M1 MAPI3
Mathématiques Appliquées Pour
l'Ingénierie l'Industrie et l'Innovation

Segmentation d'image – Mumford-Shah

Auteurs :

RAYNAL Adrien
REVAILLER Wendy
WILLEMOT Sophie
ZOUITINE Mehdi

Encadrant :

MALGOUYRES François
MARÉCHAL Pierre
MESCAM Muriel

Le 27 avril 2020

Table des matières

1 Introduction

La segmentation d'images est l'un des grands domaines du traitement d'images et de la vision par ordinateur. Cette opération a pour but de rassembler des pixels entre eux suivant des critères pré-définis. On marque ainsi les pixels correspondant à un objet dans une image. Les pixels regroupés forment une partition de l'image.

La segmentation est notamment utilisée pour simplifier ou changer la représentation d'une image en quelque chose de plus facile à analyser. On l'utilise beaucoup en imagerie médicale, dans la recherche d'image par contenu, dans la détection d'objet, la reconnaissance faciale (doigts et iris également), en vidéo surveillance, ...

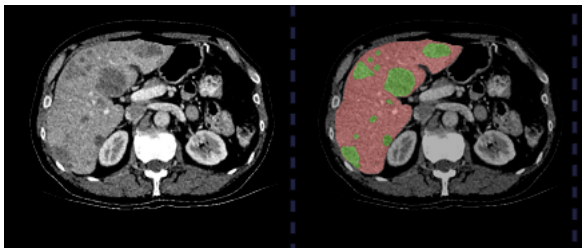

FIGURE 1: Segmentation en imagerie médicale pour déceler des tumeurs

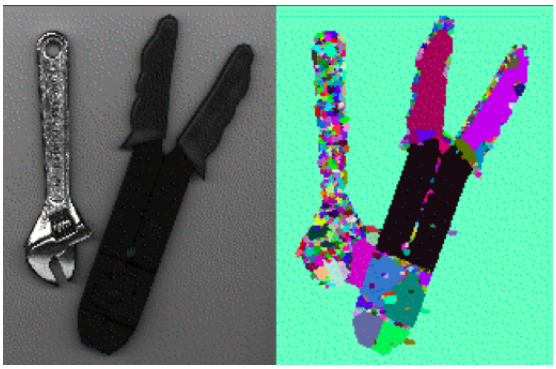

FIGURE 2: Segmentation d'objets

L'homme sait naturellement séparer des objets dans une image, grâce à la compréhension des objets et de la scène, mais la construction d'algorithme de segmentation de haut niveau représente un grand domaine de recherche dans le traitement d'image. Il existe 3 grandes classes de segmentation aujourd'hui :

— la segmentation fondée sur les régions,
— la segmentation fondée sur les contours,
— la segmentation fondée sur la classification/seuillage de pixels.

Dans ce projet, nous utilisons la méthode des contours avec le modèle de Mumford-Shah. Ce modèle a été implémenté par le groupe grâce à la plateforme GitHub. Le code source est disponible à cette adresse : https://github.com/MehdiZouitine/Segmentation_Mumford_Shah.

2 Modèle théorique de Mumford-Shah

Ce modèle est l'un des plus étudié. La fonctionnelle de Mumford-Shah a été introduite en 1989. D'un point de vue continue, une image est vue comme une fonction $g : \Omega \to \mathbb{R}$, où Ω est par exemple un rectangle, et g associe à chaque point $x \in \Omega$ de l'image une valeur $g(x)$ qui représente un niveau de gris. La fonction g n'est pas régulière, elle est très souvent discontinue. C'est cet effet de discontinuité qui nous intéresse. En effet, nous voulons trouver pour une image $g : \Omega \to \mathbb{R}$ l'ensemble des contours des objets que représente l'image. Ces contours sont dont localisés aux points de discontinuités de g : il y a une franche discontinuité dans les niveaux de gris.

Pour résoudre ce problème, Mumford et Shah introduisent une fonctionnelle qui par minimisation, va chercher les points de g les plus discontinus.
Le problème s'écrit alors :

$$\min_{(u,K)\in\mathcal{A}(\Omega)} \quad J(u,K) := \int_{\Omega\setminus K} |u-g|^2 dx + \int_{\Omega\setminus K} ||\nabla u||^2 dx + \mathcal{H}^1(K),$$

avec

$$\mathcal{A}(\Omega) = \{(u,K) : K \subset \Omega \quad \text{est fermé et} \quad u \in C^1(\Omega\setminus K)\}.$$

Nous pouvons expliciter chacun des termes qui composent cette fonctionnelle :

