Segmentation d'image – Mumford–Shah

Zahraa Asfour Maxime Chourré Esteban Marco
Université Paul Sabatier UT3

15 avril 2022

Résumé

Ce rapport consiste en la segmentation d'une image selon la méthode de Mumford-Shah.

Mots Clef

Segmentation, Mumford-Shah, Image

Table des matières

1 Introduction

En cours (chapitre 6.1 et 7), nous avons décrit des méthodes permettant de segmenter des images. Le projet consiste à implémenter la méthode de Mumford-Shah à l'aide de la méthode d'optimisation de forme par ensembles de niveaux. Il faudra aussi étudier plusieurs méthodes d'initialisation de l'algorithme, illustrer ses propriétés de convergence et illustrer ses résultats sur plusieurs images tests.

2 Nos données sur le modèle

Le modèle de Mumford-Shah consiste à minimiser la fonction F en $\Omega \subset \{1, ..., N\}^2$. Pour faire évoluer Ω, il suffit de trouver le minimiseur φ^* de G. φ^* nous permettra ainsi de redéfinir Ω. On a :

$$\begin{aligned} F : \mathbb{R}^{N^2} &\to \mathbb{R} \\ w &\mapsto F(w) \end{aligned}$$

tel que :

$$\begin{aligned} F(w) &= P(\Omega) + \lambda \sum_{m,n=1}^{N} |\nabla w_{m,n}|^2 + \mu \|w - u\|^2 \\ &= P(\Omega) + \lambda \left(\|\nabla_x w\|^2 + \|\nabla_y w\|^2 \right) + \mu \|w - u\|^2 \end{aligned}$$

où λ, $\mu \geq 0$ et

$$P(\Omega) = \sum_{((m,n),(m',n')) \in \partial\Omega} dl((m,n),(m',n')) - \lambda |\nabla w_{m,n}|^2$$

avec

$$dl((m,n),(m',n')) = \frac{\exp\left(-\frac{(u_{m,n} - u_{m',n'})^2}{2sigma^2}\right)}{\sqrt{(m-m')^2 + (n-n')^2}}$$

On définit la fonction G comme suit :

$$\begin{aligned} G : \mathbb{R}^{N^2} &\to \mathbb{R} \\ \varphi &\mapsto G(\varphi) \end{aligned}$$

avec

$$\begin{aligned} G(\varphi) &= \sum_{((m,n),(m',n')) \in \partial\Omega} dl((m,n),(m',n')) - \lambda |\nabla w_{m,n}|^2 \\ &= \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} \left(dl((m,n),(m',n')) - \lambda |\nabla w_{m,n}|^2 \right) \mathbb{1}_{\{((m,n),(m',n')) \in \partial\Omega\}} \\ &= \sum_{\substack{m,n=1 \\ (m,n)\in\Omega}}^{N} \sum_{\substack{m',n'=1 \\ (m',n')\notin\Omega}}^{N} \left(dl((m,n),(m',n')) - \lambda |\nabla w_{m,n}|^2 \right) \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}} \\ &= \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}} \left(dl((m,n),(m',n')) - \lambda |\nabla w_{m,n}|^2 \right) \mathbb{1}_{\{\varphi_{m,n} > 0\}} (1 - \mathbb{1}_{\{\varphi_{m',n'} > 0\}}) \end{aligned}$$

Définissons $\mathcal{H}_\varepsilon(\varphi_{m,n})$ qui remplacera $\mathbb{1}_{\{\varphi_{m,n}>0\}}$.

$$\begin{aligned} &\mathcal{H}_\varepsilon : \mathbb{R} \to \mathbb{R} \\ &\qquad \varphi \mapsto \mathcal{H}_\varepsilon(t) = \begin{cases} 1 & \text{si } t \geq \varepsilon \\ \frac{1}{2}(1 + \frac{t}{\varepsilon} + \frac{1}{\pi}\sin(\frac{\pi t}{\varepsilon})) & \text{si } |t| \leq \varepsilon \\ 0 & \text{si } t \leq -\varepsilon \end{cases} \end{aligned}$$

Sa dérivée ${\mathcal{H}'}_\varepsilon(t)$ est donnée par :

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

alors $\forall \in \mathcal{R}^{N^2}$

$$G(\varphi) = \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} \mathbb{1}_{\{(m',n')\in\sigma(m,n)\}}(dl((m,n),(m',n')) - \lambda|\nabla w_{m,n}|^2)\mathcal{H}_\varepsilon(\varphi_{m,n} > 0)(1 - \mathcal{H}_\varepsilon(\varphi_{m',n'} > 0))$$

DEFINITIONS :

— $\Omega = \{(m,n) \in 1,...,N^2; \varphi_{m,n} \geq 0\}$ où $(\varphi_{m,n})_{1\leq m,n\leq N} \in R^{N^2}$ est une image ;

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

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

REMARQUES :

— $(u_{m,n})_{m,n} \in 1,...,N^2$ est l'image qu'on veut segmenter ;

— $\forall (m,n) \in 1,...,N^2, \sigma(m,n)$ contient 4 éléments, ou encore (m,n) a au plus 4 voisins. (En référence à la définition ci-dessus) ;

— $\forall m,n \in \{1,...,N\}, \nabla w_{m,n} = \begin{pmatrix} \nabla_x w_{m,n} \\ \nabla_y w_{m,n} \end{pmatrix} = \begin{pmatrix} w_{m+1,n} - w_{m,n} \\ w_{m,n+1} - w_{m,n} \end{pmatrix}$

3 Étude de $F(w)$

3.1 Continuité

On remarque que F est composée de fonctions continues. En effet, lorsqu'on développe F de façon plus explicite, on peut remarquer que F est un polynôme de degré 2. Donc on peut dire que F est continue et de classe C^∞. Elle est alors différentiable.

3.2 Calcul du gradient de $F(w)$

Écrivons F sous la forme $F = f + E$ pour faciliter le calcul du gradient.

