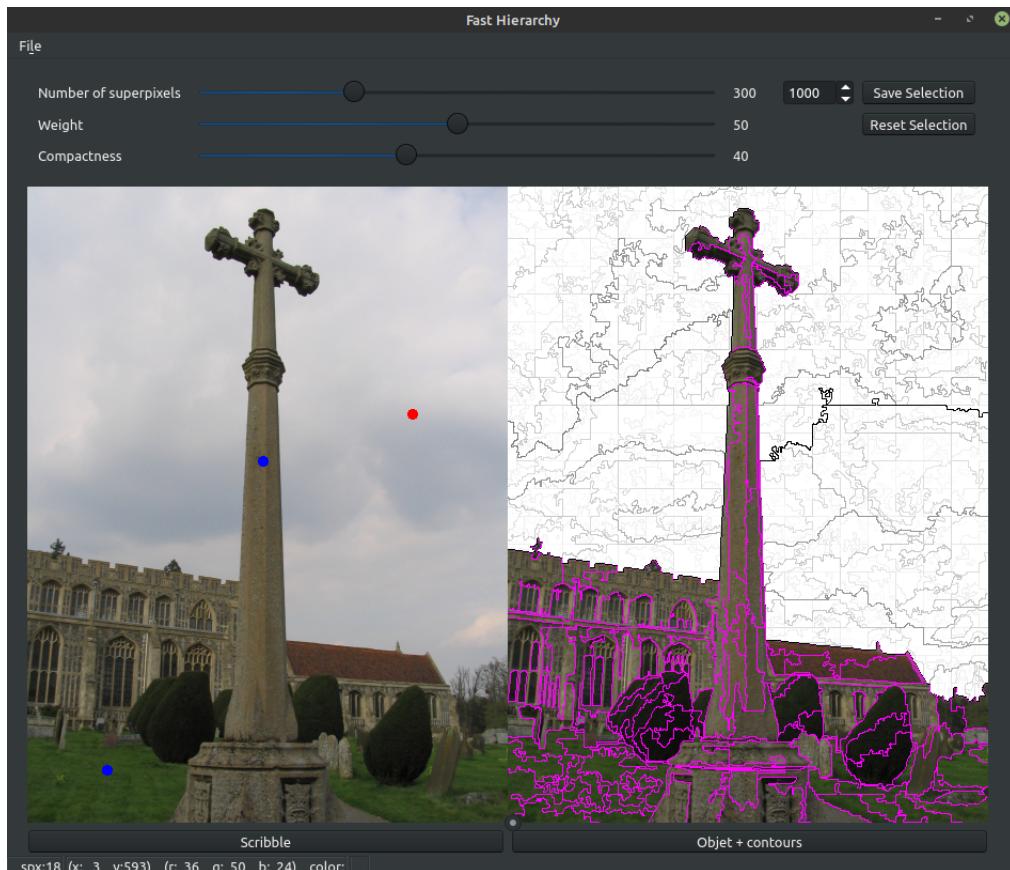


FIGURE 2.5 – Exemple de segmentation binaire avec pose de labels sur l'image `cross.png`. L'utilisateur place des marqueurs binaires sur l'image de gauche, et la segmentation binaire résultante est affichée sur la partie de droite, en plus de la carte de saillance. Dans cet exemple, quasiment l'intégralité du fond a été retiré, en seulement trois clics.

En faisant varier les paramètres des superpixels, les segmentations obtenues à partir des traits ou des points de pinceaux manuels sont globalement satisfaisantes. La pose de labels étant une interaction subjective, nous avons cherché à mettre en place une méthode quantitative pour évaluer efficacement la précision de notre méthode de segmentation binaire.

2.4.1 Pipeline de test Python

Pour permettre la mise en place de tests sur les marqueurs binaires et la segmentation obtenue, nous avons implémenté une version du logiciel sans interface graphique, exécutable en ligne de commande en spécifiant en entrée l'image à segmenter, les paramètres des superpixels et les matrices `<object>` et `<background>` contenant les marqueurs binaires à utiliser. L'image binaire résultante est alors disponible en sortie de l'algorithme.

À partir de cette version sans interface graphique, nous avons développé un pipeline Python permettant de comparer la segmentation binaire en sortie avec la vérité terrain (cf Figure 2.6).

Cette comparaison est réalisée à partir du coefficient de Dice, qui permet de mesurer la similarité entre deux échantillons discrets X et Y , selon la formule suivante :

$$s = \frac{2 \times |X \cap Y|}{|X| + |Y|}$$

$|X \cap Y|$ correspond au nombre de vrais positifs (dans notre cas, le nombre de pixels ayant été labélisés correctement en objet ou en fond) et $|X|$ et $|Y|$ au nombre d'éléments dans chaque échantillon (ici, cela correspond au nombre total de pixels dans l'objet).



FIGURE 2.6 – Comparaison entre la vérité terrain et la segmentation binaire en sortie de l'algorithme, `cross.jpg`; l'image en sortie c) est obtenue après la pose de marqueurs sur l'image a) (ici, en trois clics utilisateurs), l'image d) correspond à la binarisation de l'image c). Sur cet exemple, le score Dice entre les images b) et d) est égal à 98,56%.

À partir de ce pipeline de tests, une première idée pour tester notre algorithme de segmentation était de générer aléatoirement un nombre plus ou moins important de marqueurs situés dans l'objet et dans le fond, en se basant sur la vérité terrain de plusieurs images de tests. Ces résultats, disponibles en Table 2.1, permettent de montrer les limites de notre approche, mais ne représentent pas les résultats obtenus dans le cas où un utilisateur applique ces marqueurs.

TABLE 2.1 – Comparaison des scores Dice en fonction du nombre de marqueurs aléatoires, placés respectivement dans l'objet et dans le fond à partir de la vérité terrain, `banana.bmp`

Nombre de marqueurs					
1	5	20	50	200	1000
51.43	68.23	82.37	86.12	91.88	94.71

Après une recherche de l'état de l'art plus poussée sur cet aspect, il s'est avéré que les articles GrabCut [4] et Geodesic Star [6] fournissent des jeux de données permettant de mesurer la précision de nos segmentations. Ces datasets contiennent des images de tests, la vérité terrain de segmentation binaire pour ces images, ainsi que des marqueurs objet et fond annotés manuellement.

Pour établir un score à partir de ces jeux de données, nous parcourons chaque image contenant les marqueurs, et procédons à l'extraction des pixels correspondants au label fond et objet dans les matrices <object> et <background>, que nous envoyons en entrée de l'algorithme de segmentation binaire avec l'image initiale, et les paramètres des superpixels. Dans un premier temps, nous avons fait varier ces paramètres pour déterminer ceux qui optimisaient le score de segmentation (cf Figure 2.7).

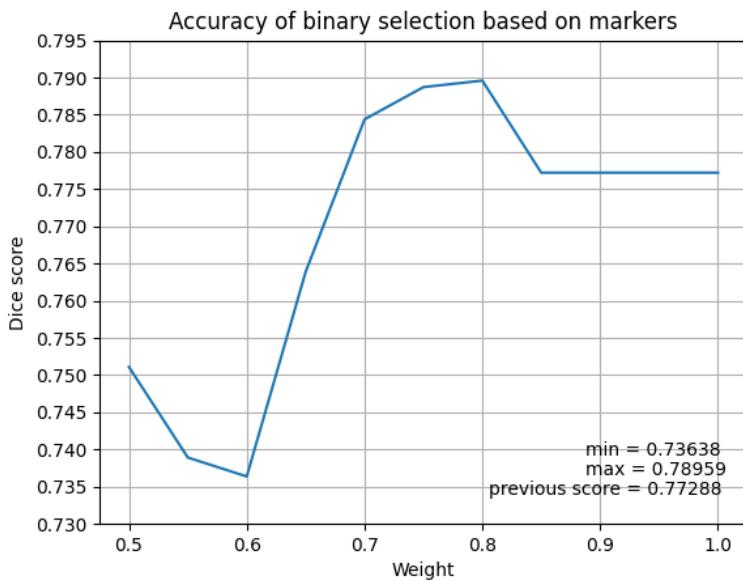


FIGURE 2.7 – Score Dice en fonction du poids distance-couleur des superpixels, dataset Geodesic [6]

À partir de ces résultats, nous en avons déduit que la segmentation est optimale lorsque le poids entre la distance couleur et la distance spatiale est fixé à 0.8.

Récemment, des méthodes de génération de marqueurs précis ont été développées, en se basant sur le squelette des objets [15]. Ces marqueurs sont notamment disponibles pour les images du jeu de données GrabCut ², et fournissent des informations supplémentaires en comparaison avec les marqueurs originels (cf Figure 2.8).

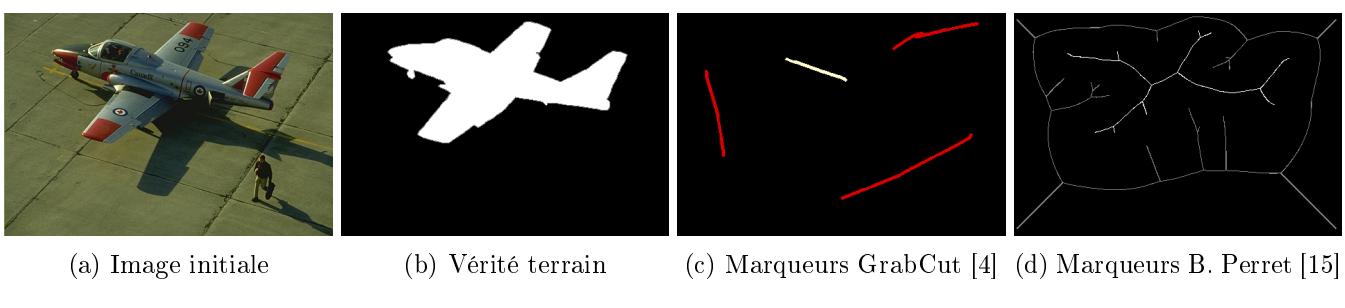


FIGURE 2.8 – Comparaison visuelle entre différents marqueurs.

²<https://perso.esiee.fr/~perretb/markerdb/>, dernier accès le 27/08/2020

Après avoir testé ces différents types de marqueurs sur le dataset GrabCut avec les paramètres optimisés précédemment, il en résulte que les marqueurs de B. Perret basés sur le squelette de l'objet permettent d'augmenter drastiquement la précision de notre algorithme de segmentation binaire (cf Table 2.2).

TABLE 2.2 – Comparaison des scores Dice en fonction des marqueurs, dataset GrabCut [4]

Marqueurs	Min	Max	Moyenne
GrabCut [4]	51.96	99.48	86.10
Squelette [15]	81.66	99.48	95.66

2.4.2 Conclusion

Nous avons intégré à notre logiciel Qt un outil de sélection manuelle rapide et intuitif, qui permet de sélectionner des objets déconnectés en plusieurs parties dans l'image (ce qui est impossible avec la méthode GrabCut par exemple, qui ne permet la sélection que d'un seul objet). Cet outil permet également un raffinement précis de la sélection avec un zoom relié à la descente de la hiérarchie de superpixels. Pour l'instant seuls des marqueurs binaires ont été implémentés, une amélioration possible pourrait donc être l'implémentation de marqueurs multi-labels, favorisant la sélection de plusieurs objets indépendants.

Notre algorithme de segmentation binaire basée hiérarchie de superpixels permet d'obtenir une précision avoisinant les 86% (score Dice) de façon rapide et semi-automatique, avec une pose de label reposant sur quelques traits objet et fond fournis par l'utilisateur. En se basant sur des marqueurs précis (e.g. marqueurs squelette B. Perret [15]), le score de notre algorithme dépasse les 95%. Ces résultats ne permettent pas de surpasser les algorithmes actuels de segmentation (notamment les algorithmes d'alpha-matting reposant sur des trimaps), mais constituent une base solide pour la segmentation interactive rapide.

En associant cette segmentation binaire avec des opérations de dilatation/érosion (cf Section 4.1.2) ou avec la sous-représentation duale que nous allons étudier, notre méthode pourrait par exemple être utile dans le cadre d'une génération de trimap semi-automatique.

