

UE Image Modélisation et Rendu (IMR)
Partie Traitement d'Images
Des transformations à l'analyse de scènes

Sylvie CHAMBON

21 janvier 2025

Remerciements

Ce cours reprend des paragraphes rédigés de mon manuscrit d'habilitation (2020). Je remercie donc toutes les personnes qui ont directement ou indirectement contribué à ces deux documents car elles ont également contribué indirectement à la rédaction de ce cours. Merci également aux étudiant·e·s des départements SN qui proposent régulièrement des améliorations.

Objectifs d'apprentissage et organisation

Objectifs d'apprentissage

Les objectifs d'apprentissage de ce cours sont les suivants :

- (1) **Connaître les outils fondamentaux pour analyser et transformer une image :**
 - (a) Savoir étudier la dynamique des niveaux de gris (ou des couleurs) dans une image en s'appuyant sur un histogramme ;
 - (b) Connaître les outils de convolution et de filtrage utilisés en traitement d'images ;
 - (c) Savoir utiliser les filtres de convolution correctement suivant les difficultés présentes dans l'image.
- (2) **Découvrir et apprendre les différents éléments liés à la détection de contours dans une image :**
 - (a) Apprendre à construire un filtre de convolution permettant de calculer la dérivée d'une image ;
 - (b) Connaître l'algorithme classique de détection de contours.
- (3) **Découvrir et apprendre différentes techniques de segmentation :**
 - (a) Savoir distinguer les approches fond/forme des approches pour plus de deux régions ;
 - (b) Apprendre et distinguer les différentes catégories de méthodes de segmentation ;
 - (c) Connaître en détails et savoir coder la méthode des k-moyennes.
- (4) **Savoir évaluer qualitativement et quantitativement un résultat de segmentation :**
 - (a) Comprendre la notion de segmentation de référence ;
 - (b) Connaître les différentes mesures d'évaluation.
- (5) **Comprendre les méthodes de sur-segmentation appelées superpixels**
 - (a) Comprendre la différence entre segmentation et sur-segmentation ;
 - (b) Savoir analyser les différents éléments d'une méthode de sur-segmentation : attributs et propriétés utilisées ainsi que méthode de construction ;
 - (c) Coder une approche connue : SLIC.
- (6) La partie pratique de cette matière doit permettre aux étudiant·e·s de **Concevoir une chaîne de traitement complète pour segmenter une image** : du calcul à l'évaluation.

Outils à votre disposition

À la fin de ce cours et des TPs associés, vous devez être capable d'analyser une image donnée en : identifiant les difficultés de la scène en terme d'extraction de contours ou de segmentation, et en proposant des solutions que vous savez justifier. Ces solutions doivent s'appuyer sur toutes les briques de bases apprises dans le cours. Sur moodle, nous vous proposons un **test de connaissances** afin de tester vos apprentissage. Nous verrons des exemples en cours. De plus, nous vous fournissons des **exemples de sujets d'examens** sur moodle et nous verrons des exemples en cours.

Temps de travail

Pour la session de classe inversée, voici une indication des temps de lecture de chaque partie. Ceci reste une indication, il faut retenir que chaque étudiant·e avance à son rythme et que certaines parties sont plus faciles que d'autres à lire.

- Ce chapitre : consacrer au moins 10 minutes et n'hésitez pas à relire!
- Introduction : 10 minutes
- Chapitre 1 : 1 ou 2h
- Chapitre 2 : 1 ou 2h
- Chapitre 3 : 1 ou 2h
- Chapitre 4 : 1 ou 2h
- Chapitre 5 et 6 (en bonus) : 1h

Organisation de l'UE

Voici le déroulé :

- 3 cours de traitement d'images avec une ou deux classes inversées (**avec évaluation en groupe**)
- 6 TPs de traitement d'images / + 6 TPs de traitement d'images (pour IATI uniquement)
- 1 **TP d'évaluation** avec une évaluation individuelle sur machine et/ou par binôme à l'oral
- 2 cours de modélisation / + 1 cours (pour IATI uniquement)
- 6 TPs de modélisation
- 1 **TP d'évaluation** avec une évaluation individuelle sur machine et/ou par binôme à l'oral
- 6 séances de projet en groupe
- 1 **évaluation en groupe** avec un oral
- 1 **exam final portant sur toute l'UE**

Table des matières

Table des figures	7
Liste des tableaux	8
Liste des algorithmes	9
1 Transformations d'images	16
1.1 Types de transformation	16
1.2 Transformations ponctuelles d'histogramme	17
1.2.1 Notions d'histogramme	17
1.2.2 Inversion	19
1.2.3 Seuillage/Binarisation	19
1.2.4 Étalement/Modification de la dynamique	21
1.2.5 Égalisation	21
1.3 Transformations locales (Filtrage)	22
1.3.1 Filtre linéaire (Convolution)	23
1.3.2 Filtre non linéaire	26
1.4 Résumé des filtres locaux étudiés	27
1.5 Comparaison avec les filtres ponctuels	27
1.6 Conclusion	28
2 Détection de contours	29
2.1 Algorithme de détection classique	30
2.2 Seuillage/Binarisation	30
2.3 Fermeture	31
2.4 Approche de type Canny et Deriche	32
2.5 Calcul de la dérivée d'une image	33
2.5.1 Dérivées premières	33
2.5.2 Dérivées secondes	36
3 Segmentation	38
3.1 Introduction sur la segmentation en recherche	38
3.2 Définition	39
3.3 Méthodes de segmentation	41
3.4 Approches par contours	42
3.4.1 Des approches classiques aux contours actifs	42
3.4.2 Principe des contours actifs	42
3.4.3 Étapes des approches par contours actifs	43
3.4.4 Énergie à minimiser	44
3.4.5 Choix possibles	44

3.4.6	Forme discrète de l'énergie et algorithme classique de calcul	45
3.4.7	Comment choisir ces différents paramètres ?	45
3.5	Approches par régions	46
3.5.1	Approches par histogramme	46
3.5.2	Fusion/Division	46
3.5.3	Croissance de régions	47
3.5.4	Ligne de partage des eaux	47
3.6	Approches inspirées des approches de classification	49
3.6.1	k -moyennes	49
3.6.2	<i>Mean-Shift</i> [Comaniciu 02]	50
3.6.3	<i>Support Vector Machine</i>	52
3.7	Synthèse et comparaison	53
4	Approche par superpixels	55
4.1	Introduction sur les approches de sursegmentation	55
4.2	Domaines d'application des superpixels	56
4.3	Définition/Propriétés	57
4.3.1	Propriétés d'apparence	57
4.3.2	Propriétés spatiales	58
4.3.3	Propriété d'invariance temporelle	58
4.3.4	Conclusion sur les propriétés	59
4.3.5	Construction/Modélisation	59
4.3.6	Classement et comparaison des différents constructeurs de superpixels	61
4.4	Conclusion sur les superpixels	62
5	Représentation de la couleur	63
5.1	Systèmes de primaires	63
5.2	Systèmes luminance-chrominance	64
5.2.1	Systèmes perceptuellement uniformes	65
5.2.2	Systèmes antagonistes	65
5.2.3	Autres systèmes	66
5.3	Systèmes perceptuels	66
5.4	Systèmes d'axes indépendants	66
5.5	Systèmes hybrides	67
5.6	Synthèse des systèmes de représentation de la couleur	67
6	Étude de la texture	70
6.1	Présentation de quelques opérateurs de texture	70
6.1.1	Densité de points de contour	70
6.1.2	Auto-corrélation	70
6.2	Caractérisation d'une texture par matrice de cooccurrences	71
	Bibliographie	72

Table des figures

1	De la vision humaine à la vision artificielle.	10
2	Domaines abordés par le traitement d'images et la vision par ordinateur.	11
3	Représentation discrète d'une image.	14
1.1	Classement des types de transformations.	16
1.2	Type de transformations en fonction des données prises en compte.	17
1.3	Exemple d'histogramme pour une image donnée.	18
1.4	Deux exemples différents d'images pour un même histogramme.	18
1.5	Illustrations de l'inversion, de la binarisation et de l'étalement d'histogrammes.	21
1.6	Égalisation d'un histogramme.	22
1.7	Comparaison entre étalement et égalisation d'un histogramme.	23
1.8	Application d'un filtre de convolution.	24
1.9	Illustration du choix de la taille d'un filtre gaussien en fonction du σ choisi.	25
1.10	Illustration et comparaison des comportements des filtres médian et conservateur.	27
2.1	Différents types de contours.	29
2.2	Différents types de contours.	30
2.3	Fermeture de contours.	31
2.4	Exemple de codage de Freeman.	32
2.5	Calcul des dérivées premières par différence finie, sans lissage.	33
2.6	Calcul des dérivées premières par différence finie, avec pondération.	34
2.7	Illustration du comportement de l'algorithme 1.	35
2.8	Calcul des dérivées premières avec le filtre de Scharr.	36
3.1	Exemples de segmentation.	39
3.2	Exemples de segmentation.	40
3.3	Illustration d'une approche classique par détection de contours puis fermeture.	43
3.4	Illustration de l'approche par contours actifs.	44
3.5	Illustration du comportement de l'approche de segmentation par Division/Fusion.	46
3.6	Illustration du comportement de l'algorithme par croissance de régions.	47
3.7	Illustration des résultats obtenus avec ligne de partage des eaux.	48
3.8	Exemple de segmentation de papillons par ligne de partage des eaux.	49
3.9	Évolution de la segmentation par k-moyenne.	50
3.10	Illustration du comportement des k-moyennes suivant les critères utilisés.	51
3.11	Application de l'algorithme de <i>Mean-Shift</i> sur différentes images.	52
4.1	Différences de résultats avec différents constructeurs de superpixels.	61

Liste des tableaux

1.1	Résumé des filtres locaux présentés.	27
3.1	Listes des approches présentées.	54
4.1	Classement des constructeurs de superpixels.	61
5.1	Systèmes représentatifs.	68
5.2	Systèmes représentatifs (suite).	69

Liste des algorithmes

1	Algorithme classique de détection de contours.	31
2	Segmentation par contours actifs.	45
3	Algorithme <i>mean-shift</i>	52

Introduction générale

Vision humaine, traitement d'images et vision par ordinateur

La vision par ordinateur a pour but de proposer des fondements et outils permettant d'imiter au mieux le comportement du système visuel humain afin de proposer des outils d'aide à toutes les tâches humaines prenant en compte la vision, cf. figure 1. L'outil de vision humaine est très performant mais il est quand même à souligner qu'il est victime de certaines illusions d'optique. Si nous comparons la machine et l'humain : la machine sait peu de choses mais elle ne se laisse pas influencer par des illusions d'optique. Sachant que 90% de l'information que nous percevons est visuelle, nous comprenons l'intérêt d'obtenir des images de qualité, ou du moins d'avoir un confort visuel suffisant, lorsque nous souhaitons mettre en œuvre un système d'interprétation de scènes et/ou de vision.

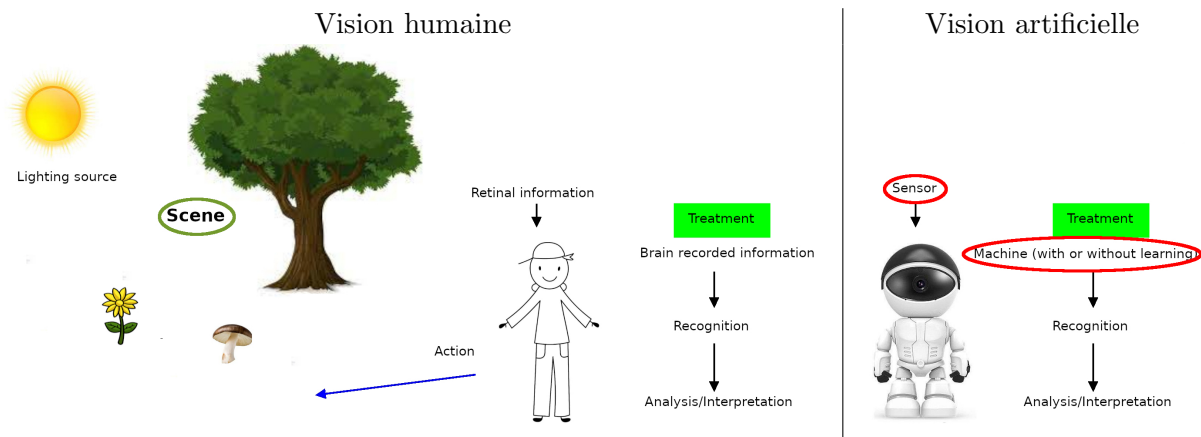


FIGURE 1 – De la vision humaine à la vision artificielle.

Les aspects abordés en traitement d'images et en vision par ordinateur couvrent de très nombreux domaines de l'acquisition d'une scène à sa compréhension. Plus précisément, le **traitement d'images** permet de modifier l'aspect de l'image afin d'en améliorer la visualisation, dans le but d'en extraire une autre information. Un exemple d'amélioration consiste, par exemple, à rehausser le contraste pour mieux faire apparaître les contours, ou encore, augmenter la dynamique des niveaux de gris d'un objet spécifique de la scène pour le mettre en évidence. Parmi les diverses disciplines, la compression et la quantification (réduction du nombre de couleurs utilisées dans l'image) permettent de réduire la quantité d'information contenue dans l'image afin d'en faciliter la transmission. En comparaison, la **vision par ordinateur** concerne l'extraction d'une information sur la scène, comme le relief, la reconnaissance d'un objet particulier, une mesure relative aux objets de la scène. La nuance entre ces deux domaines est assez délicate et, ainsi, certaines disciplines peuvent aussi bien être vues comme du traitement d'images que comme de la vision par ordinateur. Ici, nous proposons de classer les thématiques comme la figure 2.

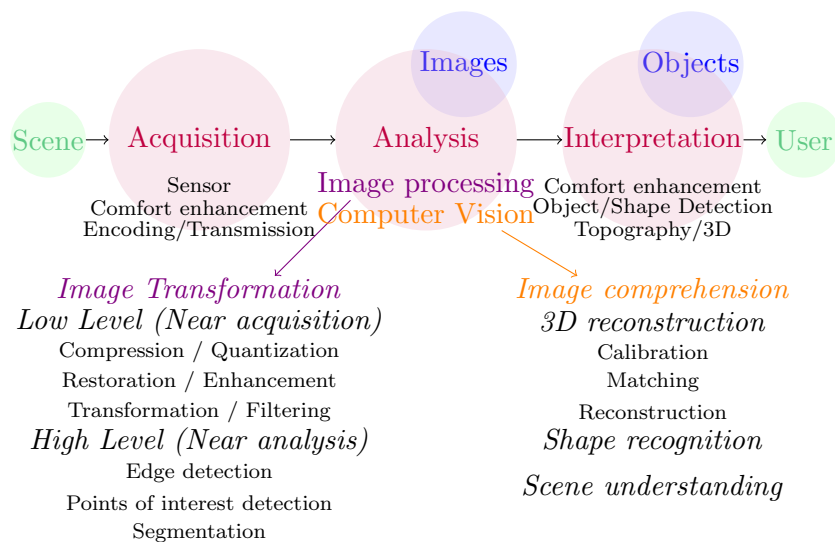


FIGURE 2 – Domaines abordés par le traitement d’images et la vision par ordinateur.

Quelle est la place de l’apprentissage ?

Cette partie permet de faire le lien avec les cours d’apprentissage profond proposé par Axel CARLIER, durant ce semestre. Le début correspond à des rappels par rapport au cours d’Axel CARLIER.

Pour commencer, il semble important de définir ce que nous appelons l’apprentissage supervisé puisque l’apprentissage profond en est un sous-domaine. En effet, il s’agit d’un type d’apprentissage supervisé faisant intervenir des réseaux de neurones. L’**apprentissage supervisé** vise à produire des programmes capables d’effectuer une tâche sans avoir explicitement codé cette tâche [Samuel 59]. Il est dit que le programme apprend de son expérience pour répondre à la tâche seulement s’il possède une mesure de performance qui augmente avec son expérience [Mitchell 97]. L’apprentissage supervisé nécessite donc deux éléments :

- (1) Un ensemble d’apprentissage, c’est-à-dire, la plupart du temps, des données annotées ;
- (2) La construction d’un prédicteur qui va minimiser la différence entre les étiquettes/les valeurs réelles et les étiquettes à prédire¹.

Pour construire ce prédicteur de nombreux algorithmes ont été introduits. Les plus connus sont l’algorithme des k -moyennes ou k -means [Duda 01, chapitre 4] et les machines à vecteur de support, ou *Support Vector Machine* [Duda 01, chapitre 5, page 259] que nous allons aborder rapidement, dans la partie sur la segmentation. Nous pouvons également citer les approches par *boosting* qui combinent différents classifieurs, le plus célèbre étant AdaBoost, *Adaptive Boosting* [Collins 02]. Mais, actuellement, les plus utilisés sont les approches s’appuyant sur des réseaux de neurones.

En informatique, le concept de **neurone** formel, étroitement lié au concept des neurones en biologie n’est pas récent. Il a été introduit par [McCulloch 43]. Ce concept a ensuite été utilisé dans un système d’apprentissage, avec la notion de **perceptron**, dans [Rosenblatt 58]. Un neurone possède plusieurs entrées qui permettent une réponse en sortie. En détail, la sortie d’un neurone correspond à un poids calculé à partir des entrées pondérées. Plus formellement, si nous notons ω_i les poids d’entrée, avec \mathbf{w} le vecteur contenant les poids, nous obtenons la sortie y à partir des entrées x_i , stockées dans le vecteur \mathbf{x} , de la manière suivante :

$$y = f(\mathbf{w}^T \mathbf{x} + b) \quad (1)$$

1. Nous parlons de classification lorsqu’il faut donner une étiquette et nous parlons de régression lorsqu’il faut donner une valeur.

où b correspond à un biais et f la fonction d'activation. Malheureusement, ce modèle de perceptron ne permet pas la résolution de problèmes non-linéaires. C'est ce qui a été présenté dans [Marvin 69] et qui a, entre autres, entraîné un désintérêt pour les réseaux de neurones. Heureusement, dans [Rumelhart 86], ces travaux sont étendus en prenant en compte plusieurs couches et les premiers réseaux opérationnels multicouches apparaissent. Dans un réseau de neurones multicouches, nous distinguons : la couche d'entrée (les données), la couche de sortie (le résultat) et toutes les couches intermédiaires qui sont dites cachées. Classiquement, l'**apprentissage profond** correspond aux réseaux avec au minimum deux couches cachées ou intermédiaires. Avant 2010, comme toute méthode d'apprentissage, les approches à base de neurones sont peu employées car elles nécessitent, entre autres, des calculs importants en terme d'apprentissage. De plus, dans de nombreuses applications, il n'existait pas de base de données annotées, de taille suffisamment importante, pour effectuer correctement la phase d'apprentissage. Récemment, ces deux verrous ont été dépassés grâce à la mise à disposition de moyens d'acquisition de données de moins en moins chers, de plus en plus faciles à manipuler, mais également par la circulation des données de plus en plus simple. Enfin, la puissance de calculs bien supérieure aux puissances possibles à la naissance des premiers réseaux de neurones a également contribué à cet essor.

Une autre avancée majeure qui a permis l'utilisation efficace des réseaux de neurones a été une meilleure compréhension et une meilleure utilisation des **fonctions d'activation**, *learning activation function*. En effet, la fonction d'activation peut être vue comme une fonction de seuillage avec trois parties : une partie non-activée (en dessous du seuil), une partie activée (au dessus du seuil) et une partie de transition (aux alentours du seuil). Ainsi, la phase d'apprentissage consiste à estimer les **poids**, les **biais** et les **paramètres** de la fonction d'activation. Une manière d'estimer la qualité d'un choix de paramètres est simplement d'évaluer la différence entre la réponse attendue et la réponse obtenue. C'est ce qui est appelé la **fonction de coût**, *cost function*. Ainsi, pour estimer les paramètres, l'approche généralement utilisée est une **descente de gradient**. Initialement, la fonction d'activation utilisée était une sigmoïde alors même que le gradient d'une sigmoïde est souvent proche de 0. Cette propriété empêche la propagation du gradient, c'est le problème de l'**evanescence des gradients**, *vanishing gradient*. C'est la raison pour laquelle une nouvelle fonction d'activation a été introduite pour permettre de résoudre ce problème : la fonction ReLU, *rectified Linear Unit* [Nair 10]. Elle est linéaire par morceau : la sortie est nulle en dessous de 0 et sinon la valeur $\mathbf{w}^T \mathbf{x} + b$ introduite dans l'équation (1) est simplement conservée. Le gradient de cette fonction est donc égal à 1 dans la zone activée et résout le problème énoncé plus haut.

Ainsi, les trois éléments cités : fonction d'activation adaptée à la descente de gradient, base de données et puissance de calculs importantes, étant mieux contrôlés et mieux compris, cela a mené au succès de l'apprentissage profond tel que nous le connaissons actuellement [LeCun 15, Goodfellow 16]. Afin d'étudier et d'utiliser des réseaux de neurones profonds, dans un contexte donné (détection, appariement, segmentation), il est important de distinguer les aspects majeurs pour construire un modèle permettant d'obtenir des résultats de qualité, à savoir :

- (1) la base de données utilisée ainsi que
- (2) le type de réseaux utilisés.

Un autre aspect à considérer concerne le type d'architecture qui peut être mise en œuvre :

- **Réseaux de neurones convolutifs**, *Convolutional Neural Network* – Il s'agit des réseaux les plus populaires et les plus anciens [Goodfellow 16, chapitre 9]. Ce qui les caractérise, en premier lieu, est d'effectuer des convolutions successives, qui permettent, suivant le filtre choisi, de mettre en évidence différents aspects (comme par exemple, dans le cadre de ce cours, un filtre de lissage ou filtre de détection de contours) et qui permettent également de réduire progressivement la taille de l'image. Par rapport à ces convolutions, il y a donc deux éléments à définir : le *padding*, la stratégie à appliquer pour filtrer sur les bords de l'image et le *stride*, le pas de convolution à appliquer (il ne s'agit plus d'utiliser une fenêtre glissante comme lorsque

nous lisons une image, comme nous allons le voir dans la suite de ce cours). En deuxième lieu, un réseau convolutif fait intervenir des étapes de *pooling*, le fait de découper l'image en une grille et chaque région de cette grille est représentée par une valeur (classiquement le maximum, d'où la notion de *max-pooling* mais il existe d'autres techniques). Cette étape correspond à un sous-échantillonnage que nous pouvons également retrouver pour d'autres types de réseaux de neurones. Ainsi, un réseau convolutif fait intervenir deux types de couches : les couches de convolution qui sont toujours associées à une couche de *pooling* (et il s'agit des couches les plus profondes) et des couches entièrement connectées, *fully connected*, qui correspondent aux couches les moins profondes et permettent de connecter tous les neurones entre eux.

- **Réseaux de neurones récurrents, *Recurrent Neural Network*, RNN** – La particularité de ces réseaux, introduits dans [Rumelhart 86], est d'être en capacité de prendre en considération des données séquentielles. Plus précisément, une séquence d'entrées notée $[x_1, \dots, x_t]$ va permettre de fournir une séquence de sortie notée $[y_1, \dots, y_t]$, tout en maintenant un état interne $[h_1, \dots, h_t]$. Typiquement, ce type de réseau permet de prendre en compte des séquences d'images, des vidéos, c'est-à-dire, plus généralement, de prendre en compte des données temporelles. Les réseaux récurrents les plus fréquemment utilisés sont les réseaux LSTM, *Long Short Term Memory*, introduits dans [Sepp 97]. Un exemple d'utilisation dans des images est la construction d'une légende de l'image qui permet de décrire la scène suivant les différents éléments qui la composent en indiquant les liens de dépendances, d'interaction entre ces différents éléments.
- **Réseaux antagonistes génératifs, *Generative Adversarial Networks*, GAN** – Ils ont été introduits par [Goodfellow 14]. Dans cette publication, les auteurs ont appliqué ce type de réseaux à la génération de nouvelles images avec un niveau de réalisme élevé. Le principe est de mettre en compétition deux réseaux différents dans un contexte proche de celui de la théorie des jeux : le premier réseau est appelé générateur, et va générer des échantillons ou des données de synthèse, comme une image de synthèse, alors que le second réseau, son adversaire, le discriminateur va tenter de faire la distinction entre une donnée de synthèse et une donnée réelle. Plus le générateur s'améliore, plus il produit des données réalistes et, en contre-partie, le discriminateur a plus de difficultés pour distinguer données réelles des données de synthèse. À l'inverse, plus le discriminateur s'améliore, plus il distingue correctement les données réelles des données de synthèse et, en contre-partie, le générateur fournit des données de synthèse qui s'éloignent des données réelles. Cette situation est assimilable à celle d'un jeu à deux joueurs à somme nulle, d'où le lien avec la théorie des jeux. Il faut donc mettre en place une stratégie où chacun minimise le gain maximal de l'adversaire. Cette définition met en avant la difficulté d'entraîner ce type de réseaux. Les travaux de [Radford 15, Radford 16] donnent un certain nombre de recommandations sur l'utilisation de ce type de réseaux de neurones. Les auto-encodeurs sont une variante des GANs et permettent, notamment, de mettre en place une méthode d'apprentissage sans annotation des données d'apprentissage [Bengio 09]. Il s'agit donc d'un réseau de neurones permettant de faire de l'apprentissage non supervisé. Ce réseau est formé d'un encodeur qui doit permettre d'extraire les caractéristiques principales des données d'entrée et ainsi construire un espace latent. Plus précisément, l'encodeur doit permettre de réduire la dimension en conservant suffisamment d'information discriminante pour qu'un décodeur puisse retrouver la donnée d'entrée. De part cette définition, l'encodeur peut être assimilé à une ACP, Analyse en Composante Principale. Enfin, le décodeur doit être capable de reconstruire la donnée d'entrée à partir de l'espace latent.

Une fois la base de données maîtrisées et le choix d'un réseau réalisé, il reste toutefois de nombreux éléments à déterminer afin de mettre en place le réseau proposé, adapté au problème que nous cherchons à résoudre :

- La méthode d'optimisation et en particulier le nombre d'epoch² et le terme d'inertie ajouté

2. Une epoch correspond à un apprentissage sur toutes les données, c'est-à-dire que l'ensemble d'apprentissage a été

(*momentum*) pour éviter la stabilisation dans un minimum local ;

- Le nombre de couches ;
- Le nombre de neurones.

Plus nous augmentons le nombre de couches et le nombre de neurones, plus nous augmentons le pouvoir de séparation des classes du réseau de neurones. Toutefois, passé un certain nombre de couches et/ou de neurones, le réseau peut être sur-appris³. De même, si le modèle est trop entraîné, cela engendre un sur-apprentissage. Il existe des techniques pour éviter le sur-apprentissage (arrêt anticipé, régularisation, connexions raccourcies). Vous pouvez trouver des détails dans [Goodfellow 16].

Il est important de noter que des tâches en intelligence artificielle peuvent être effectuées en utilisant une méthode d'apprentissage classique, sans faire intervenir de réseaux de neurones profonds. En effet, toutes les tâches où il est évident que nous pouvons identifier explicitement les éléments caractéristiques à connaître et à apprendre n'ont pas besoin de réseaux de neurones profonds. Mais, justement, dans de nombreuses tâches, il est difficile de déterminer précisément les éléments caractéristiques pour effectuer la tâche. Par exemple, lorsque nous souhaitons reconnaître des voitures, les modèles, les formes, les couleurs, sont tellement variés qu'il est difficile de définir les caractéristiques précises à exploiter.

Notations/Définitions

En continu, une image peut être vue comme une fonction, notée :

$$I : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$

$$\mathbf{p} \longmapsto I(\mathbf{p}),$$

où :

- Lorsque $n = 2$, $\mathbf{p} = (x, y)$ est le vecteur des coordonnées du pixel (PICTure ELement) ;
- En 3D, $n = 3$, nous parlons de voxel pour VOlumetric piXEL et $\mathbf{p} = (x, y, z)$.

L'espace d'arrivée varie en fonction du type d'information récupérée :

- En niveau de gris, $m = 1$;
- En couleur, $m = 3$.