On définit E comme suit :

$$\begin{aligned} E : \mathbb{R}^{N^2} &\to \mathbb{R} \\ w &\mapsto E(w) = \lambda(\|\nabla_x w\|^2 + \|\nabla_y w\|^2) + \mu\|w-u\|^2 \end{aligned}$$

Calculons le gradient de E
$\forall \omega, h \in \mathcal{R}^{N^2}$

$$\begin{aligned} E(\omega+h) &= \lambda(\|\nabla_x w + \nabla_x h\|^2 + \|\nabla_y w + \nabla_y h\|^2) + \mu\|w+h-u\|^2 \\ &= \lambda(\|\nabla_x w\|^2 + 2<\nabla_x w, \nabla_x h> + \|\nabla_x h\|^2 + \|\nabla_y w\|^2 + 2<\nabla_y w, \nabla_y h> + \|\nabla_y h\|^2)) \\ &+ \mu(\|w-u\|^2 + 2<w-u, h> + \|h\|^2) \\ &= E(w) + <2\lambda(\nabla_x^*\nabla_x w + \nabla_y^*\nabla_y w) + 2\mu(w-u), h> + \|h\|\epsilon(\|h\|) \\ &= E(w) + <\nabla E(w), h> + \|h\|\epsilon(\|h\|) \end{aligned}$$

Où $\|h\|\epsilon(\|h\|) = \lambda(\|\nabla_x h\|^2 + \|\nabla_y h\|^2) + \mu\|h\|^2$ avec $\epsilon(\|h\|) \longrightarrow 0$ lorsque $\|h\| \longrightarrow 0$

Alors $\forall \omega \in \mathcal{R}^{N^2}$

$$\boxed{\nabla E(w) = 2\lambda(\nabla_x^*\nabla_x w + \nabla_y^*\nabla_y w) + 2\mu(w-u).}$$

On définit f comme suit :

$$\begin{aligned} f : \mathbb{R}^{N^2} &\to \mathbb{R} \\ w &\mapsto f(w) = \sum_{m,n=1}^{N}\sum_{m',n'=1}^{N}(dl((m,n),(m',n')) - \lambda|\nabla w_{m,n}|^2)\mathbb{1}_{\{((m,n),(m',n'))\in\partial\Omega\}} \end{aligned}$$

Définissant $\forall m', n' \in \{1, ..., N\}$ fixé un opérateur $H^{(m',n')}$ définit par :

$$\begin{aligned} H^{(m',n')} : \mathbb{R}^{N^2} &\to \mathbb{R}^{N^2} \\ w &\mapsto H^{(m',n')}(w) \end{aligned}$$

$\forall m, n \in \{1, ..., N\}$

$$(H^{(m',n')}(w))_{m,n} = \begin{cases} w_{m,n} & \text{si } ((m,n),(m',n')) \in \partial\Omega \\ 0 & \text{sinon} \end{cases}$$

L'opérateur définit ci-dessus est auto-adjoint et idempotent. Donc $\forall m', n' \in \{1, ..., N\}$ $H^{(m',n')} = {H^{(m',n')}}^*$

Notre fonction f s'écrit alors $\forall \omega \in \mathcal{R}^{N^2}$

$$f(w) = \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) - \lambda\sum_{m,n=1}^{N}\sum_{m',n'=1}^{N}(|(H^{(m',n')}(\nabla_x w))_{m,n}|^2 + |(H^{(m',n')}(\nabla_y w))_{m,n}|^2)$$

Calculons le gradient de f
$\forall \omega, h \in \mathcal{R}^{N^2}$

$$
\begin{aligned}
f(w+h) = & \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) \\
& - \lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} (|(H^{(m',n')}(\nabla_x w + \nabla_x h))_{m,n}|^2 + |(H^{(m',n')}(\nabla_y w + \nabla_y h))_{m,n}|^2) \\
& = \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) - \lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} (|(H^{(m',n')}(\nabla_x w))_{m,n} + (H^{(m',n')}(\nabla_x h))_{m,n}|^2 \\
& + |(H^{(m',n')}(\nabla_x w))_{m,n} + (H^{(m',n')}(\nabla_x h))_{m,n}|^2) \\
& = f(w) - 2\lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} (H^{(m',n')}(\nabla_x w))_{m,n} (H^{(m',n')}(\nabla_x h))_{m,n} \\
& - 2\lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} (H^{(m',n')}(\nabla_y w))_{m,n} (H^{(m',n')}(\nabla_y h))_{m,n} + \|h\|\epsilon(\|h\|)
\end{aligned}
$$

Remarque : $(H^{(m',n')}(\nabla_y w))_{m,n}(H^{(m',n')}(\nabla_y h))_{m,n} = (H^{(m',n')}(\nabla_y w))_{m,n}(\nabla_y h)_{m,n}$.
En effet si $(H^{(m',n')}(\nabla_y w))_{m,n} = 0 \iff (H^{(m',n')}(\nabla_y h))_{m,n} = 0$. Donc il suffit d'appliquer le $H^{(m',n')}$ sur w.

Alors $\forall \omega, h \in \mathbb{R}^{N^2}$

$$
\begin{aligned}
f(w+h) = f(w) & - 2\lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} (H^{(m',n')}(\nabla_x w))_{m,n} (\nabla_x h)_{m,n} \\
& - 2\lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} (H^{(m',n')}(\nabla_y w)_{m,n} (\nabla_y h)_{m,n} + \|h\|\epsilon(\|h\|) \\
& = f(w) - 2\lambda \sum_{m',n'=1}^{N} (< H^{(m',n')}(\nabla_x w), \nabla_x h > + < H^{(m',n')}(\nabla_y w), \nabla_y h >) + \|h\|\epsilon(\|h\|) \\
& = f(w) - 2\lambda \sum_{m',n'=1}^{N} (< \nabla_x^*(H^{(m',n')}(\nabla_x w)), h > + < \nabla_y^*(H^{(m',n')}(\nabla_y w)), h >) + \|h\|\epsilon(\|h\|) \\
& = f(w) + < -2\lambda \sum_{m',n'=1}^{N} (\nabla_x^*(H^{(m',n')}(\nabla_x w)) + \nabla_y^*(H^{(m',n')}(\nabla_y w))), h > + \|h\|\epsilon(\|h\|) \\
& = f(w) + < \nabla f(w), h > + \|h\|\epsilon(\|h\|)
\end{aligned}
$$

Alors $\forall w \in \mathbb{R}^{N^2}$

$$\boxed{\nabla f(w) = -2\lambda \sum_{m',n'=1}^{N} (\nabla_x^*(H^{(m',n')}(\nabla_x w)) + \nabla_y^*(H^{(m',n')}(\nabla_y w))).}$$

Conclusion :
Le gradient de F est alors $\forall w \in \mathbb{R}^{N^2}$

$$\nabla F(w) = 2\lambda(\nabla_x^*\nabla_x w + \nabla_y^*\nabla_y w) + 2\mu(w-u) - 2\lambda \sum_{m',n'=1}^{N} (\nabla_x^*(H^{(m',n')}(\nabla_x w)) + \nabla_y^*(H^{(m',n')}(\nabla_y w))).$$

3.3 Coercitivé

Pour montrer la coercivité de F, il suffit de minorer F par une fonction qu'on sait coercive par exemple la norme.