- $\int_{\Omega\setminus K} |u-g|^2 dx$ force u, par minimisation, à ressembler le plus possible à l'image de départ g.
- $\int_{\Omega\setminus K} ||\nabla u||^2 dx$ pénalise les oscillations de u, qui par minimisation, forcera u a être la plus régulière possible. Comme g est très irrégulière au début, on ne peut pas avoir $u = g$ partout dans Ω. Ainsi, l'ensemble K possède un rôle essentiel : s'il est bien positionné sur les singularités de g, il permettra de minimiser le premier terme décrit ci-dessus.
- $\mathcal{H}^1(K)$ désigne la mesure de Hausdorff de dimension 1 de l'ensemble K. Ce dernier terme force K à être de dimension de Hausdorff 1, qui nous donne bien l'intuition d'un "contour". Par exemple, un cercle est de dimension de Hausdorff 1. Cet ensemble K devra être optimisé pour couvrir le plus possible les singularités de g, en le positionnant en priorité sur les discontinuités les plus franches. Les autres singularités non prise en compte dans ce terme, donc minimes, seront considérées en oscillations. $\mathcal{H}^1(K)$ est la mesure de Lebesgue sur $\mathbb{R}$.

2.1 Mesure et dimension de Hausdorff : compléments

Les mesures de Hausdorff sont une généralisation des notions de longueur, d'aire, de volume, ... $\mathcal{H}^n$ est la mesure de Lebesgue en dimension n pour des sous ensembles de $\mathbb{R}^n$, multipliée par une constante qui n'est autre que le volume n-dimensionnel de la boule unité.

La mesure de Hausdorff peut être définie pour tout ensemble. Pour un sous-ensemble non-vide U d'un espace euclidien de dimension n, on peut définir le diamètre de U tel que :

$$diam\ \ U = |U| = sup\{|x - y| \ : \ x, y \in U\}$$

avec la distance euclidienne usuelle.

Ensuite, si un ensemble F est recouvert par une collection dénombrable $\{U_i\}$, de diamètre au plus δ :

$$F = \cup_{i=1}^{\infty} U_i \quad \text{avec} \quad 0 < |U_i| \leq \delta. \tag{1}$$

On dit que $\{U_i\}$ est un δ-recouvrement de F.

Soit maintenant un $s \in \mathbb{R} > 0$. Pour tout $\delta > 0$, on peut définir :

$$\mathcal{H}^s_\delta(F) = inf \left\{ \sum_{i=1}^{\infty} |U_i|^s \ : \ \{U_i\}\text{est un recouvrement de } F \right\}.$$

Ainsi,

$$\mathcal{H}^s(F) = \lim_{\delta -> 0} \mathcal{H}^s_\delta(F)$$

Et $\mathcal{H}^s(F)$ est appelé la mesure de Hausdorff s-dimensionnelle de F.

Théorème 2.1 - Dimension de Hausdorff (admis) : Soit F un sous-ensemble. Il existe un unique $d \in \mathbb{R}_+ \cup \infty$ tel que :

1. $\mathcal{H}^s(F) = \infty$ pour tout $s < d$
2. $\mathcal{H}^s(F) = 0$ pour tout $s > d$

On appelle dimension de Hausdorff de F le réel d. C'est la valeur critique de s pour laquelle la mesure passe de 0 à ∞. On l'appelle également dimension fractale.

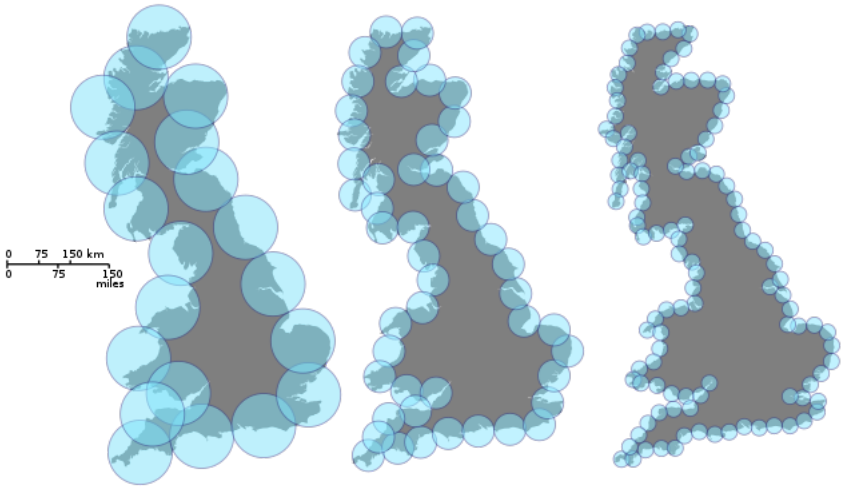

FIGURE 3: Dimension de Hausdorff de la Grande Bretagne

3 Modèle discret de Mumford-Shah

Lorsque l'on passe dans le modèle discret, la fonctionnelle introduite dans la partie précédente va être modifiée. On obtient donc une nouvelle fonctionnelle discrète qui est :

