

EA MAP 511 - RAPPORT FINAL

**Segmentation d'images par les méthodes de
Random Walker et de Machine Learning**

1 Décembre 2022

Tutrice : Stéphanie ALLASSONNIÈRE,
Elèves : Sacha BRAUN, Lucas GASCON, Hippolyte PILCHEN



TABLE DES MATIÈRES

1 Théorie de l'algorithme de Random Walker	4
1.1 Preuves Mathématiques	4
1.1.1 Un premier problème sur les chaines de Markov absorbantes	4
1.1.2 L'équivalence de ce problème avec le problème de Léo Grady	6
1.1.3 Par une méthode probabilistique en utilisant Dirichlet	8
1.2 Approche Physique : analogie du circuit électrique	9
2 L'algorithme de Random Walker	15
2.1 Code principal	15
2.2 Les résultats de notre algorithme	16
2.3 Compréhension pratique de l'algorithme	18
3 Analyse comparative : SegNet et Random Walker	22
3.1 SegNet	22
3.2 Réponses aux transformations des deux modèles	25

INTRODUCTION

Dans un contexte où les méthodes d'apprentissage profond et leurs applications se développent de plus en plus, il convient de questionner et d'analyser leurs différences par rapport à d'autres algorithmes. Nous réalisons notre étude comparative en analysant la performance de différentes méthodes pour la segmentation d'image. Ce domaine est très étudié, notamment pour la segmentation d'image à caractère médical. Pour entraîner les algorithmes d'apprentissage profond, il est important d'avoir une grande banque d'images déjà labellisées (c'est-à-dire où la segmentation a déjà été faite). Face aux coûts des médecins et à la rareté de certaines images médicales, la création d'un tel jeu d'images semble complexe et des méthodes alternatives avec des résultats différents peuvent être appliquées. Nous nous concentrerons ici sur deux méthodes de segmentation d'image : l'une fondée sur les marches aléatoires et l'autre, la méthode d'apprentissage profond SegNet, un réseau neuronal convolutif. Ce premier algorithme est utilisé pour la segmentation d'image médicale par les appareils Siemens. Il permet de faire des diagnostics précis tout en optimisant le temps de travail du médecin et ainsi le coût global de l'opération. Ainsi, nous allons démontrer les propriétés de l'algorithme de Random Walker pour ensuite analyser et comparer ses performances avec celles d'un réseau neuronal convolutif, ici SegNet.

1

THÉORIE DE L'ALGORITHME DE RANDOM WALKER

PRÉSENTATION DE L'ALGORITHME DE RANDOM WALKER

Nous présentons ici sommairement l'algorithme de random walker de Léo Grady. Nous donnons en entrée une image à l'utilisateur, qui va marquer certains pixels de cette image comme appartenant à certains labels. Ensuite, pour chaque pixel non marqué, on définit une marche aléatoire commençant à ce pixel qui se déplace de voisins en voisins. La probabilité d'atteindre un voisin dépend de la différence d'intensité entre les deux pixels, ce qui est matérialisé par une fonction de coût. Pour chaque pixel non labellisé, on regarde les probabilités pour cette marche aléatoire d'atteindre un pixel marqué, ce qui nous donne pour chaque pixel une probabilité d'appartenir à chaque label. Il suffit ensuite de conclure que ce pixel est du label pour lequel il a la plus grande probabilité d'appartenance. Ce processus est cependant très coûteux en temps à réaliser tel qu'il est ici présenté. Léo Grady énonce dans son papier que ce problème est en réalité équivalent à un problème d'algèbre linéaire, et qu'il suffit de résoudre un système linéaire pour obtenir toutes les probabilités recherchées, ce que nous allons ici démontrer.

1.1 PREUVES MATHÉMATIQUES

1.1.1 • UN PREMIER PROBLÈME SUR LES CHAINES DE MARKOV ABSORBANTES

Avant de nous intéresser au problème de Léo Grady, essayons de comprendre un peu plus le comportement des marches aléatoires lorsqu'il existe des états absorbants et des états transitoires.

Dans la suite, W représentera une marche aléatoire entre n états, avec k éléments absorbants $(a_j)_{1 \leq j \leq k} := A$, et $n - k$ éléments transitoires $(t_i)_{1 \leq i \leq n-k} := T$.

Notation 1 (Matrice de probabilité de passage d'un point à un autre) *On notera Q la matrice de probabilités telle que :*

$$\forall j \in [1, n], \forall i \in [1, n], (Q)_{ij} = P(W_1 = j | W_0 = i)$$

Notation 2 (Temps d'arrêt) *Soit $x \in [1, n]$, on notera T_A^x le temps d'arrêt*

$$T_A^x = \inf \{i \geq 0, W_i \in A | W_0 = x\}$$

Lemme 1 (Temps d'arrêt fini presque sûrement)

$$\forall x \in [1, n]$$

$$P(T_A^x < \infty) = 1$$

Preuve :

Quitte à permutez, supposons que les k états absorbants sont placés aux k premières positions. Notons $k_i = \min\{j \geq 1, \exists l \leq k, P(W_j = l | W_0 = i) > 0\}$ le nombre de pas minimal nécessaire pour atteindre un état absorbant l depuis i , où i est un état non absorbant.

Soit $p_i = P(W_{k_i} > k) < 1$ la probabilité de ne pas atteindre d'état absorbant en k_i pas, en partant de i .

Soient $M = \max_{k+1 \leq i} k_i$ et $p = \max_{k+1 \leq i} p_i$:

Alors la probabilité de ne pas atteindre d'état absorbant en au plus M pas en partant de n'importe quel point non absorbant est bornée par p . Ainsi la probabilité de ne pas atteindre d'état absorbant en au plus Mk pas est bornée par p^k qui tend vers 0, et donc $P(T_A^x < \infty) = 1$

Notation 3 (Matrice de probabilité de passage d'un point à un état absorbant) *On notera R la matrice de probabilités telle que*

$$\forall j \in [1, k], \forall i \in [1, n-k]$$

$$(R)_{ij} = P(W_{T_A^i} = j | W_0 = i)$$

Notre objectif est de trouver cette matrice de probabilité R . Sans perte de généralité, on peut ordonner les états absorbants pour les placer aux k premières positions. Ainsi, on peut réécrire la matrice Q comme une matrice par bloc de la manière suivante :

$$Q = \left(\begin{array}{c|c} I_k & 0 \\ \hline M & \tilde{Q} \end{array} \right)$$

En effet, la probabilité une fois qu'on a atteint un état absorbant d'atteindre ensuite un autre état est de 0 car celle de rester sur ce point est de 1.

Théorème 1 *Avec les notations définies ci-dessus*

$$R = (I_{n-k} - \tilde{Q})^{-1} M$$

Preuve :

Commençons par montrer que $(I_{n-k} - \tilde{Q})$ est inversible.

Soit $x \in R^{n-k}$ tel que $(I_{n-k} - \tilde{Q})x = 0$ et $x = \tilde{Q}x$. Alors par une récurrence immédiate, $x = \tilde{Q}^\infty x$ puis d'après le lemme 1 on a $\tilde{Q}^\infty = 0$ donc $x = 0$, et donc le noyau de $(I_{n-k} - \tilde{Q})$ est nul ce qui montre l'inversibilité.

Notons V_j l'ensemble des voisins d'un état j , $V_j = \{l \in [1, n], P(W_1 = l | W_0 = j) > 0\}$. Maintenant, on a, pour $i \in [k+1, n]$, $j \in [1, k]$

$$\begin{aligned}
 (R)_{i-k,j} &= P(W_{T_A^i} = j | W_0 = i) \\
 &= P(W_{T_A^i} = j \cap T_A^i = 1 | W_0 = i) + P(W_{T_A^i} = j \cap T_A^i > 1 | W_0 = i) \\
 &= (Q)_{i,j} + \sum_{l \in V_j} P(W_{T_A^l} = j | W_0 = l) (Q)_{i,l} \\
 &= (Q)_{i,j} + \sum_{l \in V_j} (Q)_{i,l} (R)_{l-k,j} \\
 &= (M)_{i-k,j} + \sum_{l \in V_j} (\tilde{Q})_{i-k,l-k} (R)_{l-k,j} \\
 &= (M)_{i-k,j} + (\tilde{Q}R)_{i-k,j}
 \end{aligned}$$

On a donc

$$R = M + \tilde{Q}R$$

Donc

$$(I_{n-k} - \tilde{Q})R = M$$

Ce qui suffit pour démontrer

$$R = (I_{n-k} - \tilde{Q})^{-1}M$$

1.1.2 • L'ÉQUIVALENCE DE CE PROBLÈME AVEC LE PROBLÈME DE LÉO GRADY

Ce problème présente des similitudes avec le problème de Léo Grady. En effet, on cherche la probabilité d'atteindre un état absorbant en partant d'un état transitoire, tandis que dans le problème de Léo Grady, on cherche à trouver les probabilités d'atteindre un état labellisé en partant d'un état non labellisé. La principale différence entre les deux problèmes est que les pixels labellisés ne sont pas définis comme absorbants avec la méthode de Léo Grady. Or, dans la preuve du théorème 1, seules les deux matrices par blocs inférieurs interviennent dans la solution. Il suffit donc d'ordonner les pixels dans le bon ordre (les pixels labellisés en premiers) pour résoudre le problème.

Dans le papier [1], on étudie la matrice L sans mentionner de matrice stochastique. Comment interpréter avec des probabilités l'impact du poids entre deux voisins sur la probabilité de passer d'un état à un autre ?