$$\begin{aligned}
\forall \omega \in \mathcal{R}^{N^2}, F(\omega) &= \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) - \lambda|\nabla w_{m,n}|^2 + \lambda \sum_{m,n=1}^{N} |\nabla w_{m,n}|^2 + \mu\|w-u\|^2 \\
&\geq -\lambda \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} |H^{(m',n')}(\nabla w_{m,n})|^2 + \lambda \sum_{m,n=1}^{N} |\nabla w_{m,n}|^2 + \mu\|w-u\|^2 \\
&= \lambda \sum_{m,n=1}^{N} (|\nabla w_{m,n}|^2 - \sum_{m',n'=1}^{N} |H^{(m',n')}(\nabla w_{m,n})|^2)) + \mu\|w-u\|^2 \\
&\geq -3\lambda \sum_{m,n=1}^{N} |\nabla w_{m,n}|^2 + \mu\|w-u\|^2
\end{aligned}$$

car on a :
$\sum_{m',n'=1}^{N}(|H^{(m',n')}(\nabla w_{m,n})|^2) \leq 4|\nabla w_{m,n}|^2$ (m,n) à au plus 4 voisins.

$$\begin{aligned}
\|w-u\|^2 &\geq (\|w\| - \|u\|)^2 \\
&= \|w\|^2 + \|u\|^2 - 2\|w\|\|u\| \\
&\geq \frac{\|w\|^2}{2} + a
\end{aligned}$$

En résolvant l'inéquation on trouve a $\leq -\|u\|^2$

Et on a : $\|\nabla_x w\|^2 + \|\nabla_y w\|^2 \leq 4\|w\|^2$ car l'image est symétrique
Alors

$$\begin{aligned}
\forall \omega \in \mathcal{R}^{N^2}, F(\omega) &\geq -3\lambda(\|\nabla_x w\|^2 + \|\nabla_y w\|^2) + \mu\|w-u\|^2 \\
&\geq -3\lambda(4\|w\|^2) + \mu(\frac{\|w\|^2}{2} + a) \\
&= \|w\|^2(-12\lambda + \frac{\mu}{2}) + \mu a
\end{aligned}$$

Alors F coercive à condition que $\mu - 24\lambda \geq 0$

$$F\text{coercive} \iff \mu \geq 24\lambda.$$

Conclusion :
Comme F est continue et coercive (avec $\mu \geq 24\lambda$), alors elle admet un minimum.

3.4 Constance de Lipschitz

Cherchons la constante de Lipschitz qui nous donnera le pas le mieux adapté pour l'algorithme de la descente de gradient à pas constant.

On a $\forall w \in \mathcal{R}^{N^2}$
$\nabla F(w) = 2\lambda(\nabla_x^*\nabla_x w + \nabla_y^*\nabla_y w) + 2\mu(w-u) - 2\lambda\sum_{m',n'=1}^{N}(\nabla_x^*(H^{(m',n')}(\nabla_x w)) + \nabla_y^*(H^{(m',n')}(\nabla_y w)))$

$\forall w, v \in \mathcal{R}^{N^2}$

$$\begin{aligned}
\|\nabla F(w) - \nabla F(v)\|^2 &= \|2\lambda(\nabla_x^*\nabla_x(w-v) + \nabla_y^*\nabla_y(w-v)) + 2\mu(w-v) \\
&\quad - 2\lambda \sum_{m',n'=1}^{N} (\nabla_x^*(H^{(m',n')}(\nabla_x(w-v))) + \nabla_y^*(H^{(m',n')}(\nabla_y(w-v))))\|^2 \\
&\leq 4\lambda^2(\|\nabla_x^*\nabla_x(w-v)\|^2 + \|\nabla_y^*\nabla_y(w-v)\|^2) + 4\mu^2\|(w-v)\|^2 \\
&\quad + 4\lambda^2 \sum_{m',n'=1}^{N} (\|\nabla_x^*(H^{(m',n')}(\nabla_x(w-v)))\|^2 + \|\nabla_y^*(H^{(m',n')}(\nabla_y(w-v)))\|^2) \\
&\leq 4\lambda^2(\|\nabla_x^*\nabla_x\|^2\|w-v\|^2 + \|\nabla_y^*\nabla_y\|^2\|w-v\|^2) + 4\mu^2\|(w-v)\|^2 \\
&\quad + 4\lambda^2 \sum_{m',n'=1}^{N} (\|\nabla_x^*\|^2\|H^{(m',n')}\|^2\|\nabla_x\|^2\|w-v\|^2 + \|\nabla_y^*\|^2\|H^{(m',n')}\|^2\|\nabla_y\|^2\|w-v\|^2) \\
&= 4\|w-v\|^2(\mu^2 + \lambda^2(\|\nabla_x^*\nabla_x\|^2 + \|\nabla_y^*\nabla_y\|^2) \\
&\quad + \lambda^2 \sum_{m',n'=1}^{N} \|H^{(m',n')}\|^2(\|\nabla_x^*\|^2\|\nabla_x\|^2 + \|\nabla_y^*\|^2\|\nabla_y\|^2))
\end{aligned}$$

Majorant $\|H^{(m',n')}\|^2$

$$\begin{aligned}
&\forall w \in (\mathcal{R}^*)^{N^2}, \forall m', n' \in \{1, ..., N\} \\
&< (H^{(m',n')}(w)), w > \leq \|w\|^2
\end{aligned}$$

Alors $\forall m', n' \in \{1, ..., N\}\ \|H^{(m',n')}\|^2 \leq 1$

Majorant $\|\nabla_x\|^2$ qui revient à majorer $\|\nabla_x^*\|^2, \|\nabla_y\|^2 et \|\nabla_y^*\|^2$ car ils ont la même norme.

$$\begin{aligned}
&\forall w \in (\mathcal{R}^*)^{N^2} \\
&< (\nabla_x(w)), w > = \sum_{m,n=1}^{N} (w_{m+1,n} - w_{m,n})(w_{m,n}) \\
&\leq \sum_{m,n=1}^{N} (w_{m+1,n})(w_{m,n}) \\
&\leq |\sum_{m,n=1}^{N} (w_{m+1,n})(w_{m,n})| \\
C.S + periodicite &\leq \|w\|\|w\| = \|w\|^2
\end{aligned}$$

Alors $\|\nabla_x\|^2 \leq 1$

Majorant $\|\nabla_x^*\nabla_x\|^2 ou \|\nabla_y^*\nabla_y\|^2$ car ils ont la même norme.

$$\begin{aligned}
&\forall w \in (\mathcal{R}^*)^{N^2} \\
&< (\nabla_x^*\nabla_x(w)), w > = \|\nabla_x(w)\|^2 \\
&Image\ periodique \leq 2\|w\|^2
\end{aligned}$$