Chapitre 3

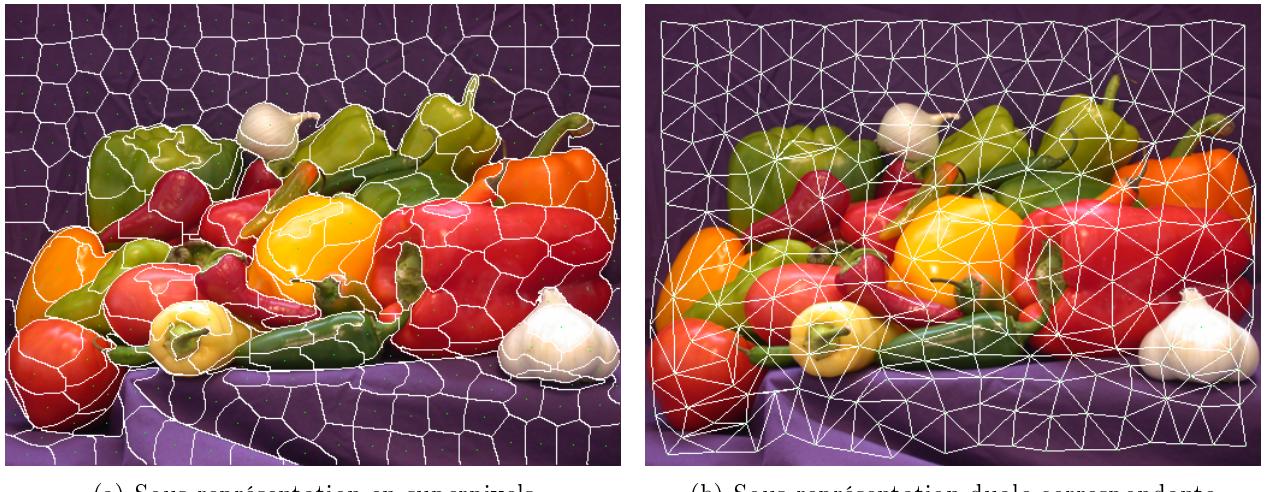
Sous-représentations duales et DSPM

La deuxième partie de ce stage est consacrée à l'étude de sous-représentations duales irrégulières. Celles-ci ont pour objectif de capturer efficacement l'information de contour disponible aux frontières des superpixels, généralement ignorée par les modèles ne considérant que l'information à l'intérieur de chaque superpixel.

Cette partie du stage a principalement été réalisée en MATLAB, en se basant sur le code de la méthode DSPM [1], fourni par mes encadrants. Dans un premier temps, des étapes de lecture, compréhension et prise en main du code ont donc été nécessaires avant de réfléchir aux sous-représentations duales.

3.1 Décomposition duale basée barycentres

La première idée de sous-représentation duale était de relier les barycentres des superpixels adjacents, pour former une carte de superpixels duraux (cf Figure 3.1).



(a) Sous-représentation en superpixels

(b) Sous-représentation duale correspondante

FIGURE 3.1 – Décomposition en superpixels et sa décomposition duale correspondante, `peppers.png`

L'intérêt d'une telle représentation est qu'elle permet de capturer rapidement et efficacement les contours des superpixels. Nous avons donc d'une part l'information intra-superpixels (en utilisant par exemple un histogramme RGB sur les superpixels), et d'autre part l'information inter-superpixels (en utilisant des descripteurs de contours sur ces nouveaux superpixels duraux formés).

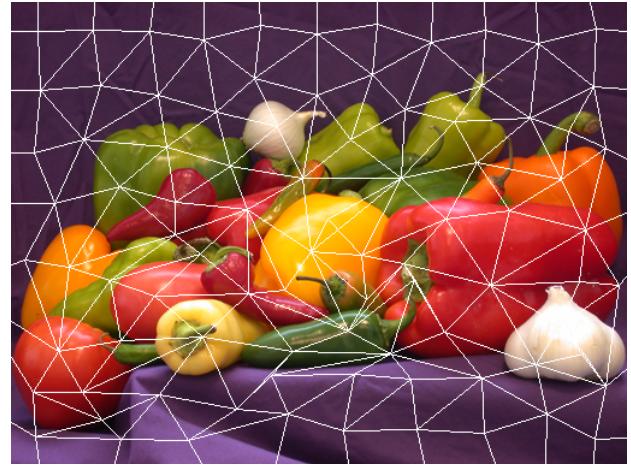
À partir de cette représentation initiale, nous avons réfléchi à plusieurs paramétrages pour optimiser la carte de superpixels duraux, tels que l'ajout d'une connexion aux bords de l'image, un seuil d'adjacence sur le voisinage, différentes méthodes de chemins entre les barycentres, et la définition d'un point central idéal pour les formes irrégulières de superpixels.

3.1.1 Connexion aux bords de l'image

Pour permettre une décomposition sur toute l'image, il était nécessaire d'ajouter une connexion aux bords, qui est inexistante si l'on relie simplement les barycentres entre eux. Pour y remédier, nous avons implémenté la fonction `SP_mean_border`, qui va calculer, pour chaque superpixel de la bordure, le point se trouvant au centre des pixels aux bords. On obtient donc un point par superpixel de la bordure, et deux points pour les superpixels se trouvant sur les coins de l'image. Il suffit ensuite de relier ces nouveaux points aux barycentres des superpixels correspondants pour former une décomposition sur toute l'image (cf Figure 3.2).



(a) Calcul des centres des bordures



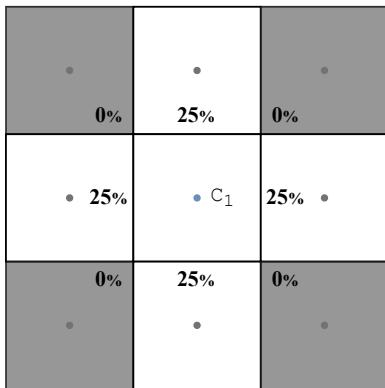
(b) Ajout à la décomposition duale

FIGURE 3.2 – Ajout de la connexion aux bords sur la représentation duale, `peppers.png`

3.1.2 Seuil d'adjacence sur le voisinage

Comme indiqué précédemment, les barycentres des superpixels *adjacents* sont reliés. Mais dans certains cas (notamment lorsque l'on diminue la valeur de la compacité), des superpixels peuvent avoir une adjacence très faible (de l'ordre de seulement quelques pixels voisins), et il serait donc préférable d'éviter de relier leurs barycentres, pour permettre d'avoir une représentation moins surchargée et plus robuste.

Ci-dessous en Figure 3.3, on peut observer différentes répartitions d'adjacence entre le superpixel central et ses voisins, en fonction de la compacité.



(a) Exemple de superpixels avec compacité élevée

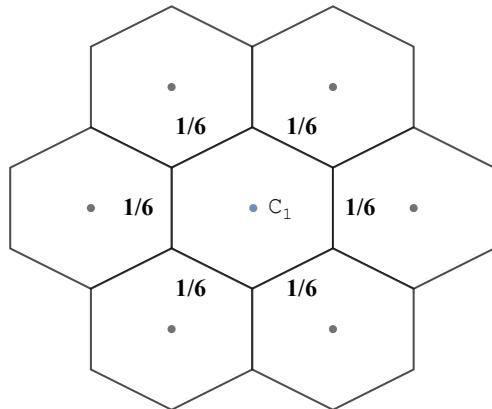


(b) Exemple de superpixels avec compacité faible

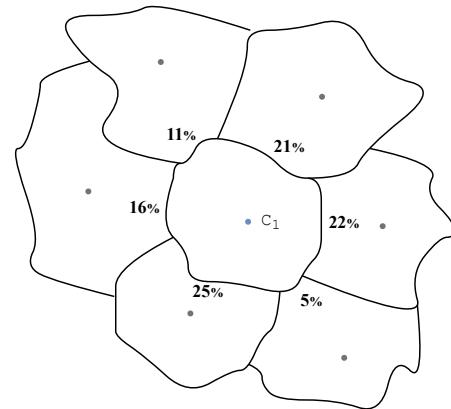
FIGURE 3.3 – Comparaison de l'adjacence des superpixels en fonction de la compacité, grille carrée

Dans la Figure 3.3a, les adjacences sont idéalement réparties, mais une compacité aussi élevée n'est pas utilisée en pratique car n'apporte pas assez d'information. La Figure 3.3b correspond à un cas

plausible d'utilisation des superpixels, et on peut y remarquer des disparités dans les adjacences. Pour pallier ces disparités, une première idée était d'initialiser les superpixels par une grille régulière en hexagones plutôt que par une grille en carrés, comme le montre la Figure 3.4.



(a) Exemple de superpixels avec compacité élevée

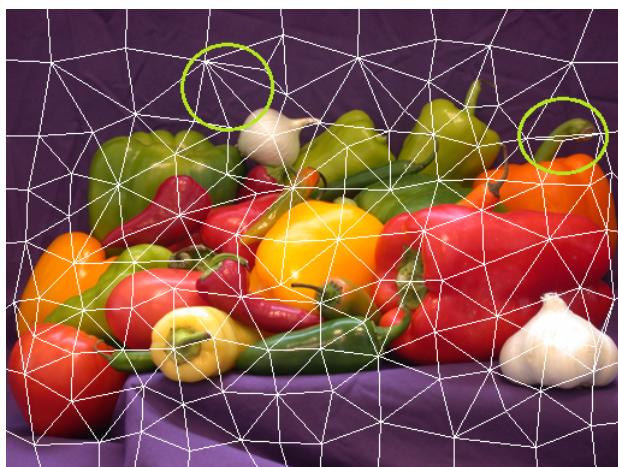


(b) Exemple de superpixels avec compacité faible

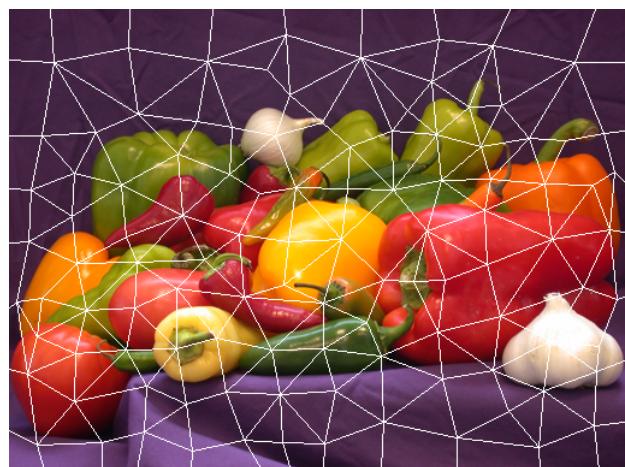
FIGURE 3.4 – Comparaison de l'adjacence des superpixels en fonction de la compacité, grille hexagonale

Mais en pratique, en utilisant une compacité assez faible (Figure 3.4b), des disparités vont se créer de façon identique à la grille carrée. Nous avons donc abandonné l'idée d'utiliser une grille hexagonale. La solution a donc été de mettre en place un seuil d'adjacence minimum pour relier les barycentres. Pour cela, nous calculons le nombre moyen de pixels frontière par superpixels, noté `mean_border_spx`. À partir de cette moyenne, on peut en déduire un pourcentage minimum qui va définir notre seuil d'adjacence. Après plusieurs tests, la valeur de 5% a été retenue ; c'est-à-dire que pour que les barycentres de deux superpixels adjacents soient reliés, il faut qu'ils aient un nombre de pixels voisins supérieur à 5% de `mean_border_spx`.

Dans la Figure 3.5, on peut observer l'impact du seuil d'adjacence sur la décomposition duale ; les cercles verts désignent les zones où l'information de contour est trop faible dans l'approche initiale, et qui ont été corrigées avec le seuil d'adjacence.



(a) Pas de seuil d'adjacence minimum



(b) Seuil d'adjacence à 5%

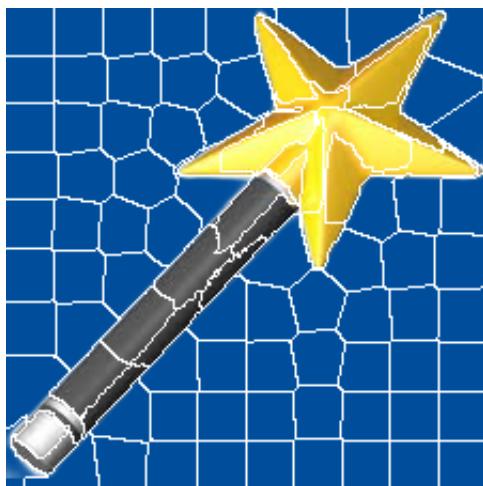
FIGURE 3.5 – Exemple de différents seuils d'adjacence pour la décomposition duale, `peppers.png`

3.1.3 Méthodes de chemin entre les barycentres

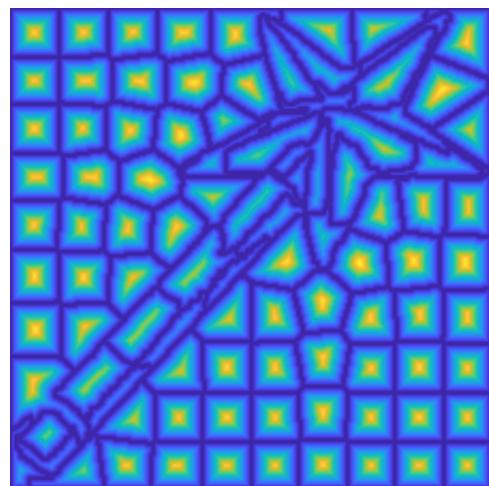
Initialement, le chemin entre les barycentres est tracé de façon linéaire (cf Figure 3.1). Notre support étant une image représentée par des pixels, l'algorithme de Bresenham [16] a été utilisé pour approximer les segments de droites entre les barycentres.