Nous allons voir que la matrice L est en réalité très proche d'une matrice stochastique

Notation 4 On notera la matrice L définie dans le papier de Léo Grady comme :

$$L = D - W$$

avec $D = \text{diag}(d_i)_{1 \leq i \leq n}$ et $W = (w_{ij})_{1 \leq i, j \leq n}$

On rappelle que $w_{ij} = 0$ si i et j ne sont pas voisins sur l'image (il est possible de se déplacer

en diagonale), et $w_{ij} = e^{-\beta(g_i - g_j)^2}$ sinon, où g_i décrit l'intensité du pixel i . On rappelle aussi que $d_i = \sum_{j \in V_i} w_{ij}$.

Enfin on rappelle la notation :

$$L = \left(\begin{array}{c|c} L_m & B \\ \hline B^T & L_u \end{array} \right)$$

Lemme 2 Avec les notations définies ci-dessus on a

$$Q = D^{-1}W$$

Où Q est la matrice stochastique qui à chaque point i du graphe associe la probabilité qu'une marche aléatoire partant de ce point à un temps $t=0$ atteigne le voisin j au temps $t=1$, pondérée par les poids des voisins. Autrement dit,

$$(Q)_{ij} = \frac{w_{ij}}{d_i}$$

Preuve : Immédiat car D est une matrice diagonale sans coefficients diagonaux nuls. Il suffit ensuite de vérifier que Q définit une matrice stochastique en sommant ses coefficients.

On note alors :

$$Q = \left(\begin{array}{c|c} K & C \\ \hline M & \tilde{Q} \end{array} \right)$$

Lemme 3 Avec les notations définies ci-dessus on a

$$L_u = D_u^{-1}(I - \tilde{Q})$$

et

$$B^T = -D_u^{-1}M$$

Preuve :

$$L = D^{-1}(I - Q) = \left(\begin{array}{c|c} D_m & 0 \\ \hline 0 & D_u \end{array} \right)^{-1} \left(\begin{array}{c|c} I - K & -C \\ \hline -M & I - \tilde{Q} \end{array} \right) = \left(\begin{array}{c|c} D_m^{-1}(I - K) & D_m^{-1}(I - \tilde{Q}) \\ \hline -D_u^{-1}M & D_u^{-1}(I - \tilde{Q}) \end{array} \right)$$

Ceci suffit pour démontrer le lien entre les deux problèmes. En effet, on a :

Théorème 2 (Solution du problème de Léo Grady) La matrice

$$m_u = -L_u^{-1}B^T$$

est telle que $(m_u)_{ij}$ donne la probabilité pour une marche aléatoire partant d'un point $i+k$ (non labellisé), de rencontrer comme premier point labellisé le point j .

Preuve :

Soit P la matrice stochastique d'une chaîne de Markov avec des états absorbants telle que :

$$P = \left(\begin{array}{c|c} I & 0 \\ \hline M & \tilde{Q} \end{array} \right)$$

Alors la matrice R d'équilibre recherchée est :

$$R = (I - \tilde{Q})^{-1}M = (D_u L_u)^{-1} * (-D_u B^T) = -L_u B^T$$

Ainsi, ce calcul matriciel donne pour chaque pixel labellisé la probabilité qu'une marche aléatoire commençant à un pixel non labellisé se termine sur ce pixel labellisé.

Pour conclure, il suffit pour chaque label de sommer les probabilités obtenues pour chaque pixels labellisés comme appartenant à ce label, ce qu'on effectue en multipliant le résultat final par x_m , et on obtient la formule recherchée :

$$X = -L_u^{-1}B^T x_m$$

Où $(x_m)_{ij} = 1$ si le pixel i est labellisé comme étant dans le label j , et 0 sinon.

Alors on a en notant $Lab(j)$ les points du label j :

$$(X)_{ij} = P(W_{T_A^i} \in Lab(j) | W_0 = i)$$

1.1.3 • PAR UNE MÉTHODE PROBABILISTIQUE EN UTILISANT DIRICHLET

Maintenant qu'on a prouvé l'équivalence entre les deux problèmes, on peut essayer de retrouver la solution au premier problème avec d'autres méthodes, par exemple avec une méthode de Dirichlet.

Notons A_k l'ensemble des pixels labellisés comme appartenant au label k , k entier.

Alors en notant f_k telle que $f_k : [1, n] \rightarrow \mathbb{R}$

On peut rechercher f_k harmonique telle que $f_k = 1_{A_k}$ pour chaque label k , d'où le théorème suivant.

Théorème 3 Soit k un label, alors :

$$f_k(x) = E[1_{A_k}(W_{T_A^x}) | W_0 = x] = P(W_{T_A^x} \in A_k | W_0 = x)$$

est solution du problème de Dirichlet :

$$\begin{cases} Qf_k = f_k \text{ sur } A_k^c \\ f_k = 1_{A_k} \text{ sur } A_k \end{cases}$$

Preuve :

L'égalité de droite est immédiate. Il suffit donc de démontrer l'égalité de gauche.

Ce résultat est démontré dans [2], l'égalité a lieu si $x \in A$. Sinon, notons $\mathcal{F}_q = \sigma(W_0, \dots, W_q)$ et on a pour $x \in A^c$

$$\begin{aligned}
f_k(x) &= E[1_{A_k}(W_{T_A^x})|W_0=x] \\
&= E[1_{A_k}(W_{1+T_A^{W_1}})|W_0=x] \\
&= E[E[1_{A_k}(W_{1+T_A^{W_1}})|\mathcal{F}_1]|W_0=x] \\
&= E[E[1_{A_k}(W_{T_A^{W_1}})|W_1]|W_0=x] \\
&= E[f_k(W_1)|W_0=x] \\
&= \sum_{1 \leq i \leq n} f_k(y)P(x,y) \\
&= Pf_k(x)
\end{aligned}$$

Alors résoudre ce problème de Dirichlet pour tous les labels nous donnera toutes les probabilités recherchées.

Essayons de résoudre ce problème de Dirichlet pour un label k , et par abus de notation notons $f = f_k$. Comme f est harmonique, on a :

$$Qf = f$$

Commençons pas noter $f = (f_m, f_u)$ en l'assimilant à un vecteur, f_m étant la partie labellisée ($f_m)_i = f_k(i) = 1_{A_k}(i)$ pour i un pixel labellisé).

On a alors, en considérant les pixels comme absorbants et en réutilisant les notations précédentes.

$$\left(\begin{array}{c|c} I & 0 \\ \hline M & \tilde{Q} \end{array} \right) \left(\begin{array}{c} f_m \\ f_u \end{array} \right) = \left(\begin{array}{c} f_m \\ f_u \end{array} \right)$$

et donc

$$f_u = Mf_m + \tilde{Q}f_u$$

soit

$$f_u = (I - \tilde{Q})^{-1}Mf_m$$

ce qui permet de trouver les valeurs de f sur les pixels non labellisés pour chaque label, et de résoudre notre problème.

1.2 APPROCHE PHYSIQUE : ANALOGIE DU CIRCUIT ÉLECTRIQUE

- ENTRÉE EN MATIÈRE

La théorie des probabilités, comme une grande partie des mathématiques, est redoutable à la physique en tant que source de problèmes et d'intuition pour résoudre certaines formules.

On observe notamment que les marches aléatoires et les circuits électriques entretiennent une relation particulière [3]. Dans notre cas, la solution du problème de Dirichlet sur un graphe arbitraire est donnée par la distribution des potentiels électriques à chaque noeud d'un circuit électrique. Ce dernier est modélisé par plusieurs noeuds reliés par des résistances de valeur l'inverse du poids affecté précédemment (ω). Les conditions limites sont représentées par une source de tension fixant la tension des noeuds labellisés.

• NOTATIONS

Pour commencer l'analyse de cette analogie, nous allons rappeler les différentes variables utilisées et en déclarer de nouvelles :

- n : le nombre de pixels
- m : le nombre de liens
- $(e_i)_{i \in \{1, \dots, m\}}$: les liens entre noeuds
- $(v_i)_{i \in \{1, \dots, n\}}$: les noeuds
- A une matrice de taille $m \times n$ telle que :

$$A_{ij} = \begin{cases} +1 & \text{si } e_i \text{ part de } v_j, \\ -1 & \text{si } e_i \text{ arrive en } v_j, \\ 0 & \text{sinon} \end{cases}$$

- $(\nu_i)_{i \in \{1, \dots, n\}}$: le potentiel à chaque noeuds $(v_i)_{i \in \{1, \dots, n\}}$
- $(u_i)_{i \in \{1, \dots, m\}}$: la tension aux bornes de chaque lien $(e_i)_{i \in \{1, \dots, m\}}$
- $(R_i)_{i \in \{1, \dots, m\}}$: la résistance placée sur chaque lien $(e_i)_{i \in \{1, \dots, m\}}$
- $(d_i)_{i \in \{1, \dots, n\}}$: la somme des $1/R_j$ pour e_j les liens reliés à v_i
- C une matrice de taille $m \times m$ telle que :

$$C_{ij} = \begin{cases} 0 & \text{si } i \neq j, \\ \omega_i \text{ ou } 1/R_i & \text{si } i = j \text{ avec } \omega_i \text{ le poids affecté au lien } e_i, \end{cases}$$

Dans toute la suite, on alternera entre les notations ω_i , qui signifie le poids du $i^{\text{ème}}$ lien, et ω_{i-j} , qui signifie le poids du lien reliant v_i à v_j .

• MODÉLISATION

Ainsi nous pouvons modéliser un circuit électrique nous permettant de retrouver la formule du Théorème 2. En effet, le problème peut être représenté par un circuit électrique composé de n noeuds liés entre eux par m résistances formant une grille (ou encore un quadrillage).

Ensuite, pour retrouver la probabilité que la marche aléatoire partant d'un pixel non marqué atteigne en premier le pixel marqué du label s , on applique une tension de 1 au pixel de label s et une tension de 0 aux pixels marqués d'un autre label.