Après échantillonnage, en **discret**, nous obtenons donc une image ou un volume, soit, une matrice de $N = L \times C$ éléments. Plus l'échantillonnage est faible, plus la résolution est élevée (images haute résolution) et inversement (basse résolution). Dans la figure 3, nous rappelons donc toutes les notations relatives aux différents cas d'images et de volumes qui peuvent être manipulés.

Type	Représentation	Machine	Oeil humain
BINAIRE	$I : \mathbb{R}^2 \longrightarrow [0; 1]$	noir = 0, blanc = 1	
NIVEAU DE GRIS	$I : \mathbb{R}^2 \longrightarrow [0, 255]$	256 niveaux (1 octet)	64 niveaux
COULEUR	$I : \mathbb{R}^2 \longrightarrow [0, 255] \times [0, 255] \times [0, 255]$	16 millions de couleurs (3 octets)	350000 couleurs
3D	$I : \mathbb{R}^3 \longrightarrow [0, 255]$ OU $[0, 255] \times [0, 255] \times [0, 255]$		

FIGURE 3 – Représentation discrète d'une image.

visité entièrement pour le calcul des gradients.

3. Nous parlons de sous-apprentissage lorsque le modèle appris explique mal l'ensemble d'apprentissage, c'est-à-dire qu'il y a beaucoup d'erreurs. Nous parlons de sur-apprentissage lorsque le modèle appris explique tellement finement les données d'apprentissage qu'il ne peut pas se généraliser à d'autres données.

Dans ce cours, nous utilisons souvent le terme norme et en particulier nous utilisons les normes L_p définies par :

$$L_P((x_1, y_1), (x_2, y_2)) = (|x_1 - x_2|^P + |y_1 - y_2|^P)^{\frac{1}{P}}. \quad (2)$$

Nous avons donc L_1 qui est la somme des différences en valeur absolue et qui est également appelée *city-block*. La norme L_2 est la distance euclidienne.

Chapitre 1

Transformations d'images

Introduction sur la transformation d'images

Les objectifs de la transformation d'images sont multiples mais nous pouvons distinguer lorsque nous souhaitons :

- (1) améliorer la visualisation, comme par exemple le contraste ;
- (2) extraire une information de plus haut niveau permettant de réaliser une analyse, voire une interprétation, de la scène étudiée, comme, par exemple, binariser l'image dans le but de séparer deux régions différentes, un objet d'un fond.

1.1 Types de transformation

Nous pouvons distinguer les approches locales/globales ou les approches spatiales/fréquentielles. Le tableau présenté dans la figure 1.1 permet de résumer l'interaction entre ces deux façons de distinguer les approches.

	SPATIALE	FRÉQUENTIELLE	MORPHOLOGIQUE
<i>Ponctuelle</i>	<i>Look Up Table / Histogrammes</i>	-	-
<i>Locale</i>	Filtrage/Convolution	Filtrage/Convolution	Morphologie mathématique
<i>Globale</i>	Filtrage/Convolution	Fourier	-

FIGURE 1.1 – Classement des types de transformations.

Nous noterons que les approches spatiales concernent aussi bien les approches basées histogrammes, filtrage ou encore convolution, alors, que les approches fréquentielles concernent essentiellement les approches par Fourier. Ce que nous appelons les LUT, *Look Up Table*, sont communément les tables de conversion utilisées pour l'affichage, notamment pour les écrans de télévision. Par exemple, pour corriger le fait que la luminosité de l'écran n'est pas linéaire, c'est ce que nous appelons la correction gamma¹.

Dans la littérature, il est fréquent de distinguer également différents types de filtres comme :

- Passe-bas : atténue le bruit et les détails (lissage) ;
- Passe-haut : accentue les contours et les détails (amplifie le bruit) ;
- Passe-bande : compromis.

1. Autrement dit, le flux d'électrons que produit le canons à électrons d'un écran n'a pas une réponse linéaire. Et souvent pour corriger cette variation une correction est appliquée. Souvent, il s'agit d'une correction gamma.

Plus précisément, les différentes transformations locales, ponctuelles et globales sont notées comme dans la figure 1.2.

Si nous notons I' l'image transformée et T la transformation :

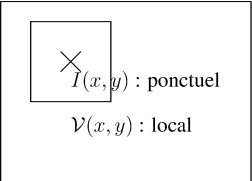
- **Ponctuelle** : $I'(x, y) = T(I(x, y))$
- **Locale** : $I'(x, y) = T(\mathcal{V}(x, y))$ avec $\mathcal{V}(x, y)$, voisinage de $\mathbf{p}(x, y)$

Voisinage : Ensemble de pixels \mathbf{p}' à une distance d donnée et tel qu'il existe un chemin de \mathbf{p} à \mathbf{p}' tel que les pixels qui le constituent appartiennent à $\mathcal{V}(x, y)$. Autrement dit, cela signifie que nous pouvons relier des points entre eux entre les deux pixels, tels que tous ces points appartiennent à la même région que les deux pixels. Nous abordons ici la notion de **connexité**. La distance d peut être :

- une distance euclidienne (L_2) ;
- de type *city-block* (L_1) ;
- ou encore *chess board* ou distance de Tchebychev :

$$L_\infty = \max(|x - x'|, |y - y'|)$$

- **Globale** : $I'(x, y) = T(I)$



$I : \text{global}$

FIGURE 1.2 – Type de transformations en fonction des données prises en compte – Il faut retenir que la plupart des approches de transformations dites locales font intervenir une **transformation de l'histogramme**. De plus, les approches globales sont le plus souvent liées à la transformée de Fourier. Chaque pixel de l'image résultat est l'application d'un filtre donné sur l'ensemble des pixels. Ce qui varie d'un pixel à l'autre, c'est le calcul qui est réalisé.

1.2 Transformations ponctuelles d'histogramme

1.2.1 Notions d'histogramme

Définition 1

- Un histogramme fournit le nombre de pixels pour chaque niveau de gris, sachant que l'image possède N pixels et que nous notons N_{\max} , le niveau de gris maximal :

$$H : [0, N_{\max}] \longrightarrow [0, N]$$

$$i \longmapsto H(i) \text{ où } H(i) = \#\{(x, y) | I(x, y) = i\}$$

- **Utilisation**
 - Cela permet d'étudier la dynamique de l'image (le **contraste**).
 - Cela donne une vue d'ensemble de la **distribution des niveaux de gris**.
 - Son analyse doit permettre de mettre en évidence les **populations significatives dans l'image**. Par exemple, un histogramme avec des valeurs très peu « étalées » sera synonyme d'image peu contrastée. Dans l'exemple de la figure 1.3, nous pouvons penser qu'il est possible d'identifier les différentes populations présentes.
 - **Par contre, l'histogramme ne permet pas de localiser les populations d'intérêt.** En effet, avec les deux exemples d'images de la figure 1.4, nous obtiendrons le même histogramme.

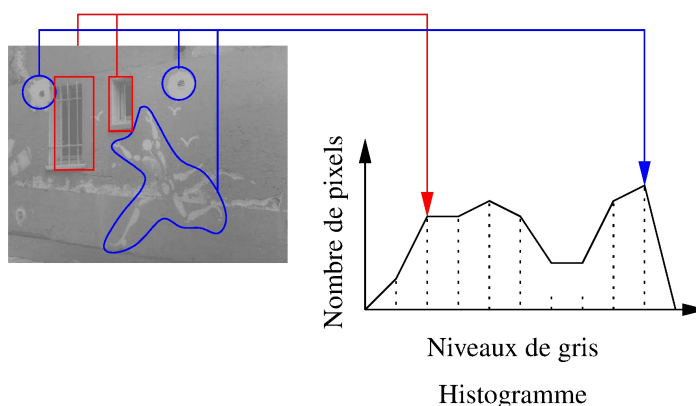


FIGURE 1.3 – Exemple d’histogramme pour une image donnée. Ceci sert à illustrer la définition, mais, il ne s’agit pas de l’histogramme réel que nous étudierons dans la figure 1.7.

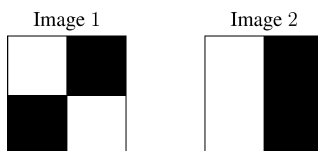


FIGURE 1.4 – Deux exemples différents d’images pour un même histogramme.

Définition 2

Un **histogramme normalisé** est défini de la manière suivante :

$$H_n(i) = \frac{H(i)}{N}.$$

Il représente une densité de probabilité. Lorsque nous souhaitons appliquer un algorithme défini dans un **cadre probabiliste**, nous utiliserons un histogramme normalisé.

Définition 3

- Un **histogramme cumulé** correspond au nombre de pixels inférieurs à chaque niveau de gris.

$$H_c : [0, N_{\max}] \longrightarrow [0, N]$$

$$i \longmapsto H_c(i) \text{ où } H_c(i) = \sum_{k=0}^i H(k) \text{ (relation avec l’histogramme).}$$

Un histogramme cumulé est bien sûr toujours croissant.

- **Propriété**

- $H_c(N_{\max}) = \sum_{k=0}^{N_{\max}} H(k) = N$

- **Propriétés de transition**

- $\begin{cases} H_c(0) = H(0) \\ H_c(i) = H_c(i-1) + H(i) \quad \forall i \in [1, N_{\max}] \end{cases}$
- $\begin{cases} H(0) = H_c(0) \\ H(i) = H_c(i) - H_c(i-1) \quad \forall i \in [1, N_{\max}] \end{cases}$

Pour effectuer certaines transformations de l'image, l'histogramme cumulé permettra plus facilement d'écrire la transformation associée.

1.2.2 Inversion

La transformation utilisée est très simple : $T(i) = N_{\max} - i$, avec, pour rappel, N_{\max} , le niveau de gris maximal. Cette transformation a surtout un intérêt purement visuel.

1.2.3 Seuillage/Binarisation

La transformation est également assez simple : $T(i) = \begin{cases} 1 & \text{Si } i > S \\ 0 & \text{Sinon.} \end{cases}$, avec S , un seuil à choisir.

La manière de choisir le seuil est délicate. Lorsque la recherche de seuil (pour la binarisation) prend en compte une information globale sur l'image, comme une moyenne, nous pouvons alors classer le seuillage dans les transformations globales. En `matlab`, c'est la fonction `im2bw` (qui effectue le seuil) qui peut être utilisée, combinée à la fonction `graythresh` qui estime un seuil en utilisant la méthode d' [Otsu 79], détaillée dans le paragraphe suivant. Plus précisément, nous donnons des détails pour deux approches pour déterminer le seuil : l'approche d' [Otsu 79] et l'approche par mélange de gaussiennes.

Approche d' [Otsu 79]

Dans cette méthode, nous faisons l'hypothèse que nous souhaitons avoir **la variance de chaque classe la plus faible possible** et, à l'inverse, **la variance entre toutes les classes la plus importante possible**. Cette intuition se vérifie mathématiquement. Pour cela, voici les éléments importants à définir :

- $p_i = H_n(i)$ avec $p_i \geq 0$ et donc $\sum_i p_i = 1$.
- C_0 et C_1 les deux classes d'objets que nous cherchons à séparer par un seuil k .
- Ainsi, la probabilité de chaque classe est :

$$w_0 = w(k) = \sum_{i=0}^k p_i \text{ et} \quad (1.1)$$

$$w_1 = 1 - w(k) = \sum_{i=k+1}^{N_{\max}} p_i \quad (1.2)$$

Nous pouvons assimiler ces probabilités aux poids de chacune des classes.

- Les moyennes pondérées respectives sont notées :

$$\mu_0 = \frac{1}{w_0} \sum_{i=0}^k ip_i = \frac{\mu(k)}{w(k)}, \text{ soit } \sum_{i=0}^k ip_i = \mu(k) \quad (1.3)$$

$$\mu_1 = \frac{1}{w_1} \sum_{i=k+1}^{N_{\max}} ip_i = \frac{\mu_T - \mu(k)}{1 - w(k)}, \quad (1.4)$$

$$(1.5)$$

avec μ_T la moyenne des niveaux de gris sur l'ensemble de l'image.

Nous pouvons facilement vérifier que : $w_0\mu_0 + w_1\mu_1 = \mu_T$ avec $w_0 + w_1 = 1$. De plus, nous notons les variances de chaque classe :

$$\sigma_0^2 = \frac{1}{w_0} \sum_{i=0}^k (i - \mu_0)^2 p_i \quad (1.6)$$

$$\sigma_1^2 = \frac{1}{w_1} \sum_{i=k+1}^{N_{\max}} (i - \mu_1)^2 p_i. \quad (1.7)$$

Par définition, la variance intra-classe, la variance des éléments au sein d'une même classe, que nous cherchons à minimiser, est donnée par :

$$\sigma_{intra}^2 = w_0\sigma_0^2 + w_1\sigma_1^2. \quad (1.8)$$

Par définition, la variance inter-classe, la variance des éléments entre les deux classes est donnée par :

$$\sigma_{inter}^2 = w_0(\mu_0 - \mu_T)^2 + w_1(\mu_1 - \mu_T)^2 = w_0w_1(\mu_1 - \mu_0)^2 \quad (1.9)$$

De plus, nous avons la propriété suivante :

$$\sigma_{intra}^2 + \sigma_{inter}^2 = \sigma_T^2, \quad (1.10)$$

où σ_T^2 est une constante et c'est pourquoi **minimiser la variance intra classe revient à maximiser la variance inter classe**. Et, finalement nous devons trouver k tel que :

$$k = \operatorname{argmax}_k [w_0(\mu_0 - \mu_T)^2 + w_1(\mu_1 - \mu_T)^2] = \operatorname{argmax}_k [w_0w_1(\mu_1 - \mu_0)^2]. \quad (1.11)$$

Le terme à minimiser est la définition mathématique de la **variance inter classe** et c'est équivalent à maximiser la variance intra classe qui correspond à la seconde expression dans cette formule.

L'algorithme à appliquer pour réaliser le calcul du seuil par Otsu peut être très simple : il suffit de parcourir l'ensemble des seuils possibles et de conserver celui qui maximise la variance inter classe. Cela reste rapide puisque qu'il n'y a que 256 niveaux de gris. Toutefois, pour que l'exécution ne soit pas coûteuse, il faut faire attention à modifier les moyennes et variances successives par un simple jeu d'ajouts/suppressions des valeurs ajoutées/supprimées.

Approche par mélange de gaussienne

Une autre approche de seuillage, très utilisée dans le domaine de la segmentation, cf. chapitre 3.3, est l'approche par mélange de gaussiennes ou *Gaussian Mixture Model*, GMM, en utilisant l'algorithme d'Espérance-Maximisation, EM. Pour déterminer le seuil, il va falloir estimer les paramètres suivants : moyennes, variances et amplitudes de chaque distribution gaussienne.

L'histogramme d'une image bimodale peut être vu comme l'estimation de la fonction densité de probabilité de l'intensité : $p(x)$. Cette densité est la superposition de deux densités unimodales correspondant aux deux régions à segmenter : $p(x) = P_1p_1(x) + P_2p_2(x)$, avec p_1 et p_2 , deux densités de probabilité gaussiennes, centrées respectivement en μ_1 et μ_2 (avec $\mu_1 < \mu_2$), et d'écart type respectif σ_1 et σ_2 . Nous avons bien sûr $P_1 + P_2 = 1$. Un seuil T peut être choisi de manière à séparer les deux modes. L'erreur de classification faite sera alors, avec E_1 , la probabilité de se tromper en classant 2 en 1, et *vice versa* : $E(T) = P_2E_1(T) + P_1E_2(T)$.

On cherche à minimiser cette erreur de la même manière sur chaque classe, on part donc de la contrainte suivante :

$$P_2E_1(T) = P_1E_2(T). \quad (1.12)$$

Le résultat de cette équation passe par la résolution d'un polynôme du second degré. En supposant des variances égales, c'est-à-dire, $\sigma_1 = \sigma_2 = \sigma$, une manière classique de calculer ce seuil est donné par :

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 + \mu_2} \ln \left(\frac{P_2}{P_1} \right). \quad (1.13)$$

Cette expression se calcule en remplaçant dans l'expression (1.12) les termes E_i par la distribution gaussienne (dont une définition est donnée au § 1.3.1).

1.2.4 Étalement/Modification de la dynamique

La transformation est déterminée en suivant les étapes suivantes :

- (1) Nous avons : $i \in [I_{\min}, I_{\max}]$ et nous souhaitons avoir : $i \in [0, N_{\max}]$.
- (2) Nous supposons une transformation affine : $T(i) = ai + b$.
- (3) Nous savons que $T(I_{\min}) = 0$ et $T(I_{\max}) = N_{\max}$.
- (4) Nous obtenons : $T(i) = \frac{N_{\max}}{I_{\max} - I_{\min}}(i - I_{\min})$.

Pour le recadrage de la dynamique, nous effectuons le même raisonnement par morceaux (type d'objets). L'étalement préserve la forme de l'histogramme. Il permet de réhausser le contraste, tout en préservant le degré d'importance/de saillance de chaque objet présent dans la scène.

Les trois transformations abordées sont illustrées dans la figure 1.5.

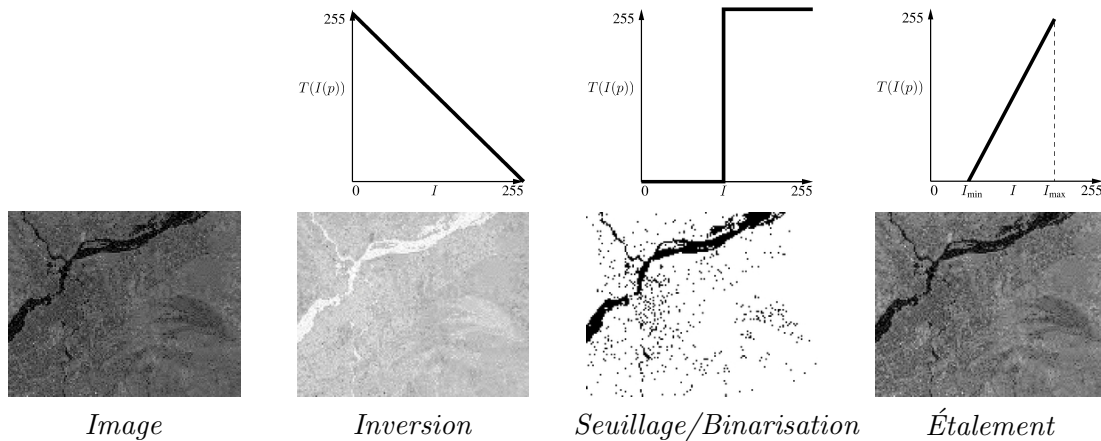


FIGURE 1.5 – Illustrations de l'inversion, de la binarisation et de l'étalement d'histogrammes.

1.2.5 Égalisation

L'objectif est d'obtenir une image où chaque objet² obtient le même niveau de contraste qu'il soit faiblement ou fortement représenté dans l'image. Nous souhaitons donc modifier l'histogramme comme illustré dans la figure 1.6.

Nous nous appuyons sur une transformation de l'histogramme cumulé, car cela se formule plus facilement :

2. Par objet, nous parlons des différents éléments importants qui composent la scène.

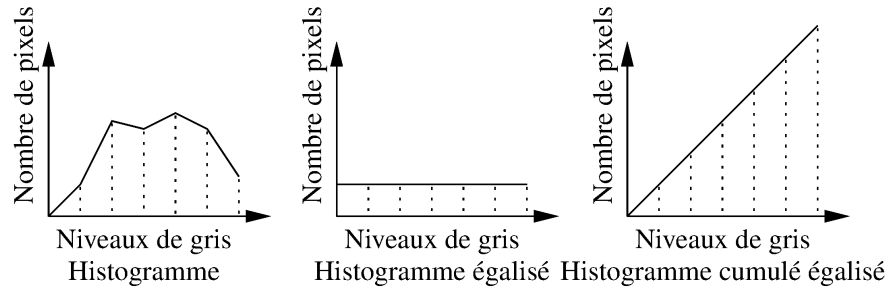


FIGURE 1.6 – Égalisation d'un histogramme.

- (1) Nous cherchons T telle que : $H'_c(T(i)) = H_c(i)$ où H'_c est l'histogramme souhaité.
- (2) De plus, nous savons que $H'_c(i) = \frac{N}{N_{\max}} \times (i + 1)$ où N est le nombre de pixels de l'image et N_{\max} le nombre de niveaux de gris.
- (3) Ainsi, nous obtenons $H'_c(T(i)) = \frac{N}{N_{\max}} \times (T(i) + 1)$.
- (4) Et nous pouvons en déduire que $\frac{N}{N_{\max}} (T(i) + 1) = H_c(i)$
- (5) D'où : $T(i) = \begin{cases} 0 & \text{Si } H_c(i) < \frac{N}{N_{\max}} \\ \frac{N_{\max} H_c(i)}{N} - 1 & \text{Sinon} \end{cases}$

Problème : En théorie, T est non définie si $H_c(i) < \frac{N}{N_{\max}}$ (cas négatif), d'où le fait d'écrire : $T(i) = 0$ dans ce cas.

À l'inverse de l'étalement, l'égalisation ré-hausse le contraste en faisant apparaître chaque population avec la même importance. Ainsi, les détails ressortent. L'histogramme égalisé n'est pas vraiment égal à celui désiré, cela vient de la perte due au problème mentionné sur les cas où T n'est pas définie. La figure 1.7 montre un exemple des différences obtenues après un étalement ou une égalisation, nous illustrons également le fait que l'histogramme obtenu après égalisation n'est pas parfaitement ressemblant à celui exposé dans la figure 1.6.

1.3 Transformations locales (Filtrage)

Introduction sur le filtrage local

Contrairement aux transformations ponctuelles, les transformations locales prennent en compte un certain voisinage de chaque pixel pour effectuer la transformation. Le principe est que le niveau de gris du pixel devient une somme pondérée des niveaux de gris de ses voisins. Les objectifs d'un filtrage sont multiples, mais, nous pouvons citer principalement :

- (1) l'amélioration de l'affichage ;
- (2) l'interprétation de l'image.

En général, nous distinguons les **filtres linéaires** qui correspondent à la notion de **convolution** en mathématique des **filtres non-linéaires** qui ne peuvent pas s'exprimer comme une convolution. Dans certains documents (livres, articles), nous pouvons aussi parler de noyau, et vous pouvez trouver l'équivalent en anglais, avec *filter/kernel*.

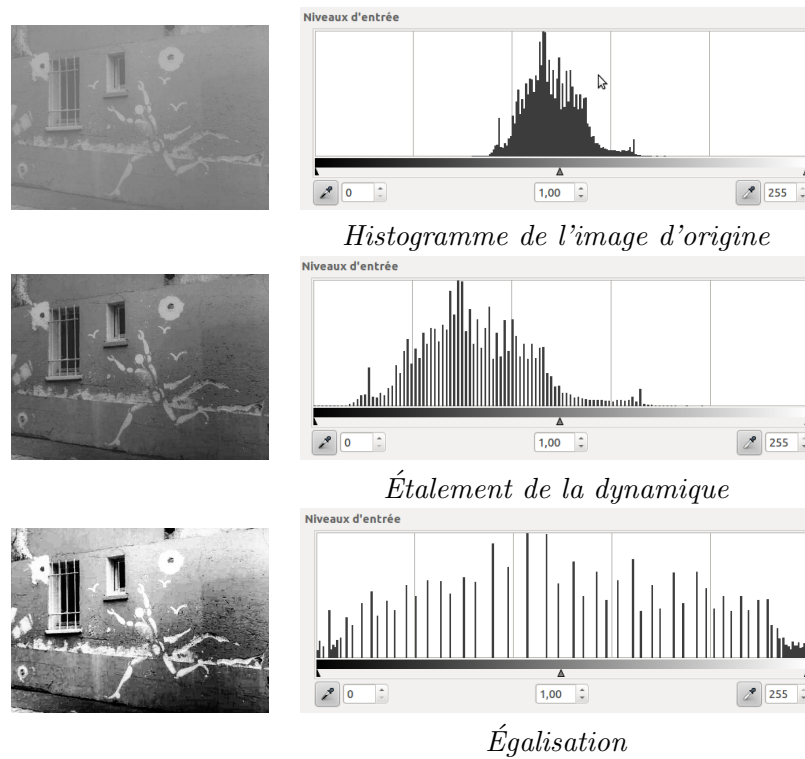


FIGURE 1.7 – Comparaison entre étalement et égalisation d’un histogramme – Pour l’étalement de la dynamique, il est important de comprendre que le nombre de niveaux de gris utilisés va rester constant, c’est seulement les valeurs minimales et maximales qui vont être étalées. D’où un histogramme étalé qui comporte des manques réguliers, c’est-à-dire des valeurs de niveaux de gris où il y a 0 pixels ayant ce niveau. Pour l’égalisation, c’est exactement la même chose, d’où cet aspect clairsemé. De plus, la transformation calculée n’étant pas définie pour certaines valeurs de niveaux de gris, certains niveaux de gris sont perdus et nous obtenons donc des valeurs qui ne sont pas strictement égales d’un niveau de gris à l’autre.

1.3.1 Filtre linéaire (Convolution)

Définitions

En continu, la convolution d’une image I par un filtre linéaire f est définie de la manière suivante :

$$I'(x, y) = (f * I)(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x', y') I(x - x', y - y') dx' dy'. \quad (1.14)$$

Il faut que $\sum f(x, y) = 1$ si nous souhaitons que la dynamique de l’image doit être conservée (niveaux de gris entre 0 et 255). De plus, la dynamique de l’image peut être recalée entre 0 et N_{\max} pour mieux visualiser le résultat. Par contre, cela pose le problème de devoir tronquer les valeurs entières.

Ce qui en discret donne la formulation suivante :

$$I'(x, y) = \sum_{x'=-k}^k \sum_{y'=-l}^l F(k - x', l - y') I(x + x', y + y'). \quad (1.15)$$

où F est le masque de convolution ou la version discrète de la **réponse impulsionnelle du filtre défini sur la fenêtre de taille** $(2k + 1) \times (2l + 1)$.

Ainsi, en pratique, l’application d’un filtre de convolution est réalisée comme illustré dans la figure 1.8.

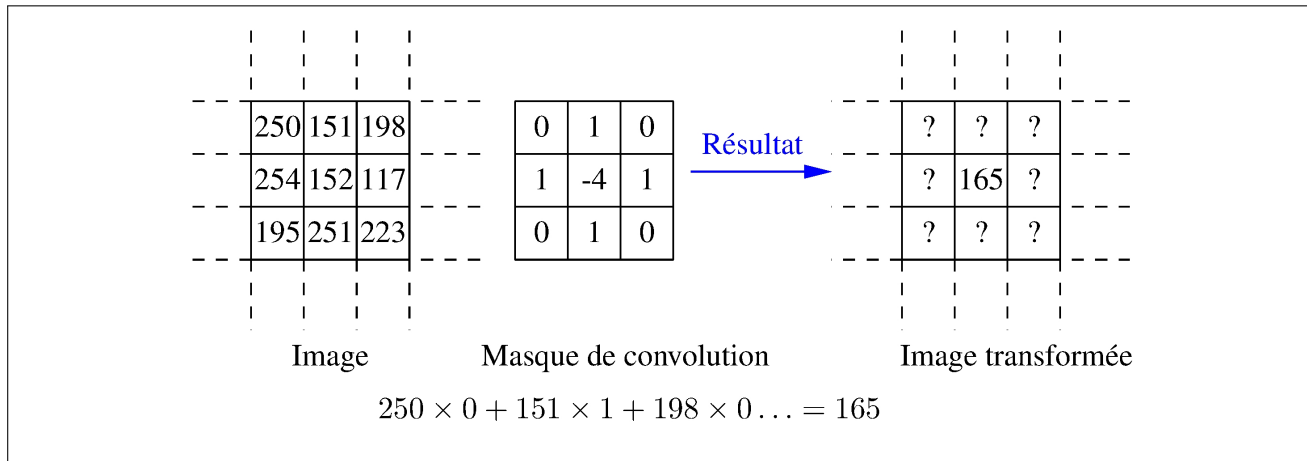


FIGURE 1.8 – Application d’un filtre de convolution.

Voisinage et bordure

Nous faisons intervenir la notion de voisinage dans cette définition où il s’agit bien de définir une propriété de liaison qui fait que nous considérons que deux pixels font partie de la même région. Généralement, deux types de critères peuvent être utilisés : un critère de similarité (niveaux de gris, couleur) ou un critère d’adjacence. Dans le cas de la convolution, c’est uniquement un critère d’adjacence.