Par la suite, nous avons également réfléchi à une approche géodésique (i.e. plus court chemin) basée sur la carte de distance aux contours de chaque superpixel. Cette contrainte va permettre d'obtenir un chemin qui va s'adapter aux bords des superpixels, et va donc permettre de mieux capturer l'information de contour.

Le principe est de calculer, pour chaque pixel à l'intérieur d'un superpixel, la plus courte distance à un pixel en dehors du superpixel actuel. La Figure 3.6 représente cette carte de distance aux bords avec une échelle de couleur (où le jaune correspond à une distance aux bords élevée, et le bleu correspond à une distance aux bords faible).



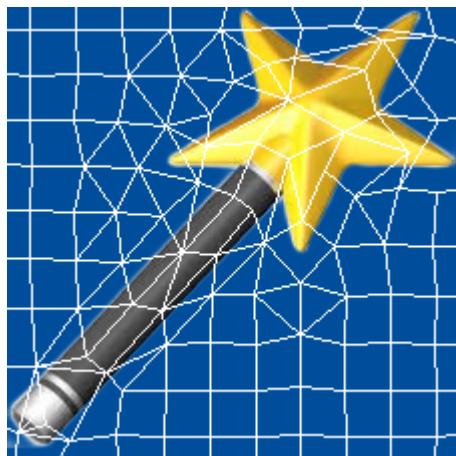
(a) Décomposition en ~ 100 superpixels



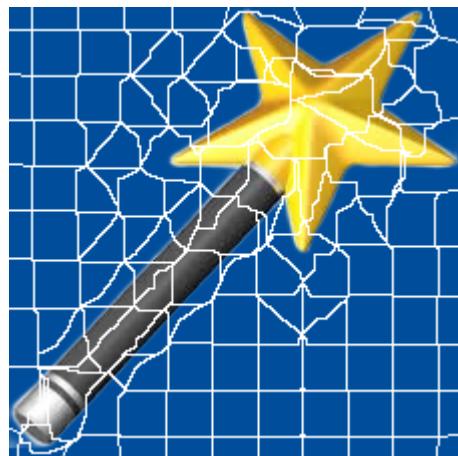
(b) Carte de distance aux contours

FIGURE 3.6 – Exemple de décomposition en superpixels et sa carte de distance correspondante, `star.png`

Nous avons ensuite utilisé une Toolbox MATLAB [17] pour établir le chemin le plus court en se basant sur cette carte de distance (cf Figure 3.7).



(a) Chemin linéaire (algorithme de Bresenham)



(b) Chemin géodésique basé distance aux bords

FIGURE 3.7 – Comparaison des méthodes de chemin entre les barycentres `star.png`

Cette méthode géodésique présente un temps de calcul supérieur à la méthode linéaire (en raison du calcul de la carte des distances et de la complexité du calcul du plus court chemin), mais présente certains intérêts topologiques ; par exemple, dans la Figure 3.7, on peut remarquer que la branche

en haut à gauche de l'étoile est traversée dans l'approche linéaire, alors qu'elle est contournée dans l'approche géodésique, ce qui permet d'obtenir des superpixels duals qui capturent plus efficacement l'information de contour.

3.1.4 Méthodes de calcul des barycentres

Notre approche se basant entièrement sur le point central des superpixels, il paraît important de bien définir le terme *central* et la façon de le calculer.

A ce jour, les articles utilisant le point central des superpixels se basent sur la moyenne des coordonnées des pixels d'un superpixel pour le calculer. On définit alors ce point comme le *barycentre* du superpixel. Or, les superpixels pouvant être des formes irrégulières, il se peut qu'en utilisant cette méthode le centre soit en dehors du superpixel, ou très proche des bords, notamment lorsque la forme est concave.

Dans notre cas, les superpixels sont *représentés* par leur point central (par exemple, dans l'approche linéaire, c'est la seule information que l'on utilise pour former la sous-représentation duale). Idéalement, il faudrait donc un point à l'intérieur de la forme, à la fois éloigné des bords, et à la fois proche de tous les points du superpixel (cf Figure 3.8).

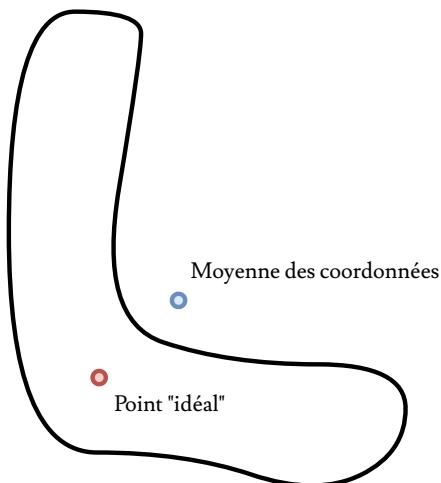


FIGURE 3.8 – Comparaison entre la moyenne des coordonnées et le point idéal d'un superpixel

Pour parvenir à calculer le centre "idéal" de chaque superpixel, nous avons mis en place un jeu de tests composé de plusieurs superpixels irréguliers ayant des formes différentes, sur lesquels nous avons testé plusieurs méthodes de calcul de centre.

pdist (*pairwise distance*) : Première méthode utilisée, correspond au point à l'intérieur du superpixel qui est le plus proche des autres points, c'est-à-dire le pixel minimisant la distance euclidienne avec tous les autres pixels d'un superpixel. Cette méthode assure d'obtenir un point à l'intérieur du superpixel, mais elle est très coûteuse et peut donner un point très proche des bords ou sur la frontière de la forme.

nemo : Nous avons ensuite pensé à utiliser la carte de distance, précédemment utilisée pour le chemin géodésique, pour obtenir le point le plus éloigné des bords ; le résultat peut sembler correct pour certains superpixels, mais pour la plupart des formes concaves (par exemple en forme de sablier), le résultat est incorrect ou indécis (car plusieurs points peuvent avoir la même distance aux bords).

Par la suite, nous nous sommes intéressés aux moments géométriques (historiquement utilisés pour la reconnaissance de caractères et pour la détection de l'orientation, cf [18]).

Dans une image binaire, les moments M d'ordre (p, q) sont définis tels que

$$M_{p,q} = \sum_{i,j \in Obj} i^p j^q.$$

Le moment d'ordre zéro $M_{0,0}$ correspond au nombre de pixels dans l'image. Les moments d'ordre *un* $M_{1,0}$ et $M_{0,1}$ permettent d'obtenir les coordonnées du barycentre, et les moments d'ordre *deux*

permettent d'obtenir les axes mineur et majeur de l'objet considéré.

1ère méthode moments : Chercher l'axe ayant la plus petite intersection avec l'objet, et définir le centre de l'intersection de cet axe en tant que centre du superpixel.

Par exemple, dans la Figure 3.9, l'axe A_1 possède une intersection de 12 pixels avec la forme, et l'axe A_2 une intersection de 40 pixels au total. Le nouveau centre (en rouge) est donc placé au centre de l'intersection de l'axe A_1 . L'ancien centre (obtenu par moyenne des coordonnées), correspond à l'intersection des axes (en bleu).

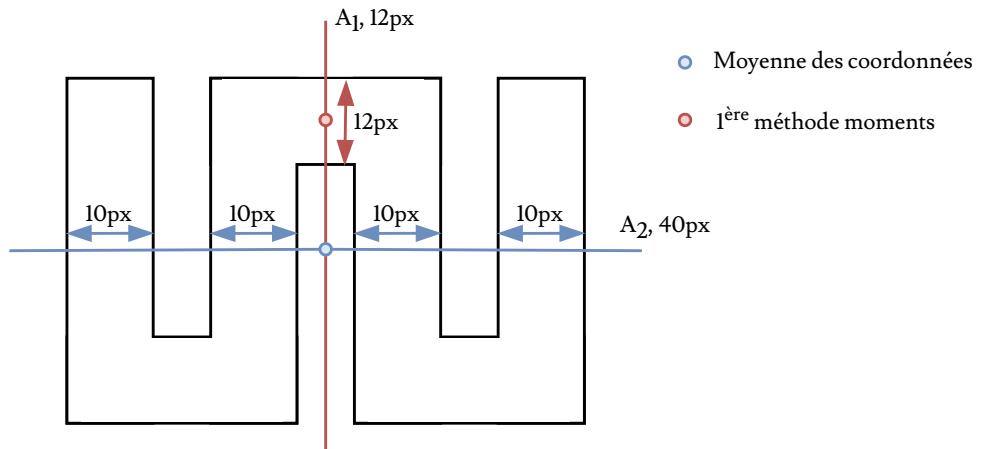


FIGURE 3.9 – Comparaison entre la première méthode des moments et la moyenne des coordonnées

Cette méthode apporte de très bons résultats, mais peut échouer dans certains cas de figure. Par exemple, dans une forme en U, si la base est large et les deux barres du U sont fines, il se peut que la plus petite intersection se trouve dans une des barres du U au lieu d'être dans la base, ce qui n'est visuellement pas satisfaisant (cf Figure 3.10). Pour pallier ce problème, une deuxième méthode a été appliquée, cette fois-ci en se basant sur la façon dont les axes découpent la forme.

2ème méthode moment : Chaque axe A_1 et A_2 coupe la forme en deux parties P_i et P_j . Si l'axe est bien défini, cela signifie que les surfaces S_i et S_j de ces deux parties sont quasiment égales (i.e. qu'elles ont le même nombre de pixels). On cherche donc à récupérer l'axe A_x définissant les parties ayant les surfaces les plus proches possibles, et on définit le centre de cet axe en tant que centre du superpixel.

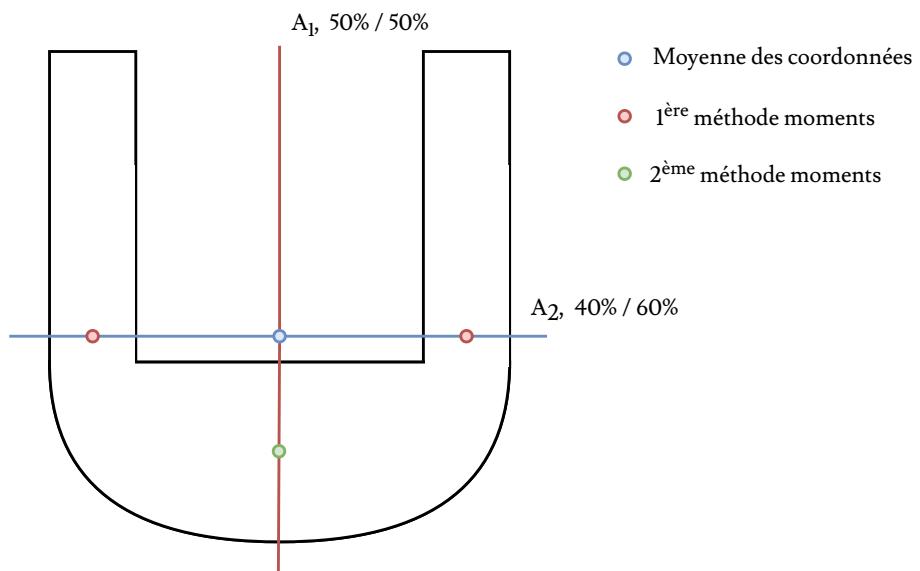


FIGURE 3.10 – Comparaison entre les différentes méthodes des moments et la moyenne des coordonnées

Par exemple, dans la Figure 3.10, l'axe A_1 découpe le superpixel en deux parties ayant une surface identique (ici, il correspond à un axe de symétrie), tandis que l'axe A_2 découpe la forme en deux parties ayant des surfaces différentes (les deux barres du U additionnées ont une surface inférieure à la base du U). Le point de la deuxième méthode va donc être placé au centre de l'intersection de l'axe A_1 .