On observe alors que la tension à chaque noeud (ν) vérifie les mêmes propriétés que la probabilité que la marche aléatoire partant d'un point x atteigne le noeud marqué de label s , v_k , avant les autres noeuds labellisés. En effet, cette probabilité est caractérisée par les trois propriétés suivantes :

- (a) $p(v_k) = 1$
- (b) $p(v_j) = 0$, pour tous v_j marqués de labels différents de s
- (c) $p(v_i) = (\sum_{j=1}^8 p(v_{i_j}) * \omega_{i-i_j}) * \frac{1}{\sum \omega}$
, pour $(v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4}, v_{i_5}, v_{i_6}, v_{i_7}, v_{i_8})$ les points adjacents à v_i .

Or, dans la modélisation de circuit électrique réalisée, le potentiel, ν , en chaque point vérifie :

- (a) $\nu_k = 1$, grâce à la source de tension
- (b) $\nu_j = 0$, pour tous ν_j labellisés autres que s car ils sont reliés à la terre
- (c) $\nu_i = (\sum_{j=1}^8 \nu_{i_j} * \frac{1}{R_{i-i_j}}) * \frac{1}{\sum \frac{1}{R}}$
, pour $(R_{i-i_1}, R_{i-i_2}, R_{i-i_3}, R_{i-i_4}, R_{i-i_5}, R_{i-i_6}, R_{i-i_7}, R_{i-i_8})$ les résistances liant v_i à ses noeuds adjacents, d'après la loi d'Ohm et la loi des courants de Kirchhoff.

Ainsi, en prenant des valeurs de résistances telles que $R_{i-j} = \frac{1}{\omega_{i-j}}$, on obtient bien que le potentiel à chaque noeud vérifie les propriétés de la probabilité décrite précédemment, d'où l'analogie.

• RÉSOLUTION DES ÉQUATIONS RÉGISSANT LE MODÈLE

Ainsi, notre problème est de trouver les probabilités décrites précédemment, associées à chaque label s pour chaque pixel. Il suffit alors de trouver les potentiels en chaque noeud du circuit modélisé précédemment, en appliquant tour à tour une tension de 1 sur un noeud labellisé et de 0 sur les autres noeuds labellisés.

Pour résoudre ce problème on introduit quelques vecteurs et matrices, dans le cas où le label s se voit affecté une tension de 1 et les autres pixels labellisés de 0 (on simplifiera en enlevant l'exposant s sur chaque vecteur) :

- $p = (u_i)_{i \in \{1, \dots, m\}}$: le vecteur des tensions aux bornes de chaque lien
- $f = (f_i)_{i \in \{1, \dots, m\}}$: le vecteur représentant une source de courant
- $b = (u_i)_{i \in \{1, \dots, m\}}$: le vecteur représentant une source de tension
- $z = (i_j)_{j \in \{1, \dots, m\}}$: le vecteur des courants présents sur chaque lien
- $x = (\nu_i)_{i \in \{1, \dots, n\}}$: le vecteur contenant la tension sur chaque noeud, i.e la probabilité cherchée

Nous pouvons ainsi réécrire les trois lois qui régissent la répartition du courant et de la tension dans notre circuit.

La loi des noeuds (Première loi de Kirchhoff) :

$$A^T z = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \begin{pmatrix} e_1 & e_2 & \cdots & e_m \\ +1 & 0 & \cdots & 0 \\ -1 & +1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_m \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

, 0 car pas de source de courant.

La loi des boucles (Deuxième loi de Kirchhoff) :

$$p = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} = Ax + b = \begin{pmatrix} e_1 \\ \vdots \\ e_m \end{pmatrix} \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ +1 & -1 & \cdots & 0 \\ 0 & 0 & \cdots & -1 \end{pmatrix} \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_n \end{pmatrix} + b$$

, b vaut 0 partout sauf aux liens reliant le noeuds labellisé s où il vaut $1/d_i$ avec v_i un noeud adjacent.

La loi d'Ohm :

$$Cp = \begin{pmatrix} 1/R_1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & \cdots & 1/R_m \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} i_1 \\ \vdots \\ i_m \end{pmatrix} = z$$

Ces trois équations régissant le circuit électrique modélisé peuvent se simplifier en une seule et même équation :

$$\begin{cases} A^T z = 0 \\ Cp = z, \\ p = Ax + b \end{cases}$$

$$\Rightarrow \begin{cases} z = CAx + Cb \\ p = Ax + b \end{cases} \Rightarrow \begin{cases} A^T CAx + A^T Cb = 0 \\ Lx = -A^T Cb \end{cases} \Rightarrow Lx = -A^T Cb \quad (1)$$

, car la matrice L définie dans les parties précédentes est égale à $A^T C A$. En effet, c'est une matrice de taille $m \times m$ définie de la manière suivante :

$$L_{ij} = \begin{cases} d_i & \text{si } i = j, \\ -\omega_{ij} & \text{si } v_i \text{ et } v_j \text{ sont adjacents,} \\ 0 & \text{sinon} \end{cases}$$

• ANALOGIE AVEC LA RÉSOLUTION MATHÉMATIQUE DU PROBLÈME

En réduisant nos matrices et vecteurs de l'équation (1) aux noeuds qui n'ont pas été labellisés, on aboutit alors à une équation semblable à $L_u x^s = -B^T m^s$, avec L_u la matrice L réduite aux pixels non marqués, x^s les probabilités que chaque pixel non marqué soit de label s et m^s le vecteur qui vaut 1 pour les pixels marqués s et 0 pour ceux d'un autre label.

Il s'agit de la solution proposée par Léo Grady, équivalente à l'équation présentée dans le Théorème 2. D'après l'analogie faite on retrouve bien le terme de gauche. Pour le terme de droite, on remarque que $B^T m^s$ correspond à la probabilité que la marche aléatoire débutant sur des pixels non marqués atteignent le pixel labellisé s en une étape.

Or, $A^T C b = \begin{pmatrix} 0 \\ \vdots \\ \frac{\omega_{i-j}}{d_j} \\ \vdots \end{pmatrix}$, avec i le noeud labellisé s et j un noeuds adjacent à ce dernier.

Ceci correspond bien à la probabilité que la marche aléatoire partant d'un noeud non labellisé d'atteindre en une étape le noeud labellisé s .

• CONCLUSION DE L'APPROCHE PHYSIQUE

Ainsi, l'analogie précédente nous permet bien de retrouver l'équation régissant cet algorithme de marche aléatoire. L'expérience physique à réaliser pour retrouver la segmentation finale est donc la suivante :

- (1) Appliquer tour à tour sur chaque noeud labellisé une tension de 1 et de 0 sur les autres noeuds labellisés.
- (2) Mesurer à chaque fois le potentiel sur chaque noeud non labellisé.
- (3) Attribuer un label à chaque noeud non labellisés : pour un noeud donné qui a atteint

un potentiel maximal, lorsqu'on a appliqué une tension de 1 au noeud de label s , celui-ci sera aussi classifié comme de label s .

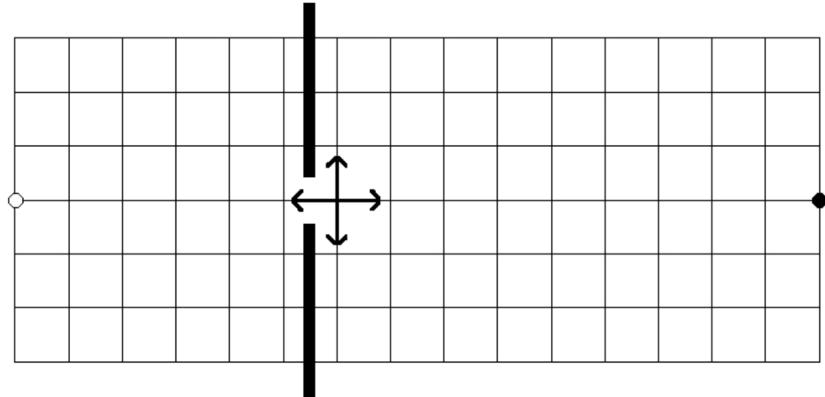


FIGURE 1 – Schema d'une "Weak Boundaries"

Cette analogie nous permet de mieux comprendre certaines propriétés de l'algorithme. Par exemple, dans le cas des "boundary weakness", à savoir un trou dans une frontière, l'algorithme de marche aléatoire est plutôt précis et parvient bien à segmenter malgré les trous dans la frontière. Pour traverser la frontière, il faut passer par ce trou de résistance faible (poids élevé) car les pixels du trou sont identiques à ceux des deux côtés de la frontière comme en témoigne l'illustration ci-dessus. Cependant, les courants de tous les noeuds adjacents à ce trou doivent alors traverser le trou.

Or la loi d'Ohm s'écrit $U_{sortie} - U_{entrée} = RI$. Bien que R soit faible, I est très élevé (loi des noeuds), donc la différence de potentiel est élevée. Ceci explique que les noeuds de part et d'autre de la faiblesse de la frontière soient labellisés différemment.

2

L'ALGORITHME DE RANDOM WALKER

2.1 CODE PRINCIPAL

Notre algorithme de labellisation est structuré en trois parties, et est disponible sur notre *repository GitHub* (https://github.com/lucasgascon/ea_research.git) :

1. Labellisation : pour se faire, on télécharge l'image, l'affiche dans une interface de jeu "pygame" comme image de fond, l'utilisateur peut alors cliquer sur des zones de l'image pour les labelliser. L'utilisateur peut choisir la "taille" de la labellisation en définissant le nombre de pixels qu'il souhaite labelliser autour de chaque clic. Nous ne développerons pas plus cet algorithme ici.
2. Application de l'algorithme random walker.
3. Affichage : dernière partie consiste à afficher les résultats en créant une nouvelle image segmentée en fonction des résultats de notre algorithme, et affichant l'image grâce à OpenCV.