Cet exemple nous permet également d’introduire le fait qu’il est nécessaire d’établir une stratégie particulière sur les bords de l’image. Voici ce que la littérature propose :

- Les pixels des bords ne sont pas traités. Ainsi, si $L \times C$ est la taille de l’image, avec L le nombre de lignes de l’image et C le nombre de colonnes, le résultat aura la taille $(L - N_l + 1) \times (C - N_c + 1)$, avec N_l le nombre de lignes du filtre et N_c le nombre de colonnes du filtre.
- Une bordure est ajoutée sur les bords pour ne pas avoir de problème de gestion des bords, ce qui revient à adapter la forme du filtre aux bords. La taille du résultat a la taille de l’image. Cette bordure peut être : de valeur nulle, tirée aléatoirement ou une répétition des bords de l’image.
- Les calculs sont effectués de manière circulaire. La taille du résultat a la taille de l’image.

Il existe d’autres approches, moins fréquentes dans la littérature, comme par exemple l’adaptation du filtre au bord de l’image. Cela signifie que la taille du masque est adaptée de manière à ce que le filtre soit bien inclus dans l’image.

Nous allons présenter quelques filtres très classiques.

Il existe en matlab la fonction `conv2(I,F,option)` qui réalise la convolution de l’image `I` par le filtre `F` (attention, l’ordre des arguments est important) en suivant l’option `option` qui peut être :

- **valid** où les bords ne sont pas traités et ainsi la taille du résultat est plus petite que la taille de l’image originale ;
- **same** où des zéros sont ajoutés sur les bords, de manière à avoir la taille du résultat exactement la même que la taille de l’image originale. Cette option permet d’effectuer un remplissage ou *padding* de manière à obtenir une image de même taille, soit, $\frac{N_c - 1}{2}$ colonnes de 0 et $\frac{N_l - 1}{2}$ lignes de 0 sont ajoutées sur chaque bord de l’image. Ainsi, au total N_c colonnes et N_l lignes sont ajoutées.
- **full** où une réponse est donnéemême pour les bordures ajoutées en fournissant un résultat de taille plus grande que la taille de l’image originale. Comme pour l’option **same**, la code matlab ajoute des 0 sur la bordure.

Filtre moyennneur

Très simplement, voici son expression :

$$F = \frac{1}{N_l \times N_c} \mathbf{1}_{N_l \times N_c}.$$

La force du lissage dépend de la taille du filtre et nous pouvons remarquer que tous les pixels ont la même influence sur le résultat obtenu.

- **Intérêts** : atténue les bruits, la texture (lissage).
- **Inconvénients** : images « floues », atténue les petits détails et reste sensible à un bruit de type « poivre et sel »³.

Filtre gaussien

Le filtrage gaussien permet d'effectuer une moyenne pondérée « plus subtile » que le filtre moyennneur. Comme le moyennneur, il peut être effectué en 1D (filtre directionnel suivant les lignes ou les colonnes) ou en 2D, et voici la formulation utilisée :

- **1D**

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

- **2D**

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}},$$

avec μ, μ_x, μ_y : moyennes et σ : écart type.

- Souvent $\mu, \mu_x, \mu_y = 0$ et σ dépend de la taille du masque et $\sigma = \frac{N_m}{4}$ où N_m : taille du masque, cf. illustration dans la figure 1.9.

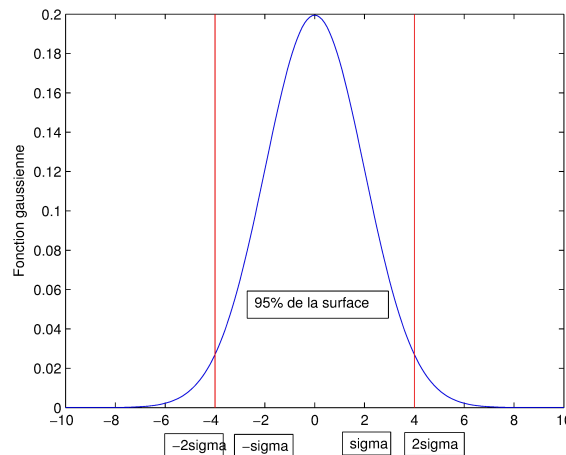


FIGURE 1.9 – Illustration du choix de la taille d'un filtre gaussien en fonction du σ choisi.

Ainsi, voici deux exemples de masque gaussien respectivement en 1D avec une taille de 5 et en 2D avec une taille de 3×3 :

3. Il s'agit d'un bruit où aléatoirement dans l'image l'intensité d'un pixel est remplacé par la valeur 255 (le sel) ou 0 (le poivre). Ce bruit est souvent lié à un problème de transmission au moment de l'acquisition des données qui peut être lié à un matériel défectueux.

$$F'_5 = (f(-2) \quad f(-1) \quad f(0) \quad f(1) \quad f(2)) \text{ et } F_{3,3}^x = \begin{pmatrix} f(-1, -1) & f(-1, 0) & f(-1, 1) \\ f(0, -1) & f(0, 0) & f(0, 1) \\ f(1, -1) & f(1, 0) & f(1, 1) \end{pmatrix} \quad (1.16)$$

Si nous supposons que $\mu = \mu_x = \mu_y = 0$, alors nous avons $f(x) = f(-x) \forall x$ et $f(x, y) = f(-x, -y) \forall x, \forall y$ et ces filtres peuvent se simplifier par :

$$F_5^i = (f(2) \quad f(1) \quad f(0) \quad f(1) \quad f(2)) \text{ et } F_{3,3}^x = \begin{pmatrix} f(1, 1) & f(0, 1) & f(1, 1) \\ f(0, 1) & f(0, 0) & f(0, 1) \\ f(1, 1) & f(0, 1) & f(1, 1) \end{pmatrix}. \quad (1.17)$$

Il est à noter que ce filtre n'est pas normalisé, c'est-à-dire que l'image obtenue ne sera plus comprise entre 0 et N_{max} . Pour cela, il faudrait diviser par la somme des éléments de F'_5 et $F_{3,3}^x$ respectivement. Soit :

$$Fnorm'_5 = \frac{1}{f(0) + 2f(1) + 2f(2)} F'_5 \text{ et } Fnorm^x_{3,3} = \frac{1}{f(0, 0) + 4f(0, 1) + 4f(1, 1)} F_{3,3}^x. \quad (1.18)$$

Sur ce filtre, nous pouvons faire les commentaires suivants :

- **Intérêts** : atténue les bruits (si nous supposons la présence d'un bruit blanc gaussien, par exemple), la texture (lissage).
- **Inconvénients** : images « floues », atténue les petits détails et reste sensible à un bruit de type « poivre et sel ».

1.3.2 Filtre non linéaire

Par opposition aux filtres linéaires, ces filtres ne sont pas applicables par convolution.

Médian

Les filtres linéaires présentés ne permettent pas d'éliminer facilement des bruits de type « poivre et sel ». Le filtre médian permet de répondre à ce problème et voici sa définition :

$$I'(x, y) = \text{med}\{\mathcal{V}(x, y)\}$$

Ce filtre possède les **propriétés suivantes** :

- (1) Un pixel **non représentatif dans le voisinage n'affectera pas** la valeur médiane.
- (2) **Les niveaux de gris isolés sont « adoucis »** (ceux ayant un niveau de gris très différent de leurs voisins).
- (3) Il n'y a **pas de création de nouveaux niveaux de gris**.

En revanche, ce filtre a pour défaut de supprimer les coins, cf figure 1.10 pour l'illustrer. En effet, en présence d'un coin, dans le voisinage pris en compte : plus de la moitié des pixels appartient au fond et un peu moins de la moitié à l'objet dont nous étudions le pixel se situant sur le coin. Ainsi, si nous prenons la médiane, son niveau de gris appartiendra au niveau du gris du fond, et non au coin. Nous remplacerons ainsi le niveau de gris du coin par le niveau de gris du fond.

FILTRE	Moyenne	Gaussien
PRINCIPE	Moyenne des niveaux de gris	Moyenne pondérée des niveaux de gris
INTÉRÊTS	Atténue bruits, texture (lissage)	Plus doux que la moyenne
INCONVÉNIENTS	Flou, atténue petits détails	Moins importants que la moyenne
FILTRE	Médian	Conservateur
PRINCIPE	Médiane des niveaux de gris	Médiane tronquée
INTÉRÊTS	Atténue bruit impulsif	Plus doux que la médiane
INCONVÉNIENTS	Supprime les coins	Non robuste aux images très bruitées

TABLE 1.1 – Résumé des filtres locaux présentés.

Conservateur

Le filtre conservateur est une version « plus douce » du filtre médiane défini par :

$$I'(x, y) = \begin{cases} I_{\min}(x, y) & \text{Si } I(x, y) < I_{\min} \\ I_{\max}(x, y) & \text{Si } I(x, y) > I_{\max} \\ I(x, y) & \text{Sinon.} \end{cases} \quad (1.19)$$

où $I_{\min/\max}(x, y) = \min / \max(\mathcal{V}(x, y))$ et $I(x, y) \notin \mathcal{V}(x, y)$.

- **Intérêt** : Mieux conserver les détails, les contours, pas de flou.
- **Inconvénient** : Fonctionne moins bien lorsqu'il y a trop de pixels erronés dans le voisinage.

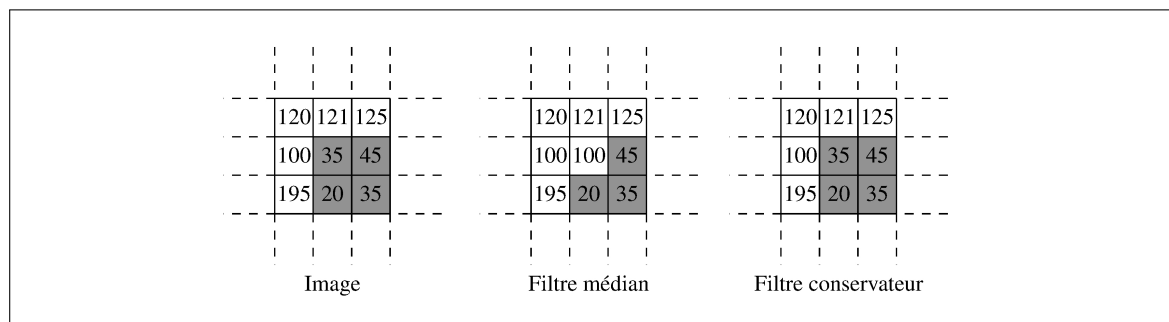


FIGURE 1.10 – Illustration et comparaison des comportements des filtres médian et conservateur.

1.4 Résumé des filtres locaux étudiés

Nous présentons dans le tableau 1.1 un résumé de tout ce qui est important à retenir sur les filtres locaux que nous avons étudiés dans ce chapitre.

1.5 Comparaison avec les filtres ponctuels

Une transformation ponctuelle permet d'étudier la dynamique de niveaux de gris de l'image, grâce à l'histogramme, et de modifier cette dynamique, notamment en améliorant le contraste de l'image alors qu'une transformation locale permet de traiter plutôt les bruits dans l'image pour en améliorer la visualisation. De plus, les filtres locaux permettent d'extraire une information de plus haut niveau

dans l'image, à savoir les dérivées qui ensuite nous permettent d'extraire les contours. C'est ce que nous étudions dans le chapitre sur la détection de contours.

1.6 Conclusion

Ce chapitre nous a permis d'introduire la notion de filtrage pour transformer/améliorer la qualité des images et nous avons abordé les filtres les plus classiques. Les notions abordées sont loin d'être exhaustives et nous avons omis notamment de parler de morphologie mathématique. Un certain nombre d'explications sur cette thématique peut être trouvé dans [\[Serra 94\]](#).

Chapitre 2

Détection de contours

Définition et notion de dérivée

Là aussi, il n'existe pas de définition universelle mais nous proposons d'utiliser cette définition :

Un contour est une frontière qui sépare deux objets/entités. En **pratique**, cela se traduit par une variation d'intensité.

Différentes conditions dans la scène peuvent amener à observer des contours, suivant la définition que nous venons de donner, dans l'image. Plus précisément, les contours observés dans l'image acquise peuvent être dus à la délimitation de la forme des objets, des discontinuités de profondeur, des changements de textures ou d'illumination.

En théorie, si nous étudions le profil d'intensité d'un contour, nous supposons qu'il peut prendre les trois formes présentées dans la figure 2.1.

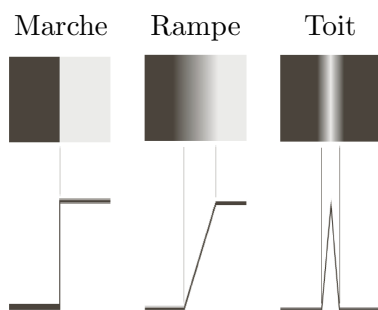


FIGURE 2.1 – Différents types de contours – Nous illustrons ici les hypothèses que nous pouvons faire sur le profil d'intensité que nous pouvons observer au niveau d'un contour. Il s'agit de la théorie. En pratique, dans de nombreux cas, nous pouvons observer une rampe alors même que nous avons plutôt une transition franche. Cela dépendra de la qualité du capteur et de la qualité de l'acquisition. Ce qui est important à retenir : les hypothèses de contours de type marche ou toit sont intéressantes car nous pouvons utiliser des outils de dérivées pour les estimer.

Parmi les trois formes possibles, la plupart du temps, **seuls les contours de type marche ou toit sont utilisés avec une préférence pour les contours de type marche**. Ces deux types de contours sont favorisés par rapport au contour de type rampe, car, les profils des dérivées sont facilement exploitables pour déterminer les points de contour. Étant donné que nous supposons que le profil d'intensité d'un contour suit soit un modèle « marche » ou « toit », nous faisons l'hypothèse que si nous arrivons à calculer la dérivée des images, l'analyse de cette dérivée nous permettra de détecter les contours, comme illustré sur la figure 2.2. En effet, nous savons que :

- Pour un contour de type marche, nous sommes en présence d'un maximum local de la dérivée première et d'un passage par zéro de la dérivée seconde.
- Pour un contour de type toit, nous observons un passage par zéro de la dérivée première et un minimum local de la dérivée seconde.

Lorsque nous faisons l'hypothèse du respect d'un des deux modèles, nous détecterons moins bien les contours qui suit le deuxième modèle. Pour expliquer cette constatation, reprenons un exemple concret : nous supposons des contours de type toit dans l'image. Ainsi, la dérivée première indiquera un passage par zéro, la dérivée seconde un minimum local. Nous pouvons choisir d'utiliser le calcul de la dérivée première et dans ce cas, nous allons chercher à déterminer les passages par zéro dans le résultat de convolution. Le résultat de convolution est l'image obtenue après application de la convolution avec un filtre de calcul de dérivée première. Une autre option est de choisir d'utiliser le calcul de la dérivée seconde et dans ce cas, nous allons chercher à déterminer les minimum dans le résultat de convolution. Et dans ce second cas, le résultat de convolution est obtenu en appliquant un filtre de calcul de dérivée seconde. Dans les deux cas, si dans l'image de départ, nous avons plutôt des contours de type marche, et non de type contour, alors, tous ces calculs seront moins adaptés et nous risquons de ne pas bien détecter les contours présents dans l'image. Par la suite, nous exposons les traitement pour modèle de type marche mais il est important de noter que tout le raisonnement que nous proposons est strictement généralisable au cas des contours de type toit.

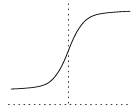
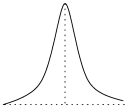
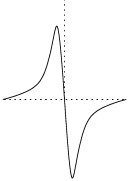
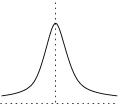
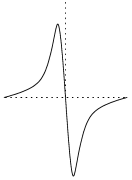
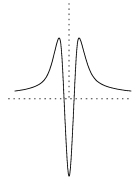
Profil	Dérivée première	Dérivée seconde
<p>Marche </p>	<p>Maximum </p>	<p>Passage par zéro </p>
<p>Toit </p>	<p>Passage par zéro </p>	<p>Minimum </p>

FIGURE 2.2 – Différents types de contours.

2.1 Algorithme de détection classique

Ainsi, un algorithme basique de calcul des contours dans une image suit les étapes suivantes décrites dans l'algorithme 1.

2.2 Seuillage/Binarisation

Comme au chapitre précédent sur la **binarisation**, le choix du seuillage est délicat et là aussi, nous pouvons utiliser une technique de détection de modes (en supposant des distributions gaussiennes). Toutefois, il existe également une autre technique de seuillage plus complète qui peut permettre de palier les défauts de contours non fermés, à savoir le **seuillage par hystérésis**. Nous supposons deux niveaux de seuils définis ainsi : S_h , un seuil haut, et S_b un seuil bas : $S_b < S_h$. Voici les étapes :

Algorithme 1 : Algorithme classique de détection de contours – L’algorithme proposé permet de détecter les contours de type marche, mais nous pouvons facilement l’adapter aux contours de type toit. En effet, l’algorithme proposé va varier en fonction : (1) du filtre utilisé pour le calcul de la dérivée, (2) de la manière d’effectuer le seuillage de l’image, (3) de l’utilisation ou non d’une étape de fermeture. La plupart du temps, lorsque les dérivées premières sont utilisées, et si nous souhaitons prendre en compte les deux directions possibles pour cette dérivée, le seuillage est effectué avec l’image des normes.

- 1 Fonction_de_calcul_de_contours (I, I_c)
Entrées : I : Image à traiter
Sorties : I_c : Image binaire représentant les contours
- 2 **pour chaque** $(x, y) \in I$ **faire**
- 3 Calcul du gradient et évaluation de sa norme et de son orientation :
 - Norme : $\|\nabla I(x, y)\| = \sqrt{(I_x^2 + I_y^2)}$
 - Orientation : $\theta = \arctan\left(\frac{I_y}{I_x}\right)$
 ou calcul du laplacien
- 4 Seuillage de la norme ou de l’orientation ou détection des passages par zéro
- 5 Fermeture des contours

- (1) Sélection de $I_h = \{I(x, y) \mid \|\nabla I(x, y)\| \geq S_h\}$ (ce sont les points fiables, où nous sommes sûrs qu’il s’agit d’un contour) ;
- (2) Sélection de $I_b = \{I(x, y) \mid \|\nabla I(x, y)\| \geq S_b, \text{ avec } I(x, y) \notin I_h \text{ et } \exists I(k, l) \in \mathcal{V}(x, y) \mid I(k, l) \in I_h\}$
- (3) $I_h = I_h \cup I_b$

Les étapes (2) et (3) sont répétées de manière itérative jusqu’à ce que $I_b = \emptyset$. Cette technique permet d’obtenir des contours plus fermés et permet plus de liberté dans le choix des seuils. Toutefois, dans de nombreuses applications, S_h et S_b restent délicats à choisir. Il est aussi possible de définir une relation entre les 2 seuils pour limiter les possibilités ou introduire des contraintes.

2.3 Fermeture

L’étape de fermeture est nécessaire pour obtenir des contours fermés. Un exemple très simple de manière d’effectuer la fermeture est présenté dans la figure 2.3.

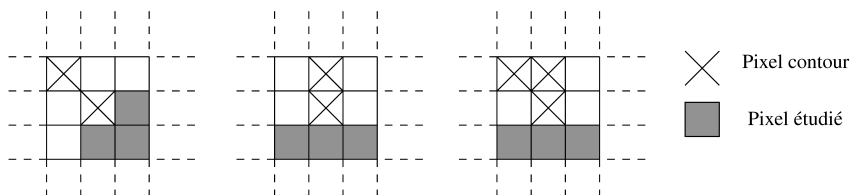


FIGURE 2.3 – Fermeture de contours – Dans cet exemple, les croix représentent les pixels détectés comme appartenant à un contour. Nous allons étudier certains pixels du voisinage qui n’ont pas été détectés comme pixels contours afin de décider de les ajouter ou non comme pixels contours. Pour choisir les pixels en gris, nous regardons le gradient associés aux points contours, cela nous donne la direction dans laquelle on peut étudier/questionner les pixels. Ensuite, pour chaque pixel gris, nous allons appliquer un test (souvent un seuillage, plus souple que le premier ayant permis de détecter des points contours) pour décider si nous ajoutons ou non ce pixel aux pixels contours.

Motivation – Le seuillage

- (1) Introduit des manques, dus à la présence de bruit ou à des occultations ;
- (2) Introduit des erreurs dues à des leurres dans les images.

Principe – La fermeture des contours permet d'obtenir des régions fermées interprétables comme projections des objets de la scène (segmentation). Les principes appliqués sont les suivants :

- (1) Utilisation de la direction du gradient ;
- (2) Suppression des contours non-fermés.

Représentation des contours par le codage de Freeman – Il s'agit d'une représentation compacte d'un contour en numérotant les 8 directions possibles pour un point de contour. Plus précisément, un nombre de directions possibles (4 ou 8) est choisi et ces directions sont numérotées. Ensuite, il faut choisir un pixel initial, en considérant qu'un élément de contour relie 2 pixels connexes. Le codage est établi en suivant ces étapes :

1. Choix d'un pixel initial du contour et d'un sens de parcours ;
2. Codage de la direction qui permet de passer au pixel contour suivant.

L'étape (2) est réalisée jusqu'à revenir au point initial. Un exemple de codage obtenu est donné dans la figure 2.4.

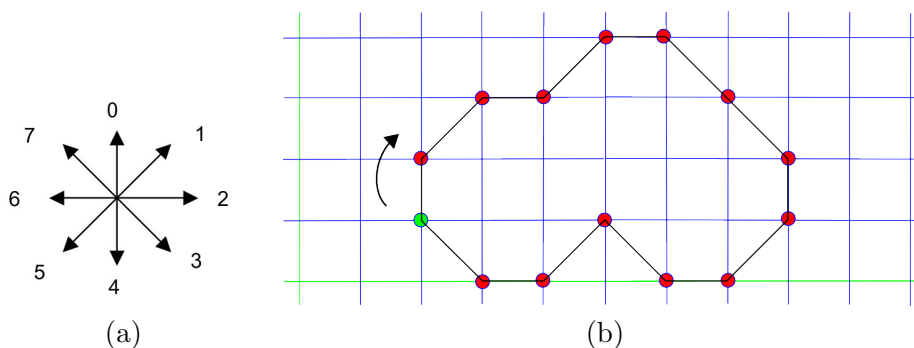


FIGURE 2.4 – Exemple de codage de Freeman – Si nous supposons le codage des directions données en (a), le code de Freeman associé au contour (b) est, dans le sens indiqué par la flèche : 0-1-2-1-2-3-3-4-5-6-7-5-6-7.

2.4 Approche de type Canny et Deriche

Dans la littérature, l'opérateur de Canny [Canny 86] est très connu et c'est pourquoi nous en faisons une présentation succincte. Une variante de cet opérateur est également présentée dans [Deriche 87]. L'auteur a conçu cet opérateur afin qu'il soit optimal suivant trois critères clairement explicités :

- (1) **Bonne détection** : Il doit y avoir une faible probabilité de manquer un point de contour mais également une faible probabilité de détecter un point qui n'est pas un contour.
- (2) **Bonne localisation** : Les points détectés comme contours doivent être aussi proches que possible des points de contour réels.
- (3) **Unicité de la réponse** : Un point du contour ne doit être détecté qu'une seule fois par le filtre mis en œuvre. Ce critère est inclus implicitement dans le premier critère puisque si deux contours sont détectés là où il n'y en a qu'un, une des deux réponses doit être considérée comme fausse.

En théorie, à chaque critère est associé une formule mathématique. La maximisation de ces critères conduit à la résolution d'une équation différentielle dont la solution est le filtre f , qui permet la détection du contour.

En pratique, cela se traduit par l'application de l'algorithme classique, présenté dans l'algorithme 1, en effectuant au préalable un filtrage de l'image par un masque de lissage gaussien, vu en page 25. La taille du filtre gaussien est choisie de manière optimale pour prendre en compte le bruit. Plus précisément, le calcul de la dérivée est réalisé très simplement, une fois l'image lissée, en utilisant une différence finie comme vue dans la figure 2.5. Cette première partie assure la **bonne localisation**. La modification majeure de l'algorithme classique est d'introduire une étape de **suppression des non-maxima locaux**. Cela signifie qu'à ce stade, seuls les maxima locaux sont conservés. Cette modification doit permettre l'**unicité de la réponse**. Puis, la sélection des points de contour est réalisée en utilisant un seuillage par hystérésis sur la norme et sur l'orientation du gradient. Ceci permet de prendre en compte l'aspect de **bonne détection**.

Finalement, cet algorithme possède trois paramètres à fixer :

- (1) La taille du filtre gaussien ;
- (2) Les seuils pour le seuillage par hystérésis.

2.5 Calcul de la dérivée d'une image

2.5.1 Dérivées premières

Le principe est de calculer des différences finies pour approximer les dérivées, exactement comme cela peut être fait en mathématiques pour estimer les dérivées de fonctions. Les filtres proposés dans la littérature permettent donc de calculer des différences. Toutefois, plusieurs directions sont possibles, c'est la raison pour laquelle nous avons toujours deux filtres : un pour calculer la dérivée suivant les lignes et un suivant les colonnes.

Calcul des dérivées premières par convolution sans lissage – Tous les **filtres basiques** présentés dans la figure 2.5 permettent de détecter des contours de type marche, suivant les lignes et les colonnes. Ils sont **rapides**, mais, ils présentent l'inconvénient d'être **sensibles aux bruits**. Pour ne pas être sensible aux bruits, il est nécessaire de **filtrer l'image**. Ainsi, d'autres filtres avec **pseudo-lissage** ont été proposés.

Nom	F_x	F_y
Différences de pixels 1	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$
Différences de pixels 2	$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$
Roberts	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

FIGURE 2.5 – Calcul des dérivées premières par différence finie, sans lissage.

Calcul des dérivées premières par convolution avec pondération – Nous pouvons noter que les filtres de **Prewitt et, Sobel et Frei-Chen**, présentés dans la figure 2.6, correspondent respectivement à **une moyenne et à des moyennes pondérées des différences de pixels**.

L'avantage du filtre de **Frei-Chen** est qu'il obtient de meilleurs résultats en présence de **contours non verticaux ou horizontaux**. Malgré le pseudo-lissage (lié au fait de prendre une moyenne pondérée) introduit, ces filtres permettent d'obtenir des résultats encore très sensibles aux bruits. La figure 2.7 permet d'illustrer l'influence des différents filtres sur le résultat obtenu.

Nom	F_x	F_y
Prewitt	$\frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$	$\frac{1}{3} \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$
Sobel	$\frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$	$\frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$
Frei-Chen	$\frac{1}{4} \begin{pmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{pmatrix}$	$\frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{pmatrix}$

FIGURE 2.6 – Calcul des dérivées premières par différence finie, avec pondération.

Ainsi, dans cette catégorie, le dernier filtre que nous pouvons citer est celui de Scharf, donné dans la figure 2.8. Ce filtre est celui qui se rapproche le plus d'un filtre gaussien (que nous allons étudier dans la section suivante), tout en gardant une arithmétique entière rapide et simple à implémenter.

Calcul des dérivées premières par convolution avec lissage – En théorie et en pratique, il est préférable de d'abord **filtrer l'image pour la lisser**, par exemple avec un filtre gaussien, déjà vu au chapitre précédent. Plus formellement, voilà ce que nous souhaitons calculer, avec \times l'opérateur de convolution :

$$I' = (I \times f)'. \quad (2.1)$$

Or, en utilisant la transformée de Fourier, sachant que f est la réponse impulsionnelle d'un filtre, nous savons que :

$$(I \times f)' = I \times f'. \quad (2.2)$$

De même :

$$(I \times f)'' = I \times f''. \quad (2.3)$$

D'où le fait que nous pouvons calculer la dérivée tout en effectuant un lissage si nous filtrons avec la dérivée d'un filtre de lissage.

Ainsi, l'étape (1) de l'algorithme 1 présenté est finalement un filtrage de l'image suivi du calcul du gradient.