Ces deux méthodes ont ensuite été fusionnées pour permettre d'obtenir de bons résultats peu importe la forme rencontrée, selon la condition suivante : si les surfaces S_i, S_j formées par chaque axe A_1 et A_2 sont similaires, alors on applique la 1ère méthode des moments, sinon, la 2ème méthode est appliquée pour trouver l'axe découpant la forme de façon optimale. Le point résultant de ces méthodes est nommé point *moments*.

mix : Enfin, le point central final, va correspondre au point maximisant la distance aux bords et minimisant la distance au point *moments* :

$$mix = \alpha \times (dist_{moments}) + (1 - \alpha) \times (1/dist_{borders}), \quad \alpha \in [0, 1]$$

Par exemple, dans la Figure 3.11, on observe une forme de superpixel relativement complexe. La moyenne des coordonnées est en dehors de la forme, le point *pdist* est trop proche des bords et le point *nemo* est éloigné des bords mais excentré par rapport à la forme globale. Le point *moments* est relativement bien placé, mais en ajoutant la pondération avec la carte de distance (ici, $\alpha = 0.5$), on obtient le point *mix*, qui semble être le point le plus proche du centre "idéal".

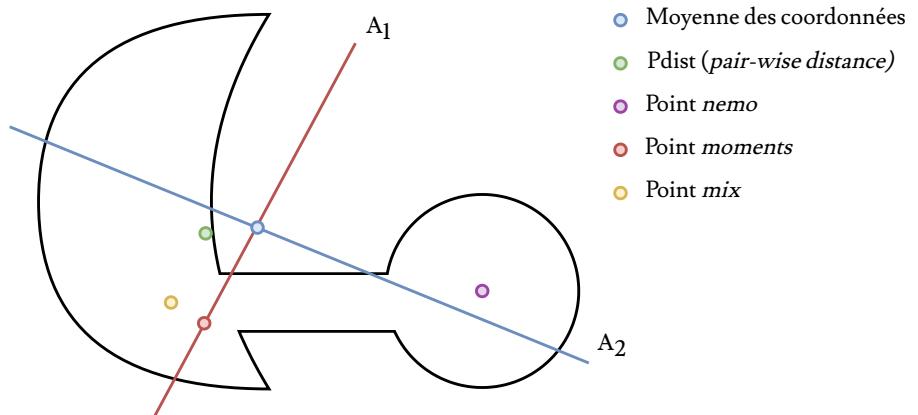


FIGURE 3.11 – Comparaison entre les différentes méthodes de point central sur un superpixel irrégulier

3.1.5 Conversion en carte de superpixels

Suite aux étapes précédentes, nous obtenons une nouvelle décomposition duale, disposant désormais d'une connexion aux bords, d'un seuil minimum d'adjacence, et basée sur une nouvelle méthode de barycentres, qui peuvent être reliés par différents types de chemins (approche linéaire ou approche géodésique).

Cette sous-représentation duale nécessite donc d'être transformée en superpixels duals pour permettre d'être utilisée avec des descripteurs de contours. Pour cela, nous utilisons la fonction `spx_from_closed_contours`, qui va faire correspondre chaque pixel de l'image à un superpixel labélisé (les superpixels étant dans notre cas les zones fermées de la décomposition, cf Figure 3.7 par exemple). Nous disposons donc désormais d'une carte S_{dual} de taille $(h \times w)$ représentant les superpixels duals.

3.2 Décomposition duale basée dilatation des bords

Nous avons également réfléchi à un autre type de décomposition duale, qui cette fois ne se base pas sur les centres mais directement sur les contours des superpixels. La méthode consiste à labéliser différemment chaque contour des superpixels, et de "dilater" ces contours par convolution jusqu'à ce qu'ils rencontrent les autres contours, ce qui va former des superpixels duals. Un exemple visuel des étapes est présenté dans la Figure 3.12.

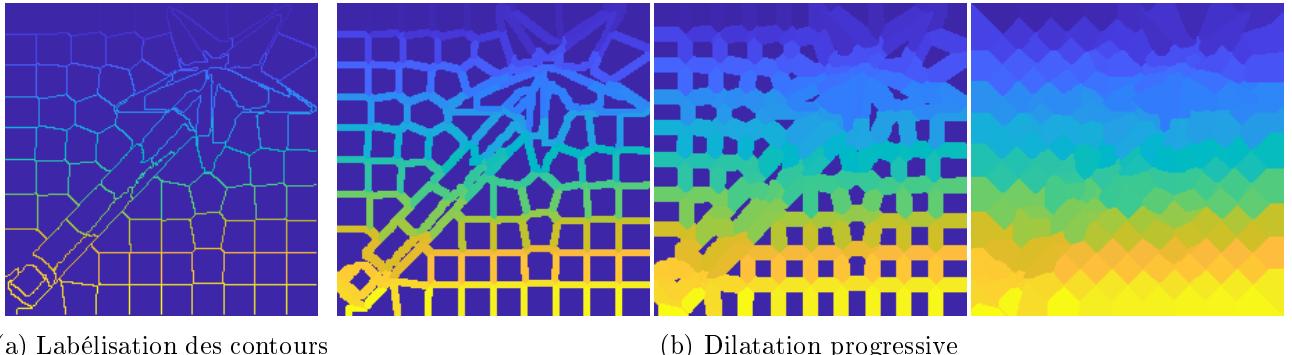


FIGURE 3.12 – Étapes de dilatation des bords des superpixels, `star.png`

Il nous suffit ensuite de considérer ces "rencontres" de contours dilatées comme des superpixels pour obtenir directement une carte de superpixels duals (cf Figure 3.13). Par la suite nous avons également ajouter un seuil de nombre de pixels au-dessous duquel les contours ne sont pas labélisés, pour éviter d'obtenir des superpixels duals trop petits.

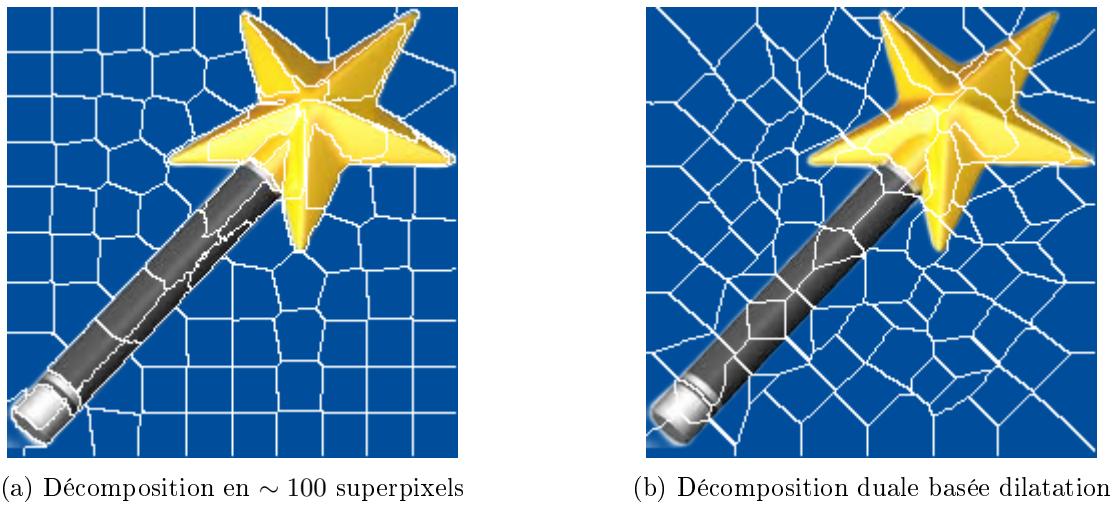


FIGURE 3.13 – Décomposition en superpixels et sa décomposition basée dilatation correspondante, `star.png`

Nous disposons alors d'une représentation qui n'est pas dépendante des centres des superpixels et qui va capturer l'intégralité des contours.

A partir de ces différentes sous-représentations duals, l'objectif est désormais de les implémenter dans le code du DSPM pour permettre d'établir une comparaison des résultats avec l'existant.

3.3 Intégration au pipeline DSPM

Dans un premier temps, nous allons présenter la structure du code du DSPM, pour pouvoir comprendre l'intérêt de la décomposition duale, et quelles sont les modifications à effectuer pour l'intégrer au pipeline DSPM.

3.3.1 Structure du code

Comme indiqué dans la partie 1.4, le SPM et le SPM dual (noté DSPM) s'appuient sur le principe du PatchMatch, un algorithme de correspondance fournissant pour chaque patch d'une image A , une correspondance dans une image B . Pour tester simplement le DSPM, il suffit de passer en entrée deux fois la même image A , mais décomposée avec deux algorithmes de superpixels différents. Par exemple, dans la Figure 3.14, l'image est décomposée avec les algorithmes SLIC et SNIC (Achanta et al., 2017 [19], une version non-itérative et plus rapide de SLIC). La correspondance est ensuite effectuée entre les superpixels de A et de B avec les méthodes SPM et DSPM. La valeur en sortie de ces méthodes correspond alors au déplacement entre les superpixels mis en correspondance.

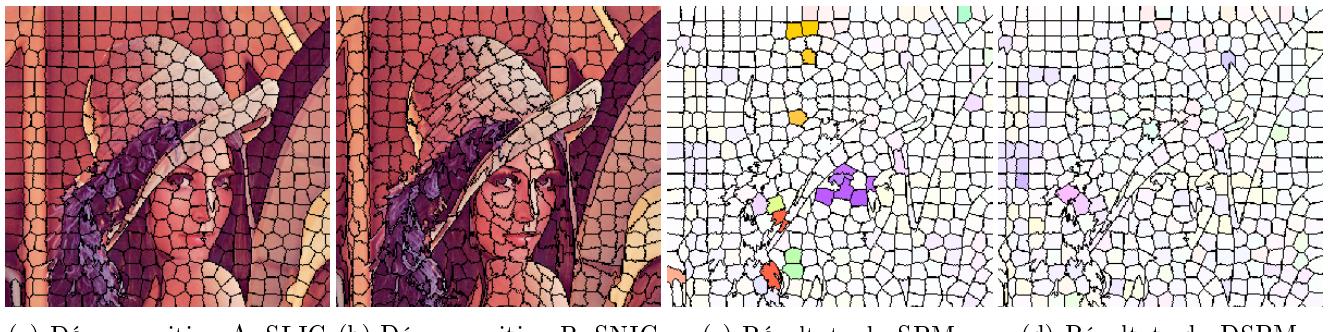


FIGURE 3.14 – Comparaison des résultats entre le SPM et le DSPM © 2020 PRL. Le déplacement entre les superpixels matchés est représenté à l'aide du code couleur de flot optique standard (des couleurs fortes représentent des déplacements plus grands).

Le DSPM repose sur plusieurs étapes fondamentales :

1. Configuration des paramètres (Images, algorithmes de superpixels, nombre et compacté des superpixels, nombre d'itérations, taille des SuperPatchs)
2. Calcul des cartes de superpixels
3. Calcul des caractéristiques des superpixels (Matrice d'adjacence, barycentres, interfaces duales, SuperPatchs)
4. Calcul des descripteurs de superpixels (Histogramme RGB pour l'information intra-superpixel, et histogramme de gradient orienté (HoG [11]) pour l'information inter-superpixel des interfaces duales)
5. Matching A→B (Fonction `SPM_dual(A,B)`)
6. Conversion de la carte de correspondance en déplacement, pour permettre l'obtention d'un score

La fonction `SPM_dual(A,B)` correspond à l'algorithme principal, divisé en 3 étapes :

Initialisation : Chaque patch de A est associé à un patch aléatoire de B .

Propagation : Amélioration des correspondances en utilisant le principe du PatchMatch (i.e. les bonnes correspondances entre patchs peuvent être propagées aux patchs adjacents) et en se basant sur la fonction `patch_dist(P_A , P_B)`, permettant de calculer la distance entre 2 patchs P_A et P_B en comparant les descripteurs inter et intra-superpixels.

Recherche aléatoire : Échantillonnage aléatoire autour de la meilleure correspondance pour améliorer les résultats et éventuellement sortir d'une zone de minimum local.

Ces deux dernières étapes sont effectuées itérativement afin d'améliorer les correspondances de patchs. Maintenant que nous connaissons la structure du code, l'idée est de remplacer les interfaces duales précédemment utilisées par la décomposition duale, pour permettre de tester cette nouvelle approche et déterminer si elle permet d'améliorer les résultats du DSPM.

3.3.2 Calcul de la décomposition duale

Dans un premier temps, il est nécessaire de déterminer les superpixels duaux des images A et B . Lors de l'étape 2 du DSPM, nous faisons appel à la fonction `dual_contours()`, qui prend en entrée les cartes de superpixels S_A et S_B ainsi que le choix de la méthode à utiliser (approche linéaire, géodésique ou dilatation), et retourne en sortie les cartes de superpixels duaux S_{dualA} et S_{dualB} .