$$P(\Omega) + \lambda \sum_{\substack{m,n=1 \\ ((m,n),(m+1,n)) \notin \partial\Omega \\ ((m,n),(m,n+1)) \notin \partial\Omega}}^{N} |\nabla w_{m,n}|^2 + \mu ||w-u||_2^2$$

où $\lambda \geq 0$, $\mu \geq 0$ sont des paramètres et $u \in \mathbb{R}^{N^2}$ est l'image à segmenter.

On peut faire une analogie entre les termes de la nouvelle fonctionnelle discrète et la fonctionnelle continue. On a le premier terme de la somme qui est très proche d'un terme H^1, sauf que l'on retire de la somme les points dont le calcul du gradient fait intervenir un voisin de l'autre côté de la frontière de Ω. Ainsi, les points qui ont un très fort gradient tendent à appartenir à la frontière. Le dernier terme constitue l'attache aux données.

Pour résoudre notre problème de segmentation, nous allons donc vouloir minimiser cette fonctionnelle en $\Omega \subset \{1, ..., N\}^2$ et en $w \in \mathbb{R}^{N^2}$ en appliquant un algorithme de descente sur Ω et w.

3.1 Notion de périmètre

Comme nous l'avons vu, notre fonctionnelle contient un terme de périmètre, il est donc important de pouvoir le définir et de savoir le calculer. En géométrie, le périmètre d'un ensemble est la longueur de sa frontière. Nous allons donc appliquer cette définition sur un ensemble discret.

Pour cela, nous avons besoin de la notion de voisinage d'un pixel. Le voisinage d'un pixel, appelé $\sigma(m,n)$, est un ensemble de pixels considérés comme voisins du pixel en question.

Les voisinages les plus utilisés sont la 4-connexité, que nous allons utiliser dans la partie implémentation et définie comme ci dessous :

$$\sigma(m,n) = \{(m',n') \in \{1, ..., N\}^2, |m-m'| + |n-n'| = 1\}$$

et la 8-connexité définie comme ci-dessous :

$$\sigma(m,n) = \{(m',n') \in \{1, ..., N\}^2, max(|m-m'|, |n-n'|) = 1\}$$

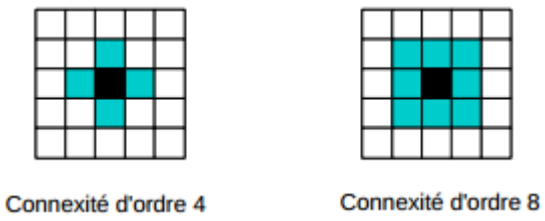

FIGURE 4: 4-connexité et 8-connexité

Nous allons donc pouvoir définir la frontière d'un ensemble comme les points qui ne sont pas dans l'ensemble mais qui sont le voisinage d'un point qui y appartienne.

On a alors la définition mathématique suivante :

$$\partial\Omega = \{((m,n),(m',n')) \in (\{1,...,N\}^2)^2, (m',n') \in \sigma(m,n) \; et \; (m,n) \in \Omega \; et \; (m',n') \notin \Omega\}$$

Pour avoir le périmètre, nous devons calculer la longueur de cette frontière. Pour cela, nous allons utiliser des éléments de longueur $dl(a,b)$ (où a,b sont des pixels) entre deux pixels. En sommant tous les éléments de longueur des points (a,b) appartenant à la frontière, nous obtenons le périmètre.
Le périmètre de Ω est donc définie par :

$$P(\Omega) = \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')),$$

où $dl((m,n),(m',n')) \geq 0$ sont des éléments de longueur. Plus la frontière de Ω contient d'éléments, plus la longueur du contour sera grande.

Dans la suite du projet, nous allons faire deux suppositions qui vont simplifier les calculs et les algorithmes qui sont : $\forall((m,n),(m',n')) \in (\{1,...,N\}^2)$

— $dl((m,n),(m',n')) = dl((m',n'),(m,n))$,
— et si $(m',n') \notin \sigma(m,n)$ alors $dl((m,n),(m',n')) = 0$.

3.2 Représentation de Ω

Pour optimiser Ω, on va procéder par des ensembles de niveau. Les méthodes basées sur les ensembles de niveaux représentent l'ensemble $\Omega \subset \{1,...N\}^2$ comme un ensemble de niveau d'une image $\phi \in \mathcal{R}^{N^2}$. On pose donc :

$$\Omega = \{(m,n) \in \{1,...,N\}^2, \phi_{m,n} \geq 0\}.$$

Ainsi, faire évoluer Ω revient à faire évoluer l'image ϕ pour minimiser une énergie analogue mais portant sur ϕ.