Intéressons nous plus en détail à la fonction principale de notre algorithme. Nous proposons ci-dessous le code utilisé pour la fonction principale. Nous avons commenté au maximum notre algorithme et invitons les lecteurs curieux à le consulter sur le *repository GitHub* pour plus de détails, notamment sur toutes les fonctions appelées dans cet algorithme.

```

def random_walk(img, matrice_label, beta):

    # On construit la matrice L
    L = get_matrice_L(img/255, beta)
    print("Matrice L generee")

    # On permute cette matrice pour placer les vecteurs labellises devant
    vector_labellise = matrix_to_vector(matrice_label)
    perm, vectorlabellise_ordonne = get_permutation(vector_labellise)
    L = permuteL(L, perm)
    print("Permutation effectuee")

    # On extrait les matrices necessaires a la resolution de l'équation
    Lu, B, M = getMatricesToSolve(L, vectorlabellise_ordonne)
    print("Extraction realisée")

    # On resout le système pour obtenir la matrice des probabilités d'appartenir
    # a chaque label
    xu = solve(Lu, B, M)
    print("Résolution du système ok")
  
```

```

# On reorganise les résultats en transformant les probabilités en
# labellisation et reorganise les résultats dans l'ordre
x = transformEnLabel(M,xu)
print("Analyse des résultats effectuée")
x = permutationVecteur(x, perm[::-1])
print("Résultats reordonnés")
imgLabel = resize_vector_to_matrix(x, img)
print("labellisation des données terminées")

# On renvoie un array de la taille de l'image à labelliser où toutes les
# valeurs sont le label auquel les points appartiennent probablement
return imgLabel

```

2.2 LES RÉSULTATS DE NOTRE ALGORITHME

Une bonne nouvelle est que notre algorithme fonctionne. Cependant, le temps de calcul est considérablement plus faible sur des images de petites tailles (explication dans la partie suivante). Pour cette raison, nous présentons quelques résultats obtenus sur des petites images.

Nous avons commencé par tester notre algorithme sur des images simples, avec un nombre très faible de couleurs (deux ou trois maximum).

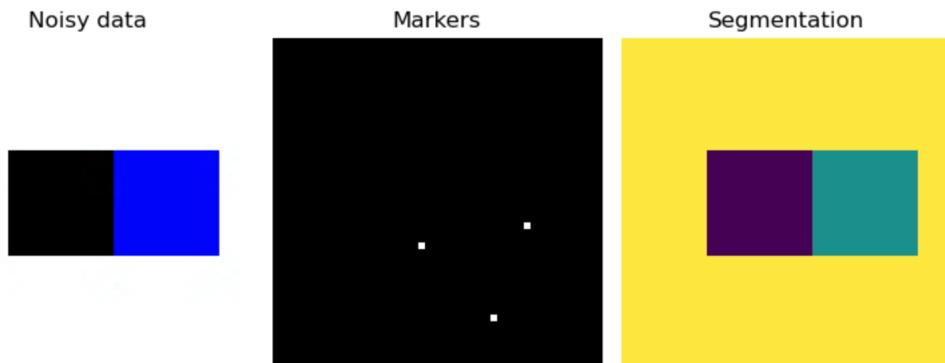


FIGURE 2 – Segmentation d'une première image simple

En adaptant la valeur du beta ($\beta=10$), avec seulement trois pixels labellisés (onglet "Markers"), un pour chaque label ; on arrive à segmenter l'image de gauche avec les trois labels parfaitement identifiés.

On essaie ensuite de labelliser une image binaire, pour observer la capacité d'une frontière à contenir les marches aléatoires.

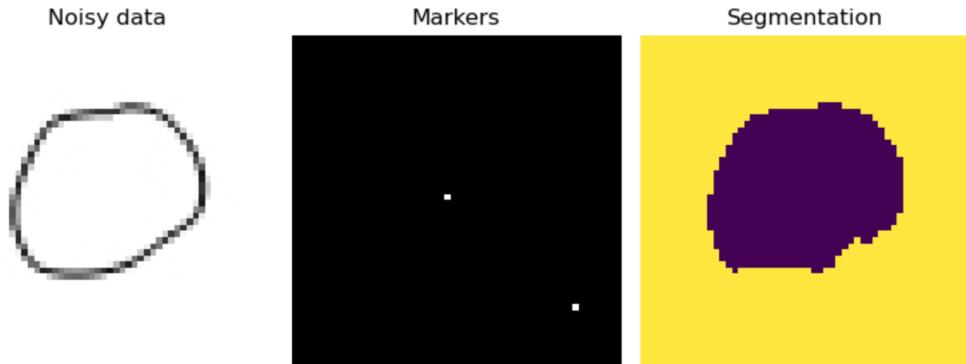


FIGURE 3 – Segmentation d'une image binaire

Là encore, notre algorithme fonctionne (après l'ajustement $\beta=100$). La segmentation arrive à bien rester contenue dans le cercle blanc, avec encore une fois deux pixels labellisés, un pour chaque label.

Enfin, nous avons essayé de labelliser une image plus complexe en deux labels distincts. Pour cela, nous avons labellisé plus de pixels car les différences de couleur au sein d'un même label étaient plus prononcées. Avec une segmentation adéquate, nous arrivons au résultat très satisfaisant ci-dessous.



FIGURE 4 – Segmentation d'une image médicale

• OPTIMISATIONS ET LIMITES DE NOTRE ALGORITHME

Après des premiers tests, la segmentation de notre algorithme s'est avérée concluante. Cependant, pour des images trop grandes, notre ordinateur n'était pas en mesure d'effectuer la segmentation dans un délai satisfaisant. Nous avons identifié le problème, qui venait de la permutation de la matrice L . En effet, si on segmente une image avec $n = w \times h$ pixels, cette matrice est une matrice de taille $n \times n$. Notre fonction "permuteL(L, perm)" génère les matrices

de permutation nécessaires à la permutation de L , puis effectue la permutation $L' = PLP^T$ (car L est symétrique et donc diagonalisable en base orthonormée. Ce calcul consiste donc à deux multiplications de matrices de taille $n \times n$. Pour donner un ordre d'idée, pour une image de taille 720×480 , on multiplie deux matrices de taille 345600×345600 , soit 4×345600^3 opérations. Après réflexions, nous avons repensé notre algorithme, et une solution pour éviter cette inversion aurait été de travailler avec des graphes et non pas des matrices. En effet, comme la plupart des coefficients de notre matrice L sont nuls, la permutation s'effectuerait en un temps considérablement plus faible. Nous avons cependant fait le choix de conserver notre algorithme qui fonctionne sur des petites images, pour nous concentrer sur la comparaison entre l'algorithme implémenté sur skimage et le modèle de SegNet.

2.3 COMPRÉHENSION PRATIQUE DE L'ALGORITHME

L'algorithme de random walker dépend principalement de deux paramètres choisis par l'utilisateur :

1. Le choix du paramètre beta
2. La labellisation des données

L'objectif de cette partie est de comprendre comment optimiser le choix de ces paramètres.

• CHOIX DU PARAMÈTRE BETA

Commençons par comprendre l'importance du choix du paramètre β . Cette variable intervient à un unique moment : définir les probabilités de passage d'un voisin à un autre.

On rappelle la fonction de poids entre deux pixels d'intensités i et j voisins :

$$w_{ij}(\beta) = \exp(-\beta||i - j||^2)$$

Pour un pixel i , la probabilité qu'une marche aléatoire commençant à ce pixel au temps 0 atteigne le voisin j au temps 0 est :

$$p_{i,j} = \frac{w_{ij}}{\sum_{k \in V_i} w_{ik}}$$

Un première remarque que l'on peut faire est que pour un certain pixel, le pixel le plus probable d'être atteint est celui qui a la différence d'intensité avec le pixel i la plus faible. Plus précisément, cette probabilité croît avec β , et tend vers 1 lorsque β tend vers l'infini (et qu'aucun autre pixel voisin de i n'a la même différence d'intensité entre les deux pixels).

Autrement dit, un β élevé favorisera les chemins à faible différence d'intensité sur l'image, ce qui aura pour effet de rendre les contours des images plus droits, moins arrondis.

Afin de confirmer ces analyses, nous avons segmenté plusieurs images avec notre algorithme. Ce sont des images que nous avons nous-mêmes prises, que nous avons ensuite segmentées en créant un masque sur Photoshop. Nous avons ensuite souhaité utiliser une métrique pour

analyser nos résultats : nous avons opté pour le pourcentage de pixels mal positionnés sur une image. Notre objectif est donc de minimiser cette fonction.

Cette image de chat illustre parfaitement la polarisation des résultats lorsque β augmente.