Calcul des dérivées premières en utilisant la dérivée gaussienne – La définition de la fonction gaussienne en 1D est donnée par:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}. \quad (2.4)$$

Ainsi, le filtre dérivé en 1D correspondant sera défini par :

$$f'(x) = \frac{-x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}}. \quad (2.5)$$

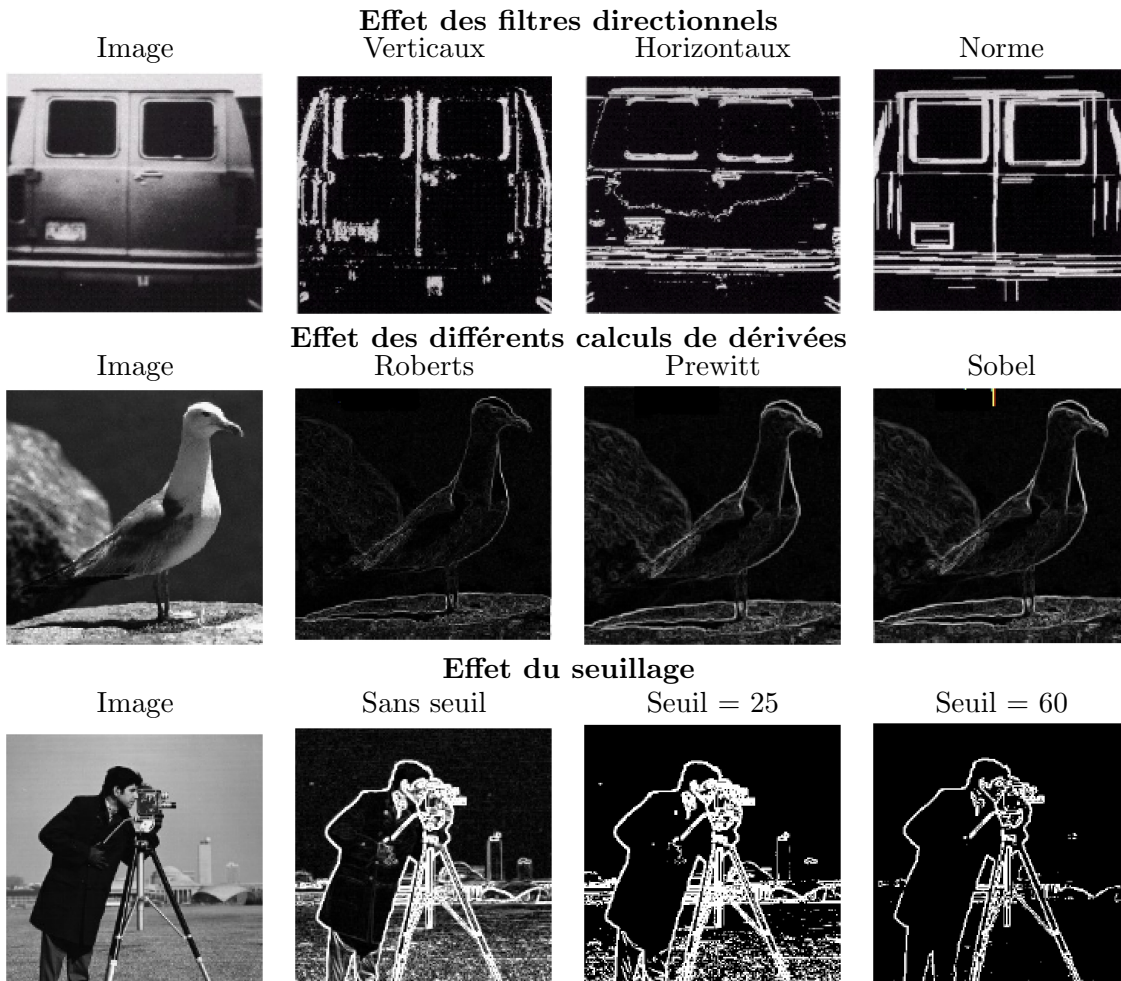


FIGURE 2.7 – Illustration du comportement de l’algorithme 1 suivant les choix réalisés pour le calcul de la dérivée et pour le seuillage.

La définition de la fonction gaussienne en 2D est donnée par :

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad (2.6)$$

Ainsi, les filtres dérivés en 2D, en x , et en y respectivement, seront définis par :

$$f_x(x, y) = \frac{-x}{2\pi\sigma^4} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \text{ et } f_y(x, y) = \frac{-y}{2\pi\sigma^4} e^{-\frac{(x^2+y^2)}{2\sigma^2}}. \quad (2.7)$$

À partir de ces formules, il est facile de définir des masques de dérivées premières gaussiennes pour une taille choisie, suivant la direction choisie (en x ou en y , soit suivant la convention choisie, suivant les lignes ou les colonnes de l’image). Nous illustrons la manière de calculer ces masques par deux exemples : en 1D avec une taille de masque de 5 et en 2D avec une taille de 3×3 . Soit, plus précisément, nous pouvons exprimer les masques de dérivées suivant les lignes par :

$$F'_5 = (f'(-2) \quad f'(-1) \quad f'(0) \quad f'(1) \quad f'(2)) \text{ et } F_{3,3}^x = \begin{pmatrix} f_x(-1, -1) & f_x(-1, 0) & f_x(-1, 1) \\ f_x(0, -1) & f_x(0, 0) & f_x(0, 1) \\ f_x(1, -1) & f_x(1, 0) & f_x(1, 1) \end{pmatrix}. \quad (2.8)$$

Comme nous avons $f'_*(-x) = -f'_*(x)$, pour $* \in \{x, y\}$, ces filtres peuvent se simplifier par :

Nom	F_x	F_y
Scharr	$\frac{1}{32} \begin{pmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{pmatrix}$	$\frac{1}{32} \begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix}$

FIGURE 2.8 – Calcul des dérivées premières avec le filtre de Scharr.

$$F'_5 = (-f'(2) \quad -f'(1) \quad 0 \quad f'(1) \quad f'(2)) \quad \text{et} \quad F_{3,3}^x = \begin{pmatrix} f_x(-1,1) & f_x(-1,0) & f_x(-1,1) \\ 0 & 0 & 0 \\ -f_x(-1,1) & -f_x(-1,0) & -f_x(-1,1) \end{pmatrix}. \quad (2.9)$$

Il est à noter que ce filtre n'est pas normalisé, c'est-à-dire que l'image obtenue ne sera plus comprise entre 0 et N_{max} . Pour cela, il faudrait diviser par la somme des éléments positifs de F_5 et $F_{3,3}$ respectivement. Soit :

$$Fnorm'_5 = \frac{1}{f'(1) + f'(2)} F_5^i \quad \text{et} \quad Fnorm_{3,3}^x = \frac{1}{f_x(-1,0) + 2f_x(-1,1)} F_{3,3}^x. \quad (2.10)$$

Nous ne détaillons pas l'expression pour les masques de dérivées suivant les colonnes puisqu'ils se déduisent très simplement en utilisant la transposée, soit par : $F_5'^T$ et $F_{3,3}^x T$.

À présent que nous avons détaillé la manière de calculer les dérivées premières, nous allons aborder la façon de calculer les dérivées secondes.

2.5.2 Dérivées secondes

Calcul des dérivées secondes par convolution – Pour calculer la dérivée seconde, nous utilisons le laplacien défini par la somme des dérivées secondes partielles, soit :

$$\nabla^2 I(x, y) = I_{xx}(x, y) + I_{yy}(x, y). \quad (2.11)$$

Les trois filtres que nous présentons sont une approximation du calcul du laplacien. Ils prennent en compte le fait que nous devons avoir un passage par zéro au niveau du pixel étudié. C'est pourquoi la somme des poids fait 0 et il faut des coefficients opposés entre le centre et le reste des pixels du masque.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

Une fois de plus, ce type de filtre est très sensible aux bruits. En effet, il s'agit de la dérivée seconde donc tout bruit affectera le résultat de manière plus importante que lorsqu'on calcule la dérivée première. C'est la raison pour laquelle, dans la littérature, la dérivée seconde d'une gaussienne est exploitée, en suivant le même principe que celui exposé dans le dernier paragraphe de la section 2.5.1.

Calcul des dérivées secondes par laplacien de gaussien Le laplacien de gaussien ou *Laplacian of Gaussian*, *LoG*, est défini par :

$$LoG(x, y) = g_{xx}(x, y) + g_{yy}(x, y), \quad (2.12)$$

où g_{xx} , respectivement g_{yy} , sont les dérivées secondes de la gaussienne, en x , respectivement en y . Après développement de cette formule, nous obtenons :

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2 + y^2}{2\sigma^2}}. \quad (2.13)$$

Il existe une manière de calculer une approximation du laplacien, qui est plus robuste aux bruits et moins coûteuse en temps de calculs que d'appliquer directement cette formule. Il s'agit de calculer des différences de gaussiennes ou DoG, *Difference Of Gaussian*. Cet aspect sera étudié en 3A, notamment avec la notion de détecteur de points d'intérêt multi-échelle.

Chapitre 3

Segmentation

3.1 Introduction sur la segmentation en recherche

La segmentation d'images est un outil indispensable dans de nombreuses applications, comme illustré dans [Szeliski 10, chapitre 5]. Classiquement, nous pouvons distinguer les approches par contours des approches par régions, comme exposé dans les travaux de [Brun 96, partie 2]. Toutefois, dans la littérature, les différents outils de segmentation et de vision ont été combinés pour tenter d'améliorer les résultats obtenus, en utilisant de multiples segmentations [Mathevet 99], en fusionnant les résultats de manière spatio-temporelle [Chen 98, Chen 99, Barnes 09], et en combinant les approches régions et les approches contours [Kermad 97, Sebari 07, El Merabet 13]. Une autre manière d'exploiter différents outils est de fusionner des informations de nature différente, en particulier des données 2D et 3D [Rosenhahn 07, Bray 06, Kholi 08, Dambreville 08]. D'autres travaux permettent également l'utilisation d'attributs d'origines différentes ou encore la prise en compte de recouvrement spatial entre différentes images pour effectuer une segmentation. Par exemple, les publications de [Heitz 08, Tu 10] permettent d'utiliser une information contextuelle (les objets à segmenter ont des positions relatives particulières ou connues, les uns par rapport aux autres) alors que les travaux de [Zhuge 04, Weippl 07] font intervenir la sémantique (les régions sont associées à un mot clé, une définition). Nous pouvons citer les approches hiérarchiques qui font intervenir la notion de super-pixels [Ren 03, Fulkerson 09, Achanta 12]. Enfin, la notion d'apprentissage apparaît depuis longtemps dans ce type de méthodes [Collins 02] et l'utilisation de l'apprentissage profond a connu un essor très important depuis 2010, avec des approches célèbres comme celle de [Krizhevsky 12]. Face à l'ampleur des applications visées et des approches développées pour y répondre, il est difficile d'évaluer, de comparer les résultats et, ainsi, de choisir une approche adaptée. En conséquence, de nombreux jeux de données ont été créés et mis à disposition, comme, par exemple, dans la publication de [Martin 01]. Avec l'arrivée de l'apprentissage profond, de nombreuses bases de données conséquentes sont apparues avec notamment la base *ImageNet* [Russakovsky 15].

Dans la littérature, **face à un problème de segmentation, les étapes suivies sont la plupart du temps les suivantes : pré-traitement des données, binarisation ou classification et enfin post-traitement**. Lorsque la segmentation fait appel à un apprentissage, le pré-traitement consiste, en partie, à sélectionner et traiter une base de données, et, dans le cas particulier de l'apprentissage profond, ce pré-traitement inclut une préparation des données qui permettra de les rendre exploitables dans le réseau de neurones mis en œuvre. Tous ces pré-traitements et étapes de préparation peuvent avoir un impact important sur la qualité du résultat. Le post-traitement permet souvent d'extraire des résultats plus interprétables pour réaliser la reconnaissance, l'identification de l'objet d'intérêt ou des objets de la scène. Il est aussi souvent proposé pour tenter de corriger les erreurs de segmentation et cela peut avoir un impact important sur la qualité du résultat obtenu.

Ce cours est une introduction à ce domaine et nous allons donc nous concentrer sur les approches classiques. Les approches par apprentissage seront vues avec Axel CARLIER ou Sandrine MOUYSET.

3.2 Définition

Qu'est ce que c'est ?

Nous souhaitons définir la segmentation ainsi :

La segmentation a pour but de définir une partition de l'image sachant que chaque partition représente une information homogène et cohérente de la scène et qui a du sens.

Plus précisément, l'objectif est d'attribuer une étiquette à chaque pixel de l'image, sachant que tous les pixels qui obtiennent la même étiquette partagent des caractéristiques communes. Il y a autant de définitions de la segmentation que de manières de définir qu'une information est homogène et cohérente, cf. figure 3.1. C'est pourquoi, souvent, on parle de problème de segmentation mal posé. Il n'existe pas de segmentation idéale et le choix d'une méthode est lié à, entre autres :

- La nature des images (éclairage, texture, ...);
- La forme des primitives à extraire;
- Des contraintes de temps.

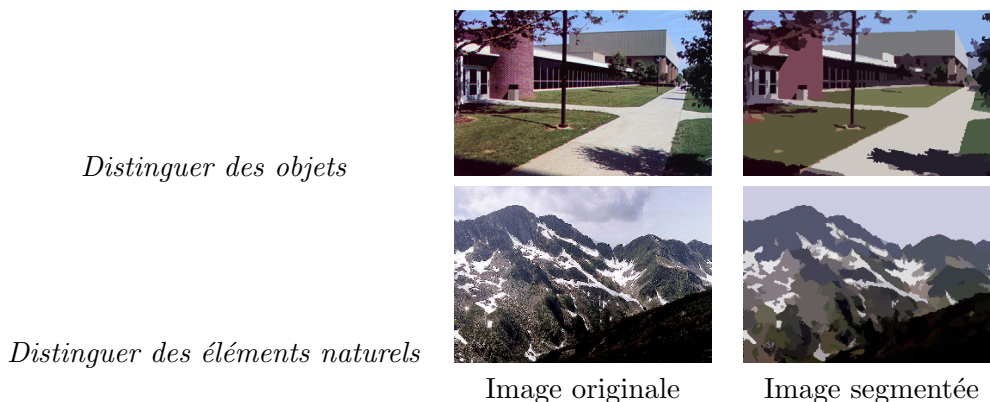


FIGURE 3.1 – Exemples de segmentation.

Domaines de la segmentation – Dans la littérature, il est possible de distinguer les segmentations fond/forme (2 classes seulement) des segmentations à plusieurs objets. Une autre façon de classer les approches est de considérer ces 3 catégories :

- La **segmentation supervisée**, qui utilise une base d'apprentissage pour avoir des connaissances *a priori* sur les régions recherchées.
- La **segmentation semi-supervisée**, où la seule connaissance *a priori* utilisée est le nombre de régions recherchées.
- La **segmentation non supervisée**, où aucune connaissance *a priori* n'est utilisée.

Définition – Plus formellement, voilà une manière de définir une segmentation :

Segmenter consiste à créer une partition de l'image I en sous-ensembles R_i appelés régions tels que :

$$\forall i R_i \neq \emptyset$$

$$\forall i, j \text{ avec } i \neq j R_i \cap R_j = \emptyset$$

$$I = \cup_i R_i$$

Une région est donc un ensemble de pixels ayant des propriétés communes qui les différencient des régions voisines : niveaux de gris, couleurs, textures, formes. Une région peut être connexe ou non.

Pourquoi ?

Ils existent de nombreuses applications qui s'appuient sur une segmentation d'images, en particulier en imagerie médicale pour délimiter différents organes. Nous pouvons distinguer les cas de segmentations de type fond/forme, avec une seule forme à reconnaître comme dans l'exemple 3.2.(a). Nous pouvons aussi envisager des cas de segmentations d'une même forme sur un fond mais cette forme peut être présente en plusieurs exemplaires ou parties, comme la détection d'une fissure (la forme à segmenter) dans une chaussée (le fond texturé), ou des insectes à identifier sur une image de piège, cf. figure 3.2.(c) et (d). Enfin, nous pouvons avoir des cas de segmentation de scènes qui comportent plus d'éléments distincts, comme des scènes urbaines, cf. figure 3.2.(d).

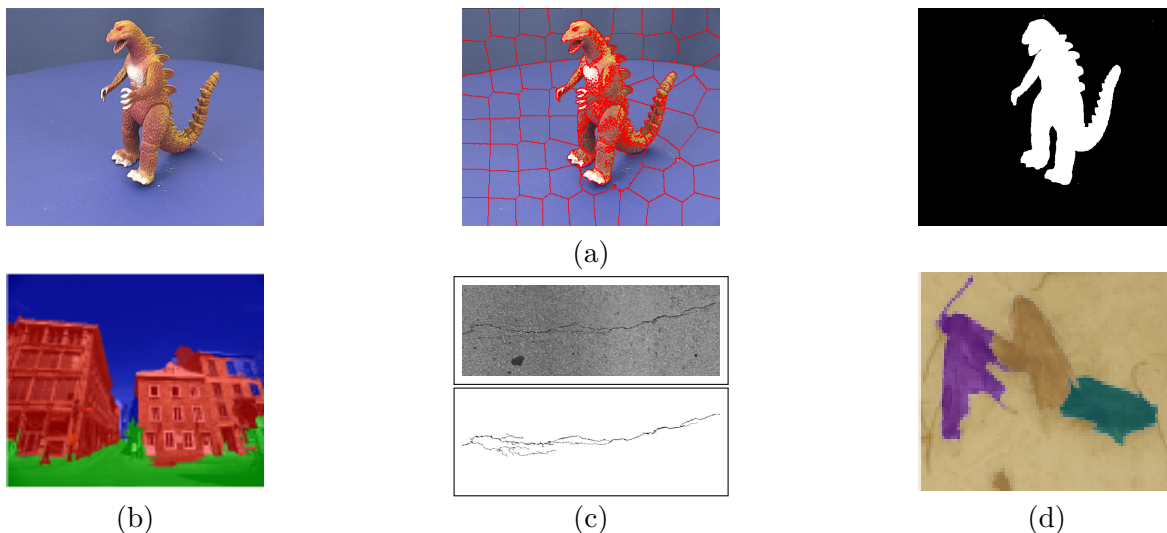


FIGURE 3.2 – Exemples de segmentation – En (a), dans le cadre de travaux pratiques proposés à l'ENSEEIH, nous sur-segmentons une image d'origine (à gauche) afin d'en extraire une segmentation fond/forme (à droite). En (b), nous présentons un résultat de segmentation de scènes urbaines [Bauda 15] (il s'agit des différentes couleurs appliquées sur la scène). En (c), il s'agit d'un résultat présenté dans [Amhaz 16] de segmentation de fissures de la chaussée (indiquée en noire en bas). Enfin, en (d), nous montrons un résultat de détection de papillons sur une image de piège (il s'agit d'un extrait et la segmentation obtenue est représentée par les couleurs ajoutées sur l'image d'origine) [Bakkay 18].

Comment ?

Il est difficile de définir un algorithme générique. Dans cette introduction, nous avons déjà présenté les principales familles mais les détails sont fournis dans la section 3.3.

Qualité du résultat ?

La littérature est abondante sur ce domaine de l'évaluation de la qualité de la segmentation, comme dans [Zhang 96, Y. J. Zhang 01], et les travaux [Pont-Tuset 15] donnent une bonne synthèse de cet aspect, pour le lecteur intéressé. La plupart des évaluations supposent que nous possédons une segmentation de référence faisant office de vérité terrain. Nous supposons donc que nous avons deux résultats de segmentation à comparer et que nous appelons la segmentation de référence, c'est-à-dire la segmentation idéale que nous souhaitons obtenir avec notre algorithme de segmentation automatique, et la segmentation estimée. Nous souhaitons évaluer la qualité de la segmentation estimée, en fonction,

de la segmentation de référence. Dans les explications que nous donnons, nous supposons que nous avons à travailler avec une segmentation fond/forme : soit deux classes. Une classe représente la fond et nous supposons que nous la représentons en blanc, et une classe représente la forme, dans la couleur noire. Une première étape pour évaluer la qualité d'un résultat consiste à évaluer les pourcentages de : vrais positifs, *True Positives* (TP), faux positifs, *False Positives* (FP), faux négatifs, *False negatives* (FN) et vrais négatifs, *True Negatives* (TN). Pour évaluer le pourcentage de TP, FP, TN, FN, nous supposons donc qu'il y a deux cartes de segmentation à comparer, avec chacune des valeurs de couleur blanche ou noire. Voici donc les définitions de ces 4 éléments :

- Un vrai positif est un pixel noté en noir dans la segmentation de référence et dans la segmentation estimée.
- Un vrai négatif est un pixel noté en blanc dans la segmentation de référence et dans la segmentation estimée.
- Un faux positif est un pixel noté en noir dans la segmentation estimée alors qu'il est blanc dans la segmentation de référence.
- Un faux négatif est un pixel noté en blanc dans la segmentation estimée alors qu'il est noir dans la segmentation de référence.

Les valeurs trouvées pour chacune de ces quatre catégories de pixels peuvent être combinées afin de déterminer une performance globale. Plus précisément, voici les critères qui peuvent être évalués :

- La précision (P) permet de mettre en évidence la proportion de vrais positifs parmi les positifs (la segmentation estimée) :

$$P = \frac{TP}{TP + FP} \quad (3.1)$$

- La sensibilité (S) met en évidence la proportion de bonnes détections par rapport à la forme à segmenter. Cela nous donne une indication indirecte sur la proportion de manques :

$$S = \frac{TP}{TP + FN} \quad (3.2)$$

- Le coefficient de similarité ou *DICE Similarity Coefficient*¹ (*DSC*) est la moyenne harmonique entre la précision et la sensibilité :

$$DSC = \frac{2TP}{2TP + FP + FN} \quad (3.3)$$

Dans le cas d'une segmentation parfaite, $DSC = 1$ alors que $DSC = 0$, quand la segmentation est entièrement fautive. Comme ce coefficient permet de combiner précision et sensibilité, il est très utilisé.

3.3 Méthodes de segmentation

Il est difficile d'établir une classification exhaustive des méthodes de segmentation, mais nous avons fait le choix de distinguer les trois méthodes suivantes :

- (1) Les approches **contours**, où nous cherchons les contours entre objets peu semblables, c'est-à-dire les contours entre zones hétérogènes. La technique la plus utilisée s'appuie sur la notion de contours actifs (*snakes*) [Kass 88].
- (2) Les approches **régions**, où nous cherchons à regrouper les pixels similaires pour former des régions homogènes, comme :
 - (a) La fusion/division de régions (*split and merge*) [Horowitz 76].
 - (b) La croissance de régions, comme la ligne de partage des eaux, *watershed* [Vincent 91].

1. Dans la littérature, nous parlons aussi de mesure/score f_1 .

- (3) Les approches s'appuyant sur le principe de la **classification** [Duda 01], les plus connues et les plus utilisées étant :
- (a) Les approches bayésiennes [Domingos 97];
 - (b) Les approches par Espérance-Maximisation (EM) [Dempster 77];
 - (c) Les k -moyennes (k -means) [MacQueen 67];
 - (d) Le *mean-shift* [Comaniciu 02];
 - (e) Les approches par *Support Vector Machine* (SVM) [Boser 92].

Il existe également des approches hybrides ou duales, qui mélangent les deux premières approches. Il est intéressant de noter que le système visuel humain est plus apte à détecter des contours que des zones homogènes. Enfin, quelle que soit la méthode choisie, les critères utilisés (couleur, position, etc.) sont déterminants.

Actuellement, il existe des approches plus avancées permettant de combiner plus d'information notamment :

- en exploitant des segmentations multiples ;
- en s'appuyant sur la notion de sur-segmentation par superpixels permettant une première estimation fiable de la segmentation finale ;
- en combinant des informations de nature différentes plus riches que le niveau de gris ou la couleur comme des informations 3D ou des informations contextuelles.

Nous présentons dans la suite les approches par contours, les approches par régions et enfin les approches inspirées des approches de classification. Comme technique plus récente, nous aborderons, dans le chapitre suivant, les approches par superpixels.

3.4 Approches par contours

3.4.1 Des approches classiques aux contours actifs

Les plus anciennes s'appuient sur des approches bas niveau déjà étudiées dans le chapitre 2, cf. figure 3.3. Les techniques les plus avancées dans ce domaine concernent les contours actifs [Kass 88]. Ils sont très souvent employés en imagerie médicale où les contours sont très faiblement contrastés.

Pour toutes ces approches de contours actifs, nous faisons l'hypothèse qu'il existe un contour idéal pour segmenter la forme et l'approche va tenter d'estimer ce contour idéal. Il s'agit d'un algorithme itératif qui partant d'un contour initial va le faire évoluer en utilisant la notion d'énergie. Ce que nous appelons **énergie** permet de déterminer l'ensemble des pixels qui forment le contour, tout en minimisant un critère attendu. Cette énergie est donc relative à un contour, c'est-à-dire à l'ensemble des pixels qui sont associés à ce contour. Une définition subjective et intuitive de l'énergie est la suivante : nous pouvons dire que l'énergie correspond à une sorte de mesure d'adéquation entre le contour choisi et les propriétés attendues pour ce contour. L'analogie avec l'énergie vient du fait que cette mesure représente la quantité d'énergie qu'il va falloir dépenser pour répondre aux propriétés attendues. Par exemple, en évaluant la distance à un contour, et en l'ajoutant au calcul de l'énergie, nous quantifions ainsi l'énergie, c'est-à-dire la distance, à dépenser pour que le contour respecte cette propriété et, par une sorte d'équivalence, nous évaluons à quel point le contour ne respecte pas cette propriété. En résumé, le rôle de l'énergie est de savoir lorsque le contour estimé est le plus proche du contour idéal. Par la suite, nous allons détailler les termes qui composent l'énergie à minimiser mais ce qu'il faut retenir c'est que les termes qui composent l'énergie sont calculés en fonction du contour. Ainsi, toute règle ou contrainte appliquée ou ajoutée à l'énergie traduit donc directement ou indirectement une règle ou une contrainte sur le contour lui-même.

3.4.2 Principe des contours actifs

Un **contour actif** ou *snake* possède les caractéristiques suivantes :

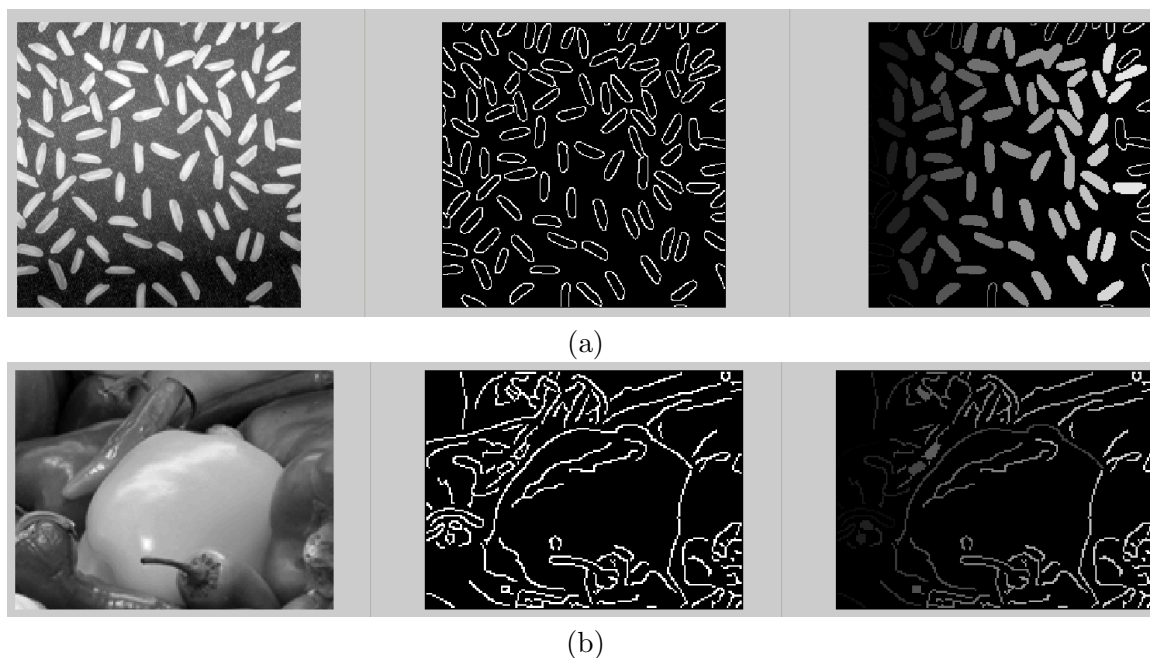


FIGURE 3.3 – Illustration d’une approche classique par détection de contours puis fermeture (colonne du milieu). Pour les deux exemples fournis, nous supposons que nous voulons segmenter les différents objets qui composent la scène : soit les grains de riz, en (a), et les différents poivrons, (b). En (a), cette approche propose des résultats exploitables alors qu’en (b), nous montrons un cas où l’approche ne fonctionne pas.

