RAPPORT DE STAGE
« Etude Diachronique, Trait de côte et fissuration »

Stagiaire : Mehdi ElMouatadil, Etudiant GM3
Maître de Stage : Mr. Cyrille Fauchard, Chef de l'équipe ENDSUM

Réalisé au Cerema Normandie-Centre,

17 Juin 2019 - 23 Aout 2019

Table des matières

1 Introduction

Une méthodologie de détection et de détermination automatique des fractures à partir de données topo-bathymétriques fournies par le Réseau d'Observation du Littoral (ROL) sur les platiers rocheux au Nord de la France a été proposé conjointement par le Cerema Normandie-Centre, le laboratoire LETG-Caen Géophen de l'université Caen Normandie, le Laboratoire de Mathématiques de l'INSA de Rouen Normandie et le laboratoire M2C.

Cette étude permettra à terme de comprendre, quantifier et surtout anticiper les phénomènes d'érosion dans les zones concernées et se base essentiellement sur des algorithmes mathématiques de traitement de données et de segmentation d'images.

De ce fait, le Cerema Normandie-Centre a mis en place un stage afin de tester la pertinence de cette méthodologie en l'appliquant sur trois sites-tests : Fécamp, Villers Sur Mer et Quiberville.

Dans un premier temps, une étude précise des notions théoriques requises à la mise en place des algorithmes de travail a été réalisé, et ensuite, les données fournies par le ROL des trois zones-tests ont été traitées en appliquant deux méthodologies différentes de détection de fracturations (semi-automatique et automatique).

2 Remerciement

Avant toute chose, j'aimerais commencer ce rapport de stage par des remerciements pour toutes les personnes qui m'ont, que ce soit de près ou de loin, aidé tout au long de cette expérience professionnelle.

En particulier, je tiens à remercier chaleureusement mon maître de stage, Mr Cyrille Fauchard, chef de l'Unité Electromagnétisme appliqué (ERA 23), qui m'a gentiment permis d'embarquer dans cette aventure très enrichissante et qui m'a soutenu pendant toute la durée de mon stage, n'hésitant jamais à répondre à mes questions et interrogations concernant mon travail.

Je tiens aussi à remercier mon professeur, Mr Nicolas Forcadel de l'INSA de Rouen, qui m'a aidé à décrocher ce stage .

Finalement, j'aimerais également adresser mes remerciements aux membres de l'équipe, à savoir, Raphaël Antoine, Vincent Guilbert, Bruno Beaucamp ainsi que Cyrille Leden, qui m'ont été d'une très grande aide tout au long de ces deux mois de stage en répondant toujours présent quand j'en avais besoin, et qui ont tout fait pour que mon insertion soit la plus fluide possible.

3 Présentation de l'entreprise

Le CEREMA Normandie-Centre, organisme du Ministère de l'Écologie, de l'Énergie, du Développement Durable et de la Mer, fait partie du Réseau Scientifique et Technique (RST) de ce ministère.

Le CEREMA Normandie-Centre est un centre public de recherche et d'ingénierie pour tous les acteurs de l'aménagement : services de l'État, collectivités territoriales, organismes para-publics ou privés.

Service de proximité, le CEREMA a constitué, depuis sa création en 1973, une connaissance fine du territoire dans sa zone d'action (régions de Hautes et Basse-Normandie, région Centre, ainsi que les territoires d'outre-mer suivants : la Martinique, la Guadeloupe, la Guyane et St Pierre et Miquelon).

Le CEREMA dispose d'une large gamme de compétences techniques, il bénéficie de son ancrage sur le territoire (un site à Rouen, un site à Blois) et peut s'appuyer sur la mobilisation du Réseau Scientifique et Technique pour réunir et assembler les moyens et compétences dont il ne disposerait pas en propre.

Le CEREMA intervient pour environ 80% de son activité pour l'État :

— Direction centrale Services techniques centraux
— Services déconcentrés : les Directions Régionales de l'Environnement, de l'Aménagement et du Logement (DREAL)
— Les Directions Départementales du Territoire (DDT)
— Les Directions Interrégionales des Routes (DIR)

Assurant des missions de service public, il se tourne également vers les besoins des collectivités et des professionnels. L'offre du CEREMA privilégie l'accompagnement des politiques publiques de l'État : la protection de l'environnement, la prévention des risques, la solidarité territoriale, la sécurité des déplacements, etc.

Il rassemble 2791 personnes sur toute la France qui interviennent dans le domaine de la ville et de l'aménagement du territoire, des transports, des infrastructures et ouvrages d'art, de l'environnement, de l'exploitation et de la sécurité routière ainsi que l'informatique.

Il propose pour chacun des domaines une palette de prestations variées : études, recherches, méthodologies, expertises, conseils, assistances, animations de réseaux, formation, avis techniques, essais de laboratoire et sur sites ou contrôles de chantier.

Ses domaines d'activités sont principalement liés à l'environnement, la prévention des risques et la sécurité routière ainsi que les infrastructures et le bâtiment.

Organisation Le Cerema Normandie-Centre se compose de :

— 3 départements : le Département Aménagement Durable des Territoires (DADT), le Département Infrastructures de Transport Multimodales (DITM)

et le Département Expérimentation, Recherche, Développement et Innovation (DERDI)
— 2 laboratoires : le Laboratoire Régional de Rouen (LRR) et le Laboratoire Régional de Blois (LRB)
— 1 secrétariat général (SG)

ENDSUM : Évaluation Non Destructive des StrUctures et des Matériaux
L'équipe du projet ENDSUM s'attache à mettre au point des méthodes de caractérisation et de diagnostic à grand rendement, moins invasives que les outils actuels, n'altérant pas les milieux auscultés et permettant le remplacement des méthodes basées sur des sources radioactives par de nouveaux développements technologiques à destination des gestionnaires d'infrastructures et des bureaux d'études.

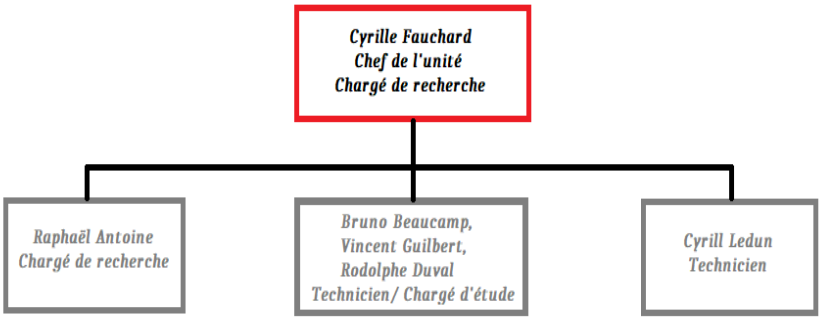


FIGURE 1 : Organigramme de l'équipe

4 La Level Set Method

4.1 Théorie

La « Level-Set Method », ou plus communément appelée la méthode des surfaces de niveau est une méthode en analyse numérique initialement introduite par Osher et Sethian en 1999, permettant l'étude et le suivi de l'évolution temporelle d'interfaces mobiles, séparant deux régions ou milieux fermés. Elle est très utilisée en traitement d'images, et plus particulièrement en imagerie médicale.

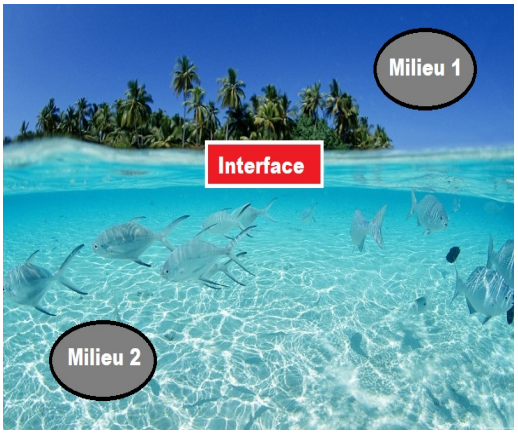


FIGURE 2 : Exemple simple d'interfaces, la surface de la mer

L'idée qui donne toute l'importance à cette méthode réside dans le fait d'évaluer une courbe paramétrique fermée qui évolue dans le temps et perpendiculairement à elle-même, suivant une équation d'Hamilton Jacobi donnée. L'évolution temporelle de cette courbe nous permettra donc de réaliser une segmentation de l'image en question.