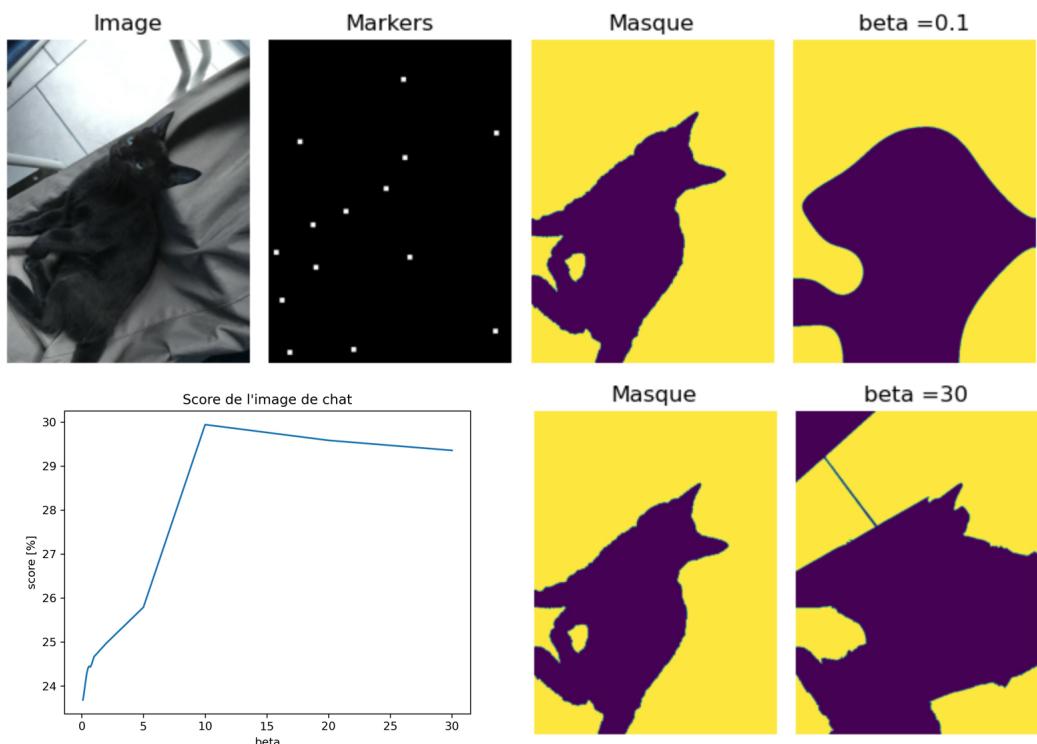


FIGURE 5 – Segmentation d'une image à faible contraste

En effet, lorsque β est assez grand, l'algorithme trouve un chemin dans les plinthes du carrelage, ce qui fausse la segmentation.

Notre première conclusion est la suivante : pour les images avec un faible contraste, privilégier un β petit permettra d'obtenir une segmentation plus approximative, mais qui conservera l'idée principale de la forme que l'on cherche à segmenter.

Nous avons ensuite essayé de segmenter des images avec un contraste plus prononcé, pour observer les réponses de notre algorithme à ce genre d'images. Pour cela nous avons pris en photo une poubelle, un carton, et une table dans une salle. Voici les résultats obtenus.

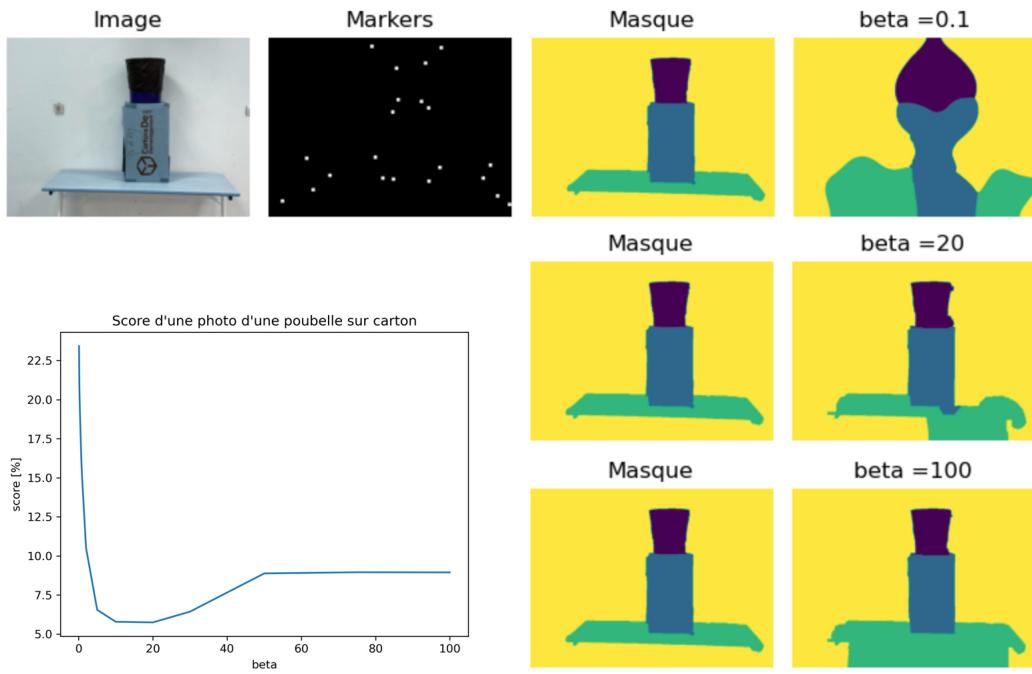


FIGURE 6 – Segmentation d'une poubelle sur un carton

On confirme encore une fois la densification des segments droits à mesure que β augmente. Le score obtenu a retenu notre attention. Si ce score est minimal pour $\beta = 20$, à l'oeil nu la segmentation obtenue pour $\beta = 100$ semble beaucoup plus satisfaisante et naturelle. Après discussion avec notre tutrice Stéphanie Allassonnière, nous avons conclu que la minimisation de notre fonction score ne devait pas prendre le pas sur une analyse visuelle cohérente de l'image. Par ailleurs, en regardant la position des markers utilisés pour labelliser l'image avant utilisation de l'algorithme, on peut déjà prédire que le passage des barreaux de la table va être impossible avec un β trop grand, car les chemins passant de la table au bord de l'image en empruntant ces barreaux seront largement privilégiés. Ces barreaux deviennent alors des obstacles infranchissables nous empêchant d'atteindre la partie du mur située entre les deux barreaux (pour une marche aléatoire commençant sur le mur à l'extérieur de ces barreaux). Ceci revêt l'importance de la labellisation et illustre une méthode permettant de maximiser le résultat lors de la labellisation.

• LABELLISER INTELLIGEMMENT LES DONNÉES

Tout d'abord, il est important de noter que plus le nombre de pixels labellisés augmente, plus notre algorithme sera efficace. Cependant, l'objectif étant de labelliser automatiquement les données, il ne faut pas non plus créer le masque de labellisation à la main et labellisant toute l'image. Comment trouver le juste milieu ?

Tout d'abord, nous conseillons de labelliser les extrémités des formes à segmenter, et insister

en particulier sur les zones à faible niveau de contraste et les points aux extrémités des formes (le coin d'une table par exemple).

Ensuite, en fonction du β choisi, on peut identifier les éventuelles barrières qui seraient compliquées à franchir, et labelliser sur ces barrières, et autour d'elles.

En reprenant la même image que précédemment, et en ajoutant 3 points à la labellisation réalisées, voici les résultats que nous obtenons :

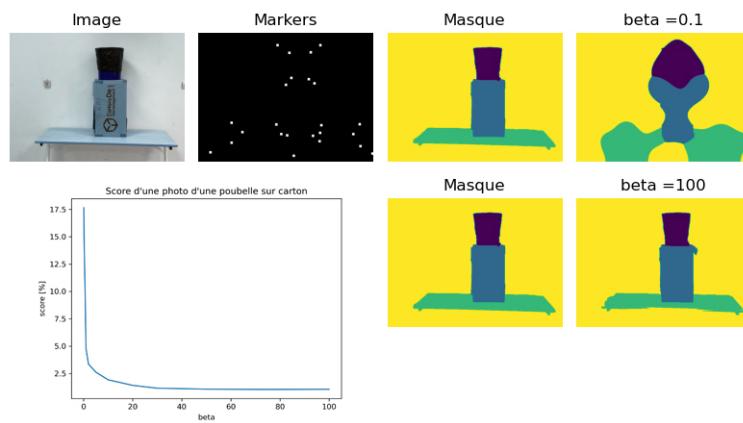


FIGURE 7 – Segmentation d'une poubelle sur un carton avec une meilleure segmentation

On observe qu'avec cette segmentation, l'algorithme fonctionne presque parfaitement avec près d'1% des pixels mals segmentés seulement.

3

ANALYSE COMPARATIVE : SEGNET ET RANDOM WALKER

3.1 SEGNET

- PRÉSENTATION DE L'APPROCHE

SegNet est un réseau de neurones convolutifs profonds pour la segmentation sémantique par pixel. Ce réseau a la même topologie que VGG16, à savoir le même nombre de couches cachées et de neurones dans chacune de ses couches cachées. Cependant dans SegNet les "couches entièrement connectées" (*fully connected layers*) ont été retirées ce qui rend ce modèle beaucoup plus facile à entraîner que les autres architectures récentes.