Ensuite, il faut calculer les descripteurs de contours sur ces cartes de superpixels, nécessaires pour la comparaison des patchs entre A et B . Pour cela, nous avons implémenté la fonction `compute_sp_descriptors_dual`, qui va se baser sur les histogrammes de gradient orienté (HoG). Deux méthodes ont été principalement utilisées :

- Calculer les HoG dans toute l'image (méthode globale nommée *features_dual*) et ensuite associer une partie de l'histogramme à chaque superpixel dual ;
- Calculer les HoG séparément dans chaque superpixel dual (méthode locale nommée *matlab_imgradient*).

Ces deux méthodes prennent en entrée les images A , B , et les cartes duales S_{dualA} , S_{dualB} et retournent en sortie les histogrammes S_{HoG_dualA} et S_{HoG_dualB} .

Enfin, le matching $A \rightarrow B$ peut être effectué, en faisant appel à la fonction `SPM_dual_proj`, similaire à la fonction précédente `SPM_dual` mais en utilisant cette fois la nouvelle représentation duale. Il nous reste alors une dernière modification à effectuer, qui est la façon dont les patchs sont comparés.

3.3.3 Nouvelle méthode de comparaison des patchs

Dans le DSPM, la fonction `patch_dist(P_A , P_B)` permet de calculer la distance entre le patch P_A de l'image A et le patch P_B de l'image B .

Un patch est initialement basé sur l'information intra-superpixels fournie par les descripteurs des superpixels (histogramme RGB) et l'information inter-superpixels fournie par les interfaces duales (histogramme de gradient).

L'information intra-superpixels n'ayant pas été modifiée par rapport au DSPM, nous avons repris la même fonction `basic_dist`, qui parcourt et projette chaque superpixel $S_i \in P_A$ sur les superpixels $S_j \in P_B$, à partir des coordonnées des barycentres. Le superpixel S_i est ensuite comparé au superpixel S_j le plus proche de la projection de S_i , en établissant une distance (notée *dist_spx*) entre leurs histogrammes RGB. Cette méthode, présentée dans l'article du DSPM, permet de réduire considérablement les temps de calcul de distance, tout en augmentant potentiellement la précision, en comparaison avec une méthode de recherche exhaustive.

L'information inter-superpixels, quant à elle, dépend désormais de la nouvelle représentation duale. Dans un premier temps, nous utilisons la fonction `get_dual_adj`, qui assigne à chaque patch les superpixels duaux contenus à l'intérieur, et les stocke dans les matrices `map_SP_dualA` et `map_SP_dualB` (de taille $nb_spx \times nb_spx_dual$).

Ensuite, pour comparer les superpixels duaux de deux patchs P_A et P_B , nous utilisons la fonction `closest_intersect` ; pour chaque superpixel dual de P_A , on cherche le superpixel dual de P_B le plus proche spatialement, et on établit ensuite la distance (notée *dist_duale*) entre leurs histogrammes de gradient.

Idéalement, cette distance devrait être calculée de la même manière que dans la fonction `basic_dist`, en se basant sur les projections de superpixels duaux, mais nous ne sommes pas parvenus à l'implémenter, en raison d'un mauvais paramétrage dans les projections des barycentres duaux. Enfin, à partir des distances inter-superpixels et intra-superpixels, nous pouvons formuler la distance globale entre deux patchs, de la même manière que dans le DSPM :

$$dist = \alpha \times dist_spx + (1 - \alpha) \times dist_duale, \quad \alpha \in [0, 1]$$

3.4 Résultats et tests

Après avoir terminé l'implémentation du nouveau pipeline DSPM avec la décomposition duale, nous avons réalisé des tests sur plusieurs images pour mesurer la précision et optimiser les paramètres de ce nouvel algorithme.

3.4.1 Corrections des cartes de superpixels duals

Dans un premier temps, les résultats du nouveau DSPM étaient bien en deçà du DSPM classique, et ce peu importe le choix des descripteurs. Après vérification des cartes des superpixels duals des approches linéaire et géodésique, il s'est avéré que certaines zones à l'intersection de plusieurs superpixels étaient considérées à tort comme des superpixels par la fonction `spx_from_closed_contours` (cf Section 3.1.5). En effet, cette fonction prend en entrée seulement l'image où les barycentres ont été reliés avec un chemin linéaire ou géodésique ; il se peut donc que des conflits se créent à l'échelle pixellique au niveau des intersections (par exemple, une zone de 1 pixel formée par plusieurs chemins se croisant va être considérée comme un superpixel). Pour corriger cela, nous avons ajouté un seuil sur le nombre de pixel minimum pour former un superpixel (cf Figure 3.15).

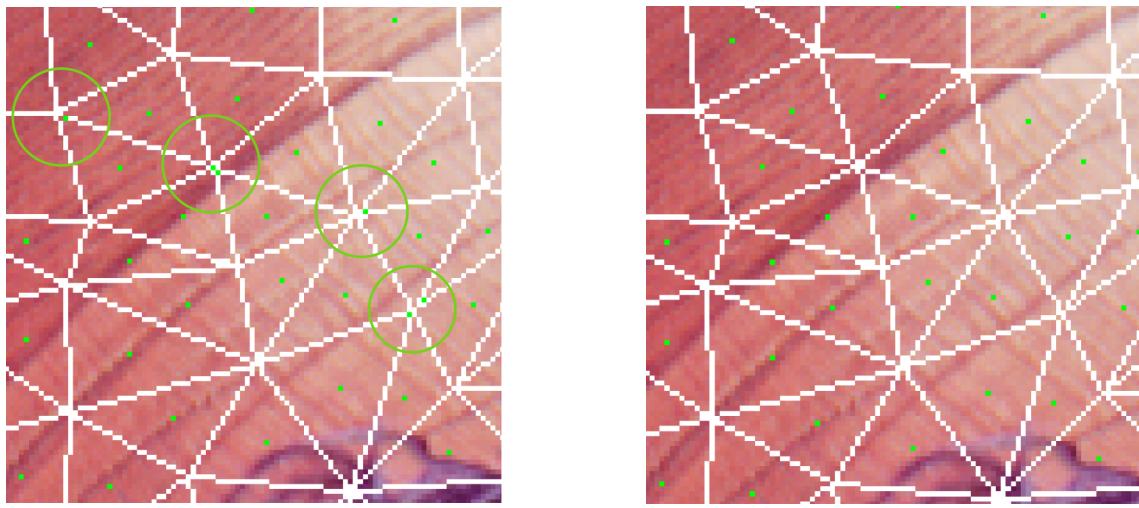


FIGURE 3.15 – Décomposition duale linéaire avant et après correction de la conversion en superpixels, `lena.png`. Les points verts correspondent aux barycentres des superpixels, et les cercles verts désignent les zones considérées à tort comme des superpixels.

Une méthode similaire a été appliquée à la décomposition duale basée dilatation, comme indiqué dans la Section 3.2. Après correction, les résultats du nouveau DSPM avoisinaient ceux du DSPM classique ; les mauvais résultats du nouveau DSPM étaient donc expliqués par ces superpixels non désirés, leur taille réduite ne permettant pas de fournir une information de contour suffisante.

3.4.2 Résultats avec la méthode DSPM

Pour comparer directement les résultats des approches DSPM et DSPM basé décomposition duale, nous avons repris les paramètres utilisés dans l'article du DSPM, en appliquant les algorithmes sur plusieurs images de test. De plus, la principale méthode à tester étant la comparaison des patchs, nous avons appliqué les algorithmes avec le mode `exhaustive_matching`, qui permet de tester toutes les combinaisons de patchs possibles entre deux images, sans dépendre du PatchMatch. En effet, le principe aléatoire du PatchMatch peut entraîner un biais dans les calculs, notamment lors des étapes d'initialisation et de recherche aléatoire ; il peut donc être judicieux de ne pas l'utiliser pour tester de façon exhaustive seulement la comparaison de patchs.

La Figure 3.16 ci-dessous correspond à une moyenne des scores sur les 5 images de test en fonction de la compacité des superpixels et de la méthode duale utilisée parmi les suivantes :

- Interfaces duales de l'article DSPM (en bleu) ;
- Approche duale basée chemin linéaire entre les centres (en vert) ;
- Approche duale basée chemin géodésique entre les centres (en magenta) ;
- Approche duale basée dilatation des bords (en rouge).

Le score correspond au déplacement moyen entre les patchs des images comparées (un score faible indique donc un meilleur résultat). Les paramètres sont définis tels que dans l'article du DSPM, avec comme descripteurs l'histogramme RGB pour l'information intra-superpixels, et les HoG locaux (*matlab_imgradient*) pour l'information inter-superpixels des méthodes duales.

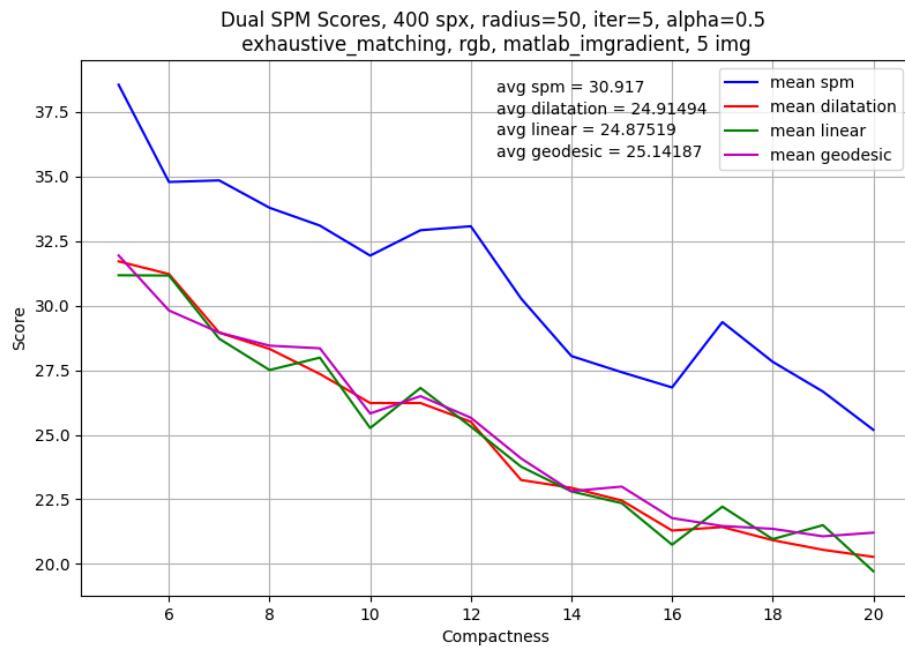


FIGURE 3.16 – Comparaison des scores entre les différentes méthodes de DSPM

À partir de ces résultats, nous observons que le nouveau DSPM basé sur les décompositions duales apporte de meilleurs résultats que le DSPM classique, et ce peu importe la représentation duale utilisée. À noter que ces approches duales ajoutent un temps de calcul non négligeable au DSPM (cf Table 3.1).

TABLE 3.1 – Scores et temps de calculs

Méthode	Score moyen	Temps de calcul
DSPM	30.917	~4.5s
DSPM linéaire	24.875	~7s
DSPM dilatation	24.915	~15s
DPSM géodésique	25.142	~110s

Ces temps de calcul correspondent à toutes les étapes nécessaires au DSPM (calcul des cartes de superpixels, calcul des descripteurs, matching et obtention du score) entre deux images *A* et *B*.

3.4.3 Nouvelle méthode de barycentre appliquée aux SuperPatchs du DSPM

Dans la partie 3.1.4, nous avons élaboré une nouvelle méthode de calcul de barycentres (nommée *mix*), adaptée pour les superpixels irréguliers. Or, le DSPM se base sur la correspondance de SuperPatchs, qui sont définis par un rayon autour du barycentre de chaque superpixel. Dans les articles SPM et DSPM, ces barycentres de superpixels sont simplement définis comme la moyenne des coordonnées des

pixels les composant (méthode nommée *mean*). Il pourrait donc être judicieux d'employer la nouvelle méthode *mix* pour la définition des SuperPatchs, ce qui apporterait en théorie de meilleurs résultats car les centres seraient systématiquement bien placés à l'intérieur des superpixels, contrairement à la méthode *mean* qui peut placer les barycentres à l'extérieur des superpixels.

Dans un premier temps, nous avons comparé les résultats des DSPM *mean* et DSPM *mix* pour l'étiquetage de visages sur la base de données LFW (*Labeled Faces in the Wild*, [20]). Cette base de référence contient des images de taille 250×250 , décomposées en environ 225 à 250 superpixels, ainsi que des vérités terrains d'étiquetage (cheveux, visage, fond) associées à chacun de ces superpixels. Ces résultats sont disponibles en Table 3.2.