DEFINITION : ***Soit*** $\zeta(t)$ ***une interface évoluant dans le temps, et soit*** $\phi(0)$ ***une fonction level-set pour*** $\zeta(0)$***. Alors*** $\phi(t)$ ***est solution de l'équation Hamilton Jacobi suivante :***

$$\frac{\partial \phi}{\partial t} = V.\nabla \phi$$

avec $\phi(0) = \phi_0$ ***et*** $V(x,t)$ ***la vitesse d'évolution de l'interface.***

EXEMPLE D'APPLICATION DE LA LEVEL-SET METHOD : La figure suivante montre comment l'interface utilisée dans le cadre de la Level-Set Method permet de reconstituer un cortex cérébrale à partir d'une IRM.

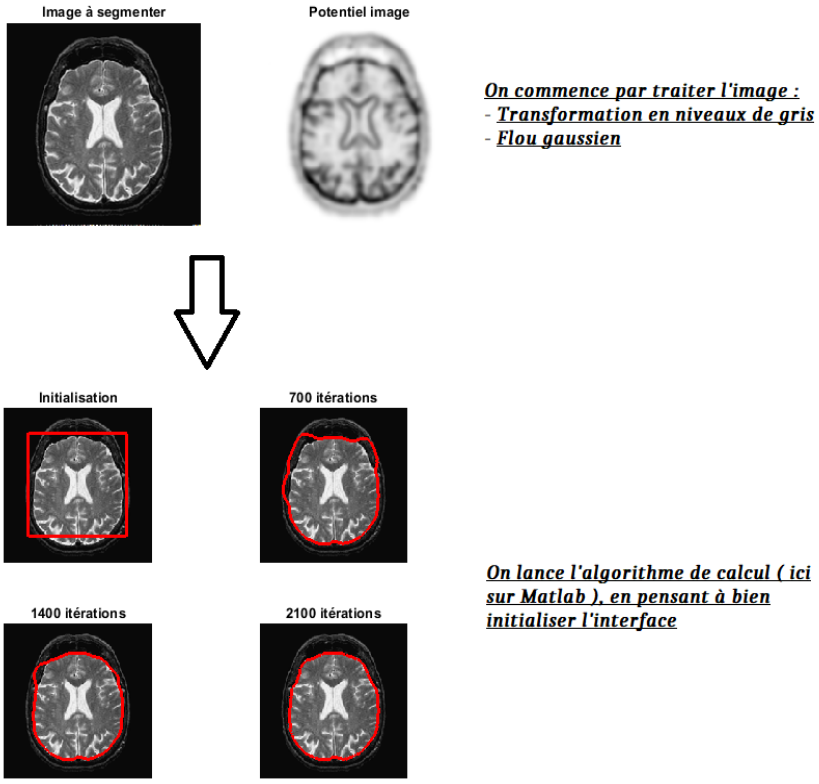


FIGURE 3 : Segmentation d'une image grâce à la Level-Set Method [*Source* : *Gabriel Peyré, Numerical Tours*]

En terme général, les level-sets d'une fonction h désignent les surfaces dans lesquelles la fonction considérée est constante. Par exemple, si on considère la fonction en deux dimensions suivante :

$$h(x,y) \quad = \quad x^2 + y^2$$

Les level-sets sont donc des cercles autour de l'origine, la fonction étant égale au rayon (qui est bien entendu constant) pour tout (x,y) appartenant à la surface du cercle en question.

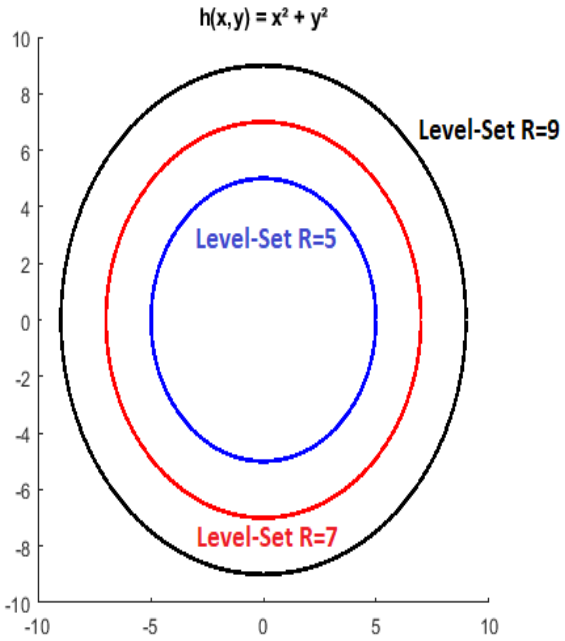


Figure 4 : Exemple de Level-Sets en deux dimensions [*Source* : *Auteur*]

Incrémentons maintenant la dimension de travail. Un level-set au niveau c sera donc donné par la coupure en surface du modèle suivant le plan $z = c$. En effet, prenons en compte cette fois-ci la fonction suivante :

$$H(x,y,z) \quad = \quad x^2 + y^2 + z^2$$

Le level-set $z = 5$ est le cercle donné par $x^2 + y^2 + 25$ (ou $H(x,y,5)$). En modifiant progressivement la valeur de z, on obtient donc au fur et à mesure

des slices, ou coupure en surface, nous permettant d'avoir un visuel du modèle donné par la fonction étudiée et de ses contours.

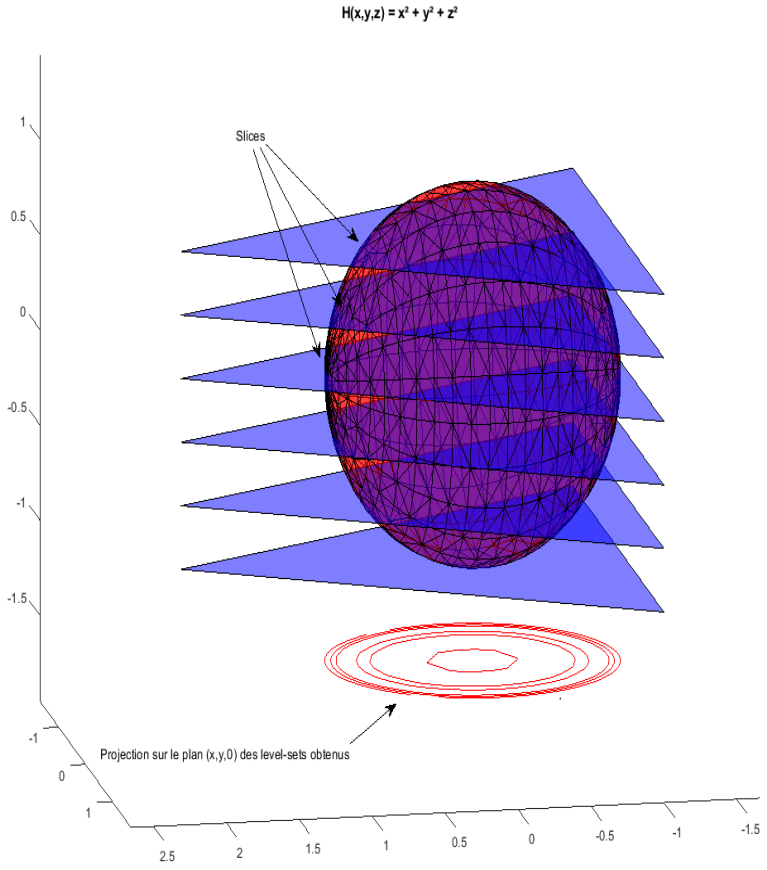


Figure 5 : Exemple de Level-Sets en trois dimensions [*Source* : *Auteur*]

Dans les deux cas précédents, les fonctions régissant les modèles étudiés sont connues. Cependant, en pratique (comme dans le cas du cortex cérébral traité en début de chapitre) , elles ne le sont pas forcément. Essayons donc maintenant de comprendre comment on peut traiter ces cas grâce à cette méthode.

Tout d'abord, considérons une interface fermée ζ qui se propage suivant un

vecteur normale à celle-ci à une vitesse V, et notons β_1 et β_2 les deux milieux séparés par ζ. La méthode des surfaces de niveau consiste donc à considérer la courbe ζ comme étant un des level-sets d'une fonction de dimension supérieure $\phi(x,y,t)$ lipschitzienne de $\mathbb{R}^2 * \mathbb{R}$ dans $\mathbb{R}^2$. Celle-ci est nommée fonction Level-Set et l'interface ζ en un point $\alpha = (x_\alpha, y_\alpha)$ est alors donné par :

$$\zeta(\alpha) \quad = \quad \{(x,y) \in \mathbb{R}^2 | \phi(x,y,t) = 0\}$$

DEFINITION : ***Soient*** E ***et*** F ***deux ensembles respectivement définies par les distances*** $dist(E)$ ***et*** $dist(F)$***.***

Soient $(E, dist(E))$ ***et*** $(F, dist(F))$ ***deux espaces métriques. On appelle lipschitzienne toute fonction de*** E ***dans*** F ***vérifiant :***

$$(f(x), f(y))_{dist(F)} \quad \leq \quad k * (x,y)_{dist(E)}$$

$\forall (x,y) \in E^2$***, avec*** f ***la fonction considérée et*** $k \in \mathbb{R}^+$***.***

Ainsi, afin de caractériser le déplacement de l'interface ζ considérée, car rappelons-le, cela constitue l'objectif final de la Level-Set Method, il suffit de déterminer l'évolution de la fonction ϕ par rapport au temps t.