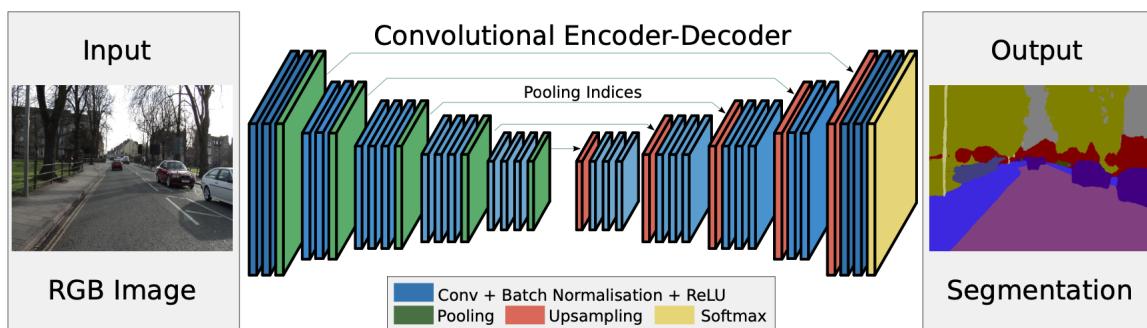


FIGURE 8 – Illustration de l'architecture de SegNet[4]

A défaut de trouver une version pré-entraînée de SegNet sur internet nous avons essayé de ré-entraîner ce modèle nous-même. Notre objectif était de pré-entraîner SegNet sur une base de données d'images relativement générale, afin d'avoir une base intéressante pour notre modèle, et être en mesure de faire de l'apprentissage par transfert (*transfer learning*) sur un faible nombre d'images et d'être ainsi capable d'obtenir un modèle fiable avec peu de données de notre problème spécifique. Nous avons alors confronté des difficultés. En effet, dans le *repository* [5] au sein duquel nous avons récupéré le modèle SegNet non-entraîné avec différentes (*backbones*) (SegNet50, VGG16, vanilla), étaient proposées des versions pré-entraînées de PSPNet (un autre modèle d'apprentissage profond ayant une fonction similaire à SegNet) sur la base de données ADE20K qui contient plus de 20.000 images de scènes. Pour entraîner SegNet sur cette base de données chaque cycle de traitement de l'ensemble des données d'apprentissage (*epoch*) prenait plus de 5h sur nos machines ce que l'on a considéré trop long. Ainsi nous avons pris la décision, en collaboration avec Stéphanie Allassonnière, d'utiliser également la version pré-entraînée de PSPNet dans notre étude quand nous souhaitions disposer d'un modèle déjà pré-entraîné.

D'autre part, l'une des limites de ces modèles pour la généralisation à d'autres types d'images que celles d'entraînement est que dès l'entraînement, des caractéristiques de la prédiction sont fixées : la taille de l'image de sortie, ainsi que le nombre de labels possibles.

• ENTRAÎNEMENT DE SEGNET SUR DES DONNÉES DE SANTÉ

Nous avons essayé d'entraîner sur des images de santé une version de SegNet avec ResNet50 comme encodeur. Notre approche a consisté à entraîner ce modèle dont les poids sont initialement aléatoires (car non pré-entraîné) sur un petit nombre d'images de lésions et de voir la qualité des prédictions sur des images similaires. Pour chacune des images ci-dessous SegNet a été entraîné sur 10 *epochs*, et en ne cherchant à prédire que deux labels, l'objectif étant de différencier la lésion de l'arrière-plan. Les images sont issues du ISIC Challenge Dataset 2016. Les remarques faites sur les prédictions ci-dessous seront essentiellement qualitatives. En effet, les résultats que l'on obtenait en utilisant une métrique qui calcule la part de pixels bien labellisés n'étaient pas intéressants. On justifie cela par l'importance relative que l'on apporte à la bonne labellisation des pixels : si l'on labellise parfaitement la lésion et l'arrière-plan proche mais qu'il y a beaucoup de pixels mal labellisés sur les bords de l'image, on considère le résultat plus satisfaisant que si on se trompe sur la forme de la lésion, bien que cela puisse conduire à des métriques similaires.

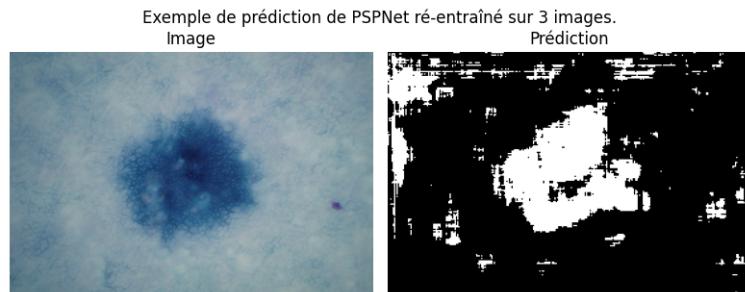


FIGURE 9 – Prédiction en entraînant le modèle sur 3 images

La figure 6 montre que dès une base de données d'apprentissage de 3 images, ce qui peut parfois être la seule chose dont l'on dispose pour des cas spécifiques en santé, on a une prédiction relativement précise de la forme de la lésion si celle-ci a une couleur bien différentiable de l'arrière-plan. Cependant il y a beaucoup d'erreurs de labellisation sur les bords de l'image, ce qui sera également le cas même si la base de données d'apprentissage est de plus grande taille. Un opérateur, ou un expert de la santé pour des tâches plus complexes, peut alors apporter son expertise pour corriger les erreurs de labélisation.

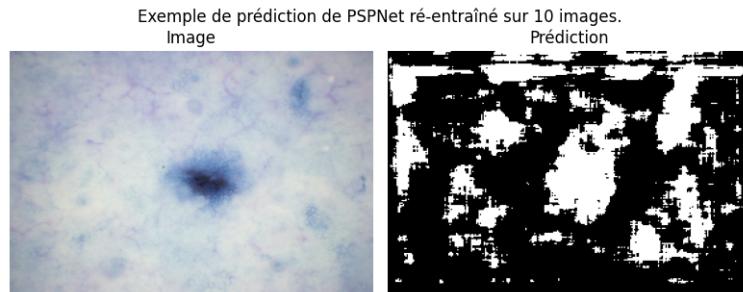


FIGURE 10 – Prédiction en entraînant le modèle sur 10 images

La figure 7 présente une lésion qui est très mal segmentée. La cause de cette mauvaise segmentation réside potentiellement dans les veines qui sont très marquées sur la photo et qui sont de couleurs similaires à la lésion, induisant le modèle en erreur. Cette segmentation est inexploitable.

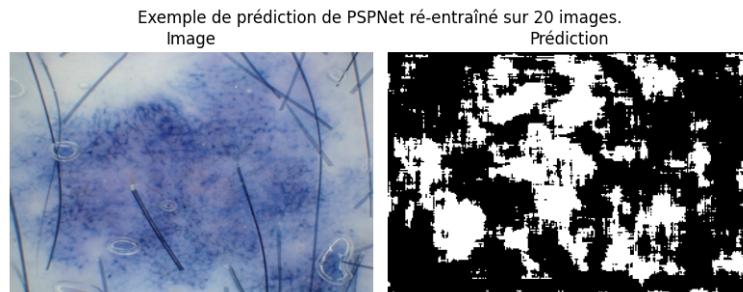


FIGURE 11 – Prédiction en entraînant le modèle sur 20 images

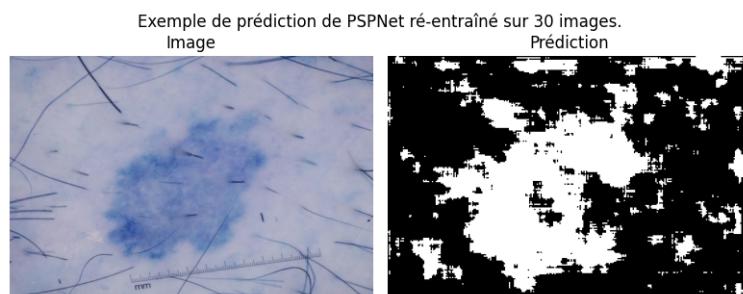


FIGURE 12 – Prédiction en entraînant le modèle sur 30 images

On remarque dans les figures 9 et 10 que les poils détériorent la segmentation, mais que l'impact est de moindre importance si ces derniers sont sur la lésion que si ils sont sur le côté de la lésion. En effet, comme expliqué précédemment, le plus important est de pouvoir déterminer la forme de la lésion avec la prédiction du modèle, que les bords de l'image soit bruité par l'erreur causée par la pilosité nous importe peu.

Finalement une expertise métier est souvent nécessaire en plus du modèle, bien que celui-ci obtienne des résultats très satisfaisants sur des images simples, proches en caractéristiques de la base de données d'apprentissage. Il est clair que ce modèle, entraîné en tout et pour tout sur au plus 30 images, ne permet pas de prédire la segmentation d'autre chose que des lésions dont l'image est similaire à celles de la base de données d'apprentissage. Cependant on remarque qu'avec très peu de données d'entraînement et un modèle qui n'est pas pré-entraîné on peut déjà avoir des résultats satisfaisants. La taille de la base de données d'apprentissage ainsi que le travail de traitement de l'image de segmentation prédictive dépendront de la précision attendue.

3.2 RÉPONSES AUX TRANSFORMATIONS DES DEUX MODÈLES

- PRÉSENTATION DE L'APPROCHE

Une fois le réseau de neurones convolutifs profonds entraînés, nous avons pu comparer les segmentations obtenues sur des images bruitées ou transformées.

Tout d'abord, nous avons comparé le résultat des deux algorithmes sur des photos que nous avons nous même prises, en utilisant le PSPNet pré-entraîné sur une large base de donnée (ADE20K). Le paramètre β est choisi égal à 10 pendant toute l'expérience. Par ailleurs, afin de mesurer les performances des segmentations en plus de l'aspect visuel, nous avons implémenter deux métriques :

– Pour l'algorithme de marche aléatoire : nous comptons le nombre de pixels mals attribués et divisons cette somme par le nombre total de pixels, le tout en pourcentage. Le score va donc de 100% à 0% de la pire segmentation à la meilleure possible.