TABLE 3.2 – Précision d'étiquetage sur la base LFW

Méthode	Précision à l'échelle des superpixels	Précision à l'échelle des pixels
DSPM mean	95.24%	95.59%
DSPM mix	95.15%	95.51%

Les précisions à l'échelle des superpixels et à l'échelle pixellique de la méthode *mix* sont légèrement inférieures à celles de la méthode initiale se basant sur la méthode *mean*. Cela pourrait s'expliquer par la taille réduite des superpixels de la base LFW, ainsi que par leur compacité relativement élevée. En observant les cartes de superpixels, on s'aperçoit en effet que très peu de superpixels présentent des formes irrégulières (la quasi-totalité des barycentres est bien placée), ce qui remettrait en question la pertinence d'utiliser la méthode *mix*.

Par la suite, nous avons également comparé les résultats du DSPM *mean* et DSPM *mix* avec et sans décompositions duals, avec les paramètres et images de tests utilisés précédemment pour comparer les approches duals. Ces résultats sont disponibles dans la Figure 3.17 (Traits continus : DSPM *mean*, pointillés : DSPM *mix*).

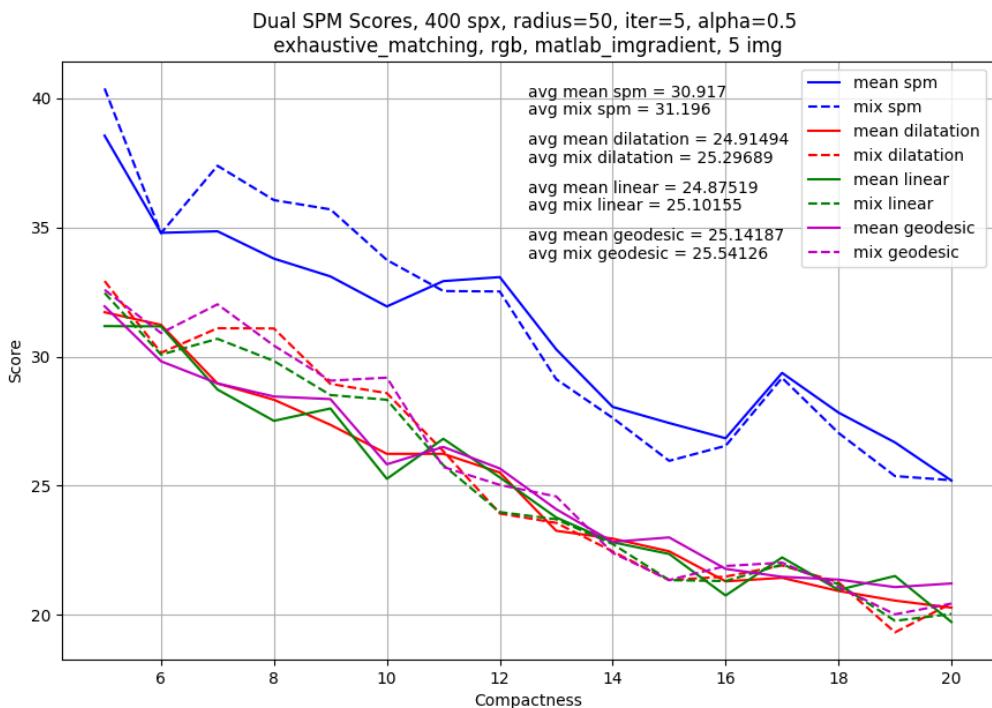


FIGURE 3.17 – Score SPM

Globalement, les résultats sont similaires, mais le DSPM *mix* est en moyenne en deçà du DSPM *mean*, et ce peu importe l'approche utilisée. Plusieurs hypothèses ont été formulées pour expliquer pourquoi les barycentres *mix* n'apportent pas de meilleurs résultats dans les conditions actuelles :

- Les barycentres de la méthode *mean* sont bien répartis dans toute l'image, car ils se basent seulement sur les coordonnées des pixels de chaque superpixel, tandis que les barycentres de la méthode *mix* peuvent présenter des disparités dans la répartition sur l'image, en raison de leur dépendance aux formes des superpixels. Ces disparités de répartition de barycentres vont ensuite impliquer des disparités dans les SuperPatchs, ce qui pourrait expliquer pourquoi la correspondance entre les patchs est moins précise.
- La méthode *mix* est particulièrement adaptée pour le calcul de barycentre des superpixels irréguliers ayant des formes complexes. Or, la forme des superpixels est directement lié au paramètre de compacité, en entrée des algorithmes SLIC et SNIC par exemple. Il se pourrait donc que la méthode de barycentre *mix* ne soit utile que si la compacité des superpixels est très faible, ce qui n'est pas compatible avec le principe du DSPM, qui apporte de bons résultats avec une compacité assez élevée. Des tests ont été effectués sur l'image `peppers.png` décomposée en 300 superpixels avec différentes compacités (cf Table 3.3), et apportent des résultats en accord avec cette hypothèse.

TABLE 3.3 – Scores DSPM *mean* et *mix* en fonction de la compacité, `peppers.png`

Méthodes	Compacité des superpixels				
	0.1	0.5	1.0	5.0	10.0
DSPM mean	37.51	34.50	29.04	23.82	20.26
DSPM mix	35.46	32.56	28.86	25.14	22.43

3.4.4 Bilan

En conclusion, nous avons présenté plusieurs sous-représentations irrégulières duales, basées sur le chemin (linéaire ou géodésique) entre les points centraux des superpixels, et sur la dilatation des bords des superpixels. Ces différentes approches proposées fournissent des résultats surpassant ceux de l'article du DSPM, et permettent de valider notre hypothèse initiale sur l'importance de l'information de contour lors de l'utilisation de méthodes basées superpixels.

En revanche, la nouvelle méthode de point central proposée, bien que satisfaisante visuellement, ne permet pas d'améliorer les résultats du DSPM. Nous restons convaincus que cette approche basée sur le calcul des moments des superpixels présente un axe de réflexion intéressant, et pourra être utilisée dans de futures méthodes basées sur le centre de superpixels irrégulier, avec des applications dans des cas spécifiques tels que l'analyse de texture complexe, par exemple pour l'étude microscopique de tissus (histopathologie).

Dans la troisième et dernière partie de ce stage, nous allons étudier les éventuels intérêts de ces approches duales pour la génération de trimap et le matting, deux domaines étroitement liés de la segmentation d'images, qui sembleraient favorables à l'utilisation d'une information de contour supplémentaire.

Chapitre 4

Génération de trimap et matting

Dans un premier temps, une étude approfondie de l'état de l'art a été réalisée sur les articles et méthodes majeurs dans les domaines de la génération de trimap et du matting, pour permettre de comprendre les problématiques et enjeux actuels dans ce milieu de la segmentation d'images.

4.1 État de l'art

4.1.1 Matting

Dans la partie 1.4 nous avions défini le matting comme le principe de calculer, à partir d'une image et d'une trimap, la combinaison linéaire entre le fond et l'objet dans la zone inconnue de la trimap. En résumé, les méthodes de matting vont chercher à définir précisément la limite entre l'objet et le fond d'une image, en se basant sur les informations de la trimap pour limiter les zones de recherche et ainsi réduire les temps de calcul.

Les algorithmes de matting sont principalement divisés en deux catégories :

- **Sampling-based methods** : Estimer le coefficient α d'un pixel en analysant la distribution de couleurs de ses pixels voisins et d'une sélection de régions connues (samples). Le principe est donc de rassembler des samples à partir des zones fond et objet, et de déterminer les plus proches paires à partir de critères pour représenter un pixel tel un mix entre objet et fond.
- **Affinity/Propagation-based methods** : Propagation des contraintes (spatiales et couleur) depuis les régions connues vers la région d'incertitude. Le but est de définir une relation entre les pixels et un modèle de propagation pour faire évoluer le coefficient α . Ces méthodes reposent sur des formulations mathématiques précises contrairement aux méthodes basées samples qui sont définies de manière empirique.

Les méthodes de sampling apportent des résultats dépendants de la sélection des régions connues ; les zones lisses sont globalement bien représentées, contrairement aux zones complexes et irrégulières, où le manque de samples va impliquer une faible précision de segmentation.

Les méthodes de propagation permettent d'obtenir de bons résultats dans les régions complexes, mais pas dans les images avec beaucoup de textures ou de détails, car l'hypothèse mathématique exprimée est locale et manque donc de robustesse face aux variations importantes dans l'image.

Un outil intéressant pour comparer les articles de matting est la base de données *alphamatting*¹, qui recense les différents scores de segmentation (SSD, MSE, Gradient) des méthodes de l'état de l'art à partir d'un jeu de données composé d'image et de trimap, face à la vérité terrain. Des datasets d'entraînement sont également disponibles pour les méthodes basées deep learning.

Ci-dessous, une liste des principales méthodes de matting, ainsi que leur score SSD sur le dataset alphamatting :

¹<http://www.alphamatting.com/eval25.php>, dernier accès le 27/08/2020

- **2007** (Date de publication) - **45.2** (SSD) - Optimized Color Sampling for Robust Matting [21] - Un des premiers articles sur la méthode de sampling. Compare la couleur de samples spatialement proches. Le manque de samples générés implique de faibles résultats sur les zones complexes, qui ne seront pas assez représentées.
- **2011 - 33.9** - A Global Sampling Method for Alpha Matting [22] - Utilise un grand nombre de samples pour palier au problème précédent, avec une adaptation de l'étape de Random Search du PatchMatch pour déterminer rapidement les paires samples-pixel inconnues les plus proches.
- **2015 - 25.3** - Image Matting with KL-Divergence Based Sparse Sampling [23] - Utilise une sélection de sous-ensembles clairsemés (Sparse Subset Selection) pour échantillonner seulement les samples fond/objet qui décrivent le mieux les pixels inconnus. De plus, une nouvelle mesure de distance (KL-divergence) basée superpixel est utilisé pour comparer deux samples.
- **2017 - 12.2** - Designing Effective Inter-Pixel Information Flow for Natural Image Matting [24] - Se base sur une combinaison de plusieurs flux ; le flux de "mixture" qui permet de déterminer le mélange de couleurs entre l'objet et le fond, le flux K-U qui calcule le lien entre les pixels connus K (fond/objet) et les pixels inconnus U , le flux intra-U, qui détermine l'opacité des pixels en se basant sur les similarités de couleurs des pixels inconnus U , et le flux d'information local, qui utilise l'information de connectivité spatiale et le voisinage.
En résumé, la méthode combine des flux utilisés dans des articles précédents en y ajoutant une information de "mixture", qui correspond aux pixels des structures fines ou transparentes comportant des informations du fond et de l'objet.
Cet article propose, à ce jour, la méthode ayant les meilleurs résultats sans utiliser de réseaux de neurones avec entraînement. Concernant ces méthodes de deep learning, les principales sont les suivantes :
- **2017 - 10.1** - Deep Image Matting [25] - Modèle en deux parties ; une convolution avec l'image et la trimap en entrée pour estimer le coefficient α , puis deuxième convolution plus fine pour améliorer le coefficient α et obtenir une segmentation plus précise des bords.
- **2020 - 7.2** - Background Matting [26] - Réseau de neurones similaire à l'article précédent, en ajoutant en entrée le résultat d'un algorithme de reconstruction du fond. Cette information supplémentaire permet d'améliorer les résultats du Deep Image Matting.
- **2020 - 1.9** - FBA Matting [27] - Article le plus récent, apportant des résultats excellents sur le dataset alphamatting. Au lieu de simplement chercher à définir le coefficient α , la méthode estime simultanément l'objet, le fond et α , en utilisant des réseaux de neurones U-net [28] se basant sur l'état de l'art actuel.

Depuis les trois dernières années, on constate que les méthodes théoriques basées sampling ou propagation sont largement dépassées par les méthodes basées deep learning. L'intérêt de notre sous-représentation duale semblerait alors limité dans ce domaine. L'information de contour des approches duales pourrait éventuellement être intégrée en entrée d'un réseau de neurones convolutif pour le matting, mais cela s'éloignerait de la thématique du stage, qui s'intéresse à des méthodes rapides, sans apprentissage et efficaces avec peu de données d'entraînement.

En revanche, il est à noter que la majorité des articles présentés ci-dessus se basent sur l'image et sur la trimap associée. Or, ces trimaps sont souvent générées manuellement, et en comparaison avec le nombre d'articles dans le domaine de le matting, peu de méthodes proposent une génération de trimap rapide et automatisée. Dans ce contexte, nous avons étudié les principaux articles et méthodes du domaine de la génération de trimap.

4.1.2 Génération de trimap automatique et semi-automatique

Une méthode répandue dans la génération de trimap est l'utilisation d'opérations morphologiques, qui correspondent à des filtres linéaires s'appliquant sur des images binaires ou en niveaux de gris. Les opérations morphologiques principalement utilisées sont l'opération d'érosion et de dilatation, basées sur un élément structurant (le plus souvent en forme de disque, de carré, de croix..).