Le choix de cette dernière est arbitraire, du moment qu'elle vérifie la condition d'initialisation suivante : en $t = 0$, la fonction choisie se doit de correspondre au contour initial du modèle étudié. En d'autres termes, le level-set de la fonction en $t = 0$ se doit d'être « égal » au contour initial du modèle. Cette condition est primordiale quant à la réussite de la méthode.

Une fois la fonction Level-Set ϕ correctement choisie, il est possible de déterminer son évolution dans le temps grâce à l'équation de mouvement donnée par :

$$\begin{aligned} \frac{\partial \phi(\alpha(t), t)}{\partial t} &= 0 \\ \iff \frac{\partial \phi}{\partial \alpha(t)} * \frac{\partial \alpha(t)}{\partial t} + \frac{\partial \phi}{\partial t} * \frac{\partial t}{\partial t} &= 0 \\ \iff \nabla \phi * \dot{\alpha} + \dot{\phi} &= 0 \end{aligned}$$

Comme l'interface ζ se propage suivant un vecteur qui lui est normale, notons

le ici n, et à une vitesse V, la vitesse de chaque point α appartenant à ζ est donc donnée par $\dot{\alpha} = V(\alpha) * n$ avec $n = \frac{\nabla\phi}{|\nabla\phi|}$.

En reprenant les calculs précédents, on a donc :

$$\begin{aligned} \nabla\phi * \dot{\alpha} + \dot{\phi} &= 0 \\ \iff \nabla\phi * V(\alpha) * n + \dot{\phi} &= 0 \\ \iff \nabla\phi * V(\alpha) * \frac{\nabla\phi}{|\nabla\phi|} + \dot{\phi} &= 0 \\ \iff |\nabla\phi| * V(\alpha) + \dot{\phi} &= 0 \end{aligned}$$

L'équation d'Hamilton Jacobi obtenue à la fin décrit alors l'évolution dans le temps de la fonction level-set considérée. En d'autres termes, en connaissant cette dernière et ϕ en $t = 0$, il est possible de calculer ϕ en tout temps t. La recherche des level-sets (et donc des contours du modèle) se fait par la suite assez intuitivement.

4.2 Implémentation de la Level Set Method

En pratique, les fonctions considérées ne sont pas continues et se doivent donc d'être discrétisées. Dans notre cas, nous évaluons la fonction ϕ à chaque pixel de coordonnées(x, y) de l'image traitée par $\frac{\phi(x,y,t+\Delta t)-\phi(x,y,t)}{\Delta t}$.

Or, comme nous avons montré que :

$$|\nabla\phi| * V(\alpha) + \dot{\phi} = 0$$

Alors

$$|\nabla\phi| * V(\alpha) + \frac{\phi(x, y, t + \Delta t) - \phi(x, y, t)}{\Delta t} = 0$$

On peut donc calculer ϕ $\forall t$ ($\phi(x, y, 0)$ connue) grâce à :

$$\phi(x, y, t + \Delta t) = \phi(x, y, t) - |\nabla\phi| * V(\alpha) * \Delta t$$

Bien que la Level-Set Method semble être la solution magique pour la segmentation des images, elle présente plusieurs inconvénients non négligeables, les plus importants étant le coût et le temps de calcul.

En effet, l'introduction d'une fonction d'une grande dimension tend à ralentir le programme et à alourdir les calculs lorsqu'on désire appliquer la méthode sur un volume important de données. Heureusement, une alternative bien plus intéressante a été introduite en même temps que cette dernière : La Fast Marching Method.

5 La Fast Marching Method

La FMM (Fast Marching Method) est une méthode numérique permettant un suivi de l'évolution d'interfaces fermées grâce à la résolution d'une équation eikonale. Elle est très employée en traitement d'images, en mécanique des fluides numériques et en infographie. Bien que son principe et celui de la Level-Set Method soient assez identiques, il subsiste quand même quelques différences, à savoir sur le plan théorique, qui donnent par ailleurs à la FMM toute sa puissance.

5.1 Théorie

Reprenons les mêmes notations qu'auparavant :

Soit ζ une interface fermée qui se propage dans un milieu β suivant un vecteur normale et à une vitesse $V > 0$.

Afin de caractériser et suivre l'évolution de ζ, nous allons considérer le temps d'arrivée de l'interface d'un point de départ $\alpha_{départ}$ à chaqu'un des autres points de β. Notons donc $T(\alpha_{arrivée})$ la durée du trajet $[\alpha_{départ}; \alpha_{arrivée}]$ de ζ à la vitesse V.

Dans le cas de la Fast Marching Method, nous allons supposer que V ne dépend pas du temps et est donc stationnaire.

On rappelle l'équation de mouvement obtenue dans le chapitre précédent concernant la Level-Set Method :

$$|\nabla \phi| * V(\alpha) + \dot{\phi} \quad = \quad 0$$

En posant $\phi(\alpha, t) = T(\alpha) - t$, nous pouvons montrer grâce à l'équation précédente que l'amplitude du gradient du temps d'arrivée T est inversement proportionnelle à la vitesse de propagation dans ζ, c'est-à-dire que :

$$|\nabla T| \quad = \quad \frac{1}{V}$$

Comme $V > 0 \ \forall \alpha$, alors la fonction$T(\alpha)$ admet un unique minimum global en $\alpha_{départ}$ (la durée de trajet ne peut qu'augmenter en s'éloignant du point de départ). De ce fait, l'interface ne peut qu'évoluer « vers l'extérieur ».

Par ailleurs, comme V dans notre cas dépend que de α, alors $|\nabla T| = \frac{1}{V}$ est une équation dite eikonale.

Definition : ***Une équation eikonale est une équation aux dérivées partielles non linéaire très utilisée en optique géométrique et en propagation d'ondes.***
Elle s'écrit sous la forme :

$$|\nabla u(x)| \quad = \quad \frac{1}{f(x)}$$

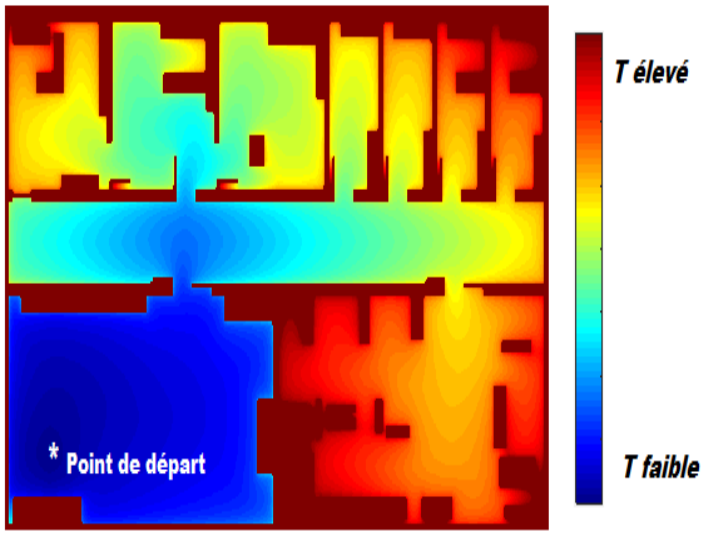
où $x \in \Omega$, Ω ***étant un ouvert de*** $\mathbb{R}^n$, ***et*** f ***une fonction positive*** $\forall x$.
La solution $u(x)$ ***nous donne le temps minimal du trajet d'un point de départ à*** x ***et à une vitesse*** f.

La résolution de cette équation constitue la finalité de la FMM dans la mesure où elle nous permet de caractériser l'évolution spatiale de T, et donc par la même occasion, celle de l'interface ζ.

L'exemple suivant montre bien cela. En effet, dans ce dernier, à chaque point α de l'image originale (en dehors des points noirs considérés ici comme des obstacles et donc le temps d'arrivée y à été initialement mis à $+inf$), l'algorithme calcule T en résolvant l'équation eikonale précédemment introduite. Ensuite, grâce à un code de couleur, il affiche les valeurs de T dans une image de sortie. Cela nous permet donc de visualiser parfaitement les variations de l'interface sous forme d'ondes dont la progression commence au point de départ et qui se poursuit ensuite plus loin.