Alors $\|\nabla_x^*\nabla_x\|^2 \leq 2$

Revenant a notre inégalité : $\forall w, v \in \mathcal{R}^{N^2}$

$$\begin{aligned}
\|\nabla F(w) - \nabla F(v)\|^2 &\leq 4\|w-v\|^2(\mu^2 + \lambda^2(\|\nabla_x^*\nabla_x\|^2 + \|\nabla_y^*\nabla_y\|^2) \\
&\quad + \lambda^2 \sum_{m',n'=1}^{N} \|H^{(m',n')}\|^2(\|\nabla_x^*\|^2\|\nabla_x\|^2 + \|\nabla_y^*\|^2\|\nabla_y\|^2)) \\
&\leq 4\|w-v\|^2(\mu^2 + \lambda^2(2+2) + \lambda^2 \sum_{m',n'=1}^{N} 1(1+1)) \\
&\leq 4\|w-v\|^2(\mu^2 + 2\lambda^2(N^2+2)) \\
&\leq L^2\|w-v\|^2
\end{aligned}$$

Alors la constante de Lipschitz est $L = 2\sqrt{(\mu^2 + 2\lambda^2(N^2+2)}$.

Conclusion :
F est de gradient L-Lipschitz

3.5 Convexité

La connexité n'est pas nécessaire pour vérifier qu'une solution existe. Mais si F est strictement convexe alors on pourra dire que la solution est unique.

F convexe si $\forall w, v \in \mathbb{R}^{N^2}, \forall \alpha \in (0,1)\ \alpha F(w) + (1+\alpha)F(v) - F(\alpha w + (1-\alpha)v) \geq 0$.
Soient $w, v \in \mathbb{R}^{N^2}$. Soit $\alpha \in [0,1]$.

$$\begin{aligned}
&\alpha\, F(w) + (1-\alpha)F(v) - F(\alpha w + (1-\alpha)v) \\
&= \alpha \left[\sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) - \lambda\,|\nabla w_{m,n}|^2 + \lambda\left(\|\nabla_x w_{m,n}\|^2 + \|\nabla_y w_{m,n}\|^2\right) + \mu\|w-u\|^2 \right] \\
&\quad + (1-\alpha)\left[\sum_{(m,n),(m',n')\in\partial\Omega} dl((m,n),(m',n')) - \lambda\,|\nabla v_{m,n}|^2 + \lambda\left(\|\nabla_x v_{m,n}\|^2 + \|\nabla_y v_{m,n}\|^2\right) + \mu\|v-u\| \right] \\
&\quad - \left[\sum_{(m,n),(m',n')\in\partial\Omega} dl((m,n),(m',n')) - \lambda\,|\alpha\,\nabla w_{m,n} + (1-\alpha)\,\nabla v_{m,n}|^2 + \lambda\Big(\|\alpha\,\nabla_x\, w_{m,n} + (1-\alpha)\,\nabla_x v_{m,n}\|^2 \right. \\
&\quad \left. + \|\alpha\,\nabla_y\, w_{m,n} + (1-\alpha)\,\nabla_y\, v_{m,n}\|^2 \Big) + \mu\,\|\alpha\, w + (1-\alpha)\, v - u\|^2 \right] \\
&= \sum_{((m,n),(m',n'))\in\partial\Omega} -\lambda\left[\alpha\,|\nabla w_{m,n}|^2 + (1-\alpha)\,|\nabla v_{m,n}|^2 - |\alpha\,\nabla w_{m,n} + (1-\alpha)\,\nabla v_{m,n}|^2\right] \\
&\quad + \lambda \sum_{m,n=1}^{N} \alpha\,(|\nabla w_{m,n}|^2) + (1-\alpha)\,|\nabla v_{m,n}|^2 - |\alpha\,\nabla w_{m,n} + (1-\alpha)\,\nabla v_{m,n}|^2 \\
&\quad + \mu\left(\alpha\,\|w-u\|^2 + (1-\alpha)\,\|v-u\|^2 - \|\alpha\, w + (1-\alpha)\, v - u\|^2\right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{((m,n),(m',n'))\in\partial\Omega} -\lambda\Big[\alpha\left|\nabla w_{m,n}\right|^2 + (1-\alpha)\left|\nabla v_{m,n}\right|^2 - \alpha^2\left|\nabla w_{m,n}\right|^2 \\
&\quad - 2\alpha\,(1-\alpha)\langle\nabla w_{m,n}, \nabla v_{m,n}\rangle - (1-\alpha)^2\left|\nabla v_{m,n}\right|^2\Big] \\
&\quad + \lambda\sum_{m,n=1}^{N}\left[\alpha(1-\alpha)\left|\nabla w_{m,n}\right|^2 + (1-\alpha)(1-(1-\alpha))\left|\nabla v_{m,n}\right|^2 - 2\alpha(1-\alpha)\left\langle\nabla w_{m,n}, \nabla v_{m,n}\right\rangle\right] \\
&\quad + \mu\left[\alpha(1-\alpha)\left\|w-u\right\|^2 + (1-\alpha)\alpha\left\|v-u\right\|^2 - 2\alpha(1-\alpha)\left\langle w-u, v-u\right\rangle\right] \\
&= \sum_{((m,n),(m',n'))\in\partial\Omega} \underbrace{-}_{<0}\underbrace{\lambda\alpha(1-\alpha)}_{>0}\left|\nabla w_{m,n} - \nabla v_{m,n}\right|^2 + \left(\sum_{m,n=1}^{N}\underbrace{\lambda\alpha(1-\alpha)}_{\geq 0}\left|\nabla w_{m,n} - \nabla v_{m,n}\right|^2\right) \\
&\quad + \underbrace{\mu\alpha(1-\alpha)}_{\geq 0}\underbrace{\left\|w+v-u\right\|^2}_{\geq 0} \\
&\geq \sum_{m,n=1}^{N}\lambda\alpha(1-\alpha)\left|\nabla w_{m,n} - \nabla v_{m,n}\right|^2 - \lambda\alpha(1-\alpha)\sum_{((m,n),(m',n'))\in\partial\Omega}\left|\nabla w_{m,n} - \nabla v_{m,n}\right| \\
&= \underbrace{\lambda\alpha(1-\alpha)}_{>0}\left[\sum_{m,n=1}^{N}\underbrace{\left(\left|\nabla w_{m,n} - \nabla v_{m,n}\right|^2 - \sum_{m',n'=1}^{N}\underbrace{H^{(m',n')}\left(\left|\nabla w_{m,n} - \nabla v_{m,n}\right|^2\right)}_{\star}\right)}_{\text{pas toujours positif si } \star\neq 0}\right].
\end{aligned}
$$

Alors :

On peut dire que notre fonction F n'est pas toujours convexe. Cela dépend de notre frontière.