Pour appliquer cette démarche nous devons construire l'énergie qui va dépendre de ϕ. Nous voulons qu'elle soit sous la forme :

$$\sum_{m',n'=0}^{N} a_{m,n}\mathbb{I}_{|\phi_{m,n}\geq 0}$$

avec pour tout $(m,n) \in \{1,...,N\}^2$

$$\mathbb{I}_{|\phi_{m,n}\geq 0} = \begin{cases} 1 & \text{si } \phi_{m,n} \geq 0 \\ 0 & si \;\; \phi_{m,n} < 0 \end{cases}$$

On va utiliser la forme :

$$E((\phi_{m,n})_{1\leq m,n\leq N}) = \sum_{m,n=0}^{N}\sum_{m',n'=0}^{N} dl((m,n),(m',n'))\mathbb{I}_{|\phi_{m,n}\geq 0}(1-\mathbb{I}_{|\phi_{m,n}\geq 0})$$

Cette fonctionnelle correspond à l'indicatrice de la frontière multipliée par les éléments de longueurs dl, que l'on définira dans la partie 4 : Algorithme de segmentation et implémentation. Or cette écriture n'est pas différentiable, donc on peut remplacer $\mathbb{I}_{|\phi_{m,n}\geq 0}$ par $H_\epsilon(\phi_{m,n})$, avec

$$H_\epsilon(t) = \begin{cases} 1 & \text{si } t \geq \epsilon \\ \frac{1}{2}(1 + \frac{t}{\epsilon} + \frac{1}{\pi} sin(\frac{\pi t}{\epsilon})) & \text{si } -\epsilon \leq t \leq \epsilon \\ 0 & \text{si } t \leq -\epsilon \end{cases}$$

Cette fonction approxime la fonction de Heaviside. Le paramètre ϵ est un seuil de tolérance : plus ϵ est petit, plus on va être précis sur la frontière.

On a donc :

$$E((\phi_{m,n})_{1\leq m,n\leq N}) = \sum_{m,n=0}^{N} \sum_{m',n'=0}^{N} dl((m,n),(m',n'))H_\epsilon(\phi_{m,n})(1 - H_\epsilon(\phi_{m',n'}))$$

Cette fonctionnelle est maintenant différentiable avec

$$H'_\epsilon(t) = \begin{cases} 0 & \text{si } t \geq \epsilon \\ \frac{1}{2\epsilon} + \frac{1}{2\epsilon} cos(\frac{\pi t}{\epsilon}) & \text{si } -\epsilon \leq t \leq \epsilon \\ 0 & \text{si } t \leq -\epsilon \end{cases}$$

3.3 Évolution de Ω

On va maintenant pouvoir appliquer un algorithme de gradient à cette fonctionnelle qui nous permettra de minimiser en Ω par l'intermédiaire de ϕ.

Son gradient en $\phi = (\phi_{m,n})_{1\leq m,n\leq N}$ est :

$$\nabla E(\phi) = \left(\sum_{m',n'=0}^{N} dl((m,n),(m',n'))H'_\epsilon(\phi_{m,n})(1 - 2H_\epsilon(\phi_{m',n'})) \right)_{1\leq m,n\leq N}$$

Ainsi, on combine alors une étape de minimisation d'une énergie sur w, puis d'une étape d'optimisation de la forme Ω, en utilisant un algorithme de descente de gradient.

En pratique, on initialisera ϕ comme égale à l'image à segmenter.

4 Algorithme de segmentation et implémentation

4.1 Algorithme de segmentation

Pour segmenter notre image, nous devons minimiser la fonctionnelle de Mumford-Shah définie ci-dessus. Pour minimiser une fonction, la démarche naturelle est d'utiliser un algorithme de descente de gradient. L'énergie n'étant pas convexe, on se contentera de minimaux locaux.

Pour minimiser la fonctionnelle de Mumford-Shah, on alterne :

— une étape de minimisation d'une énergie sur le paramètre w de la fonctionnelle,

— une étape d'optimisation de la forme Ω.

Plus concrètement, on alternera dans l'algorithme de descente, une itération de minimisation de w, puis une itération pour minimiser Ω (i.e. minimiser ϕ).

4.1.1 Minimisation en w

La minimisation des termes en w de l'énergie est très classique. Il nous suffit de calculer les gradients de chacun des termes et ensuite d'effectuer une descente de gradient classique.