- Il est caractérisé par des **énergies qui représentent ses propriétés physiques** ainsi que son **adéquation à la forme**.
- Son **énergie élastique** permet de préciser la configuration souhaitée, ce qui permet de prendre en compte une **information a priori sur la forme**.
- En lui attribuant une inertie, il acquiert un **comportement dynamique**, ce qui lui permet d’utiliser des **connaissances a priori sur le mouvement** de la forme dans une séquence d’images ou plus simplement dans la séquence de déformation.

3.4.3 Étapes des approches par contours actifs

Une approche par contour actif suit les étapes suivantes, si nous notons \mathcal{C} la courbe active (fermée ou non), comme illustrée dans la figure 3.4 :

- (1) Initialisation du contour ;
- (2) Déformation par itérations successives jusqu’à convergence vers le contour réel.

La déformation s’effectue en étudiant l’évolution d’une énergie E qui représente la qualité de la courbe vis-à-vis du contour recherché. Les itérations successives sont généralement réalisées jusqu’à ce qu’il y ait peu de changements de l’énergie de la courbe.

La première étape de cet algorithme, l’initialisation, influence beaucoup la qualité du résultat. Il faut bien noter que plus le contour est éloigné de celui recherché, plus l’algorithme sera coûteux. L’initialisation peut être réalisée par une forme simple (cercle, carré), comme sur cet exemple. Il est également possible d’envisager une détection de contours classiques (mais le bruit risque d’être trop important pour que la méthode fonctionne). Dans certains cas, c’est un utilisateur (donc une intervention humaine) qui initialise le contour.

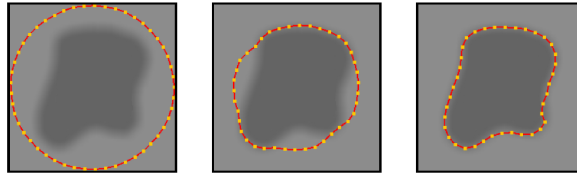


FIGURE 3.4 – Illustration de l’approche par contours actifs. De la gauche vers la droite, nous présentons : une initialisation simple par un cercle du contour, une itération de l’évolution du contour, le résultat final après convergence. Sur cet exemple, la méthode se comporte comme espéré : le contour, même initialisé simplement et loin de la forme à segmenter converge vers le contour de la forme. Il est à noter que cet exemple de scène est très simple.

3.4.4 Énergie à minimiser

Plus formellement, le contour est représenté dans le plan par une forme paramétrique (fermée ou non), notée $\mathcal{C}(s) = (x(s), y(s))$, $s \in [0, 1]$. Cette courbe est donc déformée de manière itérative afin de minimiser une fonction d’énergie E qui a pour forme générale :

$$E(\mathcal{C}) = \int_0^L \left(\underbrace{w_1 \|\mathcal{C}'(v)\|^2 + w_2 \|\mathcal{C}''(v)\|^2}_{\text{Forces internes}} + \underbrace{\mathcal{P}(\mathcal{C}(v))}_{\text{Force externe ou image}} \right) dv.$$

- $[0, L]$ est le domaine de définition de la courbe \mathcal{C} ;
- L est la longueur de la courbe.

Les deux premiers termes correspondent aux **forces internes** avec w_1, w_2 les poids de chaque élément (dérivées première et seconde). Nous parlons de forces internes car il s’agit de caractéristiques géométriques liées au contour et à sa forme. Plus précisément w_1 représente le coefficient d’**élasticité** (résistance à l’allongement) de la courbe, une variation de la longueur de la courbe est donc pénalisée, et w_2 correspond au coefficient de **flexibilité**, une variation de la courbure du contour est donc pénalisée. Ces deux termes agissent donc sur l’aspect régulier de la courbe. Là aussi, les choix pour w_1 et w_2 sont délicats et dépendent de la forme recherchée : contours plus ou moins grand, forme plus ou moins rigide. Le troisième terme correspond à la **force externe ou image** (*a priori* de forme) qui définit un potentiel d’attraction ou d’énergie d’attache aux données. Nous parlons de forces externes car elles sont liées aux attributs, souvent photométriques, que nous choisissons pour définir le contour. Il s’agit donc, la plupart du temps, d’un choix empirique en fonction des images étudiées. Dans la littérature, comme pour toutes approches en traitement d’images, on va étudié l’impact d’un paramètre sur les résultats obtenus. Plus précisément, on va faire varier chaque paramètre et on va étudier la qualité des résultats. C’est ce qui va permettre, d’une part, de conclure sur le choix optimal pour les paramètres, et, d’autre part, l’influence de ces paramètres sur la qualité du résultat. Idéalement, une approche sera jugé robuste si elle obtient de très bons résultats pour une plage de valeurs données. Par exemple, on peut faire varier w_1 et w_2 entre 0 et 1 avec un pas de 0.1 (ce qui fait 11 expériences différentes, puisque $w_1 + w_2 = 1$). On sera assez satisfait si on peut indiquer que par exemple, l’algorithme obtient de bons résultats pour la majorité des images (depuis une décennie, on attend des tests pour au minimum une centaine d’images) avec des valeurs dans un intervalle de ± 0.2 .

3.4.5 Choix possibles

Différents choix sont donc à réaliser pour mettre en œuvre une segmentation par contours actifs :

- **Force externe \mathcal{P}**

(1) *Utilisation des niveaux de gris* : $\mathcal{P}(\mathcal{C}(v)) = I(v)$;

(2) *Utilisation des gradients*: $\mathcal{P}(\mathcal{C}(v)) = \phi(\|\nabla I(v)\|)$ avec ϕ fonction décroissante.

- **Type de contours**

- *Contours géométriques* : évolution planaire de la courbe
- *Contours géodésiques* : cas particulier des contours actifs avec :

$$E(\mathcal{C}) = \int_{\omega} \left(\|\mathcal{C}'(v)\|^2 + \mathcal{P}(\mathcal{C}(v)) \right) dv$$

Un contour géodésique correspond à la distance la plus courte entre deux points en prenant en compte, c'est-à-dire en suivant, le contour (en 3D, la forme de la surface) qui relie les deux points.

3.4.6 Forme discrète de l'énergie et algorithme classique de calcul

Ceci est une définition **continue**. Pour l'appliquer, il est nécessaire de manipuler une **version discrète** de la courbe, c'est-à-dire que nous utilisons des points de contrôle sur la courbe. Le plus souvent, ces points de contrôle sont simplement choisis en prenant des points aléatoirement mais régulièrement placés sur la courbe. Cela signifie qu'il faut choisir **un pas d'échantillonnage** et une forme : polygonale, de type *spline* ... Cette notion est abordée dans le cours de Géraldine Morin en interpolation et approximation. Dans ce cas, voici le type d'algorithme qui peut être mis en œuvre, cf. algorithme 2.

Pour faire évoluer la courbe, différentes méthodes d'optimisation sont possibles : descente de gradient, recuit simulé.

Algorithme 2 : Segmentation par contours actifs – La condition d'arrêt peut être le fait que l'énergie tend vers zéro. Pour faire évoluer le point d'énergie maximale, il est déplacé dans son voisinage en utilisant, par exemple, un algorithme de descente de gradient.

- 1 Fonction segmentation_par_contours_actifs (I, \mathcal{C})
 - Entrées** : \mathcal{C} : le contour initial avec ces points de contrôle \mathbf{p}_i
 - Sorties** : Image segmentée
 - 2 Calculer l'énergie pour chaque point \mathbf{p}_i de la courbe \mathcal{C}
 - 3 Ordonner la liste des points par énergies croissantes
 - 4 **tant que** la condition d'arrêt n'est pas atteinte **faire**
 - 5 Évoluer le point d'énergie maximale \mathbf{p}_{max} d
 - 6 Attribuer la nouvelle énergie à \mathbf{p}_{max}
 - 7 Si la distance est trop importante entre \mathbf{p}_{max} et ses voisins, ajouter des points intermédiaires
-

3.4.7 Comment choisir ces différents paramètres ?

Dans ce paragraphe, nous donnons des éléments de réflexion pour faire les choix sur les paramètres. Pour la force interne, si nous imaginons que le contour initial à la bonne longueur, il faut imposer une longueur constante, si le contour doit être régulier, il faut imposer une régularité. Imaginons que nous segmentons une fleur avec des pétales bien dessinées et que nous ayons un cercle autour de cette fleur pour initialiser le contour. Comme nous souhaitons que chaque pétale soit bien découpée, il faut autoriser la longueur a évolué significativement et la courbe doit également pouvoir se déformer facilement.

Pour la force externe, si nous pensons que c'est la couleur qui apporte l'information la plus pertinente, il vaut mieux la favoriser. Même chose pour les gradients (les contours). Si nous reprenons le même exemple, et que nous avons une fleur rouge sur un fond vert, très probablement que la couleur sera suffisante comme critère de force externe. En revanche, si la différence colorimétrique est moins évidente, comme une fleur un peu sombre sur un fond végétal également sombre, le gradient ou la texture seront plus appropriés.

3.5 Approches par régions

Nous distinguons trois catégories : par histogramme, par croissance de régions, par division/fusion.

3.5.1 Approches par histogramme

Ces approches sont les plus simples, les plus adaptées dans le cas à plusieurs dimensions (hyper-spectrales, couleurs) mais malheureusement, elles sont souvent sensibles aux bruits. Le principe est d'isoler des pics de l'histogramme et d'effectuer un seuillage des zones correspondantes. Cet aspect est largement abordé dans le chapitre 1, paragraphe 1.2.3.

3.5.2 Fusion/Division

Principe – Le principe est de diviser ou agréger les régions suivant un prédicat, de manière itérative jusqu'à une certaine convergence, cf. figure 3.5.

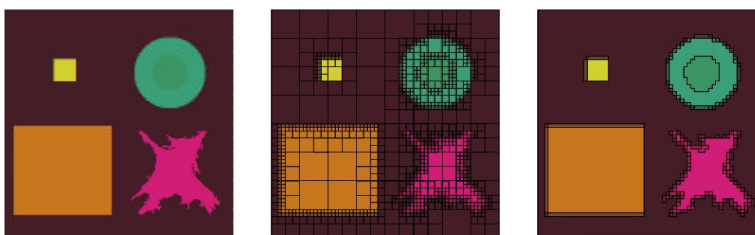


FIGURE 3.5 – Illustration du comportement de l'approche de segmentation par Division/Fusion. Pour l'initialisation, une grille régulière de l'image a été utilisée. Nous constatons sur cet exemple que si le nombre d'étapes de fusion/division est trop faible (deuxième image), le résultat est sur-segmenté.

Algorithme

Il s'écrit très simplement :

- Soit R_i , issue d'une première segmentation par division,
 - Soit $P(R_i)$ un prédicat logique donnant **vrai** si la région satisfait un certain critère (homogénéité),
- (1) Diviser en régions chaque R_i telle que $P(R_i)$ est **faux** ;
 - (2) Fusionner toutes régions R_j et R_k telles que $P(R_j \cup R_k)$ est **vrai** ;
 - (3) Arrêter dès que la fusion/division est impossible.

Le prédicat utilisé peut être par exemple le contraste de la région qui s'appuie souvent sur une analyse de l'écart-type ou de l'entropie de la région (représente la quantité d'information contenue dans la région).

Détails

Les points à déterminer pour cette approche sont :

- La manière d'enchaîner les divisions/fusions (parfois toutes les divisions puis, toutes les fusions, ou fusions/divisions simultanées) ;
- La manière de diviser les régions (souvent en 4) ;
- Le prédicat de fusion/division. Voici des exemples de prédicat :
 - (1) Le **contraste** : $P(R_i) = |\max(R_i) - \min(R_i)| < S$;

- (2) Les **différences limitées** : $\forall(x, y)$ pixels voisins $|I(x) - I(y)| < S$;
- (3) L'**écart-type**, la **moyenne**, le **nombre de pixels**, le **contour/périmètre** ...

3.5.3 Croissance de régions

Ce type d'approche est souvent utilisée en imagerie médicale où le praticien a besoin d'un outil d'aide et non d'un outil totalement automatique, cf. figure 3.6. Il s'agit donc souvent d'approches semi-supervisées puisque des points germes sont fournis pour permettre la croissance.

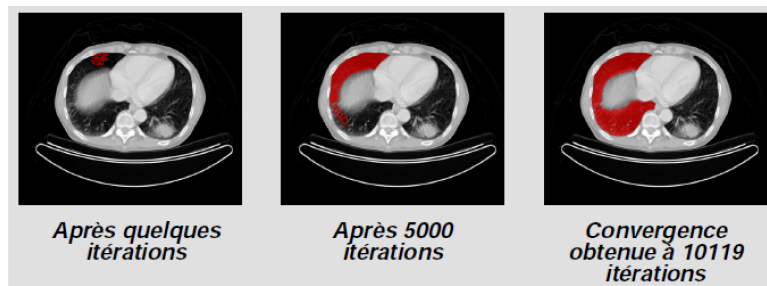


FIGURE 3.6 – Illustration du comportement de l'algorithme par croissance de régions.

Principe

Le principe est le suivant :

- (1) Les amorces sont placées : il s'agit du « cœur des régions ».
- (2) Pour chaque amorce \mathbf{a} de coordonnées (x_A, y_A) :
 - Propager l'amorce en agglomérant tous les pixels \mathbf{p} tels que : $P(\mathbf{p})$ est vraie, où P est un prédicat à définir.

Pour sélectionner des amorces à l'intérieur des régions à segmentées, nous pouvons imaginer utiliser des points qui se trouvent dans une zone homogène (variation des niveaux de gris faible dans un voisinage) ou sélectionner des points qui ne sont pas sur des contours, voire le plus éloigné possible des contours (en calculant une carte de distance aux contours). La plupart du temps, le prédicat P utilisé est le suivant : $|I(x_A, y_A) - I(x, y)| < S$. La propagation peut être séquentielle ou parallèle (pour ne pas privilégier une région). Le critère de propagation présenté est le plus simple mais il peut être adapté et prendre en considération d'autres critères : gradient, texture.

Détails

Il y a différents points à déterminer (en utilisant l'histogramme ou des *a priori*) :

- le choix des amorces (automatique ou non) ;
- le choix du seuil ;
- le choix du type de propagation.

De plus, lorsque la propagation est en parallèle (dans le cas de multiples amorces), l'ordre dans lequel sont ajoutés les pixels dans une région a une influence sur le résultat. Toutefois, cette implémentation est relativement simple et les temps d'exécution sont rapides.

3.5.4 Ligne de partage des eaux

Cette approche a été souvent utilisée dans le domaine des cartes d'élévation numérique ou *Digital Elevation Models*. Une illustration de son comportement est donnée dans la figure 3.7.

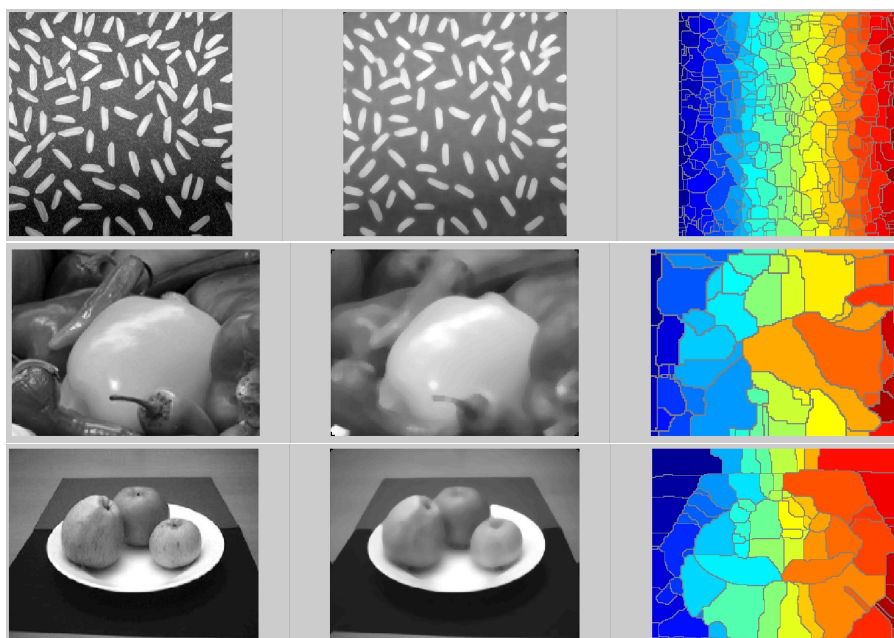


FIGURE 3.7 – Illustration des résultats obtenus avec ligne de partage des eaux. C’est surtout pour la dernière image que cette approche donne un résultat exploitable, c’est-à-dire, qui se rapproche le plus des contours de chacun des éléments de la scène. cette approche est délicate à utiliser et elle est souvent couplée à d’autres outils, d’autres techniques, en particulier en traitement d’images. Par exemple, nous l’avons utilisée pour détecter des papillons sur des images de pièges, cf. figure 3.8.

Principe

L’algorithme de la ligne de partage des eaux (LPE), proposé par [Vincent 91], utilise la **description des images en termes géographiques**. En effet, une image peut être vue comme un relief, pour peu que le niveau de gris soit assimilé à une altitude.

Une ligne de partage des eaux est définie comme la **crête formant la limite entre deux bassins versants**. Pour l’obtenir, il faut imaginer l’immersion d’un relief dans l’eau, en précisant que l’eau ne peut pénétrer dans les vallées que par ses minima. La ligne de partage des eaux est représentée par les **points où deux lacs disjoints se rejoignent au cours de l’immersion**. Une des difficultés à la mise en œuvre de cette analogie intuitive est qu’elle laisse beaucoup de liberté quant à sa formalisation. De plus, l’efficacité des lignes de partage des eaux en tant qu’outil de segmentation **dépend grandement des marqueurs de départ**, c’est-à-dire des minima. Sans traitement préalable, nous obtenons le plus souvent une sur-segmentation de l’image. Une segmentation conforme au but recherché nécessite donc un filtrage adéquat des minima, qui constituent les marqueurs de départ.

Mise en œuvre

Il est nécessaire :

- de définir une **grille** : pas de difficulté mais nécessité de définir le type de voisinage : 4/8 connexe, hexagones ;
- d’établir la notion de **chemin** : pas de difficulté mais cela dépend du voisinage choisi et la plupart du temps, la notion de recherche de chemins minimaux est utilisée [Dijkstra 59] ;
- de choisir les **minima/plateaux** : plus délicat et il est nécessaire d’effectuer des pré-traitements ;
- de définir comment construire les bassins, soit, **les lignes**. Il y a deux possibilités : par les chemins

(délicats à implémenter : ici la notion de recherche de chemin minimaux est utilisée), ou par immersions (cela se rapproche de la notion de croissance de germes). Quelque soit la technique privilégiée (chemins minimaux ou immersions), l'algorithme mis en œuvre sera itératif.

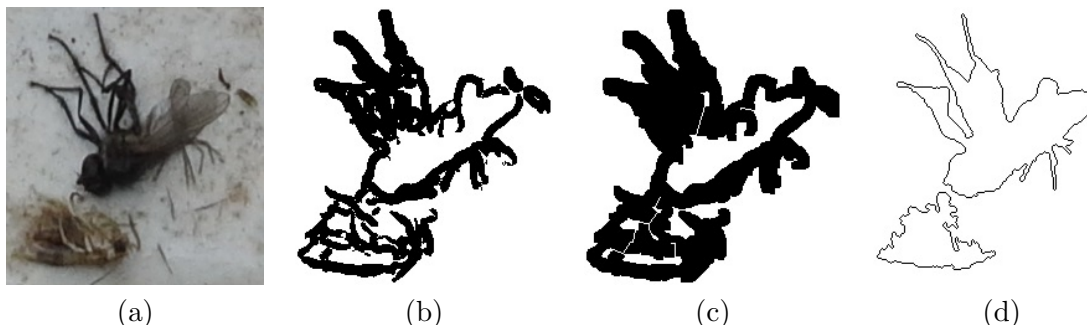


FIGURE 3.8 – Exemple de segmentation de papillons par ligne de partage des eaux – Vous pouvez voir un exemple de résultat de segmentation par ligne de partage des eaux. Dans cet exemple, l'algorithme est appliqué à toutes les imageries de l'image, comme en (a), qui correspondent à un contour détecté en (b), puis fermé par dilatation et fusion en (c). Ainsi, en (d), l'algorithme de segmentation par ligne de partage des eaux est appliqué. L'objectif de cette deuxième segmentation est de tenter de séparer les insectes qui ont été détectés comme un seul insecte car ils sont collés les uns aux autres.

3.6 Approches inspirées des approches de classification

3.6.1 k -moyennes

Principe L'algorithme des k -moyennes (ou k -means) est très connu en classification et très souvent utilisé pour la segmentation. Nous supposons le **nombre de classes connu** N . Nous cherchons donc la meilleure répartition des pixels en N classes.

Algorithme

Il s'agit d'un **algorithme itératif**, illustré dans la figure 3.9, dont les principales étapes sont :

1. Découper l'image en N régions suivant un critère \mathcal{C} (à définir).
2. Estimer les centres C_k , $k \in [1, N]$, de ces régions.
3. Tant que les centres sont modifiés, faire :
 - (a) Pour chaque pixel de l'image, trouver le centre C_{k^*} le plus proche, au sens du critère \mathcal{C} , et donner au pixel considéré l'étiquette k^* .
 - (b) Mettre à jour les centres C_k des N régions ainsi constituées.

Les régions obtenues par cette technique ne sont pas toujours connexes, cela dépend de ce qui est pris en compte dans le calcul des centres (la couleur seule, la position seule, ou une combinaison des deux). Pour conclure, il existe des **variantes de cet algorithme**, où un aspect spatial va également être pris en compte en ne prenant pas aléatoirement k couleurs, mais k **centres dans l'image**, cf. les différents résultats donnés dans la figure 3.10. Plus précisément, une version plus subtile de cet algorithme est l'algorithme de *mean-shift* qui ne nécessite pas de connaître ou du moins de choisir k (méthode par minimisation d'une fonctionnelle abordée dans le paragraphe suivant).

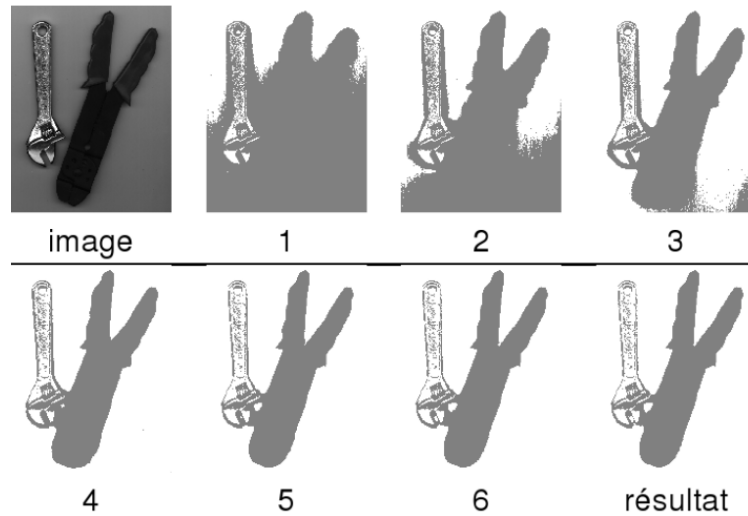


FIGURE 3.9 – Évolution de la segmentation par k-moyenne en prenant en compte uniquement la couleur. Dans cet exemple, utiliser uniquement la couleur ne permet pas de segmenter correctement la clé, ce qui est cohérent avec l’aspect visuel de la clé.

3.6.2 *Mean-Shift* [Comaniciu 02]

Historique :

Le *mean-shift* a été proposé en mathématiques pour **estimer les modes principaux d’une fonction**. Il a été repris par Comaniciu et Meer dans [Comaniciu 99, Comaniciu 02], qui ont montré l’intérêt d’appliquer cet outil à la segmentation d’images.

Principe : Cette technique de segmentation est une des plus fréquemment utilisées dans la littérature. Elle présente l’avantage d’être non supervisée, c’est-à-dire qu’il n’est pas utile de préciser à l’avance le nombre de régions recherchées. En pratique, nous disposons d’un nuage de points de \mathbb{R}^d dont nous souhaitons estimer les modes, sachant que le *mode* d’un nuage de points correspond à un **maximum local de sa fonction de densité**. Une manière de trouver les modes consiste à « **suivre la direction du gradient** » de la fonction de densité.

Approche non-paramétrique : Pour estimer les modes, dans le cas de la méthode *mean-shift*, une approche **non paramétrique** est utilisée, c’est-à-dire que nous ne faisons pas d’hypothèse sur le modèle des données. La solution pour ne faire aucune hypothèse consiste à utiliser une analyse locale par fenêtre, appelée *fenêtre de Parzen*. Soit un échantillon de n observations $\mathbf{x}_i, i = 1 \dots n$. Une estimation de la fonction de densité $\hat{f}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d$, est donnée par :

$$\hat{f}(\mathbf{x}) = \frac{1}{n h^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \quad (3.4)$$

où :

- Le *noyau* K doit posséder les propriétés suivantes : il doit être positif, borné, d’intégrale égale à 1 et à décroissance rapide.
- Le paramètre $h > 0$ permet de régler la taille du voisinage pris en compte. Ce paramètre a une influence sur le nombre de modes détectés. Plus h est petit, plus le nombre de modes détectés augmente. Cela permet de « démystifier » la segmentation non supervisée : au lieu de fixer a

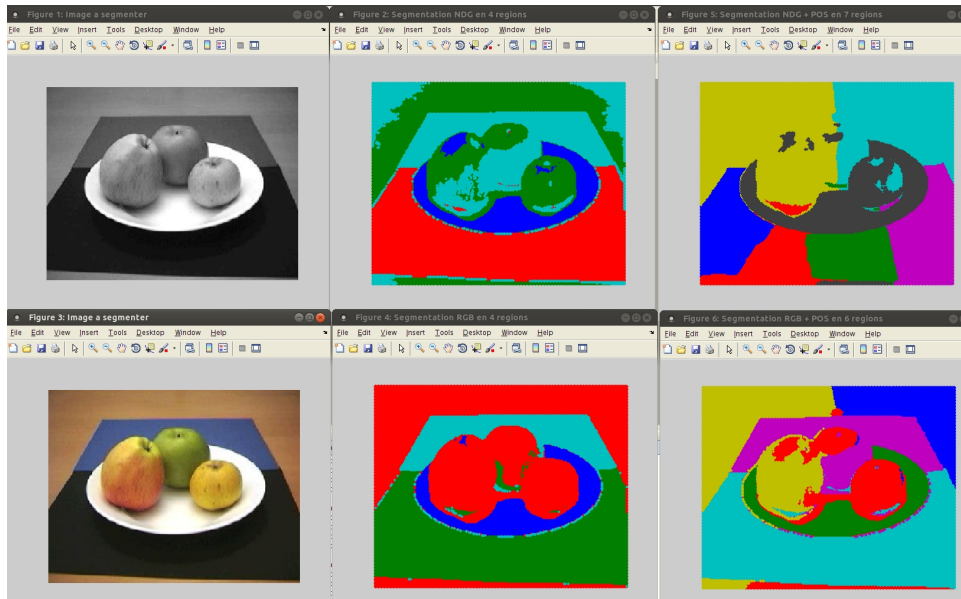


FIGURE 3.10 – Illustration du comportement des k-moyennes suivant les critères utilisés. Sur la première ligne, seuls les niveaux de gris sont utilisés alors que sur la deuxième ligne, la couleur est utilisée. La deuxième colonne ne prend pas en compte l'aspect spatial alors que la troisième combine la photométrie avec l'aspect spatial. Dans cet exemple, utiliser la couleur semble approprié. En revanche, ajouter une contrainte spatiale ne permet pas d'améliorer la qualité des résultats.

priori le nombre de régions, nous fixons un paramètre dont la valeur influe directement sur ce nombre.