4 Algorithme

4.1 Pas de plus grande descente de $F(w)$

Soit $d = -\nabla F(w)$ pour un w fixé.
Cherchons le minimiseur de

$$\begin{aligned} h :&\mathbb{R} \to \mathbb{R} \\ &\alpha \mapsto h(\alpha) = F(w + \alpha d). \end{aligned}$$

$$\begin{aligned} h(t) &= F(w + td) \\ &= \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) - \lambda\left(|\nabla w_{m,n} + t\,\nabla d_{m,n}|^2\right) + \lambda \sum_{m,n=1}^{N} |\nabla(w+td)_{m,n}|^2 + \mu\,\|w + td - u\|^2 \\ &= \sum_{((m,n),(m',n'))\in\partial\Omega} dl((m,n),(m',n')) - \lambda\left(|\nabla w_{m,n}|^2 + 2\,t\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle + t^2\,|\nabla d_{m,n}|^2\right) \\ &+ \lambda\left(\sum_{m,n=1}^{N} |\nabla w_{m,n}|^2 + 2\,t\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle + t^2|\nabla d_{m,n}|^2\right) + \mu\left(\|w-u\|^2 + 2\,t\langle w-u, d\rangle + t^2\|d\|^2\right) \\ &= t^2\left(\mu\|d\|^2 + \lambda\sum_{m,n=1}^{N}|\nabla d_{m,n}|^2 - \lambda\sum_{((m,n),(m',n'))\in\partial\Omega}|\nabla d_{m,n}|^2\right) \\ &+ 2\,t\left(\mu\langle w-u, d\rangle + \lambda\sum_{m,n=1}^{N}\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle - \lambda\sum_{((m,n),(m',n'))\in\partial\Omega}\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle\right) + C, \end{aligned}$$

où C contient les termes ne dépendant pas de t.

Alors :

$$\begin{aligned} h'(t) = 2t&\left(\mu\|d\|^2 + \lambda\sum_{m,n=1}^{N}|\nabla d_{m,n}|^2 - \lambda\sum_{((m,n),(m',n'))\in\partial\Omega}|\nabla d_{m,n}|^2\right) \\ &+ 2\left(\mu\langle w-u, d\rangle + \lambda\sum_{m,n=1}^{N}\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle - \lambda\sum_{((m,n),(m',n'))\in\partial\Omega}\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle\right). \end{aligned}$$

$$h'(t) = 0 \Longleftrightarrow t^\star = \frac{a}{b},$$

avec

$$\begin{cases} a = \lambda\displaystyle\sum_{((m,n),(m',n'))\in\partial\Omega}\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle - \lambda\sum_{m,n=1}^{N}\langle\nabla w_{m,n}, \nabla d_{m,n}\rangle - \mu\langle w-u, d\rangle \\ b = \mu\|d\|^2 + \lambda\displaystyle\sum_{m,n=1}^{N}|\nabla d_{m,n}|^2 - \lambda\sum_{((m,n),(m',n'))\in\partial\Omega}|\nabla d_{m,n}|^2 \end{cases}.$$

Alors le pas de plus grande descente est

$$\boxed{t^\star = \frac{a}{b}.}$$

4.2 Gradient de l'énergie $G(\varphi)$

Trouvons le gradient de G :

Soit $\forall m,n,m',n' \in \{1,...,N\}$ $a_{((m,n),(m',n'))} = \left(dl((m,n),(m',n')) - \lambda |\nabla w_{m,n}|^2 \right) \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}}$

$\forall \varphi \in \mathbb{R}^{N^2}$

$$
\begin{aligned}
G(\varphi + h) &= \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} a_{((m,n),(m',n'))} H_\epsilon(\varphi_{m,n} + h_{m,n}) \left(1 - H_\epsilon(\varphi_{m',n'} + h_{m',n'})\right) \\
&= \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} a_{((m,n),(m',n'))} \left[H_\epsilon(\varphi_{m,n}) + h_{m,n} H'_\epsilon(\varphi_{m,n}) + \mathcal{O}(|h_{m,n}|)\right] \\
&\left[1 - (H_\epsilon(\varphi_{m',n'}) + h_{m',n'} H'_\epsilon(\varphi_{m',n'})) + \mathcal{O}(|h_{m',n'}|)\right] \\
&= G(\varphi) + \mathcal{O}(\|h\|) + \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} a_{((m,n),(m',n'))} \\
&\left[-H_\epsilon(\varphi_{m,n}) H'_\epsilon(\varphi_{m',n'}) h_{m',n'} + h_{m,n} H'_\epsilon(\varphi_{m,n})(1 - H_\epsilon(\varphi_{m',n'}))\right]
\end{aligned}
$$

On a, par changement de borne :

$$
\begin{aligned}
&\sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} \left(-dl((m,n),(m',n')) + \lambda \left|\nabla w_{m,n}\right|^2 \right) \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}} H_\varepsilon(\varphi_{m,n}) H'_\varepsilon(\varphi_{m',n'}) h_{m',n'} \\
&= \sum_{m',n'=1}^{N} \sum_{m,n=1}^{N} \left(-dl(\underbrace{(m',n'),(m,n)}_{\text{symétrique}}) + \lambda \left|\nabla w_{m',n'}\right|^2 \right) \mathbb{1}_{\{(m,n) \in \sigma(m',n')\}} H_\varepsilon(\varphi_{m',n'}) H'_\varepsilon(\varphi_{m,n}) h_{m,n} \\
&= \sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} \left(-dl(\underbrace{(m,n),(m',n')}_{\text{symétrique}}) + \lambda \left|\nabla w_{m',n'}\right|^2 \right) \mathbb{1}_{\{(m,n) \in \sigma(m',n')\}} H_\varepsilon(\varphi_{m',n'}) H'_\varepsilon(\varphi_{m,n}) h_{m,n}.
\end{aligned}
$$

Remarque :

$$
\begin{aligned}
(m',n') \in \sigma(m,n) &\iff |m - m'| + |n - n'| = 1 \\
&\iff (m,n) \in \sigma(m',n')
\end{aligned}
$$

Alors $\mathbb{1}_{\{(m,n) \in \sigma(m',n')\}} = \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}}$

Alors

$$
\begin{aligned}
\langle \nabla G(\varphi), h \rangle = &\sum_{m,n=1}^{N} \sum_{m',n'=1}^{N} \left(-dl((m,n),(m',n')) + \lambda \left|\nabla w_{m',n'}\right|^2 \right) \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}} H_\varepsilon(\varphi_{m',n'}) H'_\varepsilon(\varphi_{m,n}) h_{m,n} \\
&+ \sum_{m,n=1}^{N} \sum_{m',n'=1} \left(dl((m,n),(m',n')) - \lambda \left|\nabla w_{m,n}\right|^2 \right) \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}} H'_\varepsilon(\varphi_{m,n})(1 - H_\varepsilon(\varphi_{m',n'})) h_{m,n}.
\end{aligned}
$$