Le terme :

$$\|w-u\|_2^2$$

a pour gradient :

$$\nabla\|w-u\|_2^2 = 2(w-u)$$

Le terme :

$$\sum_{\substack{m,n=1 \\ ((m,n),(m+1,n))\notin\partial\Omega \\ ((m,n),(m,n+1))\notin\partial\Omega}}^{N} |\nabla w_{m,n}|^2$$

a pour gradient :

$$\|C\partial_x w\|^2 + \|C\partial_y w\|^2 = C * \Bigg(-2*(w_{i+1,j} + w_{i,j+1} - 2w_{i,j}) \Bigg)_{1\leq i,j\leq N}$$

où C est un opérateur qui multiplie les pixels de $\partial\Omega$ par 0 et les autres par 1.

4.1.2 Minimisation en Ω (i.e. en ϕ)

Pour la minimisation du terme en ϕ de l'énergie, il nous suffit de calculer le gradient du terme de périmètre et de modifier les dl pour tenir compte de la contribution de $\partial_x w$ et $\partial_y w$ sur la frontière de Ω.
On rappelle que le terme

$$E\left((\phi_{m,n})_{1\leq m,n\leq N}\right) = \sum_{m,n=0}^{N}\sum_{m',n'=0}^{N} dl\left((m,n),(m',n')\right) H_\epsilon\left(\phi_{m,n}\right)\left(1-H_\epsilon\left(\phi_{m',n'}\right)\right)$$

a pour gradient

$$\nabla E(\phi) = \left(\sum_{m',n'=0}^{N} dl\left((m,n),(m',n')\right) H'_{\epsilon}\left(\phi_{m,n}\right)\left(1 - 2H_{\epsilon}\left(\phi_{m',n'}\right)\right) \right)_{1 \leq m,n \leq N}$$

avec

$$dl((m,n),(m',n') = \begin{cases} 1 - |\nabla w_{m,n}|^2 & \text{si } \Big((m,n),(m',n')\Big) \in \partial\Omega \\ 0 & \text{sinon.} \end{cases}$$

4.2 Implémentation

Pour l'implémentation, nous avons développé en Python l'algorithme de descente de gradient suivant :

Algorithm 1 Descente de gradient pour minimiser la fonctionnelle de Mumford-Shah

Input : w_0, $iterations$, η_0
Output : $w\_débruitée$, phi_finale

$w \leftarrow w_0$
$\phi \leftarrow Distribution\_Uniforme(intervalle = [-K, K], taille = N \times N)$ or w_0
$\Omega \leftarrow \{x \in N \times N,\ \phi(x) > 0\}$
$\partial\Omega \leftarrow \{(i,j) \in \Omega \mid \exists \Phi \in \mathcal{V}^{\phi}_{i,j} \notin \Omega\}$
for $idx = 1, 2, \ldots, iterations$ **do**
    $w \leftarrow w - \eta_0 * \nabla w_{part}(w, \partial\Omega)$
    $\phi \leftarrow \phi - \eta_0 * \nabla \phi_{part}(\phi, \partial\Omega)$
    $\Omega \leftarrow \{\phi(x) > 0, x \in N \times N\}$
    $\partial\Omega \leftarrow \{(i,j) \in \Omega \mid \exists \Phi \in \mathcal{V}^{\phi}_{i,j} \notin \Omega\}$
end for
return (w, ϕ)

— w_0 correspond à l'image initiale.

— $\eta_0 > 0$ correspond au pas de la descente en w et $\mu_0 > 0$ au pas de la descente en ϕ.

— $N \times N$ correspond à la dimension de l'image.

— K un réel quelconque positif, typiquement 1.

— $\mathcal{V}^{\phi}_{i,j}$ correspond au voisinage du pixel (i,j) de ϕ pour le système de voisinage choisi.

— $\nabla w_{part}(w, \partial\Omega)$ correspond au gradient de la somme des deux termes en "w" de la fonctionnelle i.e. :

$$\nabla\Big(\sum_{\substack{m,n=1 \\ ((m,n),(m+1,n)) \notin \partial\Omega \\ ((m,n),(m,n+1)) \notin \partial\Omega}}^{N} |\nabla w_{m,n}|^2 + \mu \|w - u\|_2^2 \Big)$$

— $\nabla\phi_{part}(\phi, \partial\Omega)$ correspond au gradient du terme de périmètre dans la fonctionnelle i.e. :

$$\nabla(P(\Omega))$$

qui devient par la suite via la méthode d'ensemble de niveau :

$$\left(\sum_{m',n'=0}^{N} dl\left((m,n),(m',n')\right) H'_{\varepsilon}\left(\varphi_{m,n}\right)\left(1 - 2H_{\varepsilon}\left(\varphi_{m',n'}\right)\right) \right)_{1\leq m,n\leq N}$$

Pour implémenter numériquement l'algorithme, nous avons fait cela en Python.
Pour cela, nous avons divisé notre code en deux fichiers :

— Un fichier *function.py* qui contient toutes les fonctions nécessaires à la segmentation.