– Pour l'autre modèle : nous avons implémenté une métrique semblable qui calcule le rapport de pixels bien classés par label sur le nombre total de pixels d'un label, pondéré par le nombre de pixels de l'autre label, puis nous prenons 1 moins cette valeur pour mesurer la proportion de pixels mal attribués comme pour la métrique précédente. Cette métrique a été utilisée car dans le cas de gros déséquilibres entre deux labels sans la pondération on obtenait de bons scores quand la segmentation affichait un seul label, ce qui ne nous convient pas. Ainsi, le score va de 0% (segmentation parfaite) à 100% (une segmentation qui inverse les labels).

```

cl_wise_score = tp / (tp + fn + 0.000000000001)
# avec tp et fn des vecteurs du même nombre de dimension que le nombre de
classes
n_pixels_norm = n_pixels / np.sum(n_pixels)
#nombre de pixels par classe réelle
norm_net = 1 - (cl_wise_score[0]*n_pixels_norm[1] + cl_wise_score[1]*
n_pixels_norm[0])
# on normalise pour reéquilibrer les deux classes

```

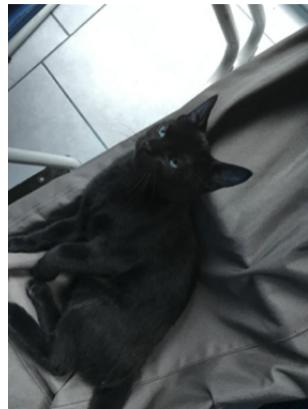


FIGURE 13 – Photo initiale du chat

Par exemple, sur l'image de chat [13] on observe que le faible contraste rend la segmentation difficile pour l'algorithme de marche aléatoire. Cependant, étant donné que le réseau de neurones n'est pas entraîné sur un jeu de données similaires à la photo du chat, le résultat est très loin de la segmentation attendue. Il prédit bien deux labels comme demandé en paramètre mais ne parvient pas à segmenter l'image de chats.

C'est pourquoi, nous allons d'abord étudier en profondeur les performances de l'algorithme de marche aléatoire face aux différents bruits et aux transformations puis on comparera ces propriétés avec celles du SegNet sur des images sur lesquelles il sera entraîné (images de lésions cutanées comme précédemment).



FIGURE 14 – Segmentation par les deux algorithmes d'une photo de chat

• PROPRIÉTÉS DE L'ALGORITHME DE RANDOM WALKER POUR LA SEGMENTATION D'IMAGES BRUITÉES ET TRANSFORMÉES

Sur les images suivantes on observera à chaque fois de gauche à droite : l'image transformée, les points labellisés, le masque et l'image segmentée. Nous avons testé notre algorithme de marche aléatoire sur des images bruitées.

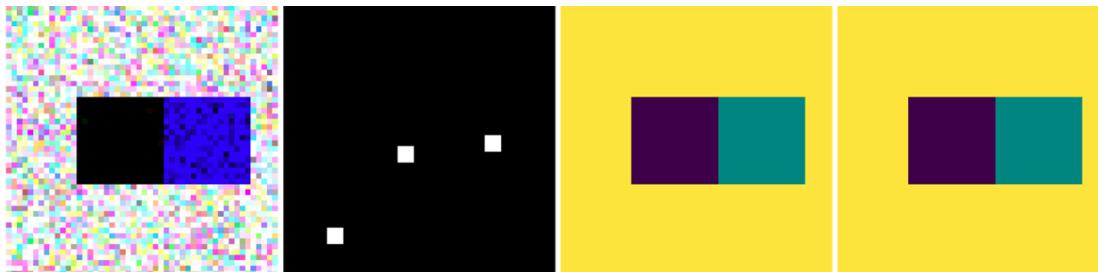


FIGURE 15 – Segmentation sur une image de deux carrées avec un bruit de type Poisson

Ici, on observe que le bruit ajouté n'a eu aucune influence sur le résultat de la segmentation. Le score de la segmentation est de 1.43%.

En revanche, l'image bruitée par impulsion aléatoirement distribuée, c'est à dire des pixels aléatoires changés par un pixel d'intensité aléatoire. On remarque que des pixels se trouvant à la frontière entre deux labels peuvent induire en erreur l'algorithme et produire une segmentation avec des pixels mal attribués comme on peut le voir ci-dessous.

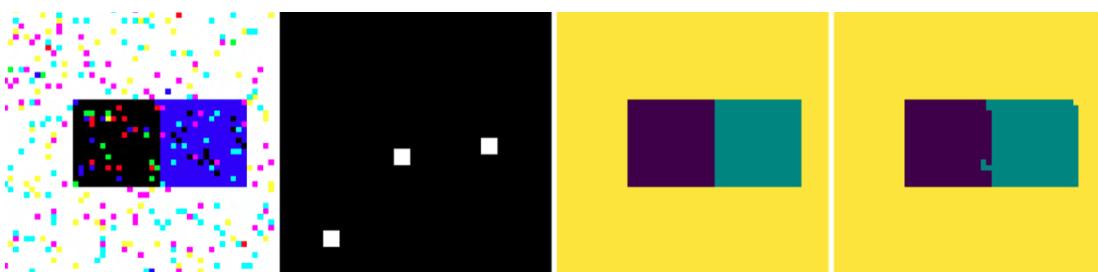


FIGURE 16 – Segmentation sur une image de deux carrées avec un bruit de type impulsion aléatoire

Une autre propriété de l'algorithme de random walker a aussi été étudiée, celle de la résistance aux *weak boundaries*. Ainsi, nous avons testé l'algorithme sur un contour de la France "troué" auquel nous avons ajouté un bruit gaussien. Comme attendu lors de la démonstration par l'approche physique, l'écart en intensité produit par la frontière (même trouée) permet de bien labelliser. En effet, la métrique nous donne 2.03% et le résultat visuel est très convaincant dans la mesure où le trou sur la frontière de gauche est conséquent.



FIGURE 17 – Segmentation sur une image du contour de la France troué et avec l'ajout d'un bruit gaussien

- **COMPARAISON DES DEUX ALGORITHMES FACE AUX BRUITS ET AUX TRANSFORMATIONS**

À présent, nous comparons les résultats du SegNet, entraîné avec 10 *epochs* sur des images de lésions, et du random walker sur deux images de lésions (autres que celles sur lequel l'algorithme s'est entraîné). Nous avons ainsi deux classes : la lésion et le reste.

Sur les images suivantes on observera à chaque fois de gauche à droite : l'image transformée, les points labellisés, la segmentation par le random walker et l'image segmentée par SegNet. Nous avons testé notre algorithme de marche aléatoire sur des images bruitées.

Tout d'abord, nous observons les résultats sans aucune modification des images. Le réseau de neurones est plus précis malgré les nombreux points marqués avec l'algorithme de random walker pour la première lésion. Les métriques pour le modèle SegNet sont ici : 1% pour la première lésion et 25% pour le second. À l'inverse, pour le random walker il y a 7,2% de pixels mal labellisés et 2.5% pour la deuxième image.



FIGURE 18 – Segmentations initiales sur les deux images de lésions

Ainsi, les résultats du modèle SegNet sont très précis pour la première image et bien moins pour la seconde. Le défaut de la segmentation pour la deuxième image est qu'il n'est pas parvenu à délimiter la bordure inférieure de la lésion. En effet, le masque de l'image fait figurer un trou dans la lésion au niveau de la bordure inférieure à gauche de la lésion. Malgré le score plus faible du SegNet sur cette image, la segmentation réalisée par le random walker nous convient

moins. Elle trace des limites droites, moins souples que le SegNet. Les différences de traitements dans l'affichage des images pour les deux modèles donnent des métriques très différentes pour les deux algorithmes, c'est pourquoi nous comparons les métriques de chaque modèle pour les images bruitées aux métriques pour les images sans bruits [18].

Nous remarquons que globalement le SegNet résiste mieux que le random walker aux bruits qui ne modifient pas fondamentalement l'image, c'est-à-dire que pour l'oeil humain la lésion est encore visible et segmentable.

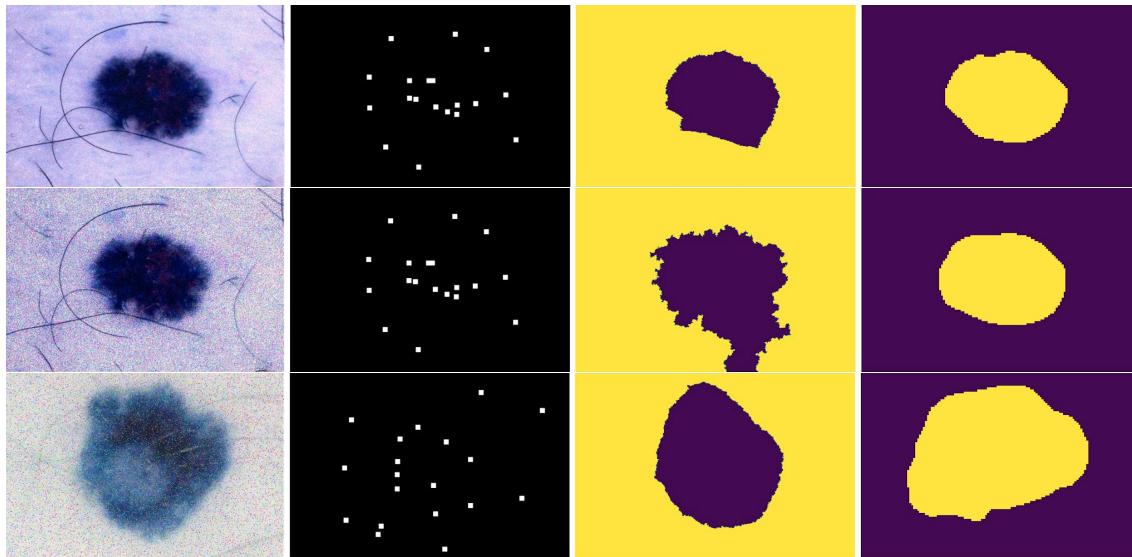


FIGURE 19 – Segmentations d’images avec l’ajout de bruits gaussien, de poisson et par impulsion, de haut en bas.

Les métriques appuient les conclusions que l’on peut faire seulement avec l’aspect visuel. SegNet est plus performant dans la mesure où la métrique évolue peu par rapport à la segmentation sans bruit. En revanche, pour le bruit par impulsion le SegNet produit une segmentation aux bordures souples mais bien loin de la forme de la lésion. Dans ce cas, l’utilisation du random walker sera alors préférable.