Alors

$$
\begin{aligned}
\forall \varphi \in \mathbb{R}^{N^2}, (\nabla G(\varphi))_{m,n} = \sum_{m',n'=1}^{N} H'_\varepsilon(\varphi_{m,n}) \Big[&dl((m,n),(m',n'))(1 - 2H_\varepsilon(\varphi_{m',n'})) + \lambda \left|\nabla w_{m',n'}\right|^2 H_\varepsilon(\varphi_{m',n'}) \\
&+ \lambda \left|\nabla w_{m,n}\right|^2 (H_\varepsilon(\varphi_{m',n'}) - 1) \Big] \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}}.
\end{aligned}
$$

$$
\begin{aligned}
\forall \varphi \in \mathbb{R}^{N^2}, \nabla G(\varphi) = \Bigg(\sum_{m',n'=1}^{N} H'_\varepsilon(\varphi_{m,n}) \Big[&dl((m,n),(m',n'))(1 - 2H_\varepsilon(\varphi_{m',n'})) + \lambda \left|\nabla w_{m',n'}\right|^2 H_\varepsilon(\varphi_{m',n'}) \\
&+ \lambda \left|\nabla w_{m,n}\right|^2 (H_\varepsilon(\varphi_{m',n'}) - 1) \Big] \mathbb{1}_{\{(m',n') \in \sigma(m,n)\}} \Bigg)_{1 \le m,n \le N}.
\end{aligned}
$$

4.3 Pas de plus grande descente de $G(\varphi)$

Soit

$$\begin{aligned} H : \mathbb{R} &\to \mathbb{R} \\ \alpha &\mapsto H(\alpha) = G(\varphi + \alpha d) \text{ avec } d = -\nabla G(\varphi). \end{aligned}$$

On a :

$$\begin{aligned} G(\varphi + \alpha d) &= G(\varphi) + \langle \nabla G(\varphi), \alpha d \rangle + \frac{1}{2}\langle \nabla^2 G(\varphi)\alpha d, \alpha d\rangle + \mathcal{O}(|\alpha d|) \\ &= G(\varphi) + \alpha\langle \nabla G(\varphi), d\rangle + \frac{\alpha^2}{2}\langle \nabla^2 G(\varphi) d, d\rangle + \mathcal{O}(|\alpha d|), \end{aligned}$$

où $\mathcal{O}(|\alpha d|) \xrightarrow[|\alpha d|\to 0]{} 0$.
Donc

$$H(\alpha) \cong G(\varphi) - \alpha\|\nabla G(\varphi)\|^2 + \frac{\alpha^2}{2}\langle \nabla^2 G(\varphi) d, d\rangle.$$

$$H'(\alpha) \cong -\|\nabla G(\varphi)\|^2 + \alpha\langle \nabla^2 G(\varphi) d, d\rangle.$$

$$H'(\alpha^\star) = 0 \Longleftrightarrow \alpha^\star = \frac{\|\nabla G(\varphi)\|^2}{\langle \nabla^2 G(\varphi) d, d\rangle}.$$

Calculons $\langle \nabla^2 G(\varphi) d, d\rangle$.
On a :

$$G(\varphi + d) = G(\varphi) + \underbrace{\langle \nabla G(\varphi), d\rangle}_{-\|\nabla G(\varphi)\|^2} + \frac{1}{2}\langle \nabla^2 G(\varphi) d, d\rangle + \mathcal{O}(|d|)$$

donc

$$\langle \nabla^2 G(\varphi) d, d\rangle \cong 2\left(G(\varphi + d) - G(\varphi) + \|\nabla G(\varphi)\|^2\right).$$

Alors le pas de plus grande descente est :

$$\boxed{\alpha^\star = \frac{\|\nabla G(\varphi)\|^2}{2\left(G(\varphi + d) - G(\varphi) + \|\nabla G(\varphi)\|^2\right)}.}$$

$\alpha^\star$ est le meilleur pas pour faire varier Ω.

5 Implémentation

Malgré les calculs très développés des gradients et du pas, l'implémentation n'est pas facile et ne fonctionne pas comme attendu.

5.1 Voisinage

Tout d'abord, il est nécessaire de connaître les voisins d'un point. Pour cela, nous avons vu en cours la formule de voisinage :

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

avec d la distance de notre voisinage. Nous utiliserons $d = 1$ dans notre cas.

Voici le code utilisé pour rechercher les voisins d'un point (m,n). Le programme va parcourir un carré de taille $2d \times 2d$ puis calcule les points correspondant à l'équation ci-dessus. Si un point se trouve en dehors de l'image de dimension $H \times L$, alors le point ne sera pas parcouru (d'où les fonctions $\min$ et $\max$).

```
def voisins(m, n, H, L, distance=1):
    liste = set()
    for i, j in product(range(max(0, m - distance), min(m + distance + 1, L)),
                        range(max(0, n - distance), min(n + distance + 1, H))):
        if abs(m - i) + abs(n - j) == distance:
            liste.add((i, j))
    return liste
```

Pour une distance de 7, nous obtenons un résultat en approximativement 90 µs.

Pour représenter Ω, nous avons décidé de stocker sa valeur dans un ensemble de couples. Par exemple, pour deux points aux coordonnées $(1,2)$ et $(1,3)$, on a $\Omega = \{(1,2),(1,3)\}$.

Ensuite, nous devons implémenter la frontière. Pour cela, nous stockons un ensemble de couples de couples, comme dans la formule du cours :

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

En Python, cela donne la fonction suivante :

```
def frontiere(points, H, L, distance=1):
    liste = set()
    for point in points:
        vois = voisins(point[0], point[1], H, L, distance)
        for v in vois:
            if not v in points:
                liste.add((point, v))
    return liste
```

Nous avons également un code pour calculer la longueur du contour dl défini par

$$dl((m,n),(m',n')) = \frac{e^{-\frac{(u_{m,n}-u_{m',n'})^2}{2\sigma^2}}}{\sqrt{(m-m')^2+(n-n')^2}}$$

```
def dl(v, points, sigma=0.2):
    m = points[0][0]
    n = points[0][1]
    m_p = points[1][0]
    n_p = points[1][1]
    s = (np.exp(-(v[m,n]-v[m_p,n_p])**2/(2*sigma**2)))/(np.sqrt((m-m_p)**2+(n-n_p)**2))
    return s
```

5.2 Choix de l'image

Afin de tester notre implémentation, nous avons choisi d'utiliser 4 images différentes. Deux photos, et deux images crées.