— Un fichier *segmenter.py* qui contient un objet *segmenter* instanciable, qui prend une image en entrée et qui retourne l'image segmentée et des statistiques sur la minimisation.

— Un notebook *demonstration.ipynb* qui présente la ségmentation et utilise le segmenteur pour les paramètres de notre choix.

Voici un aperçu de toutes les fonctions utiles à la segmentation de notre image (nom de la fonction, nom et type des paramètres et type des sorties). Le code entier avec les explications du code est disponible sur *github*.

```
def in_shape(img: np.ndarray, pixel: List)

sign = lambda x: math.copysign(1, x)

def dl(pixel1: np.ndarray, pixel2: np.ndarray, frontier: List[List]) ->
    int

def P(frontier: List[List])

def H1(w: np.ndarray, frontier: List[List]) -> float

def norm(w: np.ndarray, u: np.ndarray) -> float

def munford_shah(w: np.ndarray, u: np.ndarray, frontier: List[List]) ->
    float

def dl2(
    pixel1: np.ndarray, pixel2: np.ndarray, frontier: List[tuple], w: np
    .ndarray) -> float

def H_eps(t: float, eps: float) -> float

def H_eps_derivative(t: float, eps: float) -> float

def in_frontier(pixel: List, phi: np.ndarray) -> bool

def grad_w_part(
    w: np.ndarray, u: np.ndarray, lambda_: float, mu: float, phi: np.
    ndarray) -> np.ndarray

```

```
def get_neighbour(pixel: List, img: np.ndarray)

def grad_phi_part(
    phi: np.ndarray, w: np.ndarray, omega_frontier: List[List], eps:
    float)

def get_frontier_phi(omega: List[List], phi: np.ndarray) -> List[List]

def gradient_descent(u: np.ndarray,step_w: float,step_phi: float,eps:
    float,lambda_: float,mu: float,it: int,verbose: bool,mode: str,test:
    np.ndarray,
) -> Dict[str, Union[np.ndarray, List]]
```

L'algorithme de descente comporte plusieurs arguments, qui vont influer sur le rendu de la segmentation :

- u : l'image de base (nécessaire au calcul du terme $\|w-u\|_2^2$).
- $step_w$: le pas de la descente de gradient pour l'itération en w.
- $step_\phi$: le pas de la descente de gradient pour l'itération en ϕ.
- ϵ : la valeur de ϵ pour la fonction H_ϵ. Plus ϵ est grand, plus la segmentation sera "grossière".
- λ : le coefficient λ devant le terme : $\sum_{\substack{m,n=1 \\ ((m,n),(m+1,n))\notin\partial\Omega \\ ((m,n),(m,n+1))\notin\partial\Omega}}^{N} |\nabla w_{m,n}|^2$
- μ : le coefficient μ devant le terme : $\|w-u\|_2^2$.
- *it* : le nombre d'itérations dans l'algorithme de descente.
- *verbose* : un booléen qui indique si on veut des affichages ou non.
- *mode* : le type d'algorithme de descente que on utilise (dans ce projet nous utiliserons seleument un algorithme de descente à pas fixe).

4.3 Expérimentation

La segmentation marche extrêmement bien pour les images simples. En effet, la nuance forte et la discontinuité forte de ces images donnent de très belles segmentations.

L'image de l'arbre est nettement plus complexe : elle comporte plusieurs éléments (nuages, verdure, trous). L'algorithme donne un résultat satisfaisant mais perfectible. Il faudrait sûrement faire un pré-processing de l'image avant de la segmenter.