L'érosion d'une image binaire A par un élément structurant B est l'ensemble des pixels p tels que la fenêtre B_p est incluse dans A :

$$\epsilon_{(A,B)} = A \ominus B = \{p \mid B_p \subseteq A\}$$

La dilatation d'une image binaire A par un élément structurant B est l'ensemble obtenu en remplaçant chaque pixel p de A par sa fenêtre B_p :

$$\delta_{(A,B)} = A \oplus B = \bigcup \{B_p \mid p \in A\}$$

En résumé, l'opération d'érosion va permettre de rétrécir la figure en érodant les contours selon l'élément structurant, tandis que l'opération de dilatation va permettre d'élargir la figure en dilatant les contours (cf Figure 4.1).



(a) Image initiale



(b) Opération d'érosion



(c) Opération de dilatation

FIGURE 4.1 – Exemple d'opérations morphologiques sur une image binaire © 2008 OpenCV

Dans le cas de la génération de trimap, il est possible d'obtenir une bordure de l'objet définissant la zone inconnue de la trimap, en prenant la différence entre les opérations d'érosion et de dilatation.

Nous allons désormais présenter les principaux articles de la génération de trimap automatique et semi-automatique :

- **2013** - Automatic trimap generation for digital image matting [29] - Un des premiers articles à se pencher sur la génération de trimap de façon semi-automatique. L'algorithme se base sur la méthode GrabCut (qui nécessite une action utilisateur), couplé à une dilatation/érosion et des effets de flous pour faciliter la détection de contours.
- **2017** - Automatic trimap generation for image matting [30] - Utilisation de la carte de saillance, qui détermine les éléments attirant l'attention visuelle, calculée avec des méthodes récentes basées deep learning, pour déterminer les superpixels saillants et non-saillants. Une étape de raffinement est ensuite appliquée, en utilisant des k-means clustering sur l'objet et le fond, en se basant sur les descripteurs de contours OTC (similaires au HoG). Enfin, une binarisation est effectuée sur la segmentation, puis une étape de dilatation/érosion pour obtenir une trimap. Cette méthode est entièrement automatique, les résultats sont donc fortement dépendant du calcul de saillance, et l'algorithme ne permet pas la détection de plusieurs objets dans l'image.
- **2019** - Generating Trimaps for Image Matting Using Color Co-Fusion [31] - Génération de plusieurs "soft segmentation map" avec différentes méthodes de segmentation binaire en se basant sur des marqueurs utilisateur. Ensuite, une méthode de "fuzzy clustering" est utilisée pour diviser les cartes en 3 parties (fond - objet - inconnu), afin de former des trimaps. Enfin, les différentes

trimaps générées sont fusionnées à l'échelle des superpixels en utilisant la co-fusion (cooperative fusion of cluster images).

En résumé, les méthodes de génération de trimap actuelles permettent d'obtenir des résultats de manière automatique en se basant sur une carte de saillance générée au préalable, ou de manière semi-automatique en segmentant l'image à l'aide de marqueurs binaires annotés par l'utilisateur. Dans le dernier article présenté [31], la génération de trimap est dépendante de plusieurs segmentations binaires, et les superpixels sont seulement utilisés pour fusionner l'information des différentes cartes.

Dans notre cas, l'idée est d'utiliser seulement la hiérarchie de superpixels et les marqueurs binaires pour segmenter l'image rapidement, et générer la trimap à partir de cette segmentation binaire et éventuellement de l'information de contour fournie par les sous-représentations duales étudiées.

4.2 Génération de trimap à partir de la segmentation binaire

La méthode de binarisation avec pose de marqueurs, proposée dans la section 2.3, permet d'obtenir une segmentation rapide d'un objet dans une image. Les résultats de cette méthode ne sont pas assez précis pour rivaliser avec les méthodes de matting de l'état de l'art, mais constituent une base solide pour la génération de trimap.

Dans un premier temps, nous avons donc testé différentes implémentations de trimap en MATLAB, basées sur la segmentation binaire en sortie du logiciel Qt. La première méthode utilise les opérations morphologiques d'érosion et de dilatation pour former la zone de pixels inconnus de la trimap (partie grise). L'opération d'érosion va permettre de corriger les pixels du fond ayant été labélisés en objet, et l'opération de dilatation va permettre d'agrandir les parties potentielles de l'objet. La trimap est obtenue en définissant la zone d'incertitude comme la différence entre l'érosion et la dilatation de la segmentation binaire.

Ensuite, nous avons essayé d'utiliser les superpixels des approches duales pour former la trimap ; l'idée est d'ajouter à la zone inconnue de la trimap les superpixels duals entourant les superpixels sur les bordures de la sélection.

Par exemple, sur la Figure 4.2, à partir de la segmentation binaire obtenue à partir de la pose de marqueurs binaires (a), nous pouvons générer la trimap par opérations morphologiques (b), ainsi que la trimap superpixels duals basés sur la dilatation des contours des superpixels (c).

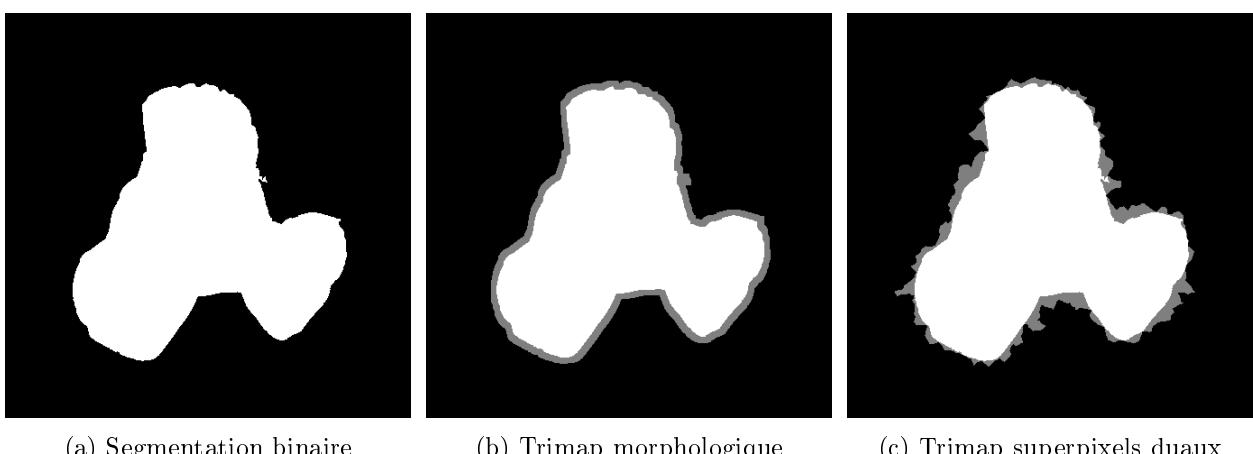


FIGURE 4.2 – Comparaison de méthodes de trimaps à partir de la segmentation binaire proposée dans la section 2.3 dans le logiciel Qt, `flower.jpg`

La trimap morphologique présente une zone d'incertitude lisse définie autour des bordures de l'objet, tandis que la trimap superpixels duals présente une zone d'incertitude dentelée. Cet aspect irrégulier peut paraître défavorable au premier abord, mais nous allons voir par la suite que ces superpixels duals peuvent apporter un intérêt non-négligeable dans la génération de trimap.

4.3 Intégration au logiciel Qt

Après avoir testé différentes implémentations de trimap en MATLAB, un des objectifs du stage est d'intégrer directement au logiciel Qt la génération de trimap.

Pour ce faire, à partir de la segmentation binaire obtenue par pose de marqueurs, les opérations d'érosion et de dilatation sont appliquées avec la bibliothèque OpenCV, et la différence de ces deux opérations est affichée en gris sur la segmentation binaire, pour définir la zone d'incertitude de la trimap.

Par exemple, dans la Figure 4.3, la segmentation binaire est effectuée à l'aide de quelques clics utilisateur définissant l'objet et le fond, et il est possible d'afficher la trimap morphologique sur la partie de droite du logiciel, en cliquant sur l'onglet "trimap" en bas à droite.

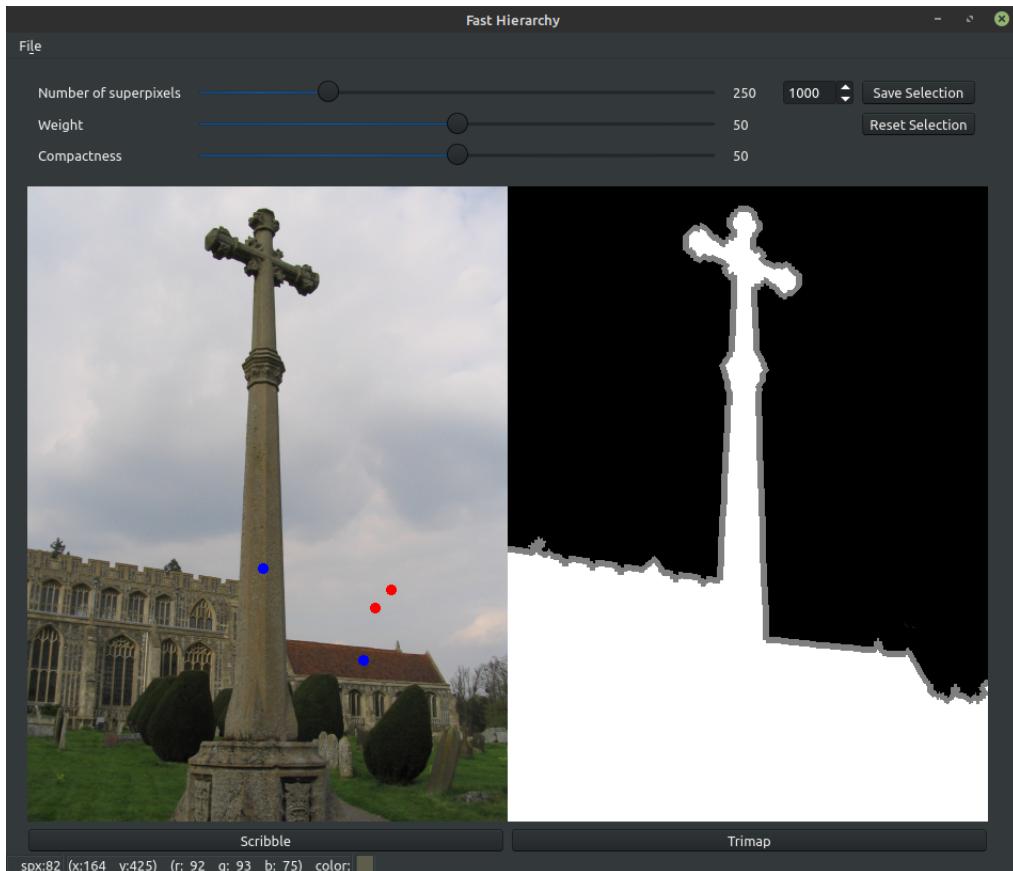


FIGURE 4.3 – Exemple de génération de trimap morphologique à partir de pose de marqueurs par l'utilisateur, `cross.png`

Récemment, un article [32] montre que l'utilisation des opérateurs morphologiques pour générer une trimap n'est pas optimale, car ces opérateurs ne prennent pas en compte la mixture de couleur des pixels venant de l'objet et du fond, et sont donc entièrement dépendants de la segmentation binaire initiale.

Dans ce contexte, l'utilisation des superpixels duals semble intéressante, car permettrait la sélection directe des régions d'incertitude, situées aux bordures de l'objet.

Nous avons commencé par implémenter l'approche duale basée dilatation des bords au logiciel Qt, dans la fonction `dualDilatation` de la structure `SuperpixelHierarchy`. Cette méthode permet d'obtenir une carte de superpixels duals, en labélisant chaque bordure de superpixel et en dilatant progressivement ces bordures jusqu'à ce que chaque pixel soit labellisé, d'une façon similaire à l'implémentation MATLAB présentée dans la partie 3.2.

Par la suite, nous avons réfléchi à l'idée d'une approche dilatation des bords pondérée par les poids de la segmentation hiérarchique. En effet, dans la structure de superpixels hiérarchiques actuelle, il est possible de récupérer la date de fusion entre deux superpixels voisins ; on dispose alors d'une information supplémentaire qui pourrait paramétriser la dilatation des bords (par exemple, une date de fusion

tardive entre deux superpixels voisins impliquerait une dilatation plus forte de leur frontière). Par manque de temps, nous n'avons pas pu finaliser cette partie du projet. A terme, il aurait également été intéressant d'ajouter des fonctionnalités basées sur cette carte de superpixels duals, telles que la sélection directe des superpixels duals pour obtenir une trimap précise.