(a) Barbara, la célèbre demoiselle (b) Jackie Chan, un célèbre acteur (c) Une étoile de mer en couleur (d) Ce smiley fait peur avec notre algorithme

FIGURE 1 – Nos images utilisées pour les tests

5.3 Fonctions et Gradients

5.3.1 Operateurs Dx et Dy

Les opérateurs Dx et Dy sont codés via deux fonctions. Ces fonctions ont été données par l'enseignant. Le paramètre $adjoint$ permet de préciser si on souhaite l'opérateur adjoint Dx^*/Dy^* ou non.

```
def grad_x(img, adjoint):
    sx , sy = np.shape(img)
    diff_x = np.copy(img)
    if adjoint==0:
        for x in range(sx):
            if x==sx-1:
                xnext=0
            else:
                xnext=x+1
            for y in range(sy):
                diff_x[x,y] = img[xnext,y]- img[x,y]
    else:
        for x in range(sx):
            if x==0:
                xprev=sx-1
            else:
                xprev=x-1
            for y in range(sy):
                diff_x[x,y] = img[xprev,y]- img[x,y]
    return diff_x

def grad_y(img, adjoint):
    sx , sy = np.shape(img)
    diff_y =  np.copy(img)
    if adjoint==0:
        for y in range(sy):
            if y==sy-1:
                ynext=0
            else:
                ynext=y+1
            for x in range(sx):
                diff_y[x,y] = img[x,ynext]- img[x,y]
    else:
        for y in range(sy):
            if y==0:
                yprev=sy-1
            else:
                yprev=y-1
            for x in range(sx):
                diff_y[x,y] = img[x,yprev]- img[x,y]
    return diff_y
```

5.3.2 Préliminaires de F

Avant de coder la fonction $F(w)$, nous avons besoin de coder la fonction $E(w)$. Le code suivant est utilisé. Il utilise $gradW$ qui est calculé en faisant $D_x w^2 + D_y w^2$.

```
def E(u, w, lmbda, mu, gradW):
    data = np.linalg.norm(w - u, ord='fro')
    return lmbda * np.sum(gradW) + mu * data * data

def grad_E(u, w, lmbda, mu):
    return 2 * lmbda * (grad_x(grad_x(w, 0), 1) + grad_y(grad_y(w, 0), 1)) + 2 * mu *
    (w - u)
```

Ensuite, il faut la fonction $P(\Omega)$. On récupère les points de la frontière de Ω et on boucle dessus.

```
def P_omega(omega, w, distance, lmbda, gradW):
    s = 0
    for p in frontiere(omega, len(w), len(w[0]), distance):
        s += dl(w, p) - lmbda * gradW[p[0][0], p[0][1]]
    return s
```

5.3.3 La fonction F et son gradient

Voici enfin le résultat tant attendu de $F(w)$. Les fonctions sont simples grâce au travail préliminaire de E et $P(\Omega)$.

```
def F(omega, v, w, lmbda, mu, distance=1):
    # Norme de grad_w ^ 2
    grad_w = grad_x(w, 0) ** 2 + grad_y(w, 0) ** 2
    val_a = P_omega(omega, w, distance, lmbda, grad_w)
    return val_a + E(v, w, lmbda, mu, grad_w)

def grad_F(w, v, lmbda, mu, omega):
    return -2 * lmbda * (grad_x(H_sum(grad_x(w, 0), omega), 1)) - 2 * lmbda *
    grad_y(H_sum(grad_y(w, 0), omega), 1) + 2 * lmbda * (grad_x(grad_x(w, 0), 1) +
    grad_y(grad_y(w, 0), 1)) + 2 * mu * (w - v)
```

5.3.4 La fonction $\mathcal{H}_\varepsilon(\varphi_{m,n})$ et sa dérivée

```
def H_eps(t, eps):
    if t >= eps:
        s = 1
    elif t <= -eps:
        s = 0
    else:
        s = 1/2 * (1 + t/eps + 1/np.pi * np.sin(np.pi / eps * t))
    return s

def H_p_eps(t, eps):
    if abs(t) < eps:
        s = 1/(2*eps) + 1/(2*eps) * np.cos(np.pi * t / eps)
    else:
        s = 0
    return s
```

5.3.5 La fonction G et son gradient

Voici un autre point intéressant. La fonction G doit parcourir tous les points de notre image, ce qui rend le programme plutôt lent à cause de ces fonctions.

```
def G(om, v, gradW, lmbda, eps):
```

```
    s = 0
    h = len(om)
    l = len(om[0])
    for m,n in product(range(h), range(l)):
        for m_p, n_p in voisins(m,n,h,l):
            s += (dl(v, ((m,n),(m_p,n_p)))-lmbda * gradW[m,n])*H_eps(om[m,n], eps)*(1
    - H_eps(om[m_p,n_p], eps))
    return s

def grad_G(om, v, gradW, lmbda, eps):
    out = np.zeros(om.shape)
    h = len(om)
    l = len(om[0])
    for m,n in product(range(h), range(l)):
        for m_p, n_p in voisins(m,n,h,l):
            out[m,n] += H_p_eps(om[m,n], eps)*(dl(v, ((m,n),(m_p,n_p))) * (1 - 2 *
    H_eps(om[m_p,n_p], eps)) + lmbda * (gradW[m_p,n_p] * H_eps(om[m_p,n_p], eps) +
    gradW[m,n] * (H_eps(om[m_p,n_p], eps) - 1)))
    return out
```

5.4 Initialisation de Ω

5.4.1 Sinus

Ici, on initalise Ω avec $\Omega_{x,y} = \sin\left(\frac{\pi}{5}x\right)\sin\left(\frac{\pi}{5}y\right)$

```
def createOmegaSinus(H,L):
    img = np.zeros((H,L), float)
    for m, n in product(range(H), range(L)):
        img[m,n] = np.sin(np.pi / 5 * m) * np.sin(np.pi / 5 * n)
    return img
```

5.4.2 Aléatoire

On peut également initialiser aléatoirement selon une loi uniforme entre -1 et 1.

```
def createOmegaAleatoire(H,L):
    return np.random.uniform(-1, 1, (H,L))
```

5.4.3 Vide

Pour tester si un algorithme est puissant, on peut ne pas initialiser Ω et laisser sa valeur vide.

```
def createOmegaVide(H,L):
    return np.zeros((H,L), float)
```

5.4.4 Zone

Enfin, afin d'aider le plus possible l'algorithme, il est possible d'initialiser selon une zone choisie.

```
def createOmegaZone(H,L,iMin,iMax,jMin,jMax):
    om = np.zeros((H,L), float)
    for i,j in product(range(H), range(L)):
        if iMin <= i <= iMax and jMin <= j <= jMax:
            om[i,j] = np.random.random(1) / 100
    return om
```