Random Walker	SegNet
2% (-20%)	25% (+0%)
10% (+300%)	10% (+60%)
7% (-1%)	35% (+3400%)

Ensuite, on remarque que lorsque le bruit est encore plus visible et que la lésion devient difficile à apercevoir, le random walker continue de donner une segmentation sensée quand le SegNet ne parvient plus à segmenter. Cette propriété du random walker peut s'avérer très intéressante dans la mesure où il parvient à détecter des formes même quand l'image est trop bruitée et que l'oeil humain aurait du mal à détecter cette même forme. Ce type de bruits, ici speckle, provient d'interférences ondulatoires et se trouve notamment dans l'imagerie médicale

par ultrason, affirmant à nouveau l'intérêt du random walker pour la segmentation d'images médicales.



FIGURE 20 – Segmentations d'images avec l'ajout de bruits de type "speckle".

Enfin, nous nous sommes intéressés à un dernier type de transformations : l'insertion de partie manquante dans les images. Nous avons réalisés cela en ajoutant une ellipse ou un rectangle sur un côté de l'image afin de voir la réaction de nos deux algorithmes. La problématique des points marqués entre alors en jeu. En effet, si l'endroit où figure la forme est labellisé comme appartenant à la lésion alors la segmentation l'inclura dans la lésion. À l'inverse, si aucun point de la forme n'est labellisé le résultat sera différent.

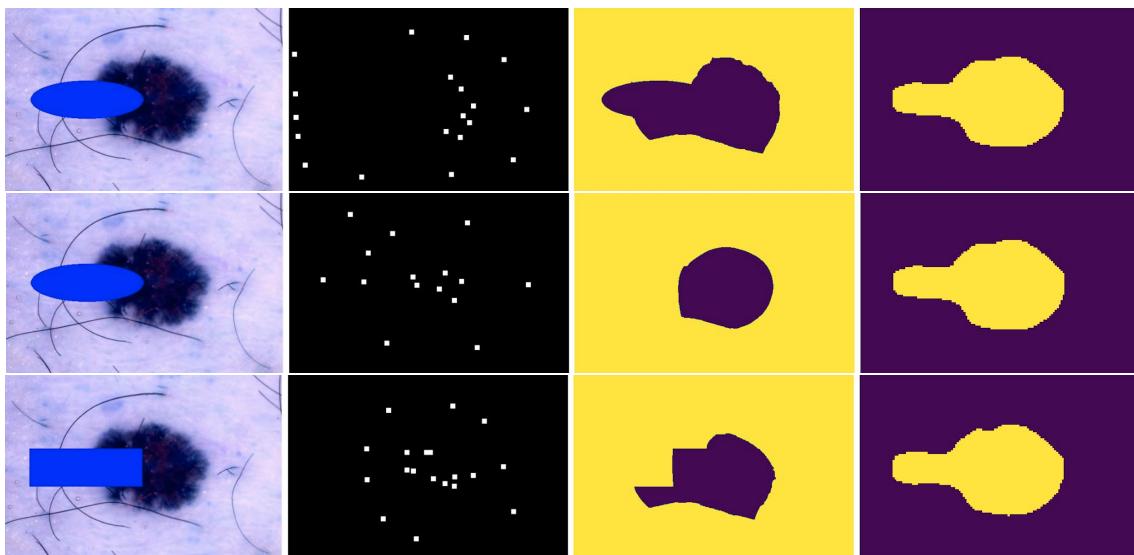


FIGURE 21 – Segmentations d'images avec insertion de formes avec, de haut en bas, la forme : non marquée, marquée comme n'appartenant pas à la lésion et marquée comme appartenant à la lésion.

Tout d'abord, remarquons que le SegNet ne parvient pas à ignorer la forme qui obstrue l'image ainsi il renvoie une segmentation similaire pour les deux formes à savoir : la lésion plus la forme.

Concentrons nous à présent sur les différentes segmentations du random walker. On observe ici sur la première image que lorsque la forme n'est pas marquée, elle est attribuée à la lésion plutôt qu'au reste de l'image. Cette attribution semble plutôt arbitraire dans la mesure où ici la forme est de couleur bleue mais si elle avait été d'une autre couleur, elle aurait pu être attribuée

au reste de l'image. Ensuite, nous avons marqué la forme comme n'appartenant pas à la lésion. On observe alors que la lésion est mieux segmentée, ignorant la forme qui obstrue. Ainsi en labellisant intelligemment l'image initiale l'algorithme pourra parvenir à segmenter la lésion malgré l'obstruction. La labellisation nécessite alors encore un expert dans le domaine mais requiert moins de temps qu'une annotation entière pour entraîner un modèle d'apprentissage profond. En revanche, lorsque la forme est marquée comme appartenant à la lésion tel que sur la troisième ligne d'images, la segmentation représente la lésion plus la forme comme pour le premier cas.

- **DISCUSSION AUTOUR DES DIFFÉRENCES DE MISE EN PLACE DES DEUX MODÈLES**

Une des principales différences entre nos deux algorithmes est la prise en main des deux modèles pour réaliser la segmentation. Le SegNet nécessite une grande base de données pour pouvoir être entraîné et ensuite réaliser des segmentations sur des images. Si les images qu'on souhaite segmenter s'éloigne trop de celles du base de données d'apprentissage, le modèle aura du mal à segmenter l'image et il y a peu de paramètres que l'on pourra modifier pour essayer d'améliorer cette segmentation. Ainsi, dans le cadre d'images de mauvaise qualité ou d'images différentes de celles d'entraînement les résultats ne conviendront pas forcément. En revanche, l'algorithme de random walker ne nécessite pas d'entraînement ce qui permet bien de segmenter différents types d'images. De plus, la labellisation initiale peut permettre de surpasser certaines difficultés comme les parties manquantes ou le bruit. Ainsi, suivant les images, l'expert sera en mesure d'influer sur la segmentation du modèle pour que celle-ci corresponde bien aux attentes. Pour le réseau de neurones, une telle manipulation sera plus compliquée. De plus, un autre paramètre joue un rôle important dans la segmentation par le random walker : le paramètre β .

CONCLUSION

Nous avons donc étudié l'algorithme de random walker sous trois prismes différents. Nous avons commencé par comprendre mathématiquement l'algorithme développé par Léo Grady. Nous avons donc développé trois différentes méthodes permettant de prouver mathématiquement que la solution du random walker est analogue à la résolution d'un système linéaire. Nous avons redémontré l'analogie avec un problème de Dirichlet mentionnée dans le papier. Enfin, nous avons analysé les analogies du problème avec des circuits électriques. Cette dernière analyse nous a permis d'acquérir une compréhension approfondie de la propriété de *weak boundaries*, et de manière générale ce travail théorique nous a permis de comprendre en profondeur l'algorithme.

Dans un second temps, nous avons recodé l'algorithme de random walker. Cette partie nous a conduit à passer du temps à comprendre chaque détail de l'algorithme, comme la permutation qui permet de placer les pixels labélisés par l'utilisateur en premières positions. Nous avons rencontré quelques difficultés liées au temps d'exécution de notre code, mais nous avons réussi à produire un algorithme fonctionnel. Nous nous sommes ensuite attardés sur les hyperparamètres, à savoir le choix du paramètre β en fonction de l'image à segmenter, ainsi que la technique qui nous a semblé la plus efficace pour labelliser les données. Nous avons déterminé certains points pour optimiser les résultats fournis par l'algorithme en fonction de la segmentation attendue par l'utilisateur. Nous avons pu comprendre ces résultats principalement empiriques grâce au travail théorique précédemment réalisé.

Enfin, dans un troisième temps, nous avons comparé l'algorithme de random walker à un modèle d'apprentissage profond : SegNet. Pour que la comparaison soit pertinente, nous avons souhaité restreindre notre étude à des données de santé. Dans un premier temps, nous avons entraîné un modèle avec peu d'images, car en santé il est souvent difficile de rassembler un grand nombre d'exemples pour l'entraînement des modèles. Dans ce cadre là, le modèle SegNet ne fournit aucun résultat satisfaisant en renvoyant des segmentations très bruitées. Pour cette raison nous avons ensuite comparé les résultats des deux modèles sur des images sur lesquelles SegNet a été entraîné. Nous observons alors des bons résultats de la part des deux algorithmes. Nous souhaitons alors comparer ces résultats sur des images transformées : ajout de bruit et de formes sur l'image. Il apparaît alors que l'algorithme de random walker est plus stable relativement aux bruitages que SegNet. De plus, la labellisation et le paramètre β permettent à l'algorithme de random walker d'être adaptable en fonction des types de transformations.

Cet enseignement nous a permis de découvrir la recherche en machine learning. Nous avons eu l'occasion d'étudier un algorithme déterministe, ce qui est particulièrement intéressant pour des problématiques de santé où les données d'apprentissage sont rares. Le fil conducteur de ce projet proposé par Stéphanie Allassonnière était de réaliser que les méthodes d'apprentissage ne sont pas inéluctablement les plus efficaces, ce que nous avons pu remarquer, à tel point que nous avons proposé une méthode sans apprentissage au sein d'un autre projet.

RÉFÉRENCES

- [1] L. Grady. Random walk for image segmentation. *Conception et évaluation d'un processus de personnalisation fondé sur des référentiels de compétences*, 2006.
- [2] T. Pierron. Chaines de markov et martingales. *ENS Ker Lann*, 2019.
- [3] P. Doyle and L. Snell. Random walks and electric networks. *Carus mathematical monographs*, 1984.
- [4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet : A deep convolutional encoder-decoder architecture for image segmentation. *Senior Member, IEEE*, 2016.
- [5] Divam Gupta. Repository github : image-segmentation-keras, 2019.