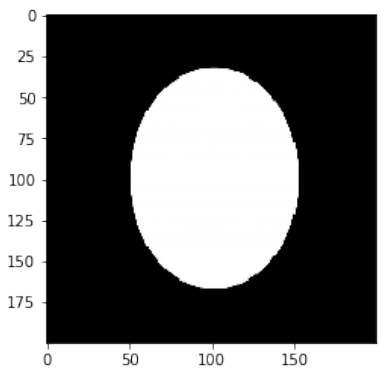

FIGURE 5: Image de base

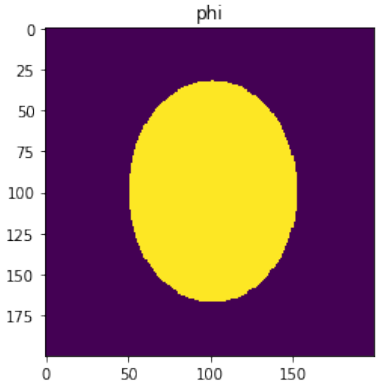

FIGURE 6: Omega

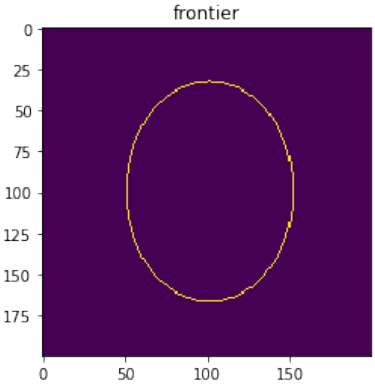

FIGURE 7: Frontière

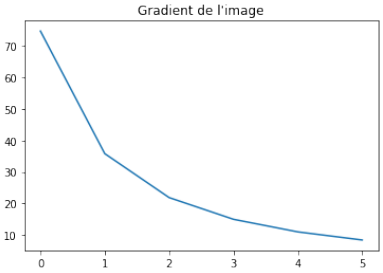

FIGURE 8: Gradient de w

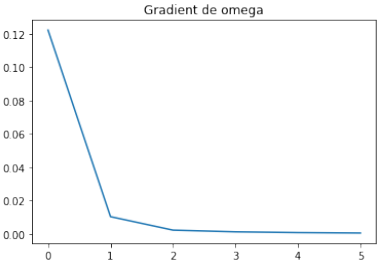

FIGURE 9: Gradient de ϕ

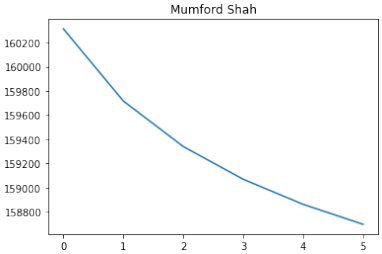

FIGURE 10: Fonctionnelle

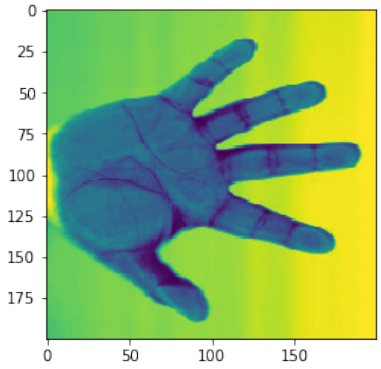

FIGURE 11: Image de base

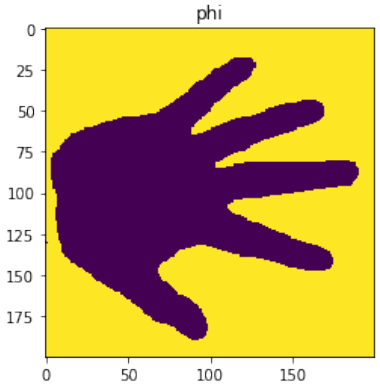

FIGURE 12: Omega

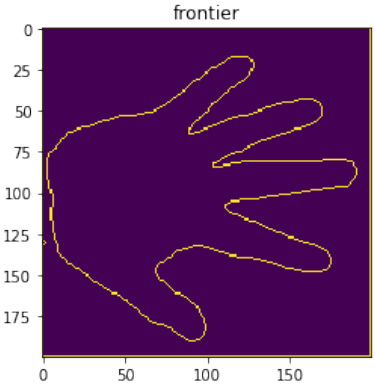

FIGURE 13: Frontière

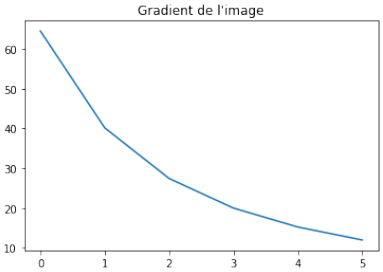

FIGURE 14: Gradient de w

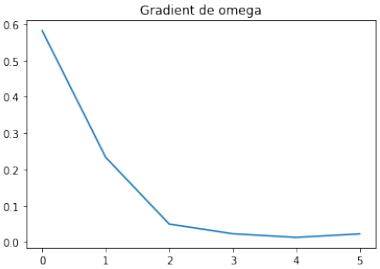

FIGURE 15: Gradient de ϕ

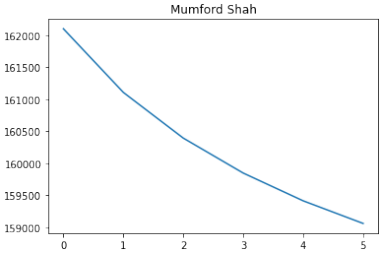

FIGURE 16: Fonctionnelle

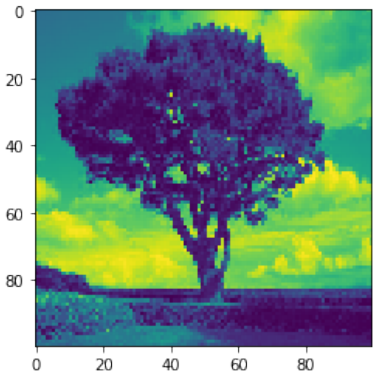

FIGURE 17: Image de base

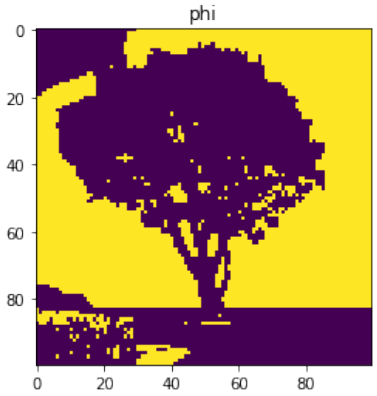

FIGURE 18: Omega

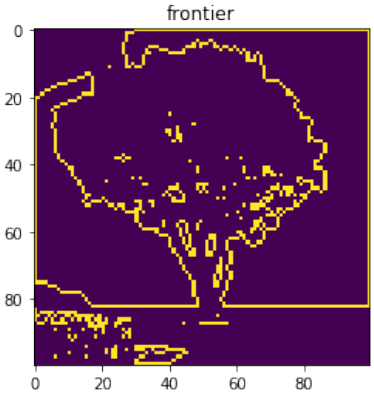

FIGURE 19: Frontière

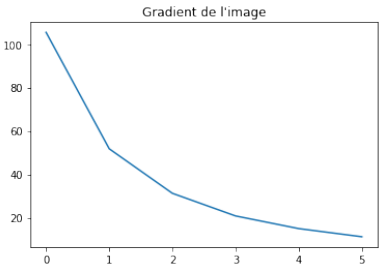

FIGURE 20: Gradient de w

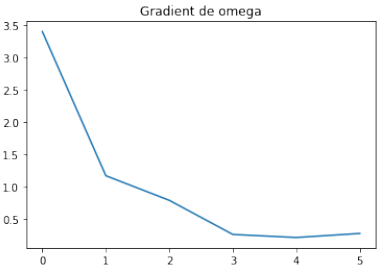

FIGURE 21: Gradient de ϕ

FIGURE 22: Fonctionnelle

5 Conclusion

Ce projet nous a permis de nous intéresser à un domaine très utile dans le traitement d'images : la segmentation. C'est un sujet très actuel et utilisé, et indispensable dans certains métiers.

Nous nous sommes heurtés à certaines difficultés lors de notre implémentation. En effet, le calcul du gradient de w n'était pas le bon au début, ce qui nous a retardé. Mais nous avons pu résoudre ce problème par la suite.