Nous pouvons choisir le **noyau d'Epanechnikov** (également appelé *noyau parabolique*), qui est un noyau à support borné défini par :

$$K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) = \begin{cases} \frac{d+2}{2c_d} \left(1 - \frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h^2}\right) & \text{si } \|\mathbf{x} - \mathbf{x}_i\| < h \\ 0 & \text{sinon} \end{cases} \quad (3.5)$$

où c_d désigne le volume de la sphère unité en dimension d et $\|\cdot\|$ désigne la distance euclidienne. L'avantage d'utiliser ce noyau est que l'estimation du gradient de la densité s'écrit alors comme suit :

$$\widehat{\nabla}f(\mathbf{x}) = \frac{1}{nh^d} \frac{d+2}{c_d h^2} \sum_{\mathbf{x}_i \in S_h(\mathbf{x})} (\mathbf{x}_i - \mathbf{x}) = \frac{1}{nh^d} \frac{d+2}{c_d h^2} n_{\mathbf{x}} \left[-\mathbf{x} + \frac{1}{n_{\mathbf{x}}} \sum_{\mathbf{x}_i \in S_h(\mathbf{x})} \mathbf{x}_i \right] \quad (3.6)$$

Dans cette expression, $S_h(\mathbf{x})$ désigne l'ensemble des \mathbf{x}_i se trouvant dans la sphère (en dimension d) de rayon h , centrée en \mathbf{x} , et $n_{\mathbf{x}} = \#\{S_h(\mathbf{x})\}$.

Pour appliquer cette approche au cas de la segmentation d'images couleur, il faut remplacer à chaque itération chaque \mathbf{x} par $\mathbf{x} + \delta_{\mathbf{x}} \widehat{\nabla}f(\mathbf{x})$, où $\delta_{\mathbf{x}} \widehat{\nabla}f(\mathbf{x})$ constitue un « petit déplacement » dans la direction du gradient, pour se rapprocher du mode le plus proche. En choisissant $\delta_{\mathbf{x}} = \left[\frac{1}{nh^d} \frac{d+2}{c_d h^2} n_{\mathbf{x}} \right]^{-1}$, cela revient d'après (3.6) à remplacer \mathbf{x} par la moyenne $M_h(\mathbf{x})$:

$$M_h(\mathbf{x}) = \frac{1}{n_{\mathbf{x}}} \sum_{\mathbf{x}_i \in S_h(\mathbf{x})} \mathbf{x}_i \quad (3.7)$$

Voilà pourquoi cette méthode s'appelle *mean-shift*, qui signifie à peu près « remplacement par la moyenne ».

Étapes

Nous pouvons alors décrire l'algorithme *mean-shift* de manière détaillée comme dans la figure 3.

Algorithme 3 : Algorithme *mean-shift*.

```

1 Fonction estimation_modes ( $I, k$ )
   Entrées :  $h_s, h_c$  : seuil colorimétrique et spatial,  $\epsilon, k_{\max}$  : paramètres d'arrêt
   Sorties : image segmentée et estimation de  $k$  le nombre de classes
2  $k \leftarrow 1$ 
3 tant que  $\exists \mathbf{x}, \|\mathbf{x} - M_h(\mathbf{x})\| > \epsilon$  et  $k \leq k_{\max}$  faire
4   pour chaque  $\mathbf{x}$  faire
5     Calcul de la moyenne  $M_h(\mathbf{x})$  : cf. (3.7)
6      $\mathbf{x} \leftarrow M_h(\mathbf{x})$ 
7    $k \leftarrow k + 1$ 

```

En pratique, pour déterminer $S_h(\mathbf{x})$, la couleur et la position sont traitées différemment : à l'intérieur d'une fenêtre de taille $(2h_s + 1) \times (2h_s + 1)$ centrée en \mathbf{x} , nous considérons les pixels \mathbf{x}_i ayant une couleur proche de celle de \mathbf{x} , c'est-à-dire tels que $\|I(\mathbf{x}_i) - I(\mathbf{x})\| \leq h_c$. Seule la couleur est modifiée au fil des itérations, ce qui signifie que \mathbf{x}, \mathbf{x}_i et $M_h(\mathbf{x})$ sont des vecteurs de \mathbb{R}^3 . L'algorithme dépend donc de quatre paramètres :

- h_s et h_c : seuil spatial et seuil colorimétrique qui définissent l'ensemble $S_h(\mathbf{x})$.
- ϵ et k_{\max} : paramètres qui permettent de contrôler l'arrêt de l'algorithme.

Une illustration du comportement de l'algorithme est donnée dans la figure 3.11.

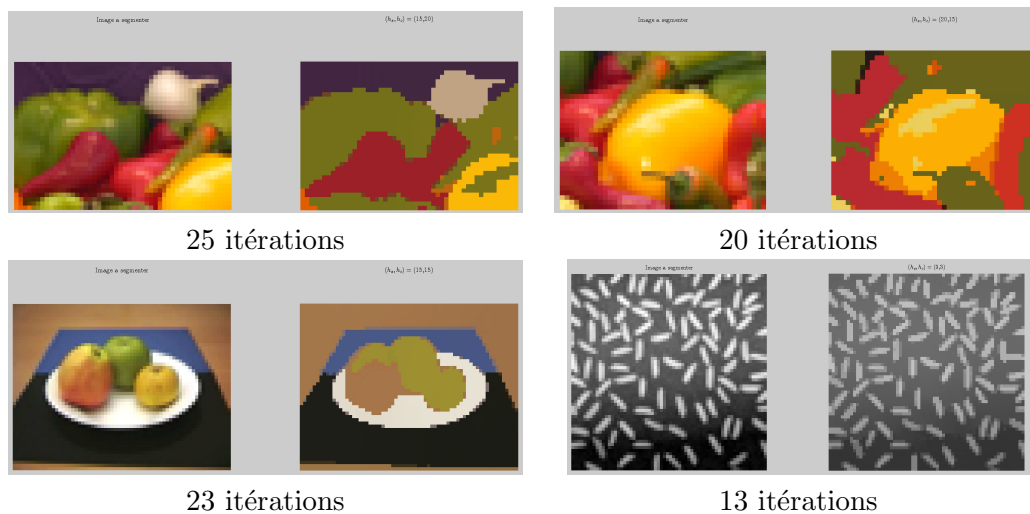


FIGURE 3.11 – Application de l'algorithme de *Mean-Shift* sur différentes images. Nous constatons que quelle que soit l'image traitée, le résultat obtenu est proche de celui attendu. En effet, chaque catégorie de poivrons est séparée, les pommes sont séparées de l'assiette et du set de table et chaque grain de riz est séparé du fond.

3.6.3 Support Vector Machine

En apprentissage supervisé, les séparateurs à vaste marge ou machines à vecteur de support, *Support Vector Machine*, ont été introduits pour permettre la séparation entre différentes classes, c'est-à-dire la classification. Cette approche s'appuie sur deux concepts : la notion de marge maximale et

la notion de fonction noyau, *kernel*, qui ne sont pas des notions récentes. Toutefois, leur utilisation conjointe comme fondement a été introduite seulement en 1992, par [Boser 92]. Depuis, cette technique a été largement employée et nous recommandons cet ouvrage pour en obtenir les détails les plus fins [Duda 01, chapitre 5, page 259].

Brièvement, l'intérêt de cette approche repose sur deux aspects pour réaliser la séparation des classes :

- (1) La capacité à traiter les cas où la séparation n'est pas linéaire en transformant l'espace de représentation des données d'entrées en un espace de dimension plus grande. Cela veut dire que nous supposons qu'il existe une dimension pour laquelle la séparation devient linéaire : c'est-à-dire une dimension où les échantillons sont clairement disjoints entre les N classes à séparer. Le principe de ces fonctions est de projeter les données dans un espace vectoriel de plus grande dimension. C'est pourquoi, dans la littérature, nous parlons de machine à noyau ou *kernel machine*. Il y a différentes possibilités pour le choix du noyau : des noyaux polynomiaux, des noyaux à fonctions de base radiales gaussiennes, des noyaux sigmoïdes.
- (2) La prise en compte d'une marge maximale entre la frontière de séparation, l'hyperplan, et les échantillons les plus proches, les vecteurs supports, contenus dans chaque classe.

3.7 Synthèse et comparaison

Dans le tableau 3.1, nous présentons un récapitulatif de toutes les approches étudiées dans ce chapitre en donnant des éléments clés permettant de faire des choix suivant l'utilisation visée.

En terme de temps d'exécution et de coût mémoire, il est difficile d'établir un classement, mais voici les éléments de comparaison que nous pouvons donner. Il est important de noter que suivant les choix effectués sur les paramètres des différentes méthodes, nous n'obtenons pas le même coût en terme de temps d'exécution ou d'espace mémoire utilisé. Nous supposons, pour tenter d'établir cette comparaison, que les choix de paramètres sont réalisés de manière à obtenir un bon compromis entre temps d'exécution et performance de l'approche. Ainsi, la segmentation par contours ou par histogramme ne va parcourir et traiter l'image qu'une seule fois. Ces deux approches sont donc les moins coûteuses, en temps et en espace mémoire. Les approches par contours actifs ou par fusion/division sont itératives mais les données à étudier/mettre à jour à chaque itération diminuent ou ne concernent pas tous les pixels de l'image, donc, elles sont moins coûteuses que toutes les autres approches qui impliquent d'examiner toute l'image, à chaque itération. Nous pouvons donc dire que les méthodes les plus coûteuses en temps de calculs sont : la ligne de partage des eaux, les k -moyennes et *mean-shift*. Ce sont également les plus coûteuses en terme d'espace mémoire.

Ce chapitre nous a permis de présenter un état de l'art des différentes approches classiques utilisées dans la littérature, en recherche, et dans l'industrie. À présent, nous pouvons aborder les techniques de sursegmentation : les superpixels.

TYPE	NOM	AVANTAGES	INCONVÉNIENTS	UTILISATION
Contours	<i>Seuillage du contour</i>	Mise en œuvre simple	Paramètres difficiles à fixer et beaucoup de pré- ou post-traitements nécessaires	Très utilisé pour des segmentations simples : objet texturé sur fond uniforme et contrasté
	<i>Contours actifs</i>	Détournage précis d'un objet	Bonne initialisation nécessaire et suivant les choix des paramètres, temps de calculs longs	Très utilisés en imagerie médicale, avec interaction utilisateur
Régions	<i>Histogramme</i>	Mise en œuvre simple	Peu de scènes avec histogramme où les modes sont bien distincts	Très utilisés pour des segmentations simples (un objet bien texturé sur un fond uniforme et contrasté)
	<i>Fusion/Division</i>	Mise en œuvre simple	Difficile de fixer les paramètres et de définir les paradigmes à utiliser	Très utilisés en imagerie médicale et avec une interaction utilisateur
	<i>Croissance de région</i>	Mise en œuvre simple	Difficile de fixer les paramètres et de définir les paradigmes à utiliser	Très utilisés en imagerie médicale et avec une interaction utilisateur
	<i>Ligne de partage des eaux</i>	Définition assez intuitive	Beaucoup de pré-traitements à réaliser difficiles à maîtriser et temps de calculs longs	Souvent utilisée couplée à d'autres outils
Classification	<i>k-moyennes</i>	Mise en œuvre simple	Nombre de classes à déterminer et temps de calculs longs	Très utilisés si nombre de classes connu ou possible à déterminer automatiquement (méthode d'Elbow [Tibshirani 01])
	<i>Mean-shift</i>	Mise en œuvre simple	Nombre de classes non nécessaire mais autres paramètres à fixer et temps de calculs longs	Très connu et très utilisé dans la littérature
	<i>Support Vector Machine, SVM</i>	Séparation non linéaire des classes	Plus délicate à mettre en œuvre, beaucoup de variantes possibles	Très connu et très utilisé dans la littérature avant l'apprentissage profond

TABLE 3.1 – Listes des approches présentées.

Chapitre 4

Approche par superpixels

4.1 Introduction sur les approches de sursegmentation

Dans la littérature, de plus en plus de papiers traitent de la construction et/ou de l'utilisation des superpixels [Achanta 12]. En vision par ordinateur, l'élément de base étudié pour réaliser une segmentation, un suivi, un appariement a très souvent été le pixel. Un certain nombre d'approches rapides et efficaces ont ainsi été mises en œuvre. Toutefois, l'information contenue/exploitée par un pixel (l'intensité/la couleur, la position) s'est révélée insuffisante car le **pixel** ne représente pas une entité physique naturelle/réaliste à lui seul et, bien souvent il en résulte une **information ambiguë et incomplète**. Ainsi, très rapidement la notion d'exploitation du voisinage a été introduite, cf. l'utilisation de **régions d'intérêt, voir de *patches***. Ces primitives, appelées **primitives intermédiaires**, permettent d'enrichir l'information exploitée en ajoutant des caractéristiques photométriques et/ou géométriques plus riches que l'intensité/la couleur et/ou la position. Cette notion de région d'intérêt s'est d'ailleurs encore perfectionnée avec la notion de points et objets d'intérêt dont le **détecteur le plus significatif et surtout le plus connu actuellement est SIFT (*Scale Invariant Feature Transform*)** (Nous aborderons cette notion dans le cours de troisième année). Ces détecteurs sont devenus de plus en plus précis, de plus en plus efficaces et ont surtout contribué à **améliorer les performances dans le cadre d'applications précises** comme la classification, la reconnaissance d'objets spécifiques et le suivi. Nous étudierons cette notion de suivi également en troisième année. Toutefois, si nous reprenons l'exemple de la détection d'objets particuliers, les modèles proposés permettent l'amélioration des approches de classification d'objets mais, du fait de la spécialisation de ces opérateurs, dans un but applicatif précis (cf. segmentation fond/forme, reconnaissance de classe d'objets), ceux-ci sont **difficiles à utiliser en dehors du contexte ou de l'application pour laquelle ils ont été proposés**. De la même façon, l'utilisation de *patches* permet une **grande stabilité** utile au cadre du suivi, par exemple, mais ceux-ci ne possèdent **pas de propriétés photométriques et/ou géométriques suffisamment spécifiques** pour servir, par exemple, de base à une segmentation de l'image.

Toutes ces raisons ont amené à exploiter une information différente, certes **moins discriminante qu'un objet ou un point d'intérêt** mais **plus riche en information photométriques et/ou géométriques, moins stable en terme de suivi** mais avec une **forme permettant une meilleure adaptation à la réalité de la scène**, à savoir des sur-segmentations de l'image, qui ont finalement été appelées des **superpixels** depuis [Ren 03].

Bien évidemment, il n'existe pas de définition classique des superpixels plébiscitée par tous les chercheurs car chacun a sa manière propre de les définir. Toutefois, nous donnons une définition suffisamment générique qui puisse être reprise par chacune des approches proposées.

Un **superpixel** est une structure locale et cohérente qui représente une zone ou une partie

d'un même objet.

Les variations de définitions d'un superpixel viennent en fait de l'interprétation mathématique des termes employés dans cette définition, à savoir : structure, locale et cohérente. Plus précisément, le sens de ces termes dépendra de l'ensemble des propriétés souhaitées.

Les différents constructeurs de superpixels présentés ont été publiés entre 2000 [Shi 00] et 2012 [Achanta 12]. Leur utilisation a pris un essor ces dernières années, notamment, de manière évidente comme première étape afin de fournir une segmentation la plus précise possible.

Toutefois, le mot *superpixel* apparaît pour la première fois dans [Ren 03]. Cependant, de nombreux travaux, sans mentionner la notion de superpixels, ont fortement influencé la naissance et la proposition de constructeurs de superpixels. En particulier, l'approche par *mean-shift* [Comaniciu 02] que nous avons déjà étudiée.

4.2 Domaines d'application des superpixels

Ainsi, les superpixels ont été utilisés dans les domaines suivants :

- Bien évidemment, comme étape de pré-segmentation ou plus exactement de sur-segmentation avant une segmentation, comme dans [Hoiem 05]. Dans cet article, les superpixels sont fusionnés en prenant en compte la catégorie associée à chaque superpixel et en vérifiant la cohérence des catégories des superpixels adjacents. Il s'agit donc de combiner segmentation sémantique et utilisation des superpixels.
- Nous avons aussi des approches d'indexation d'images qui exploitent ces outils. Par exemple, dans l'article de [Zhang 10], une image requête est mise en correspondance avec les images indexées en exploitant les superpixels.
- En détection et reconnaissance d'objets, nous pouvons citer de nombreuses approches. Dans [Mori 05], les superpixels sont utilisés pour rechercher un modèle de corps humain dans une image. L'auteur utilise les superpixels pour deux raisons : réduire l'espace de recherche du modèle et avoir un support spatial du masque le plus précis possible, par rapport à une simple boîte englobante.
- Le suivi d'objets rencontre de nombreuses difficultés comme le changement d'échelle, d'orientation ou même de luminosité de l'objet. De plus, l'objet peut, au cours de la séquence être occulté partiellement ou même totalement le temps de quelques images. La difficulté augmente encore lorsque l'objet suivi est non rigide et qu'il se déforme donc d'une image à l'autre. L'hypothèse souvent faite est que deux images consécutives sont suffisamment proches temporellement pour que les modifications de l'objet soient faibles (aussi bien du point de vue de ses modifications internes comme la rotation ou le changement d'échelle que du point de vue de ses modifications externes comme les occultations). Ainsi, les auteurs de [Ren 07] cherchent à effectuer le suivi en s'appuyant sur des superpixels.
- La construction de carte de saillance¹ peut également s'appuyer sur l'utilisation de superpixels. Les premiers algorithmes s'appuyaient sur des études psycho-visuelles, ainsi que sur des dispositifs d'*eye-tracking*. Si ces méthodes de cartes de saillance sont très utilisées, elles ont néanmoins le défaut de représenter l'information qu'elles apportent sous la forme de tâches de lumineuses : une carte de saillance apporte en soi peu d'information sur la structure des objets intéressants. C'est une des raisons pour laquelle [Lu 12] propose une méthode de calcul de carte de saillance basée sur les superpixels. L'approche des auteurs consiste à rechercher des propriétés particulières dans l'image, comme la symétrie, au niveau du superpixel, et donc à construire des

1. La carte de saillance d'une image permet de modéliser l'attention visuelle humaine sur une image donnée. Plus précisément, une carte de saillance associée à une image se présente sous la forme d'une image en niveaux de gris pour laquelle les régions les plus brillantes correspondent aux points qui attirent le plus l'attention d'un utilisateur, et donc qui révèlent les objets intéressants de l'image. S

cartes de saillance qui possèdent une cohérence liée à la structure décrite par les superpixels. Ainsi les structures d'objet apparaissent sur les cartes de saillance.

4.3 Définition/Propriétés

De manière à s'affranchir de la difficulté de donner une définition précise d'un superpixel, puisque celle-ci est étroitement liée à l'application visée, nous avons choisi de décrire les propriétés couramment utilisées dans la littérature pour leur construction avant d'introduire les principaux algorithmes de construction de la littérature. Nous présentons d'abord les propriétés héritées de la collection de pixels qui composent le superpixel, à savoir les propriétés d'**apparence** et les propriétés **spatiales**. Nous nous intéressons ensuite aux propriétés d'**échelle**, relative à leur support spatial, à la résolution de l'image ou encore au nombre de superpixels souhaité.

Par la suite, nous utilisons les notations suivantes :

- N_S : le nombre de superpixels recherché ;
- \mathcal{S} : un ensemble de superpixels, contenant $\#(\mathcal{S})$ superpixels, et S_i un superpixel d'indice i de cet ensemble ;
- \mathbf{p}_i : le pixel d'indice i du superpixel S ;
- σ_S : l'écart-type de la couleur au sein du superpixel S ;
- \mathcal{V}_S : l'ensemble des voisins d'un superpixel S donné.

4.3.1 Propriétés d'apparence

Nous distinguons les aspects suivants :

- *Statistiques sur la colorimétrie ou l'intensité* : les propriétés d'apparence d'un superpixel donné résument les propriétés des pixels qui le composent en appliquant des statistiques de premier ordre (moyenne/écart-type) sur l'intensité ou la couleur des pixels.
- *Rapport intra/inter région* : les méthodes proposées dans [Shi 00, Felzenszwalb 04] minimisent la similarité entre deux régions tout en maximisant la similarité entre pixels au sein d'une même région. Cette notion est appelée *différence intra/inter région* mais il s'agit plutôt de calculer un rapport intra/inter région. Plus formellement, dans [Shi 00], voici le critère C_{DIIR} qui est minimisé :

$$C_{DIIR}(\mathcal{S}) = \frac{1}{\#(\mathcal{S})} \sum_{S_i \in \mathcal{S}} \frac{\frac{1}{\#(\mathcal{V}_{S_i})} \sum_{S_l \in \mathcal{V}_{S_i}} \sigma_{S_i \cup S_l}}{\sigma_{S_i}} \quad (4.1)$$

Avec la méthode décrite dans [Felzenszwalb 04], le critère utilisé pour évaluer la différence intra/inter région entre deux superpixels S_1 et S_2 est donné par :

$$C_D(S_1, S_2) = \frac{\min \left(\max_{(\mathbf{p}_i, \mathbf{p}_j) \in S_1} \delta(\mathbf{p}_i, \mathbf{p}_j) + \frac{c_s}{\#(S_1)}, \max_{(\mathbf{p}_k, \mathbf{p}_l) \in S_2} \delta(\mathbf{p}_k, \mathbf{p}_l) + \frac{c_s}{\#(S_2)} \right)}{\min_{\mathbf{p}_i \in S_1, \mathbf{p}_k \in S_2} \delta(\mathbf{p}_i, \mathbf{p}_k)} \quad (4.2)$$

où c_s est une constante permettant de conditionner l'échelle des superpixels, c'est-à-dire que plus c_s est grand plus la taille des superpixels obtenus sera importante. Le terme δ définit une distance (par exemple, la différence entre les niveaux de gris de deux pixels).

- *Texture* : seul [Shi 00], à notre connaissance, introduit la notion de texture dans la distance utilisée. Dans ce travail, la texture est caractérisée en appliquant des différences de gaussiennes à différentes échelles.

4.3.2 Propriétés spatiales

Nous pouvons considérer les éléments suivants :

- *Position* : lorsqu'elle est prise en compte pour la construction des superpixels dans un critère tel qu'une distance [Shi 00, Achanta 12], la position peut contraindre les superpixels à être connexes et renforcer leur compacité. Ces deux aspects sont définis dans les deux points suivants.
- *Connexité* : un superpixel S est connexe si et seulement si pour toutes paires de pixels $(\mathbf{p}^i, \mathbf{p}^j) \in S$, il existe un chemin les reliant et qui passe uniquement par des pixels de S . Différentes approches sont proposées pour respecter cette propriété de connexité. Dans [Achanta 12], il s'agit d'une étape de post-traitement qui traite les cas de superpixels non connexes². Dans les méthodes de construction basées graphe comme [Felzenszwalb 04], des relations spatiales spécifiques sont définies par la notion de voisinage entre les pixels (les n plus proches voisins ou les 8-voisins) et ainsi la construction même du graphe respecte cette connexité.
- *Compacité* : ce terme est souvent employé dans la description des superpixels mais il est clairement défini seulement dans [Levinshtein 09] où les auteurs la caractérisent comme le rapport entre le carré du périmètre d'un superpixel et son aire :

$$C_{Comp}(S) = \frac{1}{\#(S)} \sum_{S_i \in S} \frac{P_{S_i}^2}{A_{S_i}}, \quad (4.3)$$

où P_S le périmètre du superpixel et A_S son aire. Certains auteurs proposent une construction des superpixels la plus compacte possible en exploitant la distance au centre du superpixel [Achanta 12] alors que d'autres s'appuient sur l'utilisation d'une grille pour initialiser les superpixels et limitent la déformation de cette grille [Moore 09] en exploitant les contours de l'image, c'est-à-dire que les contours des superpixels doivent correspondre aux contours de l'image. Nous pouvons également citer les approches qui imposent une contrainte sur le nombre de pixels par superpixels de manière à réguler la taille des superpixels, comme [Levinshtein 09].

- *Régularité topologique* : une autre propriété souvent favorisée est la régularité topologique [Moore 09], c'est-à-dire, le fait que chaque superpixel possède toujours le même nombre de voisins (sauf dans le cas particulier des bords). Cette propriété est intéressante si une modélisation markovienne des interactions entre primitives est envisagée. La plupart du temps pour respecter cette régularité topologique, une configuration initiale respectant cette propriété (une grille, une disposition régulière de *patches* est utilisée et le modèle d'évolution proposée s'appuie uniquement sur des modifications de la topologie de départ qui n'ont pas d'incidence sur cette contrainte de régularité.
- *Contours de l'image* : enfin, comme précédemment mentionné, de nombreux articles souhaitent favoriser le fait que les superpixels suivent au mieux les contours de l'image. Il s'agit donc de détecter les contours et de faire en sorte que les frontières des superpixels suivent au mieux ces contours [Moore 09, Levinshtein 09, Veksler 10].

4.3.3 Propriété d'invariance temporelle

Des approches permettent d'intégrer la notion d'invariance temporelle ou de cohérence temporelle des superpixels le long d'un flux vidéo ou d'un flux d'images [Grundmann 10]. Dans le cas d'un flux vidéo, la contrainte la plus utilisée s'appuie sur le fait que la position d'un objet entre deux images successives varie très peu et se retrouve dans un petit voisinage de l'image précédente.

2. Les parties non connexes les plus petites du superpixel étudié sont rattachées au superpixel voisin ayant la plus grande taille.

4.3.4 Conclusion sur les propriétés

En conclusion, il y a peu de propriétés différentes réellement abordées dans la littérature parmi les propositions de constructeurs, pourtant, les applications visées requièrent souvent plus de propriétés comme la robustesse aux diverses transformations et la cohérence spatio-temporelle.

4.3.5 Construction/Modélisation

Plusieurs techniques de construction de superpixels existent et chaque auteur prend en compte ses propres contraintes qui induisent un certain nombre de propriétés comme celles listées au § 4.3. La priorité reste souvent la réduction du temps de calcul en réduisant le nombre de primitives, dans le sens de régions, à étudier.

Approches s'appuyant sur la théorie des graphes – Dans ce contexte, une représentation par graphe se décrit ainsi : les nœuds sont les superpixels et les liens, les relations de voisinage entre superpixels. Et, en conséquence, la méthode d'optimisation utilisée pour construire le graphe est la minimisation d'un critère (qui prend en compte les propriétés attendues) par coupure de graphes, *graph-cut*. Un algorithme générique est présenté dans [Shi 00]. Nous pouvons distinguer deux types de structure :

- *Régulière* : celle de [Moore 09] commence par prendre en considération une grille régulière dans l'image en fonction du nombre de superpixels souhaités. Puis chacune des droites de cette grille va évoluer en courbes à l'intérieur d'une bande autour de la droite. Cette évolution est réalisée afin de maximiser la somme des gradients des pixels situés sur la courbe dans le but de forcer les contours des superpixels à être sur les contours de l'image. Les auteurs souhaitent que cet algorithme fournisse plus de superpixels dans les zones de l'image où la densité de contours est plus élevée. Les bandes où évoluent les courbes séparant les superpixels sont donc construits de manière à avoir chacune la même probabilité d'obtenir un contour.
- *Non régulière* : la plupart des approches proposées fournissent des graphes réguliers de superpixels et seul [Felzenszwalb 04] propose une version de son algorithme qui ne respecte pas la propriété de régularité topologique pour donner la priorité à la cohérence photométrique des superpixels.