Image Originale

$$\Downarrow$$
$$F \quad M \quad M$$
$$\Downarrow$$



Evolution Spatiale de T

Figure 6 : Calcul de T sur un exemple d'images [*Source : Gabriel Peyré, Numerical Tours*]

Comme nous pouvons le constater, l'algorithme derrière la FMM repose entièrement sur la résolution numérique de l'équation eikonale. Essayons donc maintenant de voir comment cela est possible.

5.2 Résolution de l'équation eikonale

Afin de résoudre numériquement l'équation eikonale, une discrétisation du système est nécessaire. Pour cela, comme nous travaillons majoritairement sur des données en 2D (images), nous allons créer donc des grilles de cellules de même dimension que les images étudiées. Ainsi, chaque pixel d'abscisse x_i et d'ordonnée y_j de l'image sera représenté par la cellule de coordonnées (i, j) de la grille.

Notons par la même occasion Δx et Δy le pas respectivement selon l'axe des x et des y.

On s'intéresse ensuite à la discrétisation de ∇T proposée par Sethian (1999) et donnée par la relation suivante :

$$max(D_{ij}^{-x}T, -D_{ij}^{+x}, 0)^2 + max(D_{ij}^{-y}T, -D_{ij}^{+y}, 0)^2 = \frac{1}{V_{ij}^2}$$

où

$$\begin{aligned} D_{ij}^{-x} &= \frac{T_{i,j} - T_{i-1,j}}{\triangle x} \\ D_{ij}^{+x} &= \frac{T_{i+1,j} - T_{i,j}}{\triangle x} \\ D_{ij}^{-y} &= \frac{T_{i,j} - T_{i,j-1}}{\triangle y} \\ D_{ij}^{+y} &= \frac{T_{i,j+1} - T_{i,j}}{\triangle y} \end{aligned}$$

Ensuite, en posant :

$$\begin{aligned} T &= T_{i,j} \\ T_1 &= min(T_{i-1,j}, T_{i+1,j}) \\ T_2 &= min(T_{i,j-1}, T_{i,j+1}) \end{aligned}$$

On peut réécrire l'équation eikonale précédente de la manière suivante [*Sethian*(1999)]

$$max(\frac{T - T_1}{\triangle x}, 0)^2 + max(\frac{T - T_2}{\triangle y}, 0)^2 = \frac{1}{V_{ij}^2} \quad (*)$$

5.3 Implémentation de la FMM

L'implémentation informatique de la FFM peut se faire essentiellement par la résolution de $(*)$ à chaque cellule de la grille. Cependant, pour de grandes données, le calcul risquera d'être trop coûteux en temps de calcul. C'est pourquoi un algorithme plus astucieux et surtout plus rapide a été introduit.

Tout d'abord, définissons les trois ensembles suivants :

— Les cellules se situant à l'intérieur de l'interface :

$$Accepted \quad = \quad \{c \in grille | T(c) - T(\alpha_{départ}) < 0\}$$

— Les cellules se situant à la frontière de l'interface :

$$Narrow\_Band \quad = \quad \{I, \exists J \in Voisinage(I), J \in Accepted\}$$

— Les cellules se situant loin de l'interface :

$$Far \quad = \quad \{c \in grille | T(c) - T(\alpha_{départ}) > 0$$

5.3.1 Algorithme

Initialisation : Figure 7 (1)

— L'agorithme commence par initialiser la valeur de T à 0 pour toutes les cellules appartenant à $Accepted$ (et à $+inf$ en cas d'obstacles). Au début, $Accepted$ ne contient que le point de départ choisi, c'est-à-dire, $\alpha_{départ}$.

— Il calcule ensuite les T des cellules appartenant à $Narrow\_Band$ en résolvant l'équation eikonale définie précédemment.

Itérations : Figure 7 (2) + (3)

— La cellule dont la valeur de T est minimale est noté c_{min}.

— L'algorithme intègre c_{min} dans l'ensemble $Accepted$ et résoud l'équation eikonale à chaque cellule voisine de c_{min}. Celles-ci sont ensuite intégré à l'ensemble $Narrow\_Band$.

Finalisation : Figure 7 (4)

— L'algorithme s'arrête lorsque l'ensemble $Narrow\_Band$ est vide.

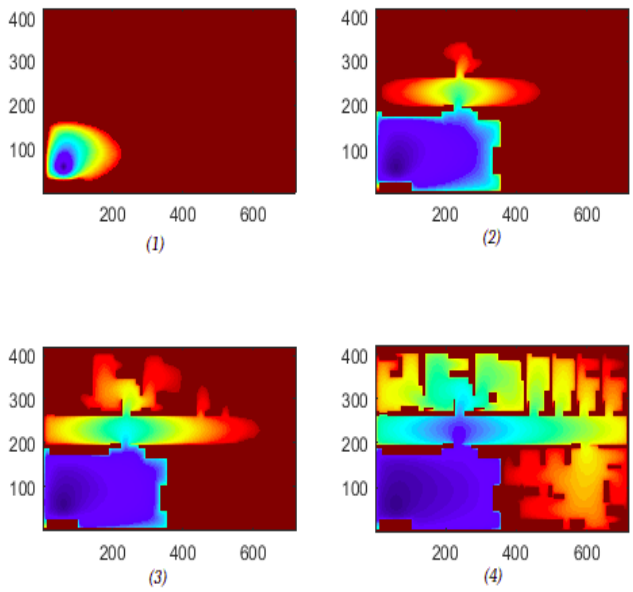


Figure 7 : Visualisation progressive de la propagation de l'interface[*Source* : *Auteur + Gabriel Peyré*]

6 Déroulement du Stage

Le but de ce stage est la détection de fracturations sur des platiers rocheux situés au nord de la France. Dans ce cadre, trois zones ont été étudiées :

- La première à Fécamp,
- La deuxième à Quiberville,
- La dernière à Villers-Sur-Mer

Les données d'entrée sont de type LIDAR (« light detection and ranging ») fournies par le Réseau d'Observation du Littoral (ROL) Normandie-Hauts-De-France. Les données finales ont été reçues par l'équipe sous la forme de plusieurs fichiers LAS, contenant chacun un modèle sous forme de nuages de points géoréférencés d'une partie d'une des trois zones étudiées.

Dans les chapitres qui vont suivre, seuls les résultats obtenus pour le sitede Fécamp seront mis en avant. En effet, cette décision a été prise afin d'avoir un rapport final plus concis et plus intéressant pour le lecteur.

Les deux autres zones ont bien entendues été traitées de la même manière.

6.1 Traitement des données

Dans un premier temps, et au vu du volume total des données (atteignant plus de 20 Go), une phase de traitement s'imposait.

En effet, après leur visualisation sur le logiciel *CloudCompare*, nous avons remarqué que la surface balayée s'étend tout au long du trait de côte, dont une grande partie était des falaises.

Étant donnée le but de ce stage (la détection de fracturations au niveau du platier rocheux), travailler sur cette portion des données, qui plus est, constitue une grande partie de leur volume total, seriat trop coûteux en temps. Ainsi, un filtrage selon l'axe des z a été réalisé et les fichiers résultants du traitement (qui sont donc beaucoup moins volumineux que ceux d'avant) ont été regroupé en un seul.

Un code Python a été donc mis au point afin de réaliser ce traitement. Il effectue successivement et dans l'ordre les commandes suivantes :

- Il demande à l'utilisateur de rentrer les bornes maximale et minimale des z au cours du filtrage.
- Il demande à l'utilisateur de choisir le répertoire contenant tous les fichiers LAS à traiter. En cas d'erreur (par exemple, si le répertoire choisi ne contient aucun fichier LAS), la commande est relancée.
- Il ouvre chaque fichier LAS du répertoire d'entrée sous forme de liste Numpy et filtre les points selon les bornes de z rentrées au départ.
- Il regroupe tous les fichiers traités en un seul et demande à l'utilisateur de rentrer son nom et celui du répertoire de sortie.
- Il ouvre le fichier-résultat sur *CloudCompare*.