De plus, ce projet nous a surtout aidé à développer le côté coopératif d'un travail d'équipe, et cela principalement grâce à la plateforme GitHub pour l'écriture du code, et la plateforme Overleaf pour l'écriture du rapport.

Ainsi, notre projet a donc été enrichissant, et nous a permis de découvrir et d'approfondir l'étude de la segmentation d'images.

Toutefois, nous aurions pu améliorer notre algorithme de segmentation pour des images plus complexes, comme évoqué dans la partie 4.3 pour l'image de l'arbre. Avec un peu plus de temps, il aurait été envisageable d'analyser nos algorithmes dans un but d'optimisation, par exemple en comparant notre algorithme de gradient avec plusieurs stratégies de pas, comme le pas d'Armijo ou en calculant le pas exact pour avoir un pas de plus grande descente. Une autre manière d'optimiser aurait été d'étudier la parallélisation de notre code.

Aujourd'hui, il existe des méthodes plus modernes de traitement d'images et notamment de segmentation, grâce au technique de deep learning. En deep learning, les réseaux de neurones à convolution sont très efficaces pour traiter et résoudre les problèmes de segmentation.

Table des figures

Références

[1] Tony F. Chan and Luminita A. Vese. A level set algorithm for minimizing the Mumford-Shah functional in image processing. 2002.

[2] Laurent D. Cohen. Segmentation d'énergie minimale. 1999.

[3] Thibaut Deheuvels. Mesure et dimension de Hausdorff.

[4] Olivier Elchinger. Les dimensions fractales. 2006.

[5] Antoine Lemenant. Initiation à la fonctionnelle de Mumford-Shah. 2012.

[6] Wikipedia. Dimension de Hausdorff.