Approches statistiques – Les algorithmes de coupure de graphe décrits précédemment sont considérés comme coûteux en temps de calcul. Ainsi, d'autres approches s'appuient sur des méthodes de classification classique comme les *k*-moyennes, *k-means*, cf. § 3.6.1, [Vedaldi 08, Achanta 12]. L'approche SLIC, *Simple Linear Iterative Clustering* proposée par [Achanta 12] est la plus connue et la plus utilisée. Dans l'algorithme introduit, une étape d'initialisation distribue de manière régulière les N_c centres des superpixels souhaités. Plus précisément, les centres, notés \mathbf{c}_k , sont les centres des cases d'une grille dont chaque case a une largeur et une hauteur de taille V qui dépend du nombre de superpixels souhaités. Puis, ces centres sont ajustés localement, dans un voisinage 3×3 , pour éviter qu'ils ne se trouvent sur un contour (déplacement vers le gradient local le plus faible). L'algorithme utilisé par la suite est un *k-means* et pour attribuer chaque pixel à un superpixel (une classe), la distance D_s , combinaison pondérée des distances dans l'espace colorimétrique et dans l'espace géométrique, est utilisée :

$$D_s(\mathbf{p}, \mathbf{c}_k) = \sqrt{d_a(\mathbf{p}, \mathbf{c}_k)^2 + \left(\frac{m}{\sqrt{V}}\right)^2 d_p(\mathbf{p}, \mathbf{c}_k)^2} \quad (4.4)$$

où d_a, d_p correspondent aux distances euclidiennes entre \mathbf{c}_k et \mathbf{p} , respectivement sur le critère d'apparence, la couleur, et le critère de position. Le poids m permet de régler l'influence de chacune des distances. Pour affecter un superpixel d'appartenance à chaque pixel de l'image, ce calcul est réalisé pour chaque centre qui se trouve dans un voisinage $2V \times 2V$. De nombreuses variantes ont été proposées

pour SLIC, comme, par exemple, celle de [Wang 12] qui modifie simplement la forme des régions initiales en prenant des hexagones. Les auteurs de [Achanta 12] eux-mêmes ont amélioré cet algorithme en proposant une version non itérative, SNIC, *Simple Non-Iterative Clustering* [Achanta 17].

Approches par croissance de germes – Un dernier groupe de méthodes s’appuient sur l’initialisation et la croissance de superpixels soumis à des contraintes. Parmi ces algorithmes, les TurboPixels de [Levinshtein 09] sont les plus connus. Tout d’abord, pour le placement des germes, les auteurs utilisent exactement la même façon de faire que SLIC, c’est la suite qui est différente. Il est important de noter que les auteurs utilisent des techniques qui s’apparentent à la notion de *Level Set* ou de courbe de niveaux [Chan 02]. Ces approches sont elles mêmes assez proches des approches par contours actifs [Kass 88], cf. § 3.4. Le principe est de faire évoluer des courbes dans l’image qui permettront de définir les frontières des superpixels. Par cette définition, nous pouvons constater que nous nous approchons du concept des contours actifs. Nous notons \mathbf{c} un vecteur contenant les coordonnées de la courbe paramétrée par p , le paramètre qui parcourt la courbe, et t le paramètre d’évolution du temps. Le terme \mathbf{n} correspond à la normal externe associée et chaque point se déplace à la vitesse v avec l’équation d’évolution de la courbe suivante :

$$\frac{\partial \mathbf{c}}{\partial t} = v \mathbf{n}.$$

L’évolution de la courbe est implémentée en représentant \mathbf{c} par une courbe de niveau d’une fonction lisse Ψ telle que :

$$\Psi : \mathbb{R}^2 \times [0, \tau) \rightarrow \mathbb{R}^2 \quad (4.5)$$

$$\Psi_t = -v \|\nabla \Psi\| \quad (4.6)$$

En pratique cette fonction Ψ correspond à la distance euclidienne signée entre le point et la frontière. Cette distance est positive lorsque le point se trouve assigné à la région et négative sinon. Ensuite, dans cet algorithme, l’état Ψ^{n+1} des superpixels à l’itération $n + 1$ est défini en fonction de l’état des superpixels à l’itération précédente Ψ^n de la manière suivante, en suivant la discrétisation, au premier ordre en fonction de t de l’équation 4.6 :

$$\Psi^{n+1} = \Psi^n - v_I v_B \|\nabla \Psi^n\| \Delta t. \quad (4.7)$$

Chaque application de cette équation correspond à une évolution de la courbe Ψ à chaque pas de temps. Cette équation est appliquée jusqu’à ce qu’il n’y ait plus d’évolutions possibles. Le terme le plus important est le produit $v_I v_B$. Le scalaire v_I dépend de la structure locale de l’image et de la géométrie du superpixel en chaque point de la frontière alors que le scalaire v_B dépend de la proximité des points de la frontière avec les points de la frontière des superpixels dans son voisinage. Ces contraintes sont liées par exemple à la courbure des superpixels, à la position de la frontière actuelle des superpixels par rapport à des contours de l’image ou encore à la position d’un superpixel par rapport à un autre (pour empêcher deux superpixels de se recouvrir). Ainsi, v_I et v_B décrivent des vitesses et recouvrent des termes qui accélèrent ou freinent l’expansion des superpixels en fonction des contraintes extérieures. Ce modèle exploite donc simultanément des propriétés d’apparence et des propriétés spatiales.

Dans cette famille, nous pouvons également citer le constructeur SEEDS, *Super-pixels Extracted via Energy-Driven Sampling* [Van den Bergh 12]. À l’aide d’un processus itératif et d’une initialisation sur une grille régulière, seuls les pixels frontaliers à deux superpixels voisins peuvent changer de centre de rattachement en optimisant un terme prenant en compte le caractère homogène du superpixel ainsi que la régularité de sa forme. Plus précisément, l’homogénéité est relative à la fonction de densité de l’histogramme couleur au sein de la région et la régularité de la forme dépend du nombre de superpixels voisins localement (en chaque pixel frontalier).

4.3.6 Classement et comparaison des différents constructeurs de superpixels

Pour conclure, le tableau 4.1 permet de comparer tous les constructeurs présentés en prenant en compte les divers aspects abordés : les critères d'apparence utilisés, le modèle de construction utilisé. Le comportement de tous ces constructeurs est illustré dans la figure 4.1. Pour conclure sur ce travail de présentation et de comparaison, nous donnons des pistes sur la manière de choisir efficacement un constructeur de superpixels. Il n'y pas de constructeur de superpixels à privilégier de manière absolue mais le choix peut être orienté en fonction du contexte applicatif (l'image à étudier, le temps et la mémoire disponible) et de l'utilisation de ce résultat. S'il est primordial d'avoir des temps de calculs faibles, il est important de privilégier l'approche SLIC. Si les superpixels doivent être réguliers, il faut privilégier une approche comme SLIC ou [Mori 05]. Au contraire si nous savons que la scène est suffisamment complexe pour ne pas avoir des superpixels réguliers, il vaut mieux utiliser [Felzenszwalb 04].

CATÉGORIE	RÉFÉRENCE	APPARENCE			SPATIALES					TEMP.
		<i>S</i>	<i>D</i>	<i>T</i>	<i>Pos.</i>	<i>Comp.</i>	<i>Conn.</i>	<i>Rég.</i>	<i>Cont.</i>	
Graphe	[Shi 00]	–			–					
	[Felzenszwalb 04]	–	–		–					
	[Moore 09]	–					–	–	–	
	[Grundmann 10]	–	–		–					–
	[Veksler 10]	–	–			–				–
Statistiques	[Vedaldi 08]	–			–					
	[Achant 12]	–		–	–	–			–	–
	[Wang 12]	–			–	–	–	–		
Germes	[Levinshtein 09]	–			–	–	–	–	–	
	[Van den Bergh 12]	–			–	–	–	–	–	

TABLE 4.1 – Classement des constructeurs de superpixels – L'abréviation TEMP. correspond à la prise en compte de l'aspect temporel. Pour les propriétés d'apparence et spatiales, nous utilisons les notations suivantes : *S*, Statistiques sur la colorimétrie, *D*, Différence intra/inter région, *T*, Texture, *Pos.*, position, *Comp.*, compacité, *Conn.*, connexité, *Rég.*, pour le critère de régularité, *Cont.*, pour l'utilisation des contours de l'image.

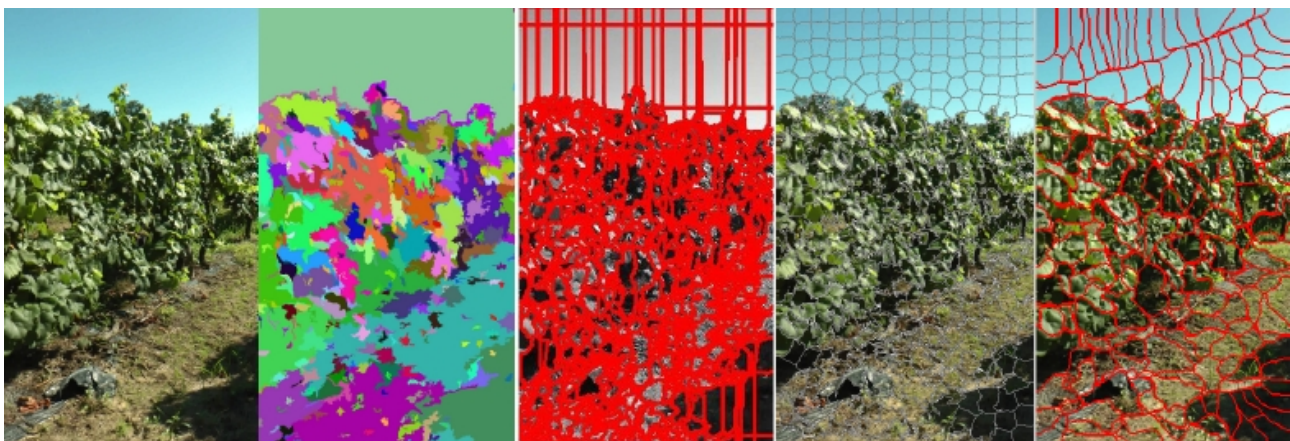


FIGURE 4.1 – Différences de résultats avec différents constructeurs de superpixels – De la gauche vers la droite, il s'agit de l'image originale, un résultat de superpixel donné par [Felzenszwalb 04], par [Moore 09], par [Achant 12] et enfin par [Levinshtein 09].

4.4 Conclusion sur les superpixels

L'analyse de l'état de l'art s'appuie essentiellement sur les publications avant 2012, mais ce domaine de recherche est toujours très actif. Ainsi, certains travaux n'ont pas été abordés mais nous donnons ici les différentes références pour obtenir plus de détails. Des travaux ont été développés sur l'utilisation de polygones pour approximer les formes [Duan 15, Forsythe 16, Bauchet 18]. D'autres approches mettent en avant l'intérêt d'utiliser une analyse globale de l'image, plutôt qu'une analyse locale, comme dans l'approche SLIC, en exploitant des outils d'analyse spectrale [Li 15]. Enfin, en s'appuyant sur le concept de la ligne de partage des eaux, *watershed*, une nouvelle forme de superpixels a été introduite : les *waterpixels* [Machairas 15].

Au cours de ce chapitre, nous avons abordé la notion d'apparence, notamment en abordant les notions de couleur et de texture. Le but des deux chapitres suivants est de développer ces deux aspects afin d'introduire les outils qui peuvent être utilisés.

Chapitre 5

Représentation de la couleur

De nombreux travaux portent sur le processus de formation des couleurs et le système de perception des couleurs de la vision humaine, cf. [Levine 85, chap. 3, 4 et 7], [Carron 95, chap. 1], [Brun 96, chap. 1], [Hardeberg 99, chap. 1] et [Chambah 01, chap. 2], mais, ici, seuls les systèmes de représentation de la couleur seront abordés. Les termes utilisés sont :

- *La luminance* – Il s’agit de la notion d’émettre plus ou moins de lumière.
- *La teinte* – Elle correspond à une couleur, une longueur d’onde dominante.
- *La saturation* – Elle indique le niveau de coloration d’une teinte.

La teinte et la saturation définissent la chrominance d’une couleur. Un système de représentation de la couleur est un système de coordonnées qui permet de représenter la couleur. Ce système, d’après la CIE (Commission Internationale de l’Éclairage), doit posséder trois composantes par analogie au système visuel humain qui comporte trois cônes différents, réceptifs à trois longueurs d’ondes différentes. Une couleur est représentée par les valeurs de ces trois composantes.

Cinq catégories de systèmes peuvent être distinguées [Vandenbroucke 00, chap. 2] :

- les systèmes de primaires ;
- les systèmes luminance-chrominance ;
- les systèmes perceptuels ;
- les systèmes d’axes indépendants ;
- les systèmes hybrides.

Les quatre premières catégories sont classées dans les systèmes classiques, par opposition aux systèmes hybrides. Nous allons présenter les systèmes les plus couramment utilisés pour chaque catégorie.

5.1 Systèmes de primaires

Ces systèmes s’appuient sur l’utilisation de trois couleurs fondamentales (les primaires). Ils se différencient par le choix des primaires et du blanc de référence (codage du blanc). Le système le plus courant est *RGB* (*Red Green Blue*) ou *RVB* (Rouge Vert Bleu), introduit par la CIE. Les composantes couleur de ce système sont définies par $R \in [R_{min}; R_{max}]$, $G \in [G_{min}; G_{max}]$ et $B \in [B_{min}; B_{max}]$. Pour modéliser certaines couleurs du domaine visible, la composante R doit parfois être négative, cf. [Brun 96, p. 15], or, généralement, les intervalles de variation sont $R \in [0; 255]$, $G \in [0; 255]$ et $B \in [0; 255]$. Ce qui signifie que certaines couleurs ne peuvent pas être modélisées. Il y a autant de systèmes *RGB* que de blancs de référence, cf. [Vandenbroucke 00, p. 195] et [Trémeau 04, p. 82].

Un autre système a été proposé par la CIE, cf. [Cie 15.2 86], il s’agit de *XYZ*. Ce système a été proposé pour prendre en compte le problème des valeurs négatives de la composante R du système *RGB*. Chaque composante de ce système est une combinaison linéaire des composantes *RGB*. Pour chaque système *RGB*, une matrice de passage a donc été définie pour passer des coordonnées R , G et B aux coordonnées X , Y et Z , cf. [Vandenbroucke 00, p. 195–197]. Ce système est rarement utilisé

directement car il sert plutôt de système de transition pour passer d'un système RGB vers un autre système. Ici, nous indiquons la transformation la plus couramment employée, cf. [Vandenbroucke 00, p. 196] et [Trémeau 04, p. 90] :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \quad (5.1)$$

Lorsque les intervalles de variation du système RGB sont $R \in [0; 255]$, $G \in [0; 255]$ et $B \in [0; 255]$, les intervalles de variation des composantes XYZ sont $X \in [0; 250.16]$, $Y \in [0; 255]$ et $Z \in [0; 301.41]$. Les trois composantes de ces systèmes sont liées à la luminance. Or, deux couleurs peuvent avoir le même caractère chromatique avec des luminances différentes. Pour ne tenir compte que de la chrominance, les coordonnées peuvent être normalisées de la façon suivante, cf. [Vandenbroucke 00, p. 52–56] :

$$r = \frac{R}{R + G + B}, \quad g = \frac{G}{R + G + B} \quad \text{et} \quad b = \frac{B}{R + G + B}. \quad (5.2)$$

$$x = \frac{X}{X + Y + Z}, \quad y = \frac{Y}{X + Y + Z} \quad \text{et} \quad z = \frac{Z}{X + Y + Z}. \quad (5.3)$$

Cette normalisation permet d'obtenir des coordonnées comprises entre 0 et 1 et comme $x + y + z = 1$, la couleur peut être représentée dans un plan. Ces nouvelles coordonnées sont indépendantes des variations scalaires de la luminance de la couleur étudiée. En effet, si chaque composante R , G et B est multipliée par le même scalaire, alors les valeurs de r , g et b ne changent pas. L'inconvénient de cette normalisation est qu'elle n'est pas définie pour $R = G = B = 0$. Dans ce cas particulier, une solution est de prendre $r = g = b = \frac{1}{3}$, cf. [Garbay 79].

Enfin, le dernier système de primaires, CMY (*Cyan Magenta Yellow*), utilisé dans [Gevers 98, Trémeau 03], s'appuie sur une synthèse soustractive et est utilisé pour la télévision et les imprimantes. Une quatrième composante peut être ajoutée (composante « noire »). La transformation suivante peut être utilisée, cf. [Trémeau 04, p. 89] :

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \quad (5.4)$$

Les intervalles de variation sont $C \in [0; 510]$, $M \in [0; 510]$ et $Y \in [0; 510]$.

Un des inconvénients d'un système de synthèse additive est qu'il ne permet pas de modéliser certaines couleurs (certaines nuances de vert s'obtiennent en ajoutant des lumières bleues et vertes et en retranchant du rouge, cf. [Brun 96, p. 15] or les couleurs sont représentées avec des valeurs positives). Dans de nombreuses applications en imagerie, comme l'affichage, l'impression, la restauration d'images, etc., il est souhaitable que les propriétés d'un système de représentation de la couleur se rapprochent le plus possible de celles de la vision humaine. Si nous notons D , la distance qui sépare deux couleurs dans un système donné et si nous appelons différence, la différence entre deux couleurs perçue par l'œil humain, un système qui se rapproche d'un système visuel humain doit posséder la propriété (P) :

$\forall \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$, trois couleurs telles que la différence de couleur perçue entre \mathbf{c}_1 et \mathbf{c}_3 est la même que celle perçue entre \mathbf{c}_2 et \mathbf{c}_3 alors $D(\mathbf{c}_1, \mathbf{c}_3) = D(\mathbf{c}_2, \mathbf{c}_3)$.

Tous ces systèmes de primaires ne possèdent pas de distance D qui vérifie la propriété (P) et constituent donc une approche limitée pour ces applications. C'est pourquoi d'autres systèmes possédant cette propriété (P) ont été proposés, comme les systèmes luminance-chrominance.

5.2 Systèmes luminance-chrominance

Parmi les systèmes luminance-chrominance, quatre types sont différenciés :

- les systèmes perceptuellement uniformes (une distance D vérifiant (P) peut leur être associée) ;
- les systèmes de télévision que nous ne détaillerons pas ici puisqu'ils sont particulièrement dédiés à la télévision et sont largement abordés dans [Vandenbroucke 00, p. 64–67] ;
- les systèmes antagonistes ;
- les autres systèmes.

Tous ces systèmes possèdent une composante de luminance et deux composantes de chrominance.

5.2.1 Systèmes perceptuellement uniformes

Parmi les systèmes perceptuellement uniformes, deux systèmes sont proposés par la CIE : CIELUV noté aussi $L^*u^*v^*$ et CIELAB noté aussi $L^*a^*b^*$, cf. [Trémeau 04, p. 95–102]. Ces deux systèmes se déduisent de XYZ par les transformations suivantes :

$$L^* = \begin{cases} 116 \left(\frac{Y}{Y_b} \right)^{\frac{1}{3}} - 16 & \text{si } \frac{Y}{Y_b} > 0.01 \\ 903.3 \frac{Y}{Y_b} & \text{sinon,} \end{cases} \quad (5.5)$$

$$u^* = 13L^*(u' - u'_b) \text{ avec } u' = \frac{4X}{X + 15Y + 3Z}, \quad (5.6)$$

$$v^* = 13L^*(v' - v'_b) \text{ avec } v' = \frac{9Y}{X + 15Y + 3Z}, \quad (5.7)$$

$$a^* = 500 \left(f \left(\frac{X}{X_b} \right) - f \left(\frac{Y}{Y_b} \right) \right), \quad (5.8)$$

$$b^* = 200 \left(f \left(\frac{Y}{Y_b} \right) - f \left(\frac{Z}{Z_b} \right) \right), \quad (5.9)$$

$$\text{avec } f(x) = \begin{cases} x^{\frac{1}{3}} & \text{si } x > 0.008856 \\ 7.787x + \frac{16}{116} & \text{sinon.} \end{cases}$$

Les termes X_b, Y_b, Z_b, u'_b et v'_b sont à associer au blanc de référence. Généralement, le blanc de référence est codé $(255 \ 255 \ 255)^T$, en RGB . Ainsi en utilisant l'équation (5.1), nous obtenons : $X_b = 250.155$, $Y_w = 255$ et $Z_w = 301.41$. Les intervalles de variation sont donc : $L^* \in [0; 100]$, $u^* \in [-131.95; 220.8]$, $v^* \in [-139.05; 121.47]$, $a^* \in [-137.72; 96.14]$ et $b^* \in [-99.23; 115.65]$.

5.2.2 Systèmes antagonistes

Les systèmes antagonistes de [Faugeras 79] ont pour but de représenter le mieux possible la perception visuelle humaine. En s'appuyant sur des études physiologiques montrant que l'activité ou la réponse des cônes récepteurs de la rétine humaine, notés LMS (*Long Medium Short*) est proportionnelle non pas à l'intensité du stimulus mais à son logarithme, Faugeras utilise le logarithme des trois signaux L, M et S . Un système semblable s'appuyant sur le système RGB a aussi été proposé par [Garbay 81] :

$$A = \frac{1}{3}(\log(R) + \log(G) + \log(B)), \quad (5.10)$$

$$C_1 = \frac{\sqrt{3}}{2}(\log(R) - \log(G)) \text{ et} \quad (5.11)$$

$$C_2 = \log(B) - \frac{\log(R) + \log(G)}{2}. \quad (5.12)$$

Les intervalles de variation sont donc $A \in [0; \log(255)]$, $C_1 \in [-\frac{\sqrt{3}}{2} \log(255); \frac{\sqrt{3}}{2} \log(255)]$ et $C_2 \in [-\log(255); \log(255)]$. Même si la modélisation de la non linéarité de la réponse des cônes de la rétine

humaine n'est pas conservée, généralement, le système de [Swain 91] est préféré car il n'utilise pas le logarithme :

$$A = \frac{R + G + B}{3}, \quad (5.13)$$

$$C_1 = \frac{\sqrt{3}}{2}(R - G) \text{ et} \quad (5.14)$$

$$C_2 = B - \frac{R + G}{2}. \quad (5.15)$$

Les intervalles de variations sont donc : $A \in [0; 255]$, $C_1 \in \left[-\frac{\sqrt{3}}{2}255; \frac{\sqrt{3}}{2}255\right]$ et $C_2 \in [-255, 255]$.

5.2.3 Autres systèmes

Enfin, d'autres systèmes semblables ont été proposés par [Simonetto 05], ou [Carron 95, p. 17]. Nous présentons le système le plus couramment utilisé, celui de Carron :

$$Y = \frac{R + G + B}{3}, \quad (5.16)$$

$$Ch_1 = R - \frac{G + B}{2} \text{ et} \quad (5.17)$$

$$Ch_2 = \frac{\sqrt{3}}{2}(B - G). \quad (5.18)$$

Ainsi, les intervalles de variation sont $Y \in [0; 255]$, $Ch_1 \in [-255; 255]$ et $Ch_2 \in \left[-\frac{\sqrt{3}}{2}255; \frac{\sqrt{3}}{2}255\right]$.

5.3 Systèmes perceptuels

Les systèmes perceptuels distinguent trois grandeurs : la luminance, la teinte et la saturation. Ils sont notés *HSI* (*Hue Saturation Intensity*). De nombreuses propositions ayant été faites pour le calcul de ces trois composantes, cf. [Hanbury 03, Vandenbroucke 00, p. 74–82], nous ne présentons que les équations les plus utilisées :

$$H = \begin{cases} \pi & \text{si } R = G = B \\ \arccos \frac{\frac{1}{2}((R - G) + (R - B))}{\sqrt{(R - G)^2 + (R - B)(G - B)}} & \text{si } B \leq G \\ 2\pi - \arccos \frac{\frac{1}{2}((R - G) + (R - B))}{\sqrt{(R - G)^2 + (R - B)(G - B)}} & \text{sinon,} \end{cases} \quad (5.19)$$

$$S = \begin{cases} 0 & \text{si } R = G = B = 0 \\ 1 - \frac{3 \min(R, G, B)}{R + G + B} & \text{sinon} \end{cases} \text{ et} \quad (5.20)$$

$$I = \frac{R + G + B}{3}. \quad (5.21)$$

Avec ces équations, les intervalles de variation sont $H \in [0; 2\pi]$, $S \in [0; 1]$ et $I \in [0; 255]$.

5.4 Systèmes d'axes indépendants

Dans les systèmes de primaires, les trois composantes sont fortement corrélées car elles possèdent toutes une information commune qui est la luminance. Des systèmes d'axes indépendants ont été proposés. Ils s'appuient sur l'utilisation de l'analyse en composantes principales qui permet d'obtenir des composantes décorrélées. Les deux systèmes les plus utilisés sont :

- Le système d' [Ohta 80] dont les composantes sont définies par :

$$I_1 = \frac{R + G + B}{3}, \quad (5.22)$$

$$I_2 = \frac{R - B}{2} \text{ et} \quad (5.23)$$

$$I_3 = \frac{2G - R - B}{4}, \quad (5.24)$$

avec $I_1 \in [0; 255]$, $I_2 \in [-\frac{255}{2}; \frac{255}{2}]$ et $I_3 \in [-\frac{255}{2}; \frac{255}{2}]$. Ce système est fondé sur une approximation de la transformation de Karhunen-Loeve, cf. [Levy 00], où I_1 est la composante la plus discriminante.

- Le système $H_1H_2H_3$ de [Braquelaire 97] dont les composantes sont définies par :

$$H_1 = R + G, \quad (5.25)$$

$$H_2 = R - G \text{ et} \quad (5.26)$$

$$H_3 = B - \frac{R + G}{2}, \quad (5.27)$$

où $H_1 \in [0; 510]$, $H_2 \in [-255; 255]$ et $H_3 \in [-255; 255]$. Dans [Levine 85, chap. 7], les auteurs montrent que les trois directions privilégiées par la vision humaine sont les directions : rouge-vert, bleu-jaune et blanc-noir. Ce système met en évidence ces trois directions privilégiées.

5.5 Systèmes hybrides

L'approche hybride de Vandebroucke [Vandebroucke 00, chap. 4] consiste à choisir, pour un ensemble d'images données, parmi toutes les composantes couleur de tous les systèmes de représentation de la couleur, les trois composantes les plus pertinentes pour une application donnée, en utilisant une technique d'apprentissage supervisé. Ces trois composantes forment le système hybride.

5.6 Synthèse des systèmes de représentation de la couleur

Parmi les systèmes qui viennent d'être brièvement présentés, nous reprenons les dix suivants (cf. tableaux 5.1 et 5.2) : RGB , XYZ , CMY , $L^*u^*v^*$, $L^*a^*b^*$, AC_1C_2 , YCh_1Ch_2 , HSI , $I_1I_2I_3$ et $H_1H_2H_3$. Ce sont les plus représentatifs et les plus utilisés en traitement d'images. La remarque qui peut être faite est la suivante : les modèles luminance-chrominance $L^*a^*b^*$ et $L^*u^*v^*$, et le modèle perceptuel HSI se rapprochent le plus de la vision humaine mais il reste à déterminer si cela a une influence sur les résultats obtenus avec des outils de détection de contours ou de segmentation tels que vus dans les chapitres précédents. Cela dépend du type d'images et de l'importance de la couleur dans les images étudiées et le traitement que nous voulons faire. Il faut également garder en tête qu'utiliser la couleur, à la place des niveaux de gris, peut avoir un impact sur le temps d'exécution puisqu'il faut traiter trois fois plus de données et que, de plus, le passage vers un de ces systèmes de représentation est coûteux en temps de calcul.