La figure suivante résume le travail réalisé par ce code pour un exemple de 21 fichiers LAS :

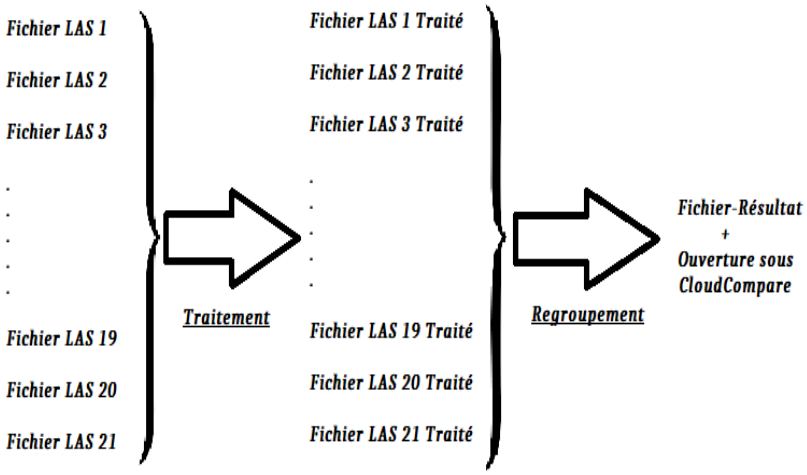


Figure 8 : Schéma général du traitement des données effectué [*Source : Auteur*]

Dans notre cas, l'intervalle de filtrage, c'est-à-dire, l'intervalle des valeurs de z des points pris en compte, est de [-11 ;0]. Les figures suivantes montrent bien le résultat du traitement effectué sur chacunes des zones étudiées :

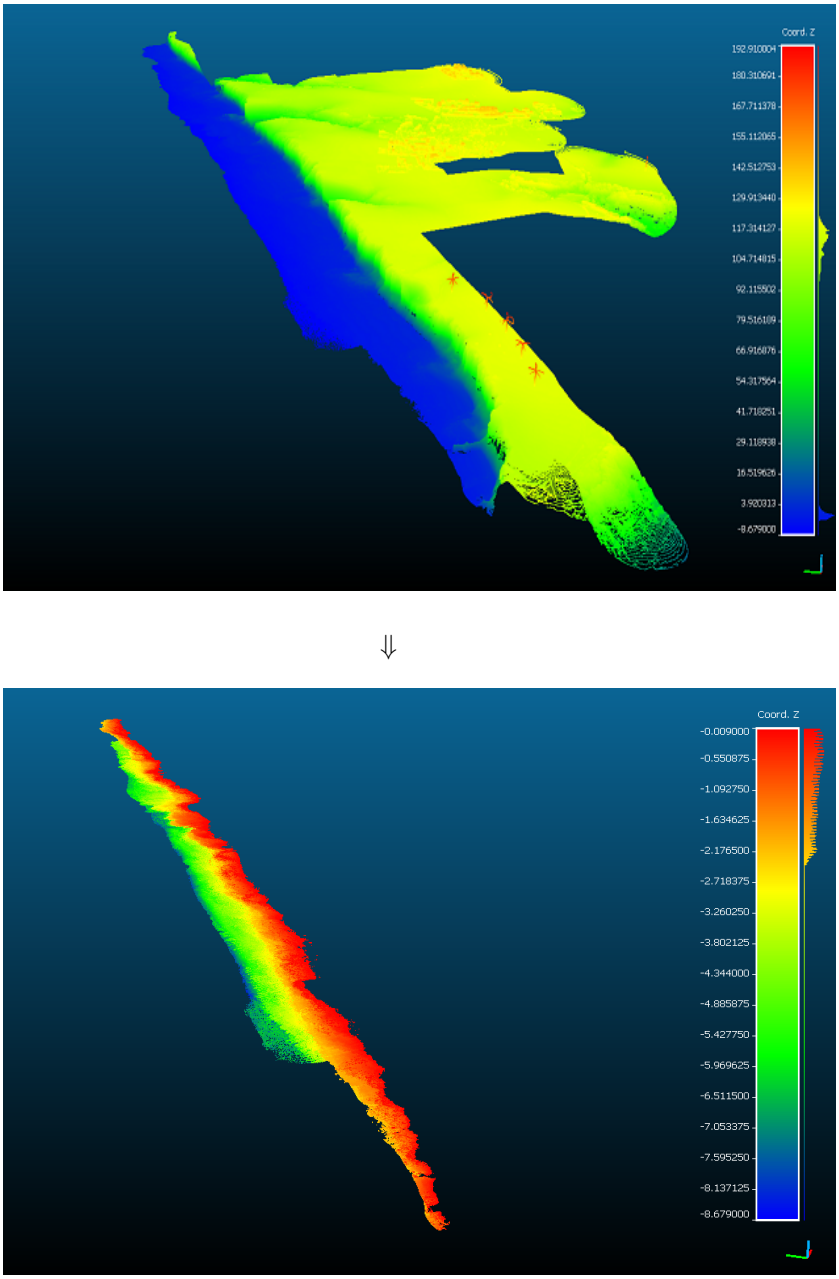


Figure 9 : Traitement des données dans la zone Fécamp [*Source : Auteur*]

Une fois que le filtrage a été correctement réalisé, la question qui se pose maintenant est celle de la détermination des fracturations sur le plateau rocheux.

Afin d'y répondre, il nous a semblé intelligent de décrémenter la dimension de travail. En effet, le fichier-résultat LAS du traitement de chaque zone, données 3D, a été transformé en un raster, données 2D. La rastérisation a été réalisé sous *CloudCompare* après interpolation du nuage de points. Les résultats sont donc les images en niveau de gris suivantes :

Figure 10 : Raster du platier rocheux situé à Fécamp [*Source : Auteur*]

Parallèlement à cela, une extraction des contours à différentes valeurs de z a été effectué une nouvelle fois grâce à *CloudCompare* sur les nuages de points traités. Les contours sont ensuite superposés sur une orthophoto (cliché aérien) de la zone étudiée,à l'aide du logiciel de traitement de données géospatiales QGis, nous permettant donc d'avoir un visuel des potentielles fracturations existantes au niveau des plaques rocheuses.

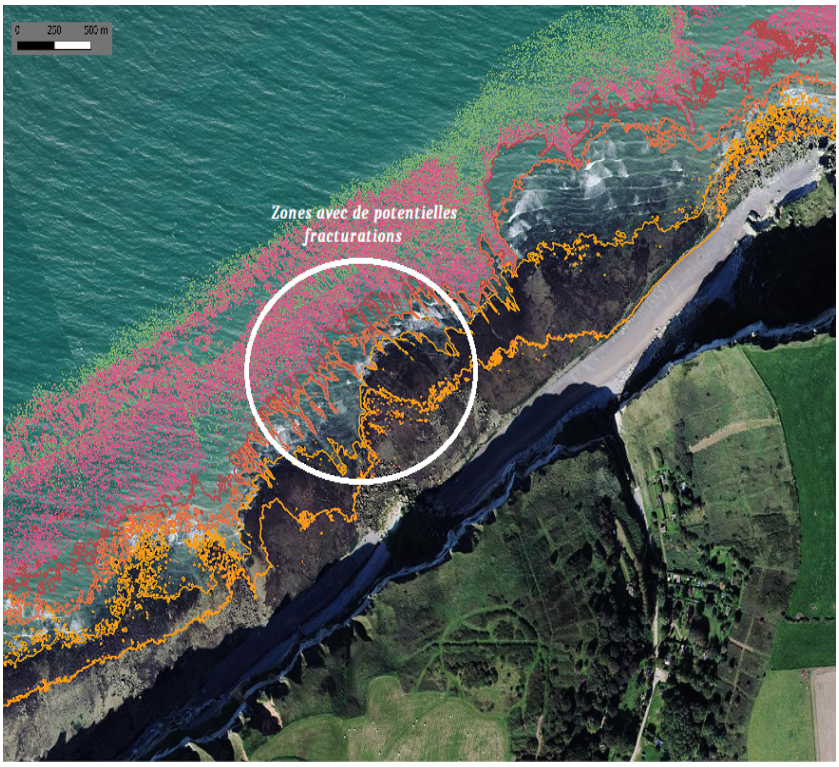


Figure 10 : Visualisation d'une partie de l'orthophoto du platier rocheux situé à Fécamp et de ses contours [*Source* : *Auteur*]

6.2 Détection Semi-Automatique

Afin de maximiser la précision de la détection des fracturations dans le raster étudié, nous avons tout d'abord effectué son découpage en parts égales. L'image initiale a donc été découpée en 8 parties.

Ensuite, un programme sous Matlab disponible au sein de l'équipe [*Source* : *PierreCharbonnier*, *FFM*] a été utilisé. Ce dernier nous permet d'effectuer une détection semi-automatique, c'est-à-dire, nécessitant de choisir en premier lieu deux points entre lesquels le programme essaira de détecter la présence ou non d'une fracturation, très précise.