Par exemple, dans la figure 4.4, on peut remarquer que le superpixel (en magenta) sur le bord de la fleur ne définit pas précisément la frontière entre la fleur et le fond de l'image. En sélectionnant le superpixel dual (en vert) qui entoure la frontière mal capturée, on définirait alors précisément la région d'incertitude qui pose problème à ce superpixel. Cette région pourrait ensuite être ajoutée à la trimap pour permettre une segmentation précise à l'échelle pixellique par un algorithme de matting.

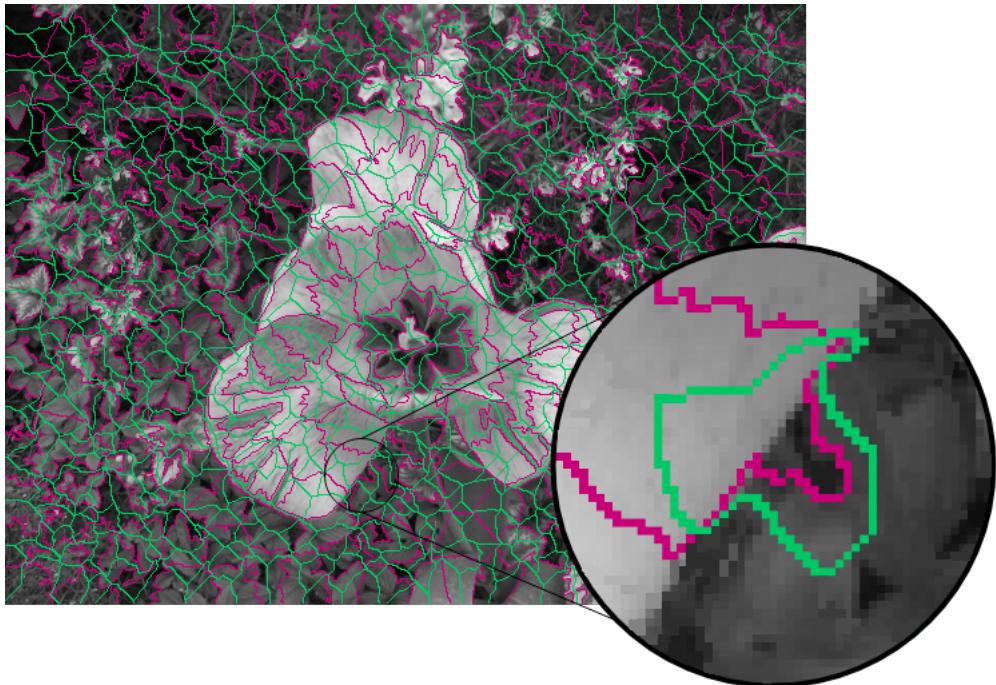


FIGURE 4.4 – Exemple de sélection de superpixel dual pour capturer la frontière mal segmentée, *flower.jpg*. Les contours des superpixels sont représentés en magenta, et les contours des superpixels duals en vert.

A notre connaissance, ce type d'approche n'a jamais été abordé dans les méthodes et articles de l'état de l'art actuel, et répond à la problématique de génération de trimap rapide et précise en permettant de sélectionner directement les régions d'incertitude situées aux bordures des superpixels.

4.3.1 Conclusion

Après une étude approfondie de l'état de l'art, il s'est avéré que les algorithmes et méthodes de matting actuels apportent des résultats satisfaisants, mais sont dépendants d'une trimap en entrée, généralement fournie par les jeux de tests ou définie manuellement par un utilisateur. Il existe donc un réel besoin en méthode de génération de trimap automatique ou semi-automatique, et à ce jour peu d'articles présentent une solution rapide et efficace. Dans ce contexte, nous avons proposé une méthode de génération de trimap morphologique basée sur la segmentation binaire de la partie 2.3. De plus, nous avons commencé à étudier l'intérêt d'une sous-représentation dual de superpixels dans ce domaine, en énonçant par exemple la dilatation des bords pondérée par la date de fusion des superpixels hiérarchiques, ou encore la sélection directe des superpixels duals, permettant de capturer intégralement l'information de contour utile aux algorithmes de matting.

Chapitre 5

Bilan

5.1 Contributions et conclusion sur les objectifs

Dans ce stage de recherche, nous avons répondu à certaines problématiques actuelles dans le domaine de la segmentation d'images, en organisant notre travail autour de plusieurs axes. Dans un premier temps, nous avons réfléchi à une approche hiérarchique de superpixels, que nous avons implémentée dans notre logiciel Qt pour permettre une segmentation interactive rapide et précise d'un objet dans une image. Par la suite, nous avons évalué l'intérêt de sous-représentations irrégulières duales, permettant d'être robuste à l'imprécision des frontières des superpixels en capturant l'information de contour, et obtenues par exemple en reliant les centres des superpixels ou en dilatant leurs contours. Une étude a été réalisée sur le calcul du point central idéal des superpixels irréguliers, et a aboutit à la création de la méthode *mix*, basée sur les axes des moments d'ordre 2 et la carte de distance aux bords. Les approches duales utilisant cette nouvelle méthode de point central ont ensuite été implémentées dans l'algorithme de correspondance de superpatchs DSPM (Giraud et al., 2020, [1]), et ont permis une amélioration de la précision de la méthode. En dernière partie de ce stage, après avoir étudié les enjeux actuels dans les domaines du matting et de la génération de trimap, nous avons implémenté une méthode de génération de trimap rapide et interactive basée sur des opérations morphologiques. De plus, nous avons proposé une application aux approches duales dans ce domaine, avec par exemple une dilatation des contours adaptative basée sur l'information hiérarchique entre les superpixels, et une sélection interactive des superpixels duaux, permettant de capturer l'information de structure des contours dans les régions d'incertitude de la trimap. Ce travail ouvre donc de nouvelles perspectives intéressantes dans la segmentation d'images, en s'appuyant sur l'importance des informations de voisinage et de contour lors de l'utilisation de superpixels.

5.2 Retours sur le déroulement du stage

Dans la première partie du stage, la continuité avec le projet de fin d'étude s'est faite naturellement, et la plupart des fonctionnalités du logiciel Qt ont été implémentées sans difficulté particulière. En effet, le langage C++, le framework Qt et la bibliothèque de traitement d'image OpenCV m'étaient déjà familier grâce au parcours Image et Son du Master Informatique effectué. La section théorique sur la pose de marqueurs binaires était la plus délicate, car nécessitait la compréhension des méthodes de parcours de graphe de la bibliothèque Hogra, pour permettre leur application à notre hiérarchie de superpixels. De plus, nous avons rencontré des difficultés sur la hiérarchie rapide basée graphe ; le code mis en ligne par les auteurs de l'article n'était pas à jour, et la correction ultérieure fournie n'avait pas exactement le comportement souhaité lors du changement des paramètres. Dans la version finale du logiciel, certaines combinaisons de paramètres entre la compacité et le poids distance-couleur peuvent donc amener à des superpixels de formes indésirables (souvent allongés verticalement).

La deuxième partie du stage reposait sur l'utilisation de l'environnement de développement MATLAB. Après avoir corrigé quelques problèmes de compatibilité entre les différentes versions de MATLAB, j'ai passé plusieurs jours à étudier l'architecture du code du DSPM, fourni par mes encadrants. Celui-ci est basé sur des fonctions C-MEX multi-processus, permettant de manipuler des fichiers sources écrits

en langage C ou C++. Lors de mon Master, nous avions eu l'occasion de travailler sur l'environnement et le langage MATLAB, et j'ai donc pu approfondir mes connaissances dans ce domaine grâce à ce stage, avec la découverte du C-MEX et de MATLAB en ligne de commande, qui permet notamment l'utilisation du debugger **gdb** en parallèle.

Concernant le sujet en lui-même, la segmentation d'images et le traitement d'images en général sont des domaines que je trouve captivants, et dans lesquels je souhaite orienter mon avenir professionnel. A cause du contexte actuel lié au COVID-19, mon stage s'est déroulé entièrement en télétravail, et je n'ai donc pas pu découvrir en détail les activités des doctorants et enseignants-chercheurs dans les laboratoires du LaBRI et de l'IMS. Malgré cela, j'affectionne particulièrement le domaine de la recherche, et la poursuite de mes études en thèse en Informatique pour l'Image est encore en cours de réflexion.

Bibliographie

- [1] R. Giraud, M. Boyer, and M. Clément, “Multi-scale superpatch matching using dual superpixel descriptors,” *Pattern Recognition Letters*, 2020.
- [2] Ren and Malik, “Learning a classification model for segmentation,” in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 10–17 vol.1, 2003.
- [3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, “Slic superpixels compared to state-of-the-art superpixel methods,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, p. 2274–2282, Nov. 2012.
- [4] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut : Interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, vol. 23, no. 3, p. 309–314, 2004.
- [5] B. L. Price, B. Morse, and S. Cohen, “Geodesic graph cut for interactive image segmentation,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3161–3168, 2010.
- [6] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman, “Geodesic star convexity for interactive image segmentation,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3129–3136, 2010.
- [7] B. Perret, “Interactive segmentation with morphological hierarchies,” 2014.
- [8] B. Perret, G. Chierchia, J. Cousty, S. F. Guimarães, Y. Kenmochi, and L. Najman, “Higra : Hierarchical graph analysis,” *SoftwareX*, vol. 10, pp. 1–6, 2019.
- [9] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “Patchmatch : a randomized correspondence algorithm for structural image editing.,” *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.
- [10] R. Giraud, V. Ta, A. Bugeau, P. Coupé, and N. Papadakis, “Superpatchmatch : an algorithm for robust correspondences using superpixel patches,” *IEEE Transactions on Image Processing (TIP)*, 2017.
- [11] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 886–893 vol. 1, 2005.
- [12] X. Wei, Q. Yang, Y. Gong, N. Ahuja, and M. Yang, “Superpixel hierarchy,” *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 4838–4849, 2018.
- [13] S. Xie and Z. Tu, “Holistically-nested edge detection,” in *IEEE International Conference on Computer Vision*, 2015.
- [14] S. Niklaus, “A reimplementation of HED using PyTorch.” <https://github.com/sniklaus/pytorch-hed>, 2018.

- [15] B. Perret, J. Cousty, J. C. Rivera Ura, and S. J. F. Guimarães, “Evaluation of morphological hierarchies for supervised segmentation,” in *Mathematical Morphology and Its Applications to Signal and Image Processing* (J. Benediktsson, J. Chanussot, L. Najman, and H. Talbot, eds.), vol. 9082 of *Lecture Notes in Computer Science*, (Reykjavik, Iceland), pp. 39–50, Springer, May 2015.
- [16] J. E. Bresenham, “Algorithm for computer control of a digital plotter,” *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.
- [17] G. Peyré, “Toolbox fast marching - a toolbox for fast marching and level sets computations.” <https://github.com/gpeyre/matlab-toolboxes>, 2008.
- [18] M-K Hu, “Visual pattern recognition by moment invariants,” *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.
- [19] R. Achanta and S. Süsstrunk, “Superpixels and polygons using simple non-iterative clustering,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4895–4904, 2017.
- [20] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild : A database for studying face recognition in unconstrained environments,” Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.
- [21] J. Wang and M. F. Cohen, “Optimized color sampling for robust matting,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.
- [22] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, “A global sampling method for alpha matting,” in *CVPR 2011*, pp. 2049–2056, 2011.
- [23] L. Karacan, A. Erdem, and E. Erdem, “Image matting with kl-divergence based sparse sampling,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 424–432, 2015.
- [24] Y. Aksoy, T. O. Aydin, and M. Pollefeys, “Designing effective inter-pixel information flow for natural image matting,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 228–236, 2017.
- [25] N. Xu, B. L. Price, S. Cohen, and T. S. Huang, “Deep image matting,” *CoRR*, vol. abs/1703.03872, 2017.
- [26] H. Javidnia and F. Pitié, “Background matting,” 02 2020.
- [27] M. Forte and F. Pitié, “ f, b , alpha matting,” 2020.
- [28] O. Ronneberger, P. Fischer, and T. Brox, “U-net : Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015.
- [29] C. Hsieh and M. Lee, “Automatic trimap generation for digital image matting,” in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pp. 1–5, 2013.
- [30] V. Gupta and S. Raman, “Automatic trimap generation for image matting,” *CoRR*, vol. abs/1707.00333, 2017.
- [31] J. Li, G. Yuan, and H. Fan, “Generating trimap for image matting using color co-fusion,” *IEEE Access*, vol. 7, pp. 19332–19354, 2019.
- [32] C. Henry and S.-W. Lee, “Automatic trimap generation and artifact reduction in alpha matte using unknown region detection,” *Expert Systems with Applications*, vol. 133, 05 2019.