RÉFÉRENCE	TYPE	DÉFINITION DES COMPOSANTES	INTERVALLES
<i>XYZ</i> [Cie 15.2 86]	Système de primaires	$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$X \in [0; 250.16]$ $Y \in [0; 255]$ $Z \in [0; 301.41]$
<i>CMY</i> [Trémeau 04, p. 89]		$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$C \in [0; 510]$ $M \in [0; 510]$ $Y \in [0; 510]$
<i>Luv</i> [Trémeau 04, p. 95–102]	Système perceptuelle-ment uniforme	$L = \begin{cases} 116 \left(\frac{Y}{Y_b}\right)^{\frac{1}{3}} - 16 & \text{si } \frac{Y}{Y_b} > 0.01 \\ 903.3 \frac{Y}{Y_b} & \text{sinon} \end{cases}$	$L \in [0; 100]$ $u \in [-131.95; 220.8]$ $v \in [-139.05; 121.47]$
<i>Lab</i> [Trémeau 04, p. 95–102]		$u = 13L(u' - u'_b) \text{ avec } u' = \frac{4X}{X+15Y+3Z}$ $v = 13L(v' - v'_b) \text{ avec } v' = \frac{9Y}{X+15Y+3Z}$ $a = 500 \left(f\left(\frac{X}{X_b}\right) - f\left(\frac{Y}{Y_b}\right) \right) \text{ et}$ $b = 200 \left(f\left(\frac{Y}{Y_b}\right) - f\left(\frac{Z}{Z_b}\right) \right) \text{ avec}$ $f(x) = \begin{cases} x^{\frac{1}{3}} & \text{si } x > 0.008856 \\ 7.787x + \frac{16}{116} & \text{sinon} \end{cases}$	$L \in [0; 100]$ $a \in [-137.72; 96.14]$ $b \in [-99.23; 115.65]$
<i>AC₁C₂</i> [Swain 91]	Système antagoniste	$\begin{pmatrix} A \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$A \in [0; 255]$ $C_1 \in \left[-\frac{\sqrt{3}}{2}255; \frac{\sqrt{3}}{2}255\right]$ $C_2 \in [-255; 255]$
<i>YCh₁Ch₂</i> [Carron 95, p. 17]	Système luminance-chrominance	$\begin{pmatrix} Y \\ Ch_1 \\ Ch_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$Y \in [0; 255]$ $Chr_1 \in [-255; 255]$ $Chr_2 \in \left[-\frac{255\sqrt{3}}{2}; \frac{255\sqrt{3}}{2}\right]$

TABLE 5.1 – Systèmes les plus représentatifs parmi ceux abordés dans ce manuscrit.

RÉFÉRENCE	TYPE	DÉFINITION DES COMPOSANTES	INTERVALLES
<i>HSI</i> [Hanbury 03]	Système perceptuel	$H_1 = \left(\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}} \right)$ $H = \begin{cases} \pi & \text{si } R = G = B \\ \arccos H_1 & \text{si } B \leq G \\ 2\pi - \arccos H_1 & \text{sinon} \end{cases}$ $S = \begin{cases} 0 & \text{si } R = G = B = 0 \\ 1 - \frac{\min(R,G,B)}{R+G+B} & \text{sinon} \end{cases}$ $I = \frac{R+G+B}{3}$	$I \in [0; 255]$ $S \in [0; 1]$ $H \in [0; 2\pi]$
$I_1 I_2 I_3$ [Ohta 80]	Système d'axes indépendants	$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$I_1 \in [0; 255]$ $I_2 \in [-127.5; 127.5]$ $I_3 \in [-127.5; 127.5]$
$H_1 H_2 H_3$ [Braquelair 97]		$\begin{pmatrix} H_1 \\ H_2 \\ H_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ -\frac{1}{2} & 0 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$	$H_1 \in [0; 510]$ $H_2 \in [-255; 255]$ $H_3 \in [-255; 255]$

TABLE 5.2 – Systèmes les plus représentatifs parmi ceux abordés dans ce manuscrit (suite).

Chapitre 6

Étude de la texture

Dans l'étude de la texture, nous pouvons considérer deux aspects :

1. Déterminer si un pixel appartient à une zone texturée ;
2. Caractériser la texture (en donner une signature permettant de la reconnaître).

Ainsi, nous appelons un opérateur de texture, une mesure permettant d'évaluer à quel point un pixel peut être considéré comme appartenant à une texture. Les opérateurs de texture utilisent différentes méthodes : les méthodes structurelles-géométriques, les méthodes statistiques, les méthodes fondées sur l'étude d'un modèle et les méthodes de filtrage. Nous donnons la description de deux de ces opérateurs, parmi les méthodes statistiques : Densité de points de contour et auto-corrélation. Ensuite, nous présentons un outils de caractérisation : les matrices de cooccurrences.

6.1 Présentation de quelques opérateurs de texture

6.1.1 Densité de points de contour

Pour un pixel donné, si le nombre de points de contour dans son voisinage est supérieur à un seuil donné alors ce pixel appartient à une zone texturée, soit :

Si $\#(\mathcal{C}(p)) > S$ avec $\mathcal{C}(p) = \{p' \in \mathcal{V}(p) | p' \text{ est un point contour}\}$ alors p est un point texturé.

Le terme \mathcal{V} définit un voisinage autour p . Pour utiliser cet opérateur, il faut donc déterminer la valeur du seuil S utilisé, ainsi que le type de voisinage \mathcal{V} considéré.

6.1.2 Auto-corrélation

Le principe est de calculer une distance (mesure de ressemblance) entre deux régions voisines dans l'image. Plus cette distance est élevée, plus la région est fortement texturée. Nous pouvons utiliser la fonction d'auto-corrélation suivante :

$$\rho(x, y) = \frac{1}{(2n+1)(2m+1)} \sum_{p=-n}^n \sum_{q=-m}^m (I(x+p-n, y+q) - I(x+p+n, y+q))^2. \quad (6.1)$$

Il s'agit de la somme des différences des niveaux de gris au carré. Le terme $I(x, y)$ est le niveau de gris du pixel de coordonnées (x, y) . La taille de chaque région considérée est $(2n+1)(2m+1)$. Si le score de corrélation est au dessus d'un seuil donné alors la région est texturée. D'autres mesures de corrélation pourraient être utilisées comme la somme des valeurs absolues des différences ou une corrélation croisée.

6.2 Caractérisation d'une texture par matrice de cooccurrences

De taille $N \times N$, avec N le nombre de niveaux de gris, elle est notée \mathbf{M}_d et chaque élément est défini par :

$$\mathbf{M}_d(i, j) = \# \{(r, s), (t, v) \mid I(r, s) = i \text{ et } I(t, v) = j \text{ et distance}((r, s), (t, v)) = \mathbf{d}\}. \quad (6.2)$$

Le terme \mathbf{d} est le vecteur distance entre les points (r, s) et (t, v) . Par exemple, \mathbf{d} peut être choisi tel que $\mathbf{d} = (1, 0)^T$. Pour déterminer la signature d'une région, Haralick propose quatorze paramètres différents à évaluer. Les plus utilisés sont :

<p style="text-align: center;">Énergie</p> $\sum_{i=0}^N \sum_{j=0}^N \mathbf{M}_d(i, j)^2$	<p style="text-align: center;">Entropie</p> $-\sum_{i=0}^N \sum_{j=0}^N \mathbf{M}_d(i, j) \log(\mathbf{M}_d(i, j))$	<p style="text-align: center;">Corrélation</p> $\left(\sum_{i=0}^N \sum_{j=0}^N ij \mathbf{M}_d(i, j) - \mu_i \mu_j \right) / (\sigma_i \sigma_j)$
$\mu_i = \frac{1}{N} \sum_{x=0}^N \mathbf{M}_d(i, x), \mu_j = \frac{1}{N} \sum_{y=0}^N \mathbf{M}_d(y, j), \sigma_i = \frac{1}{N} \sum_{x=0}^N (\mathbf{M}_d(i, x) - \mu_i)^2, \sigma_j = \frac{1}{N} \sum_{y=0}^N (\mathbf{M}_d(y, j) - \mu_j)^2.$		

Comme nous l'avons vu en introduction de ce chapitre, nous pouvons définir la texture de différentes manières. Et ainsi, nous pouvons par exemple la définir comme étant une zone possédant une certaine entropie ou au contraire une certaine auto-corrélation, donc, tous les critères présentés sont intéressants à étudier suivant le type de texture cherché.

Bibliographie

- [Achanta 12] R. ACHANTA, A. SHAJI, K. SMITH, A. LUCCHI, P. FUA et S. SUSSTRUNK. *SLIC Superpixels Compared to State-of-the-art Superpixel Methods*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 34(11):2274–2282, septembre 2012.
- [Achanta 17] R. ACHANTA et S. SUSSTRUNK. *Superpixels and polygons using simple non-iterative clustering*. Dans *IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR*, 2017.
- [Amhaz 16] R. AMHAZ, S. CHAMBON, J. IDIER et V. BALTAZART. *Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection*. *IEEE Transactions on Intelligent Transportation Systems, TITS*, 17(10):2718–2729, 2016.
- [Bakkay 18] M. C. BAKKAY, S. CHAMBON, C. LUBAT et S. N. BARSOTTI. *Automatic detection of individual and touching moths from trap images by combining contour-based and region-based segmentation*. *IET Computer Vision*, 12(2):138–145, 2018.
- [Barnes 09] C. BARNES, E. SHECHTMAN, A. FINKELSTEIN et D. B. GOLDMAN. *PatchMatch: a randomized correspondence algorithm for structural image editing*. *ACM transactions on graphics, TOG*, 28:24:1–24:11, juillet 2009.
- [Bauchet 18] J.-P. BAUCHET et F. LAFARGE. *KIPPI: KInetic Polygonal Partitioning of Images*. Dans *IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR*, 2018.
- [Bauda 15] M.-A. BAUDA, S. CHAMBON, P. GURDJOS et V. CHARVILLAT. *Geometry-based Superpixel Segmentation - Introduction of Planar Hypothesis for Superpixel Construction*. Dans *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP*, pages 227–232, 2015.
- [Bengio 09] Y. BENGIO. *Learning deep architectures for AI*. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
- [Boser 92] B. E. BOSER, I. M. GUYON et V. N. VAPNIK. *A Training Algorithm for Optimal Margin Classifiers*. Dans *Annual Workshop on Computational Learning Theory (COLT)*, pages 144–152, 1992.
- [Braquelaire 97] J. P. BRAQUELAIRE et L. BRUN. *Comparison and optimization of methods of color image quantization*. *IEEE Transactions on Image Processing, IP*, 6(7):1048–1052, juillet 1997.
- [Bray 06] M. BRAY, P. KOHLI et P. H. S. TORR. *PoseCut: Simultaneous Segmentation and 3D Pose Estimation of Humans Using Dynamic Graph-Cuts*. Dans *proceedings of European Conference on Computer Vision, ECCV*, volume *Lecture Notes in Computer Science 3952*, mai 2006.
- [Brun 96] L. BRUN. *Segmentation d'images à base topologique*. Thèse de doctorat, Université de Bordeaux I, décembre 1996.
- [Canny 86] J. CANNY. *A Computational Approach To Edge Detection*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 8(6):679–714, 1986.
- [Carron 95] T. CARRON. *Segmentation d'images couleur dans la base Teinte Luminance Saturation : approche numérique et symbolique*. Thèse de doctorat, Université de Savoie, France, décembre 1995.

- [Chambah 01] M. CHAMBAH. *Analyse et traitement de données chromatiques d'images numérisées à haute résolution. Application à la restauration numérique des couleurs des films cinématographiques*. Thèse de doctorat, Université de La Rochelle, France, décembre 2001.
- [Chan 02] T. F. CHAN et L. A. VESE. *A Multiphase Level Set Framework for Image Segmentation using the Mumford and Shah model*. *International Journal of Computer Vision*, 50(3):271–293, 2002.
- [Chen 98] C.-S. CHEN, Y.-P. HUNG et J.-B. CHENG. *A fast automatic method for registration of partially-overlapping range images*. Dans IEEE Conference Proceedings of International Conference on Computer Vision, ICCV, pages 242–248, janvier 1998.
- [Chen 99] C.-S. CHEN, Y.-P. HUNG et J.-B. CHENG. *RANSAC based DARCES – A New Approach to Fast Automatic Registration of Partially Overlapping Range Images*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 21:1229–1234, novembre 1999.
- [Cie 15.2 86] CIE 15.2. *Colorimetry, second edition*. Rapport Technique, Commission Internationale de l'Éclairage, Vienne, Autriche, 1986.
- [Collins 02] M. COLLINS, R. E. SCHAPIRE et Y. SINGER. *Logistic Regression, AdaBoost and Bregman Distances*. *Machine Learning, ML*, 48:253–285, 2002.
- [Comaniciu 99] D. COMANICIU et P. MEER. *Mean shift analysis and applications*. Dans IEEE Conference Proceedings of International Conference on Computer Vision, ICCV, volume 2, pages 1197–1203, 1999.
- [Comaniciu 02] D. COMANICIU et P. MEER. *Mean shift: a robust approach toward feature space analysis*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 24(5):603–619, 2002.
- [Dambreville 08] S. DAMBREVILLE, R. SANDHU, A. YEZZI et A. TANNENBAUM. *Robust 3D Pose Estimation and Efficient 2D Region-Based Segmentation from a 3D Shape Prior*. Dans proceedings of European Conference on Computer Vision, ECCV, volume 2, pages 169–182, octobre 2008.
- [Dempster 77] A. P. DEMPSTER, N. M. LAIRD et D.B. RUBIN. *Maximum likelihood from incomplete data via the EM algorithm*. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- [Deriche 87] R. DERICHE. *Using Canny's criteria to derive a recursively implemented optimal edge detector*. *International Journal of Computer Vision, IJCV*, 2(6):167–187, 1987.
- [Dijkstra 59] E. W. DIJKSTRA. *A Note on Two Problems in Connexion with Graphs*. *Numerische Mathematik*, 1(1):269–271, 1959.
- [Domingos 97] P. DOMINGOS et M. PAZZANI. *On the optimality of the simple Bayesian classifier under zero-one loss*. *Machine Learning, ML*, 29(2-3):103–130, 1997.
- [Duan 15] L. DUAN et F. LAFARGE. *Image partitioning into convex polygons*. Dans IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR, pages 3119–3127, 2015.
- [Duda 01] R. O. DUDA, P. E. HART et D. G. STORK. *Pattern classification*. Wiley-interscience, 2001.
- [El Merabet 13] Y. EL MERABET. *Segmentation d'images couleur par combinaison LPE-régions/LPE-contours et fusion de régions – Application à la segmentation de toitures à partir d'orthophotoplans*. PhD thesis, Université de Technologie de Belfort-Montbéliard, 2013.
- [Faugeras 79] O. FAUGERAS. *Digital Color Image Processing Within the Framework of a Human Visual Model*. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27(4):380–393, août 1979.
- [Felzenszwalb 04] P. F. FELZENSZWALB et D. P. HUTTENLOCHER. *Efficient Graph-Based Image Segmentation*. *International Journal of Computer Vision, IJCV*, 59(2):167–181, 2004.

- [Forsythe 16] J. FORSYTHE, V. KURLIN et A. FITZGIBBON. *Resolution-Independent Superpixels Based on Convex Constrained Meshes Without Small Angles*. Dans International Conference on Advances in Visual Computing, ISVC, pages 223–233, 2016.
- [Fulkerson 09] B. FULKERSON, A. VEDALDI et S. SOATTO. *Class segmentation and object localization with superpixel neighborhoods*. Dans IEEE Conference Proceedings of International Conference on Computer Vision, ICCV, pages 670–677, octobre– novembre 2009.
- [Garbay 79] C. GARBAY. *Modélisation de la couleur dans le cadre de l’analyse d’images et de son application à la cytologie automatique*. Thèse de doctorat, Institut National Polytechnique, INP, Grenoble, France, décembre 1979.
- [Garbay 81] C. GARBAY, F. BRUGAL et C. CHOQUET. *Application of Colored Image Analysis to Bone Marrow Cell Recognition*. *Analytical and Quantitative Cytology*, 3(4):272–280, juillet 1981.
- [Gevers 98] T. GEVERS, A. W. M. SMEULDERS et H. STOKMAN. *Photometric Invariant Region Detection*. Dans proceedings of British Machine Vision Conference, BMVC, pages 578–589, Southampton, Royaume-Uni, septembre 1998.
- [Goodfellow 14] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, X. BING, D. WARDE-FARLEY, S. OZAIR, A. COURVILLE et Y. BENGIO. *Generative adversarial nets*. Dans Advances in neural information processing systems, pages 2672–2680, 2014.
- [Goodfellow 16] I. GOODFELLOW, Y. BENGIO et A. COURVILLE. *Deep learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [Grundmann 10] M. GRUNDMANN, V. KWATRA, M. HAN et I. ESSA. *Efficient hierarchical graph-based video segmentation*. Dans IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR, pages 2141–2148, 2010.
- [Hanbury 03] A. HANBURY. *A 3D-Polar Coordinate Colour Representation Well Adapted to Image Analysis*. Dans proceedings of Scandinavian Conference on Image Analysis, SCIA, pages 804–811, Göteborg, Suède, juin 2003.
- [Hardeberg 99] J. Y. HARDEBERG. *Acquisition and reproduction of colour images: colorimetric and multispectral approaches*. Thèse de doctorat, École Nationale Supérieure des Télécommunications, ENST, Paris, France, janvier 1999.
- [Heitz 08] G. HEITZ et D. KOLLER. *Learning Spatial Context: Using Stuff to Find Things*. Dans proceedings of European Conference on Computer Vision, ECCV, volume 1, pages 30–43, octobre 2008.
- [Hoiem 05] D. HOIEM, A. A. EFROS et M. HERBERT. *Geometric context from a single image*. Dans IEEE Conference Proceedings of International Conference on Computer Vision, ICCV, volume 1, pages 654–661, Pittsburg, États-Unis, octobre 2005.
- [Horowitz 76] S. L. HOROWITZ et T. PAVLIDIS. *Picture Segmentation by a Tree Traversal Algorithm*. *Journal of the Association for Computing Machinery, ACM*, 23(2):368–388, 1976.
- [Kass 88] M. KASS, A. WITKIN et D. TERZOPOULOS. *Snakes : Active contour models*. *International Journal of Computer Vision, IJCV*, 1(4):321–331, 1988.
- [Kermad 97] C. KERMADE. *Image Segmentation: toward an automatic and unsupervised system through methods cooperation*. PhD thesis, Université Rennes 1, 1997.
- [Kholi 08] P. KHOLI, J. RIHAN, M. BRAY et P. TORR. *Simultaneous segmentation and 3D pose estimation of humans using dynamic graph cuts*. *International Journal of Computer Vision, IJCV*, 79(3):285–298, 2008.
- [Krizhevsky 12] A. KRIZHEVSKY, I. SUTSKEVER et G. E. HINTON. *ImageNet Classification with Deep Convolutional Neural Networks*. Dans International Conference on Neural Information Processing Systems, NIPS, pages 1097–1105, 2012.

- [LeCun 15] Y. LECUN, Y. BENGIO et G. E. HINTON. *Deep learning*. *Nature*, 521(7553):436, 2015.
- [Levine 85] M. D. LEVINE. *Vision in man and machine*. McGraw-Hill Book Company, 1985.
- [Levinshtein 09] A. LEVINSHTEIN, A. STERE, K. N. KUTULAKOS, D. J. FLEET, S. J. DICKINSON et K. SIDDIQI. *TurboPixels: Fast Superpixels Using Geometric Flows*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 31(12):2290–2297, décembre 2009.
- [Levy 00] A. LEVY et M. LINDENBAUM. *Sequential Karhunen-Loeve Basis Extraction and its Application to Images*. *IEEE Transactions on Image Processing, IP*, 9(8):1371–1374, août 2000.
- [Li 15] Z. LI et J. CHEN. *Superpixel segmentation using linear spectral clustering*. Dans *IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR*, pages 1356–1363, 2015.
- [Lu 12] Y. LU, W. ZHANG, C. JIN et X. XUE. *Learning attention map from images*. Dans *IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR*, pages 1067–1074, juin 2012.
- [Machairas 15] V. MACHAIRAS, M. FAESSEL, D. CÁRDENAS-PEÑA, T. CHABARDES, T. WALTER et E. DECENCIÈRE. *Waterpixels*. *IEEE Transactions on Image Processing, TIP*, 24(11):3707–3716, 2015.
- [MacQueen 67] J. MACQUEEN. *Some methods for classification and analysis of multivariate observations*. Dans *Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, 1967.
- [Martin 01] D. MARTIN, C. FOWLKES, D. TAL, J. et MALIK. *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*. Dans *IEEE Conference Proceedings of International Conference on Computer Vision, ICCV*, volume 2, pages 416–423, juillet 2001.
- [Marvin 69] M. MARVIN et A. SEYMOUR. *Perceptrons*. MIT Press, 1969.
- [Mathevet 99] S. MATHEVET, L. TRASSOUDAINE, P. CHECCHIN et J. ALIZON. *Combinaison de segmentations en régions*. *Traitement du Signal*, 16(2):93–104, 1999.
- [McCulloch 43] W. S. MCCULLOCH et W. PITTS. *A logical calculus of the ideas immanent in nervous activity*. *Bulletin of mathematical biophysics*, 5:115–133, 1943.
- [Mitchell 97] T. M. MITCHELL. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.
- [Moore 09] A.P. MOORE, S. J. D. PRINCE, J. WARRELL, U. MOHAMMED et G. JONES. *Scene shape priors for superpixel segmentation*. Dans *IEEE Conference Proceedings of International Conference on Computer Vision, ICCV*, pages 771–778, octobre– novembre 2009.
- [Mori 05] G. MORI. *Guiding model search using segmentation*. Dans *IEEE Conference Proceedings of International Conference on Computer Vision, ICCV*, volume 2, pages 1417–1423, octobre 2005.
- [Nair 10] V. NAIR et G. E. HINTON. *Rectified Linear Units Improve Restricted Boltzmann Machines*. Dans *International Conference on Machine Learning, ICML*, page807–814, 2010.
- [Ohta 80] Y.-I. OHTA, T. KANADE et T. SAKAI. *Color Information for Region Segmentation*. *Computer Graphics and Image Processing, CGIP*, 13(3):222–241, juillet 1980.
- [Otsu 79] N. OTSU. *A Threshold Selection Method from Gray-Level Histograms*. *IEEE Transactions on Systems, Man and Cybernetics, SMC*, 9(1):62–66, janvier 1979.
- [Pont-Tuset 15] J. PONT-TUSET et F. MARQUÉS. *Supervised evaluation of image segmentation and object proposal techniques*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 38(7):1465–1478, 2015.
- [Radford 15] A. RADFORD, L. METZ et S. CHINTALA. *Unsupervised representation learning with deep convolutional generative adversarial networks*. *CoRR (arXiv)*, abs/1511.06434, 2015.

- [Radford 16] A. RADFORD, L. METZ et S. CHINTALA. *Unsupervised representation learning with deep convolutional generative adversarial networks*. Dans International Conference on Learning Representations, ICLR, 2016.
- [Ren 03] X. REN et J. MALIK. *Learning a classification model for segmentation*. Dans IEEE Conference Proceedings of International Conference on Computer Vision, ICCV, volume 1, pages 10–17, 2003.
- [Ren 07] X. REN et J. MALIK. *Tracking as Repeated Figure/Ground Segmentation*. Dans IEEE Conference Proceedings of Computer Vision and Pattern Recognition, CVPR, juin 2007.
- [Rosenblatt 58] F. ROSENBLATT. *The perceptron: A probabilistic model for information storage and organization in the brain*. *Psychological Review*, 65(6):386–408, 1958.
- [Rosenhahn 07] B. ROSENHAHN, T. BROX et J. WEICKERT. *Three-Dimensional Shape Knowledge for Joint Image Segmentation and Pose Tracking*. *International Journal of Computer Vision, IJCV*, 73:243–262, 2007.
- [Rumelhart 86] D. E. RUMELHART, G. E. HINTON et R. J. WILLIAMS. *Learning representations by back-propagating errors*. *Nature*, 323(6088):533–536, 1986.
- [Russakovsky 15] O. RUSSAKOVSKY, J. DENG, H. SU, J. KRAUSE, S. SATHEESH, S. MA, Z. HUANG, A. KARPATY, A. KHOSLA, M. BERNSTEIN, A. C. BERG et L. FEI-FEI. *ImageNet Large Scale Visual Recognition Challenge*. *International Journal of Computer Vision, IJCV*, 115(3):211–252, 2015.
- [Samuel 59] A. L. SAMUEL. *Some studies in machine learning using the game of Checkers*. *IBM journal of Research and Development*, pages 71–105, 1959.
- [Sebari 07] I. SEBARI et D. C. HE. *Les approches de segmentation d’image par coopération régions-contours*. *Revue Télédétection*, 7:499–506, 2007.
- [Sepp 97] H. SEPP et J. SCHMIDHUBER. *Long Short-Term Memory*. *Neural Computation*, 9(8):1735–1780, 1997.
- [Serra 94] J. SERRA. *Morphological filtering: An overview*. *Signal Processing*, 38(1):3–11, 1994.
- [Shi 00] J. SHI et J. MALIK. *Normalized cuts and image segmentation*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 22(8):888–905, 2000.
- [Simonetto 05] E. SIMONETTO et A. SOADANE. *Colour Image Watermarking Using a Visual Sub-Band Decomposition*. *International Journal of Robotics and Automation, IJRA*, 20(2):101–108, 2005.
- [Swain 91] M. J. SWAIN et D. H. BALLARD. *Color Indexing*. *International Journal of Computer Vision, IJCV*, 7(1):11–32, novembre 1991.
- [Szeliski 10] R. SZELISKI. *Computer vision: Algorithms and applications*. Springer, 2010. <http://szeliski.org/Book/>.
- [Tibshirani 01] R. TIBSHIRANI, G. WALTHER et T. HASTIE. *Estimating the number of clusters in a data set via the gap statistic*. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
- [Trémeau 03] A. TRÉMEAU et P. COLANTONI. *Visualisation 3D dédiée à l’analyse des images couleur*. Dans actes du congrès francophone de Vision par Ordinateur, ORASIS, pages 261–268, Gérardmer, France, mai 2003.
- [Trémeau 04] A. TRÉMEAU, C. FERNANDEZ-MALOIGNE et P. BONTON, éditeurs. *Image numérique couleur – de l’acquisition au traitement*. Dunod, janvier 2004.
- [Tu 10] Z. TU et X. BAI. *Auto-Context and Its Application to High-Level Vision Tasks and 3D Brain Image Segmentation*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 32(10):1744–1757, 2010.

- [Van den Bergh 12] M. VAN DEN BERGH, X. BOIX, G. ROIG, B. DE CAPITANI et L. J. VAN GOOL. *SEEDS: Superpixels Extracted via Energy-Driven Sampling*. Dans proceedings of European Conference on Computer Vision, ECCV, pages 13–26, 2012.
- [Vandenbroucke 00] N. VANDENBROUCKE. *Segmentation d'images couleur par classification de pixels dans des espaces d'attributs colorimétriques adaptés. Application à l'analyse d'images de football*. Thèse de doctorat, Université des sciences et technologies de Lille 1, France, décembre 2000.
- [Vedaldi 08] A. VEDALDI et S. SOATTO. *Quick Shift and Kernel Methods for Mode Seeking*. Dans proceedings of European Conference on Computer Vision, ECCV, pages 179–192, 2008.
- [Veksler 10] O. VEKSLER, Y. BOYKOV et P. MEHRANI. *Superpixels and supervoxels in an energy optimization framework*. Dans proceedings of European Conference on Computer Vision, ECCV, pages 211–224, 2010.
- [Vincent 91] L. VINCENT et P. SOILLE. *Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 13(6):583–598, 1991.
- [Wang 12] J. WANG et X. WANG. *VCells: Simple and Efficient Superpixels Using Edge-Weighted Centroidal Voronoi Tessellations*. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 34:1241–1247, 2012.
- [Weippl 07] E. R. WEIPPL, M. KLEMEN, S. FENZ, A. EKELHART et A. M. TJOA. *The semantic desktop: a semantic personal information management system based on RDF and topic maps*. Dans VLDB conference on Ontologies-based databases and information systems, pages 135–151, 2007.
- [Y. J. Zhang 01] Y. J. ZHANG. *A review of recent evaluation methods for image segmentation*. Dans International Symposium on Signal Processing and its Applications, volume 1, pages 148–151, 2001.
- [Zhang 96] Y.J. ZHANG. *A survey on evaluation methods for image segmentation*. *Pattern Recognition, PR*, 29(8):1335–1346, 1996.
- [Zhang 10] H. ZHANG, J. XIAO et L. QUAN. *Supervised label transfer for semantic segmentation of street scenes*. Dans proceedings of European Conference on Computer Vision, ECCV, pages 561–574, mai 2010.
- [Zhuge 04] H. ZHUGE. *Retrieve images by understanding semantic links and clustering image fragments*. *Journal of Systems and Software*, 73(3):455–466, novembre 2004.