Le code en question se base essentiellement sur l'application de la Fast Marching Method présentée précédemment et celle de la descente de gradient. En effet, la FFM nous permet, entre autre, d'obtenir d'une image 2D une carte de distance (la **Figure 5** en est un bon exemple). Ensuite, par descente de gradient, on retrouve et trace la fracturation, s'il y en a, entre les deux points de départ et d'arrivée choisis.

Le code se présente comme suit :

— Lecture de l'image choisie par l'utilisateur,
— Première Phase de traitement de l'image : normalisation, filtrage gaussien et soustraction du fond permettant de réduire le bruit et de maximiser la précision de la détection des fracturations par le programme,
— Création de la carte de distance en appliquant la Fast Marching Method sur l'image traitée,
— Calcul et normalisation du gradient à chaque point de l'image rentrée par l'utilisateur,
— Choix des points de départ et d'arrivée par l'utilisateur,
— Détection de la fracturation par descente de gradient (notons que celle-ci commence au point d'arrivée),
— Affichage de la ligne suivi par le programme lors de la descente de gradient. Celle-ci représente donc la fracturation recherchée.

Les figures suivantes retracent le fonctionnement du programme sur une portion du platier rocheux situé à Fécamp :

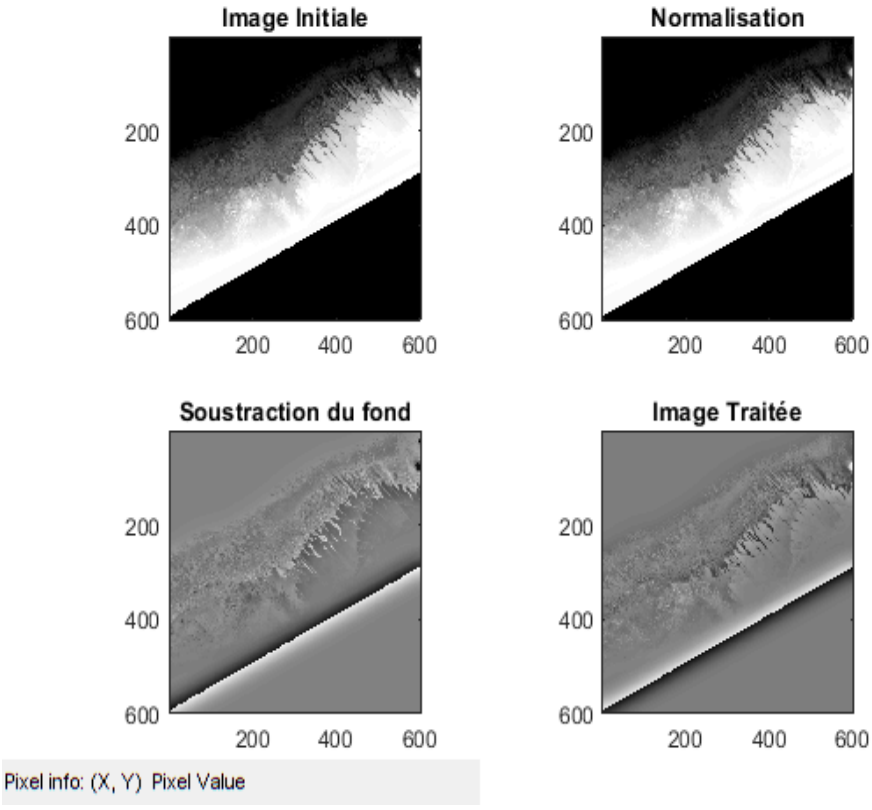


Figure 11.1 : Pré-Traitement de l'image

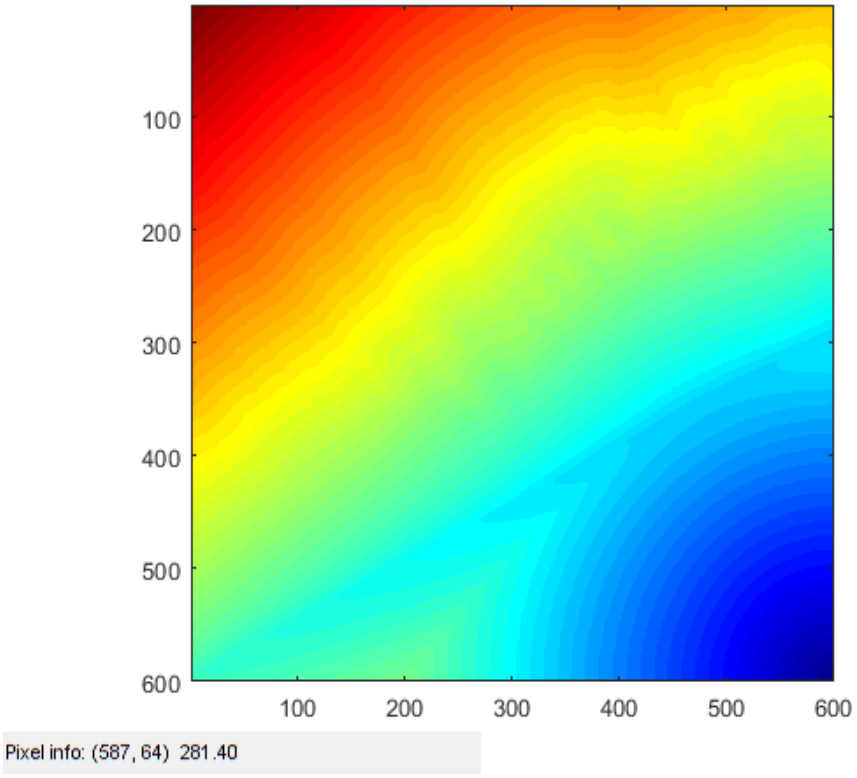


Figure 11.2 : Carte de distance, le point de départ de la FMM choisi ici est situé à (600,600)

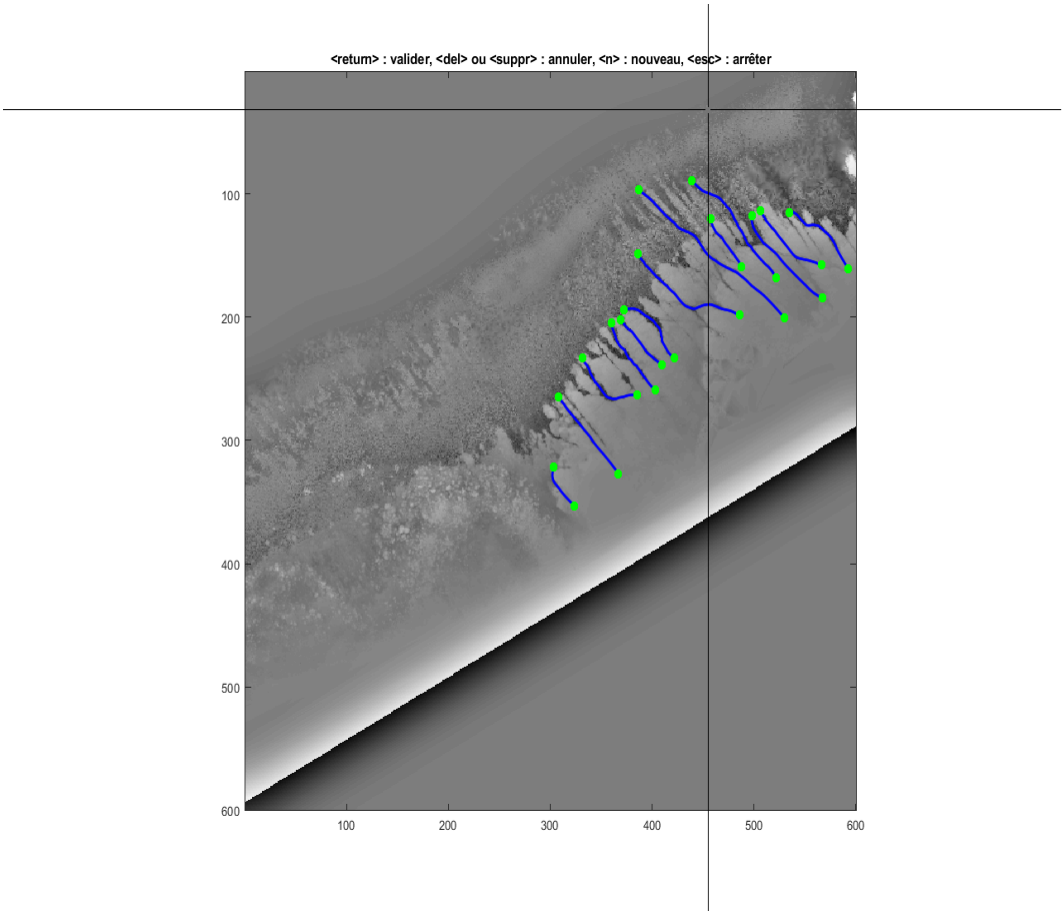


Figure 11.3 : Quelques exemples de fractures détectées par le programme

6.3 Détection Automatique des fracturations

Un second code sous Matlab a également été mis au point. Il permet de réaliser une détection automatique des fracturations sur un zone donnée, c'est-à-dire, que le choix de points de départ et d'arrivée n'est plus requis et que le programme détecte directement toutes les fracturations existantes dans l'image.