5.5 Outils pour Ω

Afin de pouvoir passer de l'ensemble Ω à l'image ϕ, il est nécessaire d'avoir des fonctions effectuant la traduction. La première permet de cacluler l'ensemble Ω à partir d'une image, et la deuxième permet d'afficher Ω sous la forme d'une image.

```
def calculOmega(om):
    omega = set()
    for m,n in product(range(len(om)), range(len(om[0]))):
        if om[m,n] > 0:
            omega.add((m,n))
    return omega

def filtreOmega(om, omega):
    out = np.zeros(om.shape)
    for m,n in omega:
        out[m,n] = 1
    return out
```

5.6 Pas optimal

Pour calculer la descente de gradient, nous devons d'abord calculer le pas optimal.

```
def scalaire(M1, M2):
    return np.matrix(M1 * M2).sum()

def pas_F(w, v, omega, d, lmbda, mu):
    gradW_x = grad_x(w, 0)
    gradW_y = grad_y(w, 0)
    gradD_x = grad_x(d, 0)
    gradD_y = grad_y(d, 0)

    a = 0
    for mn, _ in frontiere(omega, len(w), len(w[0])):
        a += gradW_x[mn] * gradD_x[mn] + gradW_y[mn] * gradD_y[mn]
    a *= lmbda
    a -= lmbda * np.sum(gradW_x * gradD_x + gradW_y * gradD_y)
    a -= mu * scalaire(w - v, d)

    b = mu * (np.linalg.norm(d) ** 2) + lmbda * np.sum(gradD_x ** 2 + gradD_y ** 2)
    s2 = 0
    for mn, _ in frontiere(omega, len(w), len(w[0])):
        s2 += gradD_x[mn] ** 2 + gradD_y[mn] ** 2
    b -= lmbda * s2

    return a/b

def pas_G(om, v, d, lmbda, gradW, eps):
    a = np.linalg.norm(d) ** 2
    b = 2 * (G(om + d, v, gradW, lmbda, eps) - G(om, v, gradW, lmbda, eps) +
    np.linalg.norm(d)**2)
    return a/b
```

5.7 Descente de gradients

Voici le code final. Malheureusement, il ne fonctionne pas, et renvoie seulement les contours de notre image. C'est un bon début, mais ce n'est pas le résultat attendu.

```
def descente(v,lmbda,mu,eps,nbIter):
    if mu < 24*lmbda:
        raise Exception("Mu doit etre 24 fois plus grand que lambda !!!!")

```

```
    H = len(v)
    L = len(v[0])

    w = np.copy(v)
    om = createOmega(H, L)

    for it in range(nbIter):
        omega = calculOmega(om)

        dF = - grad_F(w, v, lmbda, mu, omega)
        pasF = pas_F(w, v, omega, dF, lmbda, mu)
        w = w + pasF * dF

        gradW = grad_x(w, 0) ** 2 + grad_y(w, 0) ** 2
        dG = - grad_G(om, v, gradW, lmbda, eps)
        pasG = pas_G(om, v, dG, lmbda, gradW, eps)
        om = (om + pasG * dG)

    return om
```

6 Résultats

Voici un résultat obtenu avec l'image de Jackie Chan, après quelques itérations. Nous avons utilisé les paramètres suivants :

— $\lambda = 52000$
— $\mu = 8900000$
— $\epsilon = 20000$

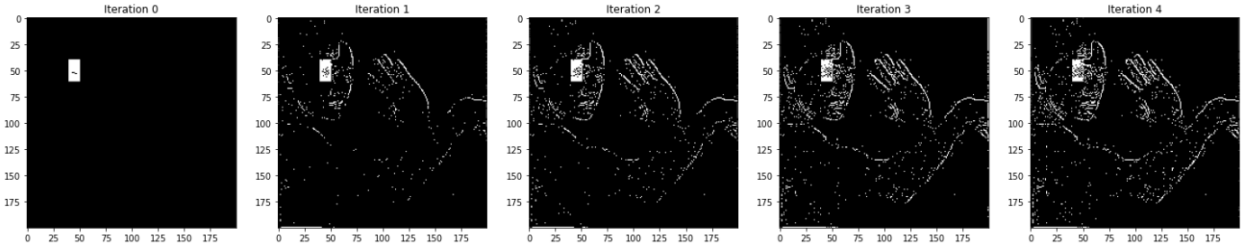

FIGURE 2 – Résultat après 5 itérations

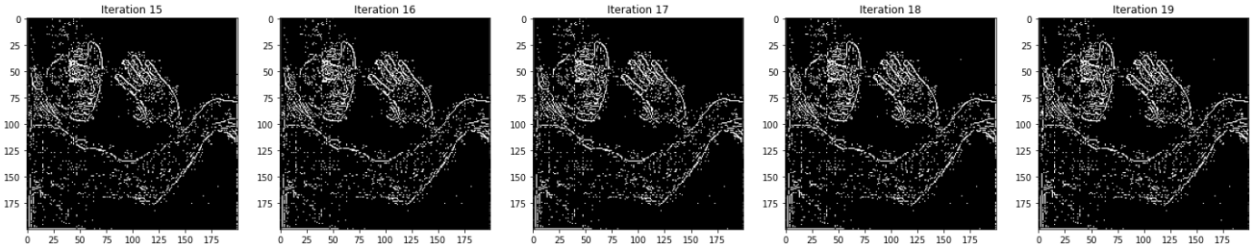

FIGURE 3 – Résultat après 20 itérations

Voici un autre résultat avec une image d'étoile. Nous avons utilisé les paramètres suivants :

— $\lambda = 500$
— $\mu = 896000$
— $\epsilon = 4000000$

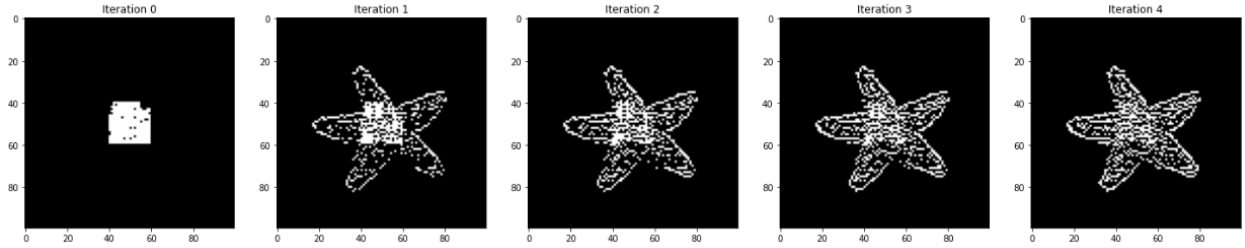

FIGURE 4 – Résultat après 5 itérations