Cette fois-ci, le travail consiste à réaliser une segmentation d'images, non pas grâce à la Fast Marching Method, mais plutôt par la localisation des contours et de leur direction. En effet, la détection est rendue possible essentiellement grâce au filtrage de l'image considérée selon le filtre de Sobel [*Source* : *OpenCV*].

Explication : Une image est considérée par le programme comme étant une matrice de même taille et où chaque case correspond à un pixel de l'image. Ainsi, le principe du filtre de Sobel consiste à approximer la dérivée horizontale et verticale de chaque point grâce à un calcul de convolution comme suit [*Source* : *OpenCV*, *Sobel Derivatives*] :

— Dérivée Horizontale :

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * M$$

— Dérivée Verticale :

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * M$$

où M désigne la matrice correspondant à l'image traitée.

La norme du gradient est ensuite donnée par :

$$|G| = \sqrt{G_x^2 + G_y^2}$$

Enfin la direction du gradient est déterminée en utilisant la formule suivante :

$$\theta = atan2(G_y, G_x)$$

De ce fait, l'application de ce filtre à l'image permet de calculer le gradient de l'intensité de chacun de ses pixels et donc de déterminer les endroits avec de fortes variations de niveaux de gris ainsi que la direction du gradient, c'est-à-dire, l'angle formé par ces variations.

Par la suite, une sélection des variations se fait suivant les valeurs des directions. Dans le cas de la zone étudiée à Fécamp, comme les fracturations ont un angle par rapport à l'axe des abscisses appartenant à [90°C ;180°C], la sélection se fait donc suivant cet intervalle et on affiche ensuite que les variations dont les angles y sont compris. La détection automatique des fracturations est donc possible.

La figure suivante montre le résultat de l'application du programme sur le raster du platier rocheux situé à Fécamp :

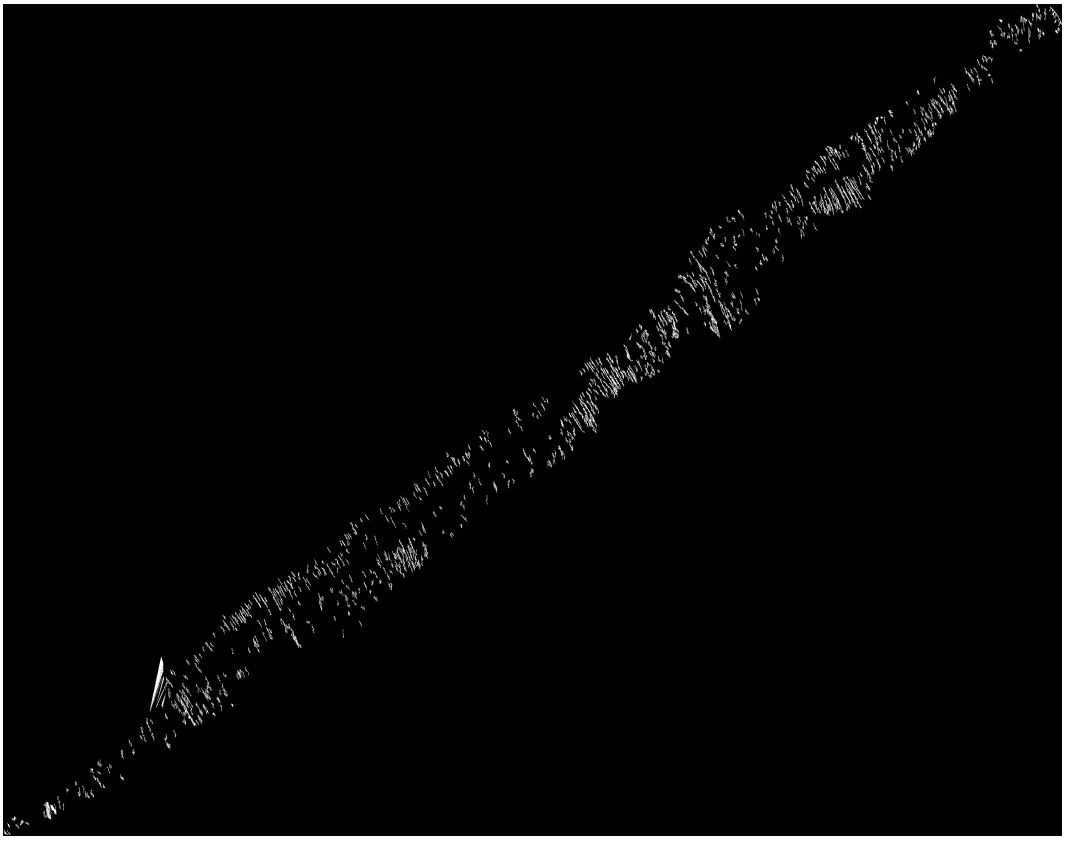


Figure 12 : Détection automatique des fracturations [*Source* : *Auteur*]

La rosace suivante nous donne l'orientation des fracturations détectées par le programme :

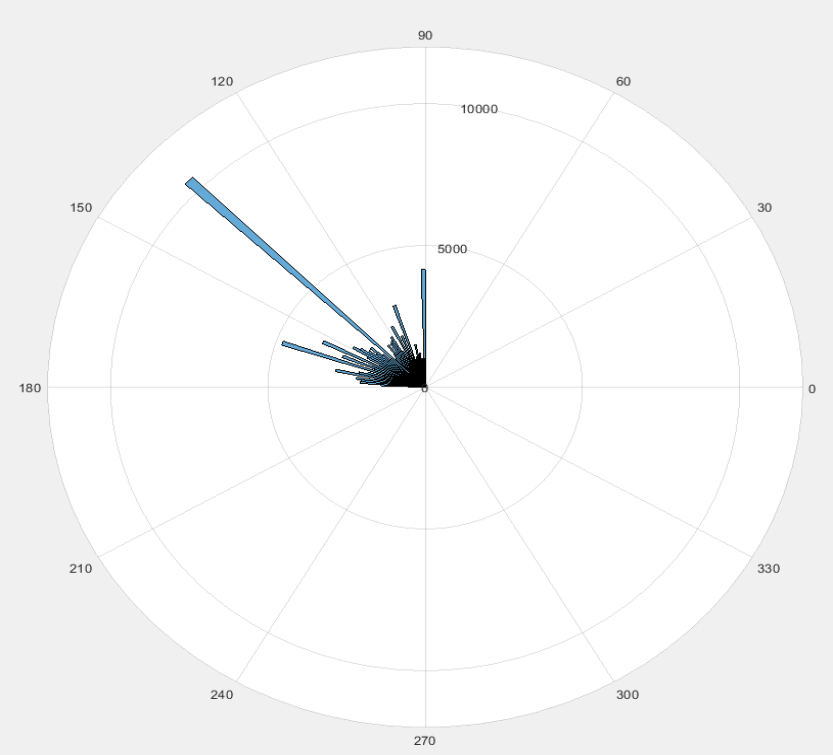


Figure 13 : Rosace des principales directions de fracturations [*Source : Auteur*]

On remarque donc que la grande partie des fracturations forme un angle d'environ 135° par rapport à l'axe des abscisses, soit un angle de 310° Nord.

Par la suite, les fracturations représentées dans la **Figure 12** sont ensuite placées sur l'orthophoto de la zone étudiée :

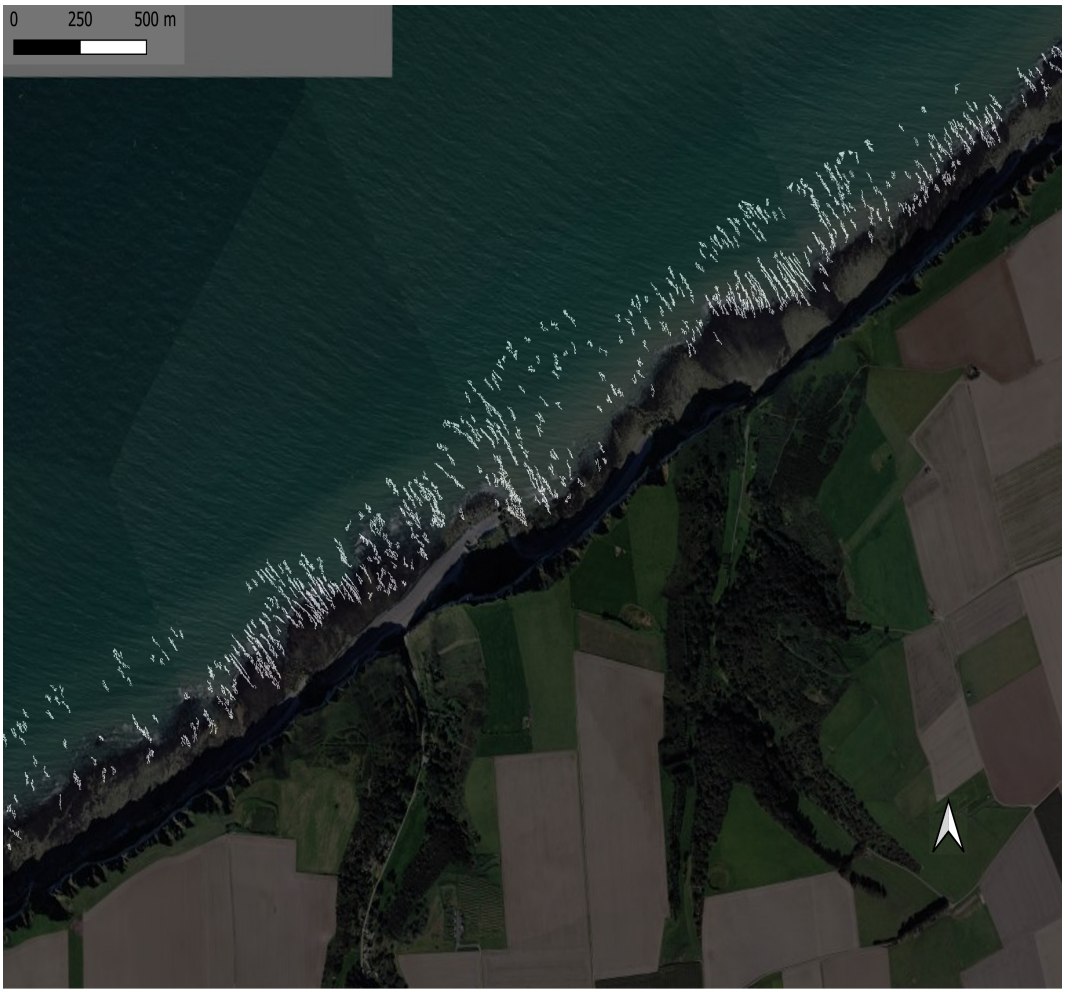


Figure 14 : Superposition des fracturations détectées sur l'orthophoto de la zone concernée [*Source : Auteur*]

On remarque donc que cette méthode nous donne des résultats assez satisfaisants quant à la détection automatique des fracturations dans une zone donnée. En effet, on voit bien dans la figure précédente que le programme développé détecte la majeure partie des fissurations présentes au niveau du platier rocheux de la zone étudiée.

Il est quand même important de noter quelques inconvénients concernant cette méthode :

— Une connaissance en amont d'un intervalle des directions des fracturations est requise. Celle-ci peut être fournie par le géologue.

— Elle est moins précise que la détection semi-automatique utilisant la FMM. En effet, après examination des résultats, on a pu remarquer que quelques fractures n'ont pas pu être détectées. Cependant, une optimisation du programme pourrait réduire ce manque de précision et rendre cette méthode plus efficace.

Finalement, cette méthode constitue une réelle alternative à la Fast Marching Method, surtout si on cherche un moyen de détection des fracturations qui soit à la fois automatique et applicable sur de larges volumes de données.

7 Conclusion et Perspectives

On présente dans ce rapport une étude sur la détection de zones de fracturations à partir de données Lidar fournies par le Réseau d'Observation du Littoral. Trois sites ont été étudiés : Fécamp, Villers-Sur-Mer et Quiberville.

L'utilisation de la Fast Marching Method a permis, en plus d'obtenir des résultats très précis, d'éviter un grand nombre d'inconvénients, notamment, ceux liés au coût et au temps de calcul. Cependant, l'aspect « semi-automatique » constitue un frein non négligeable à cette méthode, surtout dans le cas où l'étude de détection de fracturations porte sur de grandes zones.

Afin de remédier à ce problème, une deuxième méthode permettant cette fois-ci une détection automatique des fracturations, a donc été mis en place. Celle-ci se base essentiellement sur une étude de contours et l'application de filtres, à savoir, le filtre de Sobel.

Étant une alternative très intéressante à la Fast Marching Method, cette dernière n'en est pas moins privée d'inconvénients. En effet, malgré son aspect automatique et autonome, elle manque quand même de précision comparée à la méthode citée plus haut. Une des perspectives pour ce stage serait alors de tenter d'optimiser le programme derrière cette méthode, permettant ainsi une détection d'une part précise, et d'autre part automatique des fracturations.

8 Références bibliographiques

8.1 Level Set Method

- Frederic Gibou, Ronald Fedkiw et Stanley Osher, « *A review of level-set methods and some recent applications* » , 15 Janvier 2018 :
https ://www.sciencedirect.com/science/article/pii/S0021999117307441

- Ilda Reis, Joao Manuel R. S. Tavares, Renato Natal Jorge (University of Porto), « *An Introduction to the Level Set Methods and its Applications* » , Avril 2012 :
https ://www.researchgate.net/publication/37650267_An_Introduction_to_the_Level_Set_Methods_and_its_Applications

- Gilbert Strang (Massachusetts Institute of Technology), « *Level Sets and the Fast Marching Method* », 2006 :
https ://ocw.mit.edu/courses/mathematics/18-086-mathematical-methods-for-engineers-ii-spring-2006/readings/am57.pdf

- Paul Vigneaux (Université de Bordeaux), « *Méthodes Level Set pour des problèmes d'interface en microfluidique* », 12 Juillet 2007 :
https ://tel.archives-ouvertes.fr/tel-00189409/document

- Charles Dapogny et Emmanuel Maitre (Université Joseph Fourier), « *An introduction to the Level Set Method* », 2018 :
http ://www-ljk.imag.fr/membres/Charles.Dapogny/cours/CoursLS.pdf

- James A. Sethian (Department of Mathematics, University of Califorina, Berkeley), « *Fast Marching Methods and Level Set Methods for Propagating Interfaces* », 2006 :
https ://math.berkeley.edu/~sethian/2006/Papers/sethian.vonkarman_1.pdf

- James A. Sethian et David Adalsteinsson (Department of Mathematics, University of Califorina, Berkeley), « *A Fast Level Set Method for Propagating Interfaces* », Septembre 1994

- EL ARWADI Toufic (Université Paul Verlaine), «*La méthode Level Set avec préservation topologique* », 2005/2006 :
https ://www.researchgate.net/profile/Toufic_El_Arwadi/publication/280612028_La_methode_Level_S

8.2 Fast Marching Method

- Nicolas Forcadel (Université Paris-Dauphine), « *Méthodes numériques pour les équations d'Hamilton-Jacobi et les lois de conservations hyperboliques, Fast Marching Method* », 2008 :

https ://uma.ensta-paris.fr/files/zidani/CEA-EDF-INRIA08/FMM_Forcadel.pdf

- Régis Monneau, « *Introduction to the Fast Marching Method* », 2010 :
https ://hal.archives-ouvertes.fr/hal-00530910/document

- Romain Echegut et Olivier Vitry (Université d'Orléans), « *Implémentation de la Fast Marching Method pour la segmentation de cervelet de souris trisomiques sur image RMN » :*
https ://idpoisson.fr/louchet/teaching/timo/Vitry-Echegut.pdf

- Luis Moreno et Santiago Garrido (Universidad Carlos III, Madrid),« *Fast Marching Method : Application of the Eikonale equation in path planning problems* », 2016 :
https ://canal.uned.es/uploads/material/Video/49784/Presentaci__n_Luis_Moreno.pdf

- Nicolas Forcadel, Carole le Guyader et Christian Gout (INSA de Rouen),« *Generalized Fast Marching Method : Applications to Image Segmentation* »

8.3 Filtre Sobel

- Wikipédia, « *Sobel Operator* » :
https ://en.wikipedia.org/wiki/Sobel_operator

- OpenCV, « *Sobel Derivatives* » :
https ://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html

- Samta Gupta, Susmita Ghosh Mazumdar, « *Sobel Edge Detection Algorithm